ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

A THESIS PRESENTED TO THE
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FULFILLMENT OF THE THESIS
REQUIREMENT FOR THE DEGREE OF
PHILOSOPHIAE DOCTOR IN ENGINEERING
Ph.D.

BY
LUIZ EDUARDO SOARES OLIVEIRA

AUTOMATIC RECOGNITION OF HANDWRITTEN NUMERICAL STRINGS

MONTREAL, 4TH JULY 2003

THIS THESIS WAS EVALUATED

BY THE COMMITTEE COMPOSED OF :

Dr. Robert Sabourin, Thesis Supervisor
Département de Génie de la Production Automatisée, École de Technologie Supérieure

Dr. Ching Y. Suen, Thesis Co-Supervisor
Centre for Pattern Recognition and Machine Intelligence, Concordia University

Dr. Tonny Wong, President
Département de Génie de la Production Automatisée, École de Technologie Supérieure

Dr. Yoshua Bengio, External Examiner
Departement d'Informatique et Recherche Opérationnelle, Univeristé de Montréal

Dr. Jacques De Guise, Examiner
Département de Génie de la Production Automatisée, École de Technologie Supérieure

THIS THESIS WAS DEFENDED IN FRONT OF THE EXAMINATION

COMMITTEE AND THE PUBLIC

ON 18TH JUNE 2003

AT THE ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

LA RECONNAISSANCE DES CHAÎNES NUMÉRIQUES MANUSCRITES DE LONGUEUR VARIABLE

Luiz Eduardo Soares Oliveira

SOMMAIRE

La lecture automatique de champs numériques a été envisagée dans plusieurs domaines tels que le traitement de chèques bancaires, la reconnaissance de codes postaux, et le traitement de formulaires qui constituent un large éventail des problèmes implicites à la reconnaissance de l'écriture manuscrite. Par la suite, la recherche autour de cette problématique s'est rapidement développée dès lors que la puissance de calcul et les ressources de mémoire sont devenus disponibles à moindre frais.

Dans le cadre de cette thèse, nous présentons un système modulaire de reconnaissance de chaînes numériques manuscrites. Celui-ci utilise une approche "segmentation-reconnaissance" pour la segmentation, une stratégie de reconnaissance et une stratégie de post-traitement pour la vérification. Le système combine les sorties issues des modules de segmentation, de reconnaissance et de vérification au sein d'un même modèle probabiliste. Un nouveau schéma de vérification que composent deux vérificateurs traitant respectivement les cas de sur-segmentation et de sous-segmentation est présenté. Dans le but de vérifier les cas de sur-segmentation, un nouveau jeu de caractéristiques est considéré. L'étape de post-traitement utilisant un automate déterministe en association avec le module de décision globale permet le rejet ou l'acceptation de l'image traitée. Enfin, des résultats très significatifs ont été obtenus sur des images de montants numériques de chèques Bresiliens. Afin de montrer la robustesse et la moularité du système, ce dernier est testé sur la base de données de chiffres manuscrits NIST SD19.

Plus loin, nous discutons différentes stratégies que nous avons explorées dans le but d'améliorer la performance et la fiabilité du sytème. Nos efforts portent particulièrement sur la vérification, la sélection de caractéristiques et les ensembles de classifieurs. L'analyse des résultats d'expériences obtenus aussi bien sur les chiffres manuscrits isolés et sur les chaînes de chiffres, met en relief la difficulté inhérente au problème de reconnaissance de chaînes numériques manuscrites.

LA RECONNAISSANCE DES CHAÎNES NUMÉRIQUES MANUSCRITES DE LONGUEUR VARIABLE

Luiz Eduardo Soares Oliveira

RÉSUMÉ

L'objet de notre travail est la reconnaissance de chaînes numériques manuscrites rédigées sans contraintes. Des chiffres manuscrits écrits sans contraintes ne sont pas transcrits soigneusement, ni écrits dans des emplacements isolés et ne respectent pas forcément une fonte spécifique. D'où le défis de reconnaître des images de nombres ou de montants que l'on peut trouver dans la vie réelle telles que des images de montants numériques de chèques ou des images de valeurs numériques dans les formulaires. Parmi les difficultés inhérentes à la reconnaissance de chaînes de chiffres figurent : le bruit, les chiffres fragmentés, les chiffres qui se recouvrent, les chiffres non isolés et la longueur inconnue des chaînes de chiffres. Aussi, la lecture des montants numériques inscrits sur les chèques peut compliquer davantage la tâche de reconnaissance en présentant des cas de classes non numériques, l'ambiguité entre les caractères non numériques et les chiffres, et la présence de ligatures éventuelles qui lient les chiffres entre eux.

Ainsi, la difficulté de notre application dépend tout aussi bien de la reconnaissance de caractères individuels qu'à la séparation des caractères afin de les isoler les uns des autres dans la chaîne, lequel processus est appelé segmentation. Deux approches différentes de reconnaissance sont bien connues dans la littérature : l'approche de "segmentation puis reconnaissance" et l'approche "segmentation guidée par la reconnaissance"(ou segmentation-reconnaissance). La première fournit une seule hypotèse de séquence, ou chaque composante de la séquence devant contenir un seul caractère est soumise au module de reconnaissance. La seconde quant à elle, est basée sur un formalisme probabiliste qui exprime la meilleure combinaison des scores de segmentation et de reconnaissance de l'image présentée à l'entrée su système. Bien que cette approche soit plus fiable que la première, son inconvénient majeure réside dans la complexité de calcul nécessaire à l'évaluation de toutes les hypothèses de segmentation possibles. En outre, le module de reconnaissance doit distinguer les différentes configurations possibles telles que : un chiffre fragmenté, un caractère isolé ou bien des caractères connectés. Dans cette stratégie, la segmentation peut être explicite lorsqu'elle utilise des règles de segmentation ou bien implicite lorsque chaque colonne de pixels est un point de segmentation potentiel.

Des travaux ultérieurs ont montré que l'approche "segmentation-reconnaissance" utilisant une segmentation explicite est plus performante que les autres. Toutefois, la précision du système reste sujet à la fiabilité de l'algorithme de segmentation (heuristiques) et à la capacité du module de reconnaissance à distinguer correctement un caractère segmenté d'un fragment de caractère (sur-segmentation) ou d'une chaîne de caractères (sous-segmentation). A la lumière de tout cela, le principal objectif de notre recherche est de développer un système de reconnaissance modulaire pour la reconnaissance de chaînes de chiffres manuscrits. Ce système intègre une approche de "segmentation-reconnaissance" où une segmentation explicite détermine les régions de découpage et fournit une représentation spatiale de la solution. Comme argumenté plus haut, notre système doit être en mesure de distinguer et de détecter les aléas éventuels que peuvent causer les phénomènes de "sur-segmentation" et de "sous-segmentation" de la chaîne, pendant la reconnaissance. Dans cette optique, nous avons proposé une stratégie basée sur la reconnaissance et la vérification. Ne considérant aucune hypothèse à priori sur l'entrée, le module de reconnaissance utilise un classifieur ad-hoc pour produire une hypothèse que le vérificateur peut soit entériner ou rejeter. Un second volet de notre recherche concerne l'optimisation du système. Pour ce faire, un effort particulier est porté sur la vérification, la sélection de caractéristiques et les ensembles de classifieurs. Les résultats d'expériences réalisées sur des chiffres isolés et sur des chaînes de chiffres, mettent en relief la difficulté associée à l'amélioration de la performance d'un tel système. Les contributions originales de ce travail sont résumées comme suit :

- Un algorithme de segmentation explicite basé sur la relation de complémentarité de différents jeux de caractéristiques structurelles, à savoir, le contour, le profil et le squelette.

- Nous avons démontré comment la vérification peut améliorer la reconnaissance et la fiabilité du système en détectant les phénomènes de sous-segmentation et de sur-segmentation. La stratégie proposée est basée sur deux vérificateurs. Le premier gère les cas de sur-segmentation, tandis que le deuxième gère les cas de sous-segmentation. Un nouveau jeu de caractéristiques qui utilise l'analyse de concavités multi-niveau et l'information contextuelle a été développé pour la détection des cas de sur-segmentation dans le premier vérificateur. Nous mettons à l'avant aussi les concepts de modularité, de niveaux de vérification, et montrons comment un tel système peut être adapté à des applications différentes.

- À la lumière de la performance des vérificateurs discutés ci-haut, nous avons élaboré une autre stratégie de vérification, que nous avons appelé "vérificateur haut-niveau", qui vise à améliorer la fiabilité du sysème. Pour ce faire, deux stratégies ont été présentées : vérificateur un-A-un et vérificateur absolu. S'en suit une analyse d'erreurs qui vise à identifier sous quelles conditions expérimentales le vérificateur haut niveau est le plus pertinent.

- Nous avons introduit une méthodologie de sélection de caractéristiques qui optimise une fonction objective multi-critère avec un algorithme génétique qui produit un ensemble de solutions alternatives et une méthode de validation croisée pour indiquer le meilleur compromis précision/complexité. Nous démontrons l'importance de la validation croisée lorsqu'un ensemble de solutions alternatives est considéré. Le résultat de classification est fourni par un réseau de neurones en conjonction avec une analyse de sensibilité.

- Nous proposons dans un autre chapitre, une stratégie de génération d'un ensemble de classifieurs basée sur la sélection de caractéristiques. Un tel schéma a comme entrée les solutions différentes que génère l'algorithme de sélection de caractéristiques. La procédure exploite le fait que des sous-ensembles de caractéristiques différents, issus du même jeu de caractéristiques, peuvent montrer un pouvoir discriminant appréciable en les combinant et ensuite améliorer la performance du système. Afin de trouver les meilleurs classifeurs de l'ensemble, une recherche de deuxième niveau est appliquée.

Nous présentons les résultats d'expérience sur des bases de données de montants numériques et de chiffres manuscrits (NIST SD19) pour démontrer l'apport de la modularité du système de reconnaissance et l'importance de la vérification sur la performance de celui-ci. Le système montre des taux de reconnaissance élevés à un seuil de rejet nul. Il exhibe enfin un compromis erreur/rejet très encourageant. Les résultats de notre système se comparent favorablement aux autres méthodes plubliées récemment dans la littérature.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Robert Sabourin and Dr. Ching Y. Suen for supervising my work over these last years. Special thanks to my early supervisor back in Brazil, Dr. Flávio Bortolozzi, who motivated me to pursue my studies.

My thanks go to Dr. Edouard Lethelier, Dr. Mounim El-Yacoubi, and Nadia Benahmed for their important contribution in this work. I'm grateful also to all the friends at LIVIA, CENPARMI, and PUCPR for their help and friendship that enriched my study these years.

I would like to thank the members of my examining committee: Dr. Tony Wong, Dr. Jacques De Guise, and Dr. Yoshua Bengio. Their comments helped to improve the quality of the final version of this thesis.

Most of all, thanks to my wife, Dr. Marisa Morita, who brought me peace, made me smile, and gave me strength all the way through. This work is dedicated to her and to our daughter Isabela.

This research has been supported by research grants from Fundação Araucária, CENPARMI, and NSERC of Canada.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND SYMBOLS

AD	Application Dependent
BP	Basic Point
CC	Connected Component
CI	Contextual Information
CP	Contour Point
CENPARMI	Centre for Pattern Recognition and Machine Intelligence
CPU	Central Processing Unit
GA	Genetic Algorithm
HMM	Hidden Markov Model
HSF	Handwritten Sample Forms
ICL	Initial Concavity Level
IP	Intersection Point
NN	Nearest Neighbor
MCA	Multi-Level Concavity Analysis
MLP	Multi-Layer Perceptron
MNIST	Modified NIST
MOGA	Multi-objective Genetic Algorithm
NIST	National Institute of Standards and Technology
NP	Nondeterministic Polynomial time
OCR	Optical Character Recognition
PDA	Personal Digital Assistants

PP	Post-Processor
RR	Recognition Rate
SS	Square Error
SI	String Image
SD	Special Database
SP	Segmentation Path
SVM	Support Vector Machines
TD	Task Dependent
TP	Terminal Point
a_{ij}	State transition probability
$b_i(k)$	Segmentation graph in the Viterbi algorithm
B	Test set
C_i	States in a segmentation graph
$d(\cdot)$	Euclidean distance
$\delta_t(i)$	Survivor score in i_t
e	Classifier or expert
e_3	Classifier trained with three non-numerical classes
e_{10}	Classifier trained with ten numerical classes
e_{13}	Classifier trained with both non-numerical and numerical classes
E_t	Estimation of the thickness of the stroke
η	Learning rate term
I	Image after pre-processing
I^*	Single optimal path

$I(p)$	Intensity of a pixel (0 or 1)
$i_{b_{(k)}}$	Initial segment of the image
$i_{e_{(k)}}$	Final segment of the image
Λ	Set of specified patterns
$\widehat{M}$	The most probable interpretation of the written amount M
μ	Momentum term
N_B	Number of images in the test set
N_{rej}	Number of samples rejected by the system
N_{rec}	Number of samples correctly classified by the system
N_{err}	Number of samples misclassified by the system
$Net_{iteration}$	Number of epochs before reducing the learning rate
Net_{step}	Constant used to reduce the learning rate
ω_i	Mutually exclusive set of the pattern space
Ω	Pattern space
p	Number of decision variables
$P(\cdot)$	Probability
$proj(BP_{ik})$	Vertical projection of BP at the step k
PP	Post-processor
π	Initial state probability
$\Psi_t(i)$	Survivor terminating in state i_t
q	Desired number of distinct Pareto-optimal solutions
$r_{b_{(k)}}$	Initial segmentation rule
$r_{b_{(k)}}$	Final segmentation rule

S	Segmentation term
S_i	Sensitivity of the network model
$Sh()$	Sharing function
σ_{share}	maximum distance allowed between two individuals to become a member of a niche
$(T_0, \ldots, T_n)$	Thresholds used for rejection
T_{error}	Fixed error rate
$T(p)$	Number of neighbors of a pixel p
v_o	Over-segmentation verifier
v_u	Under-segmentation verifier
V	Variant of a written amount
x	Input pattern

INTRODUCTION

Handwriting consists of artificial graphical marks on a surface and its purposes are to communicate something and also to expand the human memory. It is a skill closely related to the human personality, which explains its great variability. In spite of all this variability, by the time they are five years old, most of the children can recognize digits and letters. Small characters, large characters, or rotated, all are easily recognized by the young. The characters may be written on a cluttered background, on crumpled paper or may even be partially occluded. We take this ability for granted until we face the challenge of teaching a machine how to do the same. After 50 years of research, design of a general handwriting recognition machine remains an elusive goal.

At the beginning of a new millennium, technology has brought handwriting to a crossroads. Nowadays there are several ways to expand human memory and to facilitate communication. In the light of this, one may ask: Will handwriting be threatened with extinction? We could say no, and the reason that handwriting persists in the age of the digital computer is the convenience of paper and pen as compared to keyboards for numerous day-to-day situations. For example, students in a classroom are still not typing on a notebook computer. They store language, equations, and graphs with a pen on a paper. This typical paradigm has led to the concept of pen computing, where the keyboard is an expensive and nonergonomic component to be replaced by a pentip position sensitive surface superimposed on a graphic display that generates electronic ink. This led us to two different approaches: off-line and on-line handwriting. In the former the data are converted to digital format by scanning the writing on paper while in the latter the data are acquired by writing with a special pen on an electronic surface.

On-line systems for handwriting recognition are available in hand-held computers such as personal digital assistant (PDA). The performance of PDA is acceptable for processing hand-printed symbols. Off-line systems are less accurate than on-line systems because the temporal information is lost. However, they may have a significant economic impact on specialized domains such as interpreting handwritten postal address on envelopes, and reading numerical amounts on forms. The latter is the subject of this thesis.

A considerable number of paper-based documents are processed daily by computers all over the world in order to handle, retrieve, and store information. The great bulk of them are still processed manually by human operators, the most common and labor-consuming operation being document amount reading and typing. A common way to automate this process is to replace the human operator with an off-line handwriting recognition system which is able to do the operator's job. Such a system gets access to both theoretical and practical achievements of different fields such as pattern recognition, document analysis, machine learning, and artificial intelligence. Over the past years, several techniques and methodologies have been proposed in order to build faster and more reliable systems. However, notwithstanding all efforts made in this field, there is still a considerable gap between human and machine performances. Thus, the big challenge in this field is to make computers approach human performance.

Problem Statement

The focus of our work is the recognition of unconstrained handwritten numerical strings. Unconstrained handwritten numerals are handwritten numerals that are not written in separate boxes, nor written neatly, nor written with a specific type of pen. Thus, the challenge is to recognize numbers written by people in real-life situations such as numerical amounts on cheques and numerical values on forms.

Some possible difficulties contributing to the unsatisfactory performance of many methods for recognizing handwritten numerical strings are: noise, broken digits, overlapping digits, touching digits, and unknown length of the string. Figure 1 depicts some of these problems.

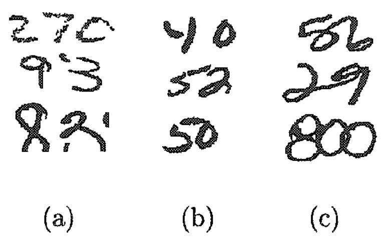


Figure 1 Examples of difficult problems when recognizing strings of digits: (a) noise, (b) broken digits, and (c) overlapping and touching digits.

When dealing with some specific applications such as numerical amounts on bank cheques, other concerns should be added to those discussed above, e.g., non-numerical classes, possibility of expressing the same amount through different variants, ambiguity among non-numerical characters and digits, and ligatures between digits and non-numerical characters. Figure 2 exemplifies some of these difficulties.

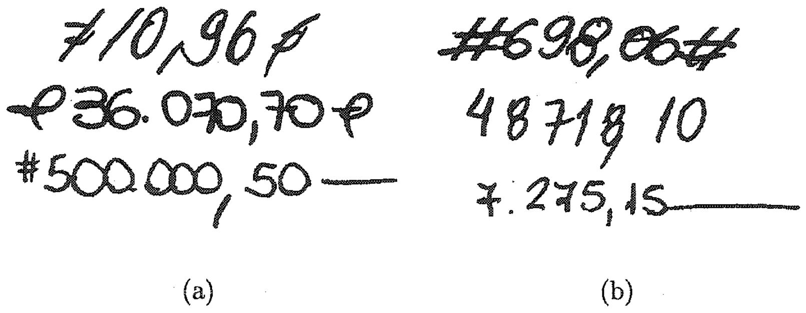


Figure 2 Examples of numerical amounts: (a) non-numerical at the beginning and the end, and (b) digits connected to non-numerical classes.

On top of all this, we still have to deal with the intrinsic ambiguity of handwritten digits, which may vary depending on the database being considered. Figure 3 exhibits such an ambiguity for the two databases used in this work. It can be observed from this Figure that the samples extracted from the numerical amounts of Brazilian bank cheques have more variability when compared to those extracted from NIST.

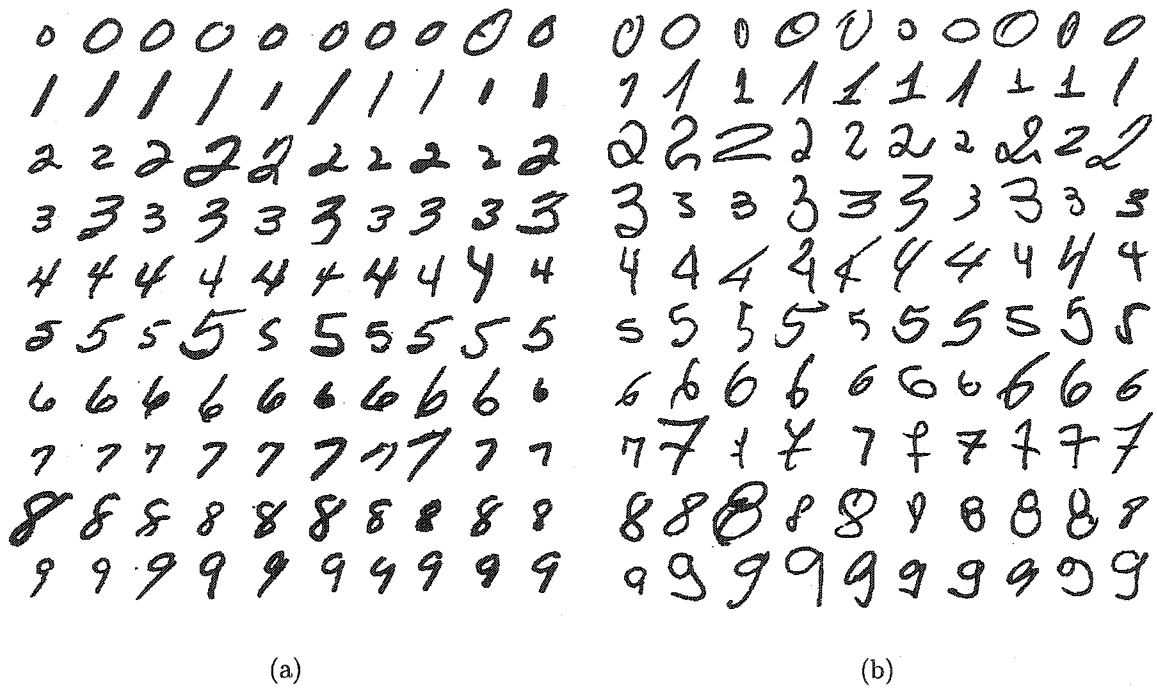


Figure 3 Handwritten digits extracted from: (a) NIST SD19 and (b) numerical amounts of Brazilian bank cheques.

Summarizing, the difficulties of handwriting recognition do not lie only in recognizing individual characters, but also to separate out characters from their neighbors within the string, a process known as segmentation. The literature shows us two different approaches for digit string recognition: segmentation-then-recognition and segmentation-based recognition. In the first approach, the segmentation module provides a single sequence hypothesis where each sub-sequence should contain an isolated character, which is submitted to the recognizer. The second strategy is based on a probabilistic assumption where the final decision must express the best segmentation-recognition score of the input image. Although this latter approach gives a better reliability than the previous one, the main drawback lies in the computational effort needed to compare all the hypotheses generated. Moreover, the recognition module has to discriminate various configurations such as fragments, isolated characters and connected characters. In this strategy, segmentation can be explicit when based on cut rules [22, 122] or implicit when each pixel column is a potential cut location [14, 95].

The literature has shown that segmentation-based recognition systems using explicit segmentation (sometimes also called heuristic over-segmentation [90]) have yielded better results than other approaches. However, in such an approach, the accuracy of the system depends upon the quality of the cuts generated by the segmentation algorithm (heuristics), and on the ability of the recognizer to distinguish correctly segmented characters from pieces of characters (over-segmentation) and multiple characters (under-segmentation). In our work we address the foregoing problem by using a strategy based on Recognition and Verification.

Goals of the Research

The primary goal of our research is to develop modular recognition system for handwritten numerical strings. This system takes a segmentation-based recognition approach where an explicit segmentation algorithm determines the cut regions and provides a multiple spatial representation. As stated before, such a system has to solve a crucial problem: distinguishing, at the recognition stage, a sequence corresponding to an inter-character segmentation from another relative to an intra-character segmentation. In order to deal with this problem we have proposed a strategy based on Recognition and Verification where the recognition function takes into account only a general-purpose recognizer while the verifiers evaluate the result produced by the recognizer.

A second aspect of our research lies in the optimization of the handwriting recognition system. To accomplish this, our efforts are geared towards the verification, feature selection, and ensemble of classifiers. Comprehensive experiments carried out on both isolated digits and strings of digits emphasize the difficulties in improving the performance of handwritten numerical string recognition systems.

Contributions

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

- An explicit segmentation algorithm which is based on the relationship of complementary sets of structural features, namely, contour, profile, and skeletal points. In [122] such an algorithm was introduced. Some results using combination of different feature sets and combination of classifiers was presented in [121]. Finally, we developed a strategy based on graphs to manage and generate all segmentation hypotheses [120].
- We demonstrated how the verification can improve the recognition and reliability rates of the system by detecting over- and under-segmentation. Firstly, we have experimented a new scheme of verification based only on one verifier [124]. Afterwards, we realized that a strategy based on two verifiers would be more reliable. The first verifier tackles over-segmentation while the second one deals with under-segmentation. A new feature set, which takes into account multi-level concavity analysis and contextual information, was developed to feed the over-segmentation verifier. We present also the concept of modular recognition system, levels of verification, and show how such a recognition system can cope with different applications. In [125] we present a complete description of the Recognition and Verification strategy.

- In light of the success obtained by the verifiers discussed above, which we have called low-level verification, we elaborate another strategy of verification, called high-level verifier, to improve the reliability of the system. Two different strategies were developed: absolute and one-to-one verifiers. A thorough error analysis is also presented in order to identify the conditions under which high-level verification is more appropriate. This work appeared in [123] and as extended version in [128].

- The issue of feature selection was first addressed in [119]. In this paper we discussed different strategies of genetic algorithms such as simple and iterative generic algorithm to perform feature selection for handwriting recognition. Thereafter, we realized that the use of a multi-objective genetic algorithms could be much more interesting since it can provide a set of different solutions (subsets of features) rather than just one solution. So we have introduced a methodology [126], that considers an efficient multi-objective genetic algorithm to generate a set of alternative solutions and a cross-validation method to indicate the best accuracy/complexity trade-off. We demonstrate that the cross-validation is very important when working with a set of alternative solutions. The classification accuracy is supplied by a neural network in conjunction with the sensitivity analysis. An extended version of this work will appear in [129].

- A strategy for generating an ensemble of classifiers based on feature selection was proposed in [127]. Such a scheme receives as input the different solutions provided by the feature selection algorithm. The idea is to take advantage of the fact that the feature selection algorithm yields different subsets of features with a high discriminant power, which use different parts of the same feature set. In order to find the best classifiers to compose the ensemble, a second level of search is performed.

- As marginal contributions, we have demonstrated the reliability and applicability of our system by using it to recognize strings of digits in different applications such as day and year on handwritten dates [113]. We also built a database of about 2,000 images of Brazilian bank cheques, which we have used to evaluate the proposed system. The specifications of this database are reported in [39] and Appendix 1.

Outline of the Thesis

This document consists of eight chapters (including the introduction and the conclusion) and three appendixes. The current chapter outlined the problem we are working with, the goals, and the main contributions of the thesis. A review of the state of the art for off-line handwriting recognition is given in Chapter 1.

In Chapter 2, we present a brief overview of the system in order to be able to introduce all definitions related to the system. Firstly, the probabilistic model that the system is based on is introduced. Then, the classifiers, training method, and decision rules are defined. Finally, the definitions about levels of verification and modularity are presented.

In Chapter 3 we describe the two first modules of the system, namely component detection and segmentation. The former performs a very simple smoothing operation before detecting all the components in the image. Then, such components are used as input to the segmentation module, which produces a list of segmentation hypotheses that will be evaluated by the recognizer.

The Recognition and Verification strategy is addressed in Chapter 4. In this chapter all modules defined in Chapter 2 are described in detail. In addition, we present the feature set used by each classifier and also describe how the low-level verifiers and general-purpose recognizer interact with each other. In order to support the ideas proposed in this thesis, Chapter 5 presents a series of comprehensive experiments, which are carried out on two different databases: numerical amounts and NIST SD19.

Chapter 6 reports our efforts towards the performance and reliability of the general-purpose recognizer. In this chapter we discuss three well known and established techniques we have investigated: high-level verification, feature selection, and ensemble of classifiers. Finally, the final chapter concludes this thesis and shows an outlook on the future works as well.

We have assumed that the reader is familiar with the theory of pattern recognition, neural networks, and genetic algorithms. For this reason, we do not devote much attention to such subjects. For those who do not feel comfortable in these fields we would suggest the reading of the following references: [33] for pattern recognition, [9, 38] for neural networks, [48, 65] for genetic algorithms, and [30] for multi-objective genetic algorithms.

CHAPTER 1

STATE OF THE ART

In this chapter we present the state of the art of handwritten isolated digit recognition and handwritten numerical string recognition, which are the two main fields related to our research. First of all we present a very brief historic about character recognition, then we address the problem of isolated digits. Thereafter, we discuss the problem of handwritten numerical string recognition and finally some applications of handwriting recognition.

1.1 Brief History

An interesting view about the evolution of the character recognition systems is presented in [4]. The authors argue that such systems have been evolved in three ages: early ages, developments, and advancements. In the early ages (1900-1980), the first character recognizers appeared in the middle of the 40s with the development of digital computers. At that time, the works were concentrated either upon machine-printed text or upon a small set of well-distinguished handwritten text or symbols. Successful, but constrained algorithms have been implemented mostly for Latin characters and numerals [149]. The commercial character recognizers were available in the 1950s, when electronic tables capturing the x-y coordinates data of pen-tip movement was first introduced [153].

The developments age took place from 1980 to 1990. With the explosion of information technology and consequently easier access to personal computers, the previously methodologies found a very fertile environment for rapid growth in many application areas. Structural approaches [145] were initiated in many systems in addition to the statistical methods [70]. The character recognition research was focused basically on the shape recognition techniques without using any semantic information. This led to an upper limit in the recognition rate, which was not sufficient in many practical applications. The state of the art of this period can be found in [112] and [153] for off-line and on-line cases respectively.

The real progress in character recognition was achieved in the advancements age (after 1990). In the early 1990s, image processing and pattern recognition were efficiently combined with artificial intelligence techniques. Efficient tools such as neural networks, support vector machines, hidden Markov models, fuzzy set reasoning, and natural language combined with more powerful computers and more accurate electronic equipments have provided quite satisfactory results for restricted applications [134]. However, there is still a long way to go in order to reach the ultimate goal of machine simulation of fluent human reading, especially for unconstrained off-line handwriting.

1.2 Handwritten Isolated Digit Recognition

Many researchers have tried to improve the performance of recognition systems by investigating features, classification methods, and different system architectures such as multi-experts and verification modules.

Selection of a feature extraction method is probably the single most important factor in achieving high recognition performance. For this reason, such a subject has gained considerable attention of the scientific community. A good survey about feature extraction can be found in [156]. The literature of handwritten digit recognition shows us basically three classes of features.

The first class are the gray-level or binary values of all the pixels in an image, usually represented by a N-dimensional vector, where N is the number of pixels in the image. Since no abstraction is applied, all the variances among the patterns are to be handled by a classification algorithm. One fast evolving approach is using pixel features as direct input to neural networks [59, 90]. The reason of success can be explained by the fact that neural networks also act as a feature extractor during the learning.

Features of the second class are the structural features of the image, which are typically perceptual entities of the character such as bends, end points, intersections, loops, measures of concavity, distance information, and directional features [67, 93, 116, 118, 151, 169]. The last class of features are the statistical features. They are results of global transformations on an image. Typical mathematical transformations include moments [6, 68], Fourier descriptors [144], and wavelet transforms [27, 133].

In order to get more reliable systems, many researchers have turned towards the combination of statistical and structural features in a same structure of classification (e.g., in a one-shot classifier). Cai and Liu in [18] present an approach that integrates both statistical and structural information for the recognition of unconstrained handwritten numerals. Heutte et al [60] propose a combination of seven different families of features to feed a linear discrimination based classifier. The performances of these systems are reported in Table I.

Alongside these investigations of feature extraction methods, several works have addressed classification methods. Different classifiers have been used for handwritten digit recognition, such as (i) template matching, (ii) statistical techniques, (iii) structural techniques, and (iv) neural networks. Template matching operations determine the degree of similarity between two vectors (groups of pixels, shapes, curvatures, etc) in the feature space. Matching techniques can be grouped into two classes: direct matching [43], deformable templates and elastic matching [71].

Statistical techniques is concerned with statistical decision functions and a set of optimality criteria, which determine the probability of the observed pattern belonging to a certain class. Several popular handwriting recognition approaches belong to this domain, as suggested in [70]:

- The k-Nearest-Neighbor (k-NN) rule is a popular non-parametric recognition method, where the posteriori probability is estimated from the frequency of nearest neighbors of the unknown pattern. Good recognition results for handwriting recognition have been reported by using this approach [53]. The problem with this method is the high computational cost when the classification is conducted. To surpass such a problem some researchers have been proposed faster k-NNs methods [108].
- The Bayesian classifier assigns a pattern to a class with the maximum posteriori probability. Class prototypes are used in the training stage to estimate the class-conditional probability density function for a feature vector [24, 33, 114].
- The polynomial discriminant classifier assigns a pattern to a class with the maximum discriminant value which is computed by a polynomial in the components of a feature vector. The class models are implicitly represented by the coefficients in the polynomial [141].
- Hidden Markov Model (HMM) is a statistical framework for modelling sequential input by state transitions. It has been widely used in speech recognition and its applications to handwritten digit recognition have been growing [14, 18, 135].
- Fuzzy set reasoning is a technique that employs fuzzy set elements to describe the similarities between the features of the characters. The literature shows different approaches such as fuzzy graphs [1], fuzzy rules [44], fuzzy logic [56], and linguistic fuzzy [87].

- One of the most interesting recent developments in classifier design was the introduction of the support vector machines (SVM) by Vapnik [157]. In the past few years, SVM has received increasing attention in the communities of machine learning and pattern recognition due to its excellent generalization performance. It is primarily a two-class classifier, but multiple SVMs can be combined to form a classification system for multi-class classification. Some SVM classification systems have been developed for handwritten digit recognition in the recent years, and some promising results have been reported [5, 17, 155].

In structural handwriting recognition the characters are represented as unions of structural primitives. It is assumed that the character primitives extracted from handwriting are quantifiable, and one can find the relations among them. Basically, structural methods can be divided into two classes: grammatical methods [145] and graphical methods [62].

In the past decade, there has been a tremendous increase of interest in neural networks as a possible solution to the problem of recognizing handwritten numerals. A neural network is defined as a computing structure consisting of a massively parallel interconnection of adaptative "neural" processors. The main advantages of neural networks are (i) the ability to be trained automatically from examples, (ii) good performance with noisy data, (iii) possible parallel implementation, and (iv) efficient tools for learning large databases. The most widely studied and used neural network is the Multi-Layer Perceptron (MLP) [9]. Such an architecture trained with the gradient descent applied to a sum-of-square error function [91] is among the most popular and versatile forms of neural network classifiers and is also among the most frequently used traditional classifiers form handwriting recognition. See [164] for a review. Other architectures include Convolutional Networks [90], Self-Organized

Maps [163], Radial Basis Function [9], Space Displacement Neural Network [106], Time Delay Neural Networks [96, 3], and Quantum Neural Networks [165].

In the recent years, combination of classifiers has been attracted theoretical and practical attention. In the long run, the combined decision is supposed to be better (more reliable) than the classification decision of the best individual classifier. Such an idea appears under a variety of names in the literature: classifiers fusion [46], classifiers combination [79, 159], mixture of experts [69], committees [9], classifiers ensembles [81, 57], etc.

The literature on theoretical classifier combination has grown rapidly [2, 79, 84, 92, 154, 159]. The papers advocate different fusion strategies, demonstrate the benefits of classifier, and provide a theoretical underpinning of the various strategies commonly used in multiple expert fusion. Some interesting results of combination scheme are also reported in practical applications. Suen et al [151] combined four structural methods using variants of majority vote. High performance was reported on CENPARMI database. Xu et al [159] used the same four classifiers and results are provided using Dempster-Shafter and Bayesian formalism. Lam and Suen [86] applied weighted voting to handwritten digit recognition using seven classifiers. Kittler et al [79] applied several classifier combination strategies such as product rule, sum rule, min rule, max rule, median rule, and majority vote to recognize handwritten numerals. Gorski [50] demonstrated that for large databases, neural networks provide compelling results as a classifier integrator.

Another strategy that can increase the recognition rate in a relatively easy way with a small additional cost is through the use of verification. Such a scheme consists of refining the top few candidates in order to enhance the recognition rate economically. Such a kind of scheme has been successfully applied to handwriting recognition in [14, 152, 167].

Table I compiles some of the results reported in the literature from 1996 until nowadays. It is important to point out that the recognition, rejection, and error rates are not directly comparable, since these results are based on different databases. The symbol "-" means that the rate was not reported by the author. For performance of the systems published before 1996, please see [94].

Table I

Compilation of some of the results reported in the literature (Rec: Recognition rate, Rej: Rejection Rate, and Rel: Reliability Rate).

Author	Year	Rec. (%)	Error (%)	Rej. (%)	Rel. (%)	Database used	No. of Images
Lee [94]	1996	97.1	2.9	-	-	CENPARMI	4,000
Gader and Khabou [45]	1996	96.3	1.0	2.7	98.9	CENPARMI	10,000
Ha and Bunke [54]	1997	99.5	0.5	-	-	NIST SD3	173,124
Cheng and Yan [23]	1998	98.5	0.9	0.6	99.1	NIST	5,278
Hu and Yan [67]	1998	96.8	0.5	2.7	99.4	NIST	10,852
Heutte et al [60]	1998	94.8	0.5	4.7	99.5	NIST SD3	53,324
Oh et al [117]	1999	97.8	2.2	-	-	CENPARMI	2,000
Liu and Nakagawa [100]	1999	98.4	1.6	-	-	CENPARMI	2,000
Park et al [132]	2000	98.2	1.8	-	-	NIST SD3	53,301
Cai and Liu [18]	2001	98.4	1.6	-	-	CEDAR	2,213
Zhou et al [166]	2001	90.0	1.0	9.0	98.9	CENPARMI	2,000
Zhang et al [162]	2001	97.8	2.2	-	-	NIST	10,000
Lu et al [103]	2002	96.4	3.6	-	-	NIST SD3	10,426
Mayraz and Hinton [107]	2002	98.3	1.7	-	-	MNIST	10,000

1.3 Handwritten Numerical String Recognition

Handwritten numerical string recognition has been a topic of intensive research in recent years due to its large number of potential applications. The recognition of numerical strings differs from that of isolated digits because it requires the segmentation of a string into separate entities representing individuals digits. It is also different from the problem of recognizing handwritten words form a dictionary in the sense that almost no contextual information is available, i.e., any digit can follow any other one. Segmentation of numerical strings is generally a difficult task because individual numerals in a string can overlap or touch each other, or a numeral can be broken into several parts.

Strategies for handwritten numerical string recognition can be divided into segmentation-then-recognition [143, 145] and segmentation-based recognition [55, 105]. In the first approach, the segmentation module provides a single sequence hypothesis where each sub-sequence should contain an isolated character, which is submitted to the recognizer. This technique shows its limits rapidly when the correct segmentation does not fit with the pre-defined rules of the segmenter. Very often, contextual information is used in the segmentation process to improve the robustness of the system.

The second strategy is based on a probabilistic assumption where the final decision must express the best segmentation-recognition score of the input image. Usually, the system yields a list of hypotheses from the segmentation module and each hypothesis is then evaluated by the recognition. Finally, the list is post-processed taking into account the contextual information. Although this approach gives a better reliability than the previous one, the main drawback lies in the computational effort needed to compare all the hypotheses generated. Moreover, the recognition module has to discriminate various configurations such as fragments, isolated characters and connected characters. In this strategy, segmentation can be explicit when based on cut rules [96, 22] or implicit when each pixel column is a potential cut location [14, 95]. A good review about segmentation can be found in [20].

Indeed, handwritten numerical string recognition is present in almost every application involving handwriting recognition, for instance, postal address [89, 77], cheque processing [96, 74, 80], and form reading [14, 55]. Such applications have been very popular in handwriting recognition research, due to the availability of relatively inexpensive CPU power, and the possibility to reduce considerably the manual effort involved in these tasks. In the following subsections we discuss two applications where handwritten numerical string recognition has been widely applied: bank cheque processing and form processing.

1.3.1 Bank Cheque Processing

Bank cheques are probably the most widespread documents. Nearly one hundred billion cheques circulate yearly all over the world [51]. The great bulk of them are still processed manually by human operators, the most common and labor-consuming operation being document amount reading and typing. Automation of bank cheque processing uses both recent theoretical achievements of pattern recognition and document analysis, and practical approaches developed in adjacent applications such as postal automation or form recognition.

Processing numerical amounts in bank cheques is a difficult task due to the nature of the handwritten material. For instance, bank cheque systems have to take into account the great variability in the representation of a numerical amount, e.g. the number of components to be identified, which is not necessary, for example, for a zip-code recognition system since the number of digits is fixed and known a priori. Another important requirement from a bank cheque system is its reliability. It has been estimated that a system becomes commercially efficient only when the error rate is 1% or lower.

Several different approaches to recognize numerical amounts on bank cheques can be found in the literature. Lethelier et al [96] present a system to recognize numerical amounts on French cheques. It is based on a segmentation-based recognition strategy where an explicit segmentation algorithm provides the cut regions. Their classifier is based on a combination of a Radial Basis Function network and a Timed Delay neural network. Besides of the ten numerical classes, this system copes with four non-numerical classes ("-", ".", ",", "F").

Kaufmann and Bunke [74] propose a system for Swiss postal cheques where the numerical amount is processed by the system presented in [55]. Since such a classifier does not tackle the cents portion and non-numerical symbols such as currency characters, horizontal lines, they are manually removed.

Dzuba et al [34] (Parascript) describe an industrial system for American personal cheques which uses a classifier based on matching input subgraph to graphs of symbol prototypes. The symbol prototype consists of the symbol graph and the description of its elements (geometrical characteristics of edges, mutual position of edges and nodes, etc.). In the prototype an obligatory subgraph is defined. The obligatory subgraph is a subgraph where each element of which should have a match in the input subgraph. Some pre-processing is done by means of Hough transformation.

Another industrial system is described in [51]. The *A2iA CheckReader*^{TM} is designed to process cheques written in French or English and it is based on some key ideas such as Hierarchical organization, soft decision-making, modularity, complementarity, adaptivity and, segmentation-based recognition. Their classifier is composed of five neural networks. Four of them are feed with different feature sets and their outputs combined into a fifth neural network. They claim that the average read rate at the document level varies from 65 to 85% with error rate fixed at 1%, but they do not specify the performance achieved by the handwritten digit classifier. The performance of some applications found in the literature, including those discussed above are reported in Table II.

It is impossible to compare the results presented in Table II, since different databases and formats are used, different non-numerical classes are involved, and different sizes

Table II

Performance of some bank cheque systems.

Author	Rec. (%)	Error (%)	Rej. (%)	Test Database	Year
Lethelier et al [96]	60.0	40.0	-	10,000 French cheques	1995
Houle et al [66]	55.0	1.0	44.0	1,000 American cheques	1996
Parascript [34]	53.0	1.0	46.0	5,000 American cheques	1997
Lee et al [98]	72.7	27.3	-	121 Brazilian cheques	1997
Suen et al [150]	62.0	1.0	37.0	400 Canadian cheques	1998
Kaufmann and Bunke [74]	79.3	20.7	-	1,500 Swiss cheques	2000

of databases are considered. However, we can observe that the recognition rates of those systems that implemented a rejection mechanism (error rate fixed at 1%) varies from 50 to 60%.

1.3.2 Form processing

Form processing such as census and tax forms also are potential application for handwritten numerical string recognition. The challenge here consists of recognizing strings of unknown length which are not neatly written. The most common database used for research in this case is the NIST database.

NIST database was originally designed for competition of First Census Optical Character Recognition Systems Conference in May 1992 organized by the National Institute for Standards and Technology (NIST). The event was for assessing the state of the art in OCR. In 1995 NIST released an upgrade called SD19 (Special Database 19) [52] (see Appendix 1). Several researchers have been used NIST databases to report results of algorithms for handwritten numerical string recognition. In the remaining of this section we discuss some of these works. The results claimed by authors are reported in Table III.

Martin et al in [104] propose the exhaustive and saccadic scan methods for integrating segmentation and recognition of handwritten strings. In the first scan method (Method 1), a neural network trained with back-propagation 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 (Method 2), 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 rates have been reported. Thus, they were calculated from the reported reject and error rates.

Keeler and Rumelhart [75] use a neural network to simultaneously segments and recognizes connected characters. Their self-organizing integrated segmentation and recognition 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. Again, the recogniton rates were calculated from the reported reject and error rates (1%).

Fujisawa et al [41] introduce 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 from a numeral string. These connected components 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.

Ha et al [55] build a system upon four main components. A pre-segmentation module divides the input numeral string into independent groups of digits which are processed by a cascade of two recognition methods. The digit detection module identifies and recognizes groups containing isolated digits and a classifier recognizes the remaining groups containing an arbitrary number of digits. The global decision module merges all results and makes an accept/reject decision. They have used about 5,000 strings of the NIST SD3 in their experiments.

Lee and Kim [95] propose a segmentation-based recognition where the segmentation is implicitly performed. They introduce a new type of cascade neural network to train the spatial dependencies in connected handwritten numerals. This cascade neural network was originally extended from MLP to improve the discrimination and generalization power. They used 5,000 strings of digits but they did not specify the used data.

In the same vein, Britto Jr. et al [14] propose a handwritten numeral string recognition method composed of two HMM-based stages. The first stage uses an implicit segmentation strategy based on string contextual information to provide multiple segmentation-recognition hypotheses. These hypotheses are verified and re-ranked by using a verification stage based on a isolated digit classifier. Such a strategy allows the use of two sets of features and numeral models: one taking into account of both segmentation and recognition aspects in an implicit segmentation-based strategy, and another considering just the recognition aspects of isolated digits. They used 12,802 strings of the NIST SD19 (hsf_7 series) and present results with and without the knowledge of the size of the string.

Table III

Recognition rates on NIST databases reported in the literature.

Authors	String Length	Number of tested strings	Zero-rejection Level%	Error Rate 2%	Error Rate 1%	Error Rate 0,5%
Ref. [41]	2	1000	89.79	-	-	-
	3	1000	84.64	-	-	-
	4	1000	80.63	-	-	-
	5	1000	76.05	-	-	-
	6	1000	74.54	-	-	-
Ref. [104] Method 1	2	1000	-	-	94.20	-
	3	1000	-	-	87.90	-
	4	1000	-	-	79.90	-
	5	1000	-	-	75.60	-
	6	1000	-	-	63.30	-
Ref. [104] Method 2	2	1000	-	-	92.60	-
	3	1000	-	-	86.30	-
	4	1000	-	-	79.50	-
	5	1000	-	-	75.80	-
	6	1000	-	-	72.20	-
Ref. [55]	2	981	96.20	94.50	93.50	91.50
	3	986	92.70	86.00	79.50	70.50
	4	988	93.20	86.50	81.00	70.00
	5	988	91.10	81.00	77.50	70.50
	6	982	90.30	80.50	75.50	66.50
Ref. [95]	2	1000	95.23	-	95.20	-
	3	1000	88.01	-	87.90	-
	4	1000	80.69	-	80.50	-
	5	1000	78.61	-	78.40	-
	6	1000	70.49	-	70.20	-
Ref. [14]	2	2370	94.81	-	-	-
	3	2385	91.61	-	-	-
	4	2345	91.25	-	-	-
	5	2316	88.30	-	-	-
	6	2169	89.07	-	-	-
	10	1217	86.94	-	-	-

As stated before, Table III summarizes the recognition rates at different error levels for the works discussed here. The symbol "-" indicates that no recognition rate has been reported for the specified error rate.

1.4 Discussion

In the foregoing sections we have presented a review of the fields related to our research. By analyzing the performances of the state-of-the-art systems, we can draw some observations:

- Researchers are getting interesting results for isolated handwritten digits. Table I shows that even for large databases (e.g., Ref. [54]) the results are compelling.
- It is difficult to carry out a deeper analysis since different databases and different sizes (e.g. from 2,000 to 173,124 in Table I and from 121 to 10,000 in Table II) are used. Sometimes the authors use just one part of the database even when the entire set is available.
- When the topic comes to numerical string recognition, performances go dramatically down due to problems such as touching digits, overlapping, and unknown number of digits. Such a fact can be clearly observed in Table III, especially for error rates fixed at low levels.
- Practical systems usually report relatively low error rates which depend on other sources such as city names in postal codes and legal amount in cheque processing.

Even with recognition rates close to 99% for handwritten isolated digits, we can affirm that there is still a considerable gap between human and machine performances. This gap is even greater when we consider the problem of strings of digits of unknown length. Nevertheless, we have seen that important contributions on handwritten numerical string recognition have been made.

Since methods based on explicit segmentation have to face touching digits usually through heuristic-based algorithms, several researchers have been investigated how to avoid it by using implicit segmentation. The first systems built upon this concept were introduced by Keeler and Rumelhart [75] and Martin et al [104]. More recently, Lee and Kim [95] used a similar idea. In all these systems, neural networks were used and they have shown to constitute a suitable framework for integrating the segmentation and recognition processes. However, the problem with these methods lies in the definition of the size of the sliding window used as input for the neural network, and corresponding scan rate. Another weakness is that when using an exhaustive scan method, too many segmentation hypotheses are yielded.

More recently, a more robust approach of implicit segmentation have been proposed in [14]. It takes into account HMMs where the implicit segmentation strategy is based on string contextual information to provide multiple segmentation-recognition hypotheses. This is a promising way to face the inherent difficulties of the numerical string recognition problem. However, this kind of systems must implement some pre-processing steps, such as slant correction and size normalization in order to provide good results.

In spite of the fact that most of the algorithms based on explicit segmentation take into consideration some heuristics, which usually are time-consuming and difficult to define, segmentation-based recognition systems with explicit segmentation still achieves better results than those mentioned above. We can cite for example Ref. [55]. Another example is the work discussed in this report. We shall see results that surpass all methods based on implicit segmentation we have found in the literature.

1.5 Summary

In this chapter we have presented a state of the art of the main topics related to our research: handwritten isolated digit recognition and handwritten numerical string recognition. We have seen that several important contributions have been made in these fields. Some recent works have been briefly described in terms of features, types of classifiers, test databases, and results. In addition, different approaches for recognizing strings of digits have been presented and discussed. In the next chapter we will introduce our system with a brief overview and present the definitions that will be used throughout this work.

CHAPTER 2

SYSTEM OVERVIEW

The system discussed in this thesis takes a segmentation-based recognition approach and a Recognition and Verification strategy. The outputs from different modules of the system such as segmentation, recognition and post-processing are combined using a probabilistic framework (described in Section 2.1.1), which adds a degree of tractability to understand the interactions among the system modules.

An explicit segmentation algorithm determines the cut regions and provides multiple spatial representation. After segmentation, different kinds of features are extracted in order to feed the recognition module, which is composed of one general-purpose recognizer and two verifiers. The goal of the verifiers is to improve the overall performance of the system by detecting over and under-segmentation. All classifiers used in the system are described in Section 2.1.2. Thereafter, the system generates a list of hypotheses through a modified Viterbi algorithm and then each hypothesis is syntactically analyzed by means of a deterministic automaton. Finally, the global decision module makes an accept/rejection decision.

Despite the fact that the verifiers presented in this work are classifiers with probabilistic outputs, they are called verifiers because they do not play the same role as the general-purpose recognizer does in the system. In Section 2.1.3 we discuss the verification and its different levels as well. In Section 2.1.4 we present the idea of modular system employed in this work.

2.1 Definitions

2.1.1 Probabilistic Model

The goal of the probabilistic model is to define a function that combines all the system modules in order to allow a sound integration of all knowledge sources used to infer a plausible interpretation. The probabilistic model that we are using has been applied to speech recognition [97], handwritten word recognition [16] and handwritten digit recognition [96]. Such a model estimates the most probable interpretation of the written amount M (noted $\widehat{M}$). Its input corresponds to an image I after pre-processing.

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

$$P(\widehat{M}|I) = \max_{M} P(M|I) \tag{2.1}$$

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

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

$$P(M|I) = \sum_{S} \sum_{V} P(M, V, S|I) \tag{2.2}$$

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

$$\sum_{S} \sum_{V} P(M, V, S|I) \approx \max P(M, V, S|I) \tag{2.3}$$

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

$$\exists V, S \quad \text{such that} \quad P(M|I) \approx P(M, V, S|I) \tag{2.4}$$

Using Bayes rule, we can write

$$P(M|I) \approx \frac{P(I|M, V, S) \times P(M, V, S)}{P(I)} \tag{2.5}$$

The probability of amount M is then:

$$P(M|I) \propto P(I|M, V, S) \times P(S|M, V) \times P(M, V) \tag{2.6}$$

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

$$P(M|I) \approx P(I|V,S) \times P(S|V) \times P(M,V) \tag{2.7}$$

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

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

$$P(I|V,S) \approx \prod_{i,k} P(i_{b_{(k)}} \dots i_{e_{(k)}} | v_k, s_k = r_{b_{(k)}} \dots r_{e_{(k)}}) \tag{2.8}$$

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

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

$$P(i_{b_{(k)}} \ldots i_{e_{(k)}} | v_k, s_k = r_{b_{(k)}} \ldots r_{e_{(k)}}) \approx P([i_{b_{(k)}} \ldots i_{e_{(k)}}] | v_k) \tag{2.9}$$

By applying Bayes rule, we have the following approximation:

$$P(M|I) \approx \prod_{i,k} \frac{P(v_k|[i_{b_{(k)}} \ldots i_{e_{(k)}}]) \times P([i_{b_{(k)}} \ldots i_{e_{(k)}}])}{P(v_k)} \times P(S|V) \times P(M,V) \tag{2.10}$$

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

2.1.2 Neural Classifiers

Although many types of neural networks can be used for classification purposes [99], we opted for a MLP which is the most widely studied and used neural network classifier. Moreover, MLPs are efficient tools for learning large databases [89]. Therefore, all classifiers presented in this work are MLPs trained with the gradient descent applied to a sum-of-squares error function [9]. The transfer function employed is the familiar sigmoid function.

In order to monitor the generalization performance during learning and terminate the algorithm when there is no longer an improvement, we have used the method of validation or early stopping. Such a method takes into account a validation set, which is not used for learning, to measure the generalization performance of the network. During learning, the performance of the network on the training set will continue to improve, but its performance on the validation set will only improve to a point, where the network starts to overfit the training set, that the learning algorithm is terminated (Figure 4). A second stop criterion is the maximum number of epochs (defined in Table V).

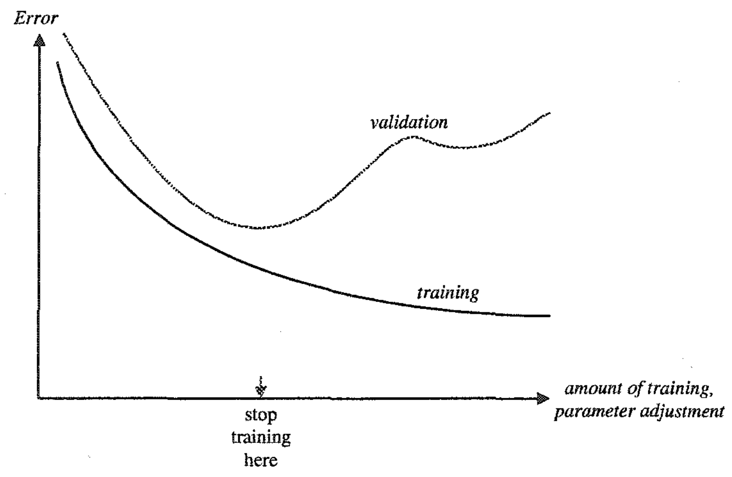


Figure 4 Using a validation database as stop criterion.

MLP is a measurement-level classifier, i.e., it attributes to each possible label a measurement value to address the degree or probability that the input sample has the label. In this work, we will interpret the measurement value as estimation of a posterior probability. In order to accurately estimate Bayesian probabilities, network output values must lie in the range (0,1) and they must sum to unity. In our MLP networks, the value of each output necessarily remains between zero and one because of the sigmoidal functions used, but the criterion used for training did not require the outputs to sum to one. Nevertheless, as shown by Richard and Lippmann in [139] the summed outputs of the MLP network are always close to one. As such, normalization techniques proposed to ensure that the outputs of an MLP network are true probabilities such as softmax [12] may be unnecessary. This is further supported by results of experiments performed by Bourlard and Morgan [10], which demonstrated that the sum of the outputs of MLP networks is near one for large phoneme-classification speech-recognition problems.

Let Ω be a pattern space which consists of N mutually exclusive sets $\Omega = \omega_1 \cup \ldots \cup \omega_N$, each of ω_i, $i \in \Lambda = \{1, \ldots, N\}$ representing a set of specified patterns called a class (e.g., $N = 10$ for digit recognition). Let x be an input pattern that should be assigned to one of the N existing classes. e means the classifier and $e(x) = (m^1(x), m^2(x), \ldots, m^N(x))$ means that the classifier e assigns the input x to each class i with a measurement value $m^i(x)$. This definition is used for all classifiers of the system. Table IV describes our classifiers as well as where they are used in the system.

Table IV

Description of the classifiers used in our system.

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

All networks presented in Table IV have one hidden layer where the units of input and output are fully connected with units of the hidden layer. The number of hidden units used in this work are 70, 80, 20, 30 and 30 for e_3, e_{10}, e_{13}, v_o, and v_u respectively. These architectures were determined empirically based on the performance on the validation set. The learning rate term (η) is set at high values in the beginning to make the weights quickly fit the long ravines in the weight space, then it is reduced each $Net_{iteration}$ epochs by multiplying itself by a constant Net_{step}. In this manner, the weights can fit the sharp curvatures. If the learning rate is very small, then the algorithm proceeds slowly, but accurately follows the path of steepest descent in weight space. If the learning rate is largish, the algorithm may oscillate ("bounce off the canyon walls"). A simple method of effectively increasing the rate of learning is to include a momentum term (μ). The idea is consistently pushing a weight in the same direction, then it gradually gathers "momentum" in that direction. Table V defines all parameters used to training the networks defined in this section.

Table V

Parameters used to train the neural networks.

Parameter	Value
Maximum Number of Epochs	200
Initial Learning Rate (η)	0.50
Momentum (μ)	0.93
$Net_{iteration}$	25
Net_{step}	0.50

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

$$e_{13}(x) = \max_{i \in \Lambda} m^i(x) \tag{2.11}$$

The goal of the verifiers v_o and v_u is to validate whether an input pattern x is an isolated character or not. We will see in Chapter 4 that this is achieved by multiplying three measurement values (classifier and both verifiers). Then, if the outputs of the verifiers that represent the posterior probability of an input pattern x be an isolated character (first output) are low, the output supplied by the classifier will be penalized, otherwise, it will be confirmed. For this reason, the decision rule used by the verifiers simply takes the measurement value produced in their first output, which contains the posterior probability of an input pattern x to be an isolated character. Thus, v_o shares the same decision rule as v_u, which is defined as:

$$v_u(x) = m^1(x) \tag{2.12}$$

2.1.3 Levels of Verification

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

Takahashi and Griffin in [152] define three kinds of verification: absolute verification for each class (Is it a "0" ?), one-to-one verification between two categories (Is it a "4" or a "9" ?) and verification in clustered, visually similar, categories (Is it a "0", "6" or "8" ?).

In addition to these definitions, we introduce the concept of levels of verification, where two levels are considered: high-level and low-level. We define as high-level verifiers those that deal with a sub-set of the classes considered by the general-purpose recognizer. The goal of the verifiers at this level is to confirm or deny the hypotheses produced by the general-purpose recognizer by recognizing them [152, 167]. We define as low-level verifiers those that deal with meta-classes of the system such as characters and parts of them. The purpose of a low-level verifier is not to recognize a character, but rather to determine whether a hypothesis generated by the general-purpose recognizer is valid or not [26].

In this work we propose two low-level verifiers to cope with the over-segmentation and under-segmentation problems. The objective of these verifiers is to validate the general-purpose recognizer hypotheses by using the following meta-classes: characters, parts of characters and under-segmented characters. Chapter 4 will present more details about these verifiers.

2.1.4 Modular System

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

We define as AD those modules that should be changed (replaced or removed) to cope with another context. The best example of an AD module is the post-processor which is usually the part of the system that has more knowledge about the application. We define as TD those modules related to the task, e.g., digit string recognition. These modules can be used for various applications. In Figure 5, the white boxes represent TD modules while the grey boxes represent AD modules.

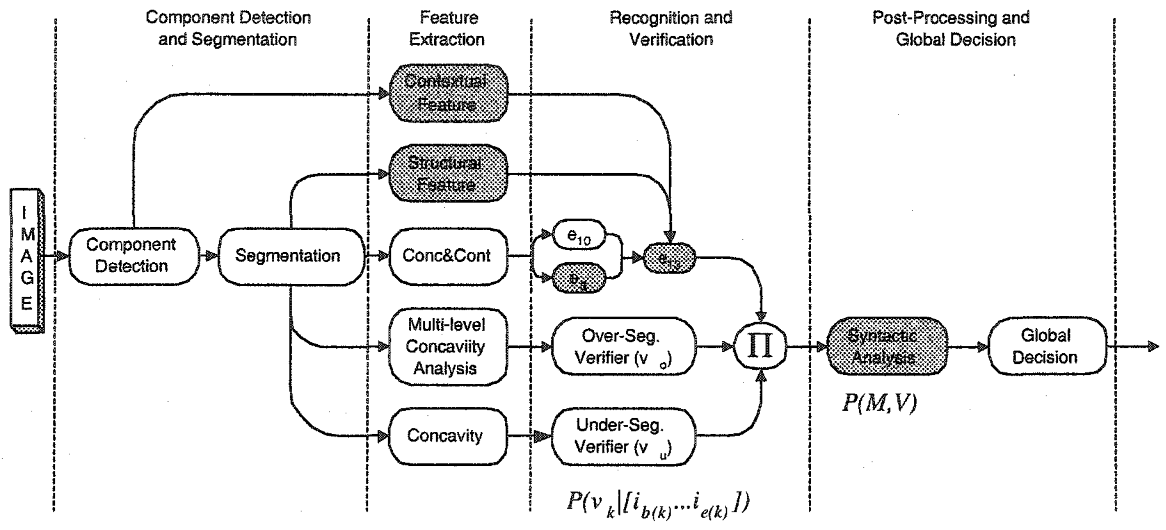

Component Detection and Segmentation
Feature Extraction
Recognition and Verification
Post-Processing and Global Decision
IMAGE
Component Detection
Segmentation
Contextual Feature
Structural Feature
Conc&Cont
Multi-level Concaviity Analysis
Concavity
e_{10}
e_3
e_{13}
Over-Seg. Verifier (v_o)
Under-Seg. Verifier (v_u)
Π
Syntactic Analysis
Global Decision
$P(M,V)$
$P(v_k|[i_{b(k)} \dots i_{e(k)}])$

Figure 5 Block diagram of a numeric string recognition system

In Figure 5 we can see two different versions of the same system. The first version, which considers all modules (white and grey boxes), was designed to process numerical amounts of Brazilian bank cheques, while the second version, which does not consider the grey boxes, was designed to process strings of digits from the NIST database. As we can notice, through the exclusion of four AD modules (*Contextual Feature*, *Structural Features*, e_3 and e_{13}) and the modification of the *Syntactic Analysis* module, we have built a new system specialized in another context. Figure 5 also presents the relationship between the system modules and its estimators. As we can see, the proposed system takes into account the estimators of recognition ($P(v_k|[i_{b_{(k)}} \dots i_{e_{(k)}}])$) and post-processing ($P(M, V)$) presented in Equation 2.10. Since our classifiers are based on neural networks, which provide direct estimation of the posterior probabilities, we decided not to use the estimator related to a priori probabilities ($P([i_{b_{(k)}} \dots i_{e_{(k)}}])$) of the images to be recognized. We will see that the verifiers can replace this estimator successfully. Lethelier et al in [96] use the empirical frequencies of segmentation configurations for each class in order to compute the probability of fragmentation of characters ($P(S|V)$). However, we have observed that our segmentation algorithm does not supply discriminative information about the classes of the system. This is illustrated in Figure 6.

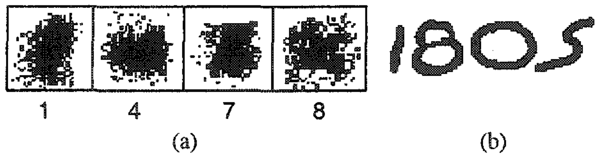


Figure 6 Problems of using segmentation estimator: (a) Distribution of the segmentation points for isolated digit classes 1,4,7 and 8 and (b) Samples of isolated digits where the segmentation will not provide any segmentation point.

Figure 6a presents the distribution of the segmentation points provided by our segmentation algorithm [122] for four different isolated digit classes (1,4,7 and 8) of the training set of the NIST database. We can observe that even for different classes the segmentation algorithm provides a very similar distribution of segmentation points, hence, it will be difficult to improve the performance of the system using the primitives generated by the segmentation algorithm. The second problem can be observed in Figure 6b. In this case, the segmentation algorithm will not produce any segmentation point due to the lack of minima and maxima on the superior and inferior contours respectively. For these reasons, we choose not to use the estimator $P(S|V)$.

2.2 Summary

In this chapter all definitions used throughout this thesis were presented. We have seen that all knowledge sources of the system are combined by means of a probabilistic model and also that all classifiers used into the system are MLPs trained with the back-propagation algorithm. In addition, the concepts about levels of verification and modular system, which are underpinning ideas in our strategy, were introduced. In the next chapter the first two modules depicted in Figure 5 are discussed.

CHAPTER 3

COMPONENT DETECTION AND SEGMENTATION

In this chapter we describe the first two modules of the system. Section 3.1 describes the component detection module which receives as input a binary image. Before detecting the components in the image, this module performs a very simple smoothing operation. Once the components have been detected, they are used as input to the segmentation module, which is discussed in Section 3.2. Such an algorithm produces a list of segmentation hypotheses that will be assessed by a recognition module.

3.1 Component Detection

The goal of this module is to divide the input numeral string, called string image (SI), into groups of components, called partial images, where each group should represent an integral number of components. This allows us to convert the recognition of a string image to that of its partial images, thus reducing the complexity of the subsequent tasks. This module operates in three steps: connected component analysis, delimiter detection, and grouping.

3.1.1 Connected Component Analysis

Before detecting the components of the image, a smoothing operation is done to regularize the edges in the image and to remove small bits of noise [148]. A 3×3 mask (see Figure 7) is passed over the entire image to smooth it. The mask begins in the lower right corner and processes each row moving upwards row by row. The pixel in the centre of the mask is the target. Pixels overlaid by squares marked "X" are ignored. If the pixels overlaid by the squares marked "=" all have the same value, i.e., all zero, or all one, then the target pixel is forced to match them, otherwise it is not changed. This test is done 4 times for each target pixel, once for each possible rotation of the mask.

X	X	X
=	T	=
=	=	=

Figure 7 3×3 mask for smoothing.

The result is that single-pixel indentations in all edges are filled and single-pixel bumps are removed. Furthermore, the mask modifies the identical image that it scans, so that lines that are one pixel thick will be completed eroded (Figure 8). This is a small price to pay for a single and efficient smoothing operation. Indeed, it is extremely rare anything other than noise is removed by this filter.

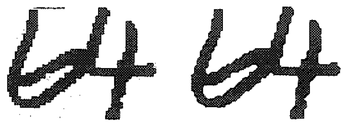

(a) (b)

Figure 8 Result of the smoothing algorithm (a) Original image and (b) Image after smoothing.

After this light pre-processing, the image is segmented into connected components (CC) and those very small are eliminated by filtering, i.e., CCs with surface smaller than 10 pixels are discarded. Other kind of CC we are interested in removing at this point is the stroke. Usually, it is found at the beginning and at the end of the numerical amount. A CC is regarded as a stroke if the following two rules are met:

1. $\frac{CC_{width}}{CC_{height}} > 4.3$.

where CC_{width} is the width of the CC and CC_{height} the height of the CC.

2. The gravity centre of the CC should be closer to the median line (SI_{median}) than either to the upper line (SI_{upper}) or to the bottom line (SI_{bottom}).

The first rule tries to identify the stroke through its basic characteristic, i.e., the relationship between width and height. Usually, strokes are much larger than higher. The second rule aims at preserving those strokes disconnected from digits such as "4" and "5". Figure 9 shows an examples of this problem. In such a case, the stroke detected at the beginning of the image satisfies both rules. The same does not happen for the stroke disconnected from the digit "5" (CC_2).

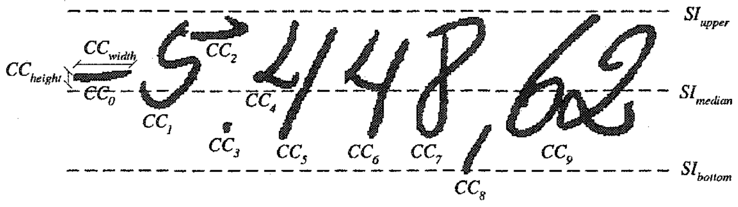


Figure 9 Numerical amount with all components detected.

3.1.2 Delimiter Detection

The second step aims at identifying specific parts of the numerical amount, namely, period (".") and comma (","). By inspecting our numerical amount database, we noticed that such components are located in the inferior part of the image. Thus, we consider as delimiters those components located entirely below the median line of the numeral string. In Figure 9 we can see both delimiters (CC_3 and CC_8) detected. The information about their location serve as some of the features used to feed the classifier. We will discuss this issue latter in Chapter 4.

3.1.3 Grouping

The last step tries to overcome the effects of fragmentation. An CC can represent either an integer number of characters or not. The second situation is critical and should be avoided. Basically, the grouping step tries to group a character composed of several CCs by detecting potential parts and grouping each of them to its nearest neighbour. We have adopted a strategy similar to the one presented by Ha et al in [55]. Such a strategy takes into account the median line of the image (SI_{mediam}). However, when dealing with a numerical amount, the SI_{mediam} could be biased by the position of the delimiters. Therefore, SI_{mediam} should be re-computed without considering the delimiters.

An CC is regarded as a broken part if at least one of the following two conditions is met:

1. The CC does not intersect the median line (SI_{mediam}) of the numeral string.
2. $\frac{max(CC_{above},CC_{below})}{min(CC_{above},CC_{below})} > 5$

   where CC_{above} and CC_{below} denote the vertical height of the part above and below SI_{mediam}, respectively.

Even when an CC intersects the median line, it may still be considered as broken part by the second condition if the intersection point is near to the top or bottom part of the CC. Thereafter, those CCs which are deemed broken parts are grouped to their neighborhood. We have to decide to which neighboring CC a broken part (CC_{broken}) should be grouped. The decision is based on the following rule:

```
IF  $CC_{left} < CC_{right}$  THEN
    Group($CC_{previous}$, $CC_{broken}$)
ELSE
    Group($CC_{broken}$, $CC_{next}$)
```

If a broken CC is on the left or right end of the numeral string, CC_{left} or CC_{right} is set to a very high value to achieve the correct grouping. Figure 10 defines the foregoing geometric quantities. The resulting partial images are ordered from left to right according to their horizontal position in the SI. In the case of Figure 10, we would expect three partial images corresponding to "4", "5", and "6", respectively.

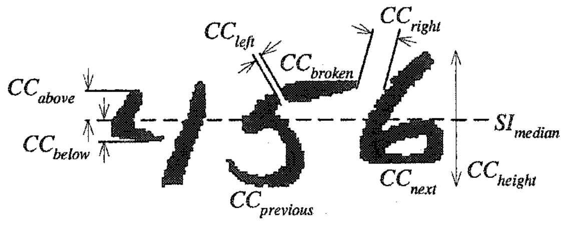


Figure 10 Definition of various geometric quantities.

All thresholds described in this section were determined based on experimentation.

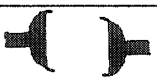
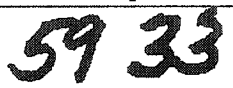
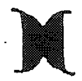
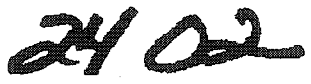
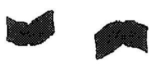
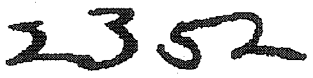
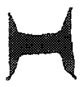
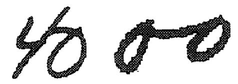
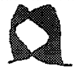
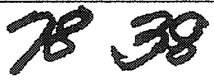
3.2 Segmentation

As stated elsewhere, our system takes into account an explicit segmentation algorithm, which determines the cut regions and provides multiple spatial representation. In this section we describe the segmentation algorithm we have developed. It makes use of three complementary sets of structural features: contour, profile, and skeleton. The final objective of this module is to provide a list of hypotheses of segmentation without any a priori knowledge of the context, such as the number of characters to be segmented.

We have focused the design of the segmentation algorithm on the limitation of heuristic rules so that it could be able to cope with the different types of connected numeral strings depicted in Table VI.

Table VI

Types of connected numeral strings [22].

Category	Style of Touching	Examples
Single touching		
		
		
		
Multiple touching		

3.2.1 Generation of the Segmentation Features

Depending on the context, a lot of features are available to find plausible segmentation points in a character image. Certainly some of the most used in the literature are the minima and maxima from contour and profile [88, 96, 145]. Indeed, these features can be obtained easily and they usually express directional variability of the character strokes and then possible cuts.

However, they are not even fully informative to localize any kind of connection between characters, mostly when the handwriting is strongly skewed or overlapped. To overcome such problems, we consider a third set of features provided by the skeleton: the intersection points. These points are often located in the neighborhood of stroke connections where contour and profile features are not always available.

The contour of an image can be defined as the image envelope. This is a bi-dimensional data where each contour point CP_i is associated with the coordinates (X_i, Y_i) of the image. To construct the profile, first we locate the leftmost and the rightmost pixels of the contour. The contour is split at these two points. To get the profile, for each x value, we select the outermost y value on each contour half. From these sets of features (contour and profile), we are able to localize the first list of potential cuts which correspond to the local minima (maxima) of the upper (lower) contour and profile (Figures 11a and 11b). We define these points as Basic Points (BP)s.

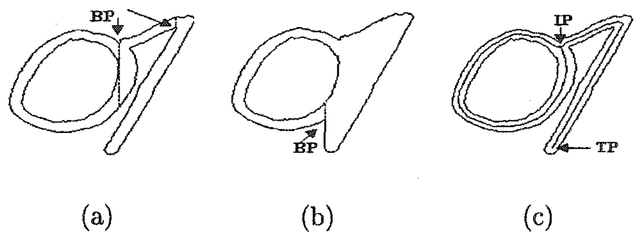


Figure 11 Features used by the segmentation algorithm. (a) Contour, (b) Profile, and (c) Skeleton.

Let us define now characteristics of the skeleton :

Definition 1: In a binary image, the intensity of a pixel p, denoted $I(p)$, is either 0 (white) or 1 (black). A pixel p is a *foreground pixel* iff $I(p) = 1$. Conversely, a pixel p is a *background pixel* iff $I(p) = 0$.

Definition 2: Using 8-Freeman directions, when the eight neighbors of a pixel p are traced clockwise, the neighborhood of p is denoted $T(p)$, such as:

$$T(p) = \sum_{i=1}^{8} I(p_i') \tag{3.1}$$

where p_i' is the i^{th} neighbor of p. We can then define the terminal point (TP) when $T(p) = 1$ and the intersection point (IP) when $T(p) > 2$. IPs and TPs are called characteristic points of the skeleton.

Definition 3: A skeleton path is a pixel sequence of the skeleton where each extremity corresponds to a characteristic point.

The skeleton image is obtained by using the thinning algorithm proposed by Jang et al [72]. The advantage of this algorithm relies on the limited number of IPs generated in the neighborhood of connected strokes of different characters. After some experiments, we realized that two kinds of IPs occur very often. The first one, which we call "Class 1", contains all configurations of IPs with one upper segment and two lower segments. The second type, which we call "Class 2", contains all configurations of IPs with two upper segments and one lower segment. Figure 12 depicts all possible configurations for each class. Since these configurations are very common, we try to take advantage of them to build the segmentation paths. We discuss this in the following subsection.

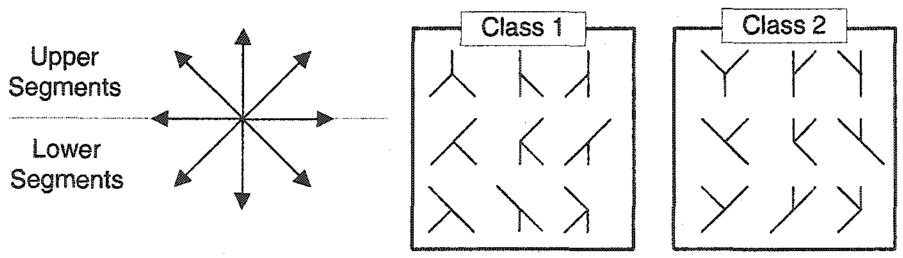


Figure 12 Two different classes of IPs.

3.2.2 Building Segmentation Paths

After observing some images of touching digits, we realized that very often the touching area contains BPs and IPs. In the light of this, we assume that if there is a BP near to an IP, this could indicate a possible connection between two digits. Thus, once BPs and IPs have been detected in the image, the next step is to establish a relationship among those points belonging to the same neighborhood.

In order to determine the proximity between two points, we used the estimation of the thickness of the strokes (E_t), obtained with the projection of the density histogram. Thus, two points BP$_i$ and IP$_j$ belong to the same neighborhood if one of the two following Equations are verified:

$$d(BP_i, IP_j) \leq E_t \tag{3.2}$$

$$d(proj(BP_{ik}), IP_j) \leq E_t \quad \text{for} \quad k = 1, 2 \ldots n \tag{3.3}$$

where d is the Euclidian distance, $proj(BP_{ik})$ is the vertical projection of BP$_i$ at the step k on the segment whose height is n pixels. Figures 13 exemplifies both configurations of neighborhood verification. In such cases, the circle stands for E_t. In Figure 13a both BP and IP are inside of the circle, so Equation 3.2 is verified. The same does not happen in Figure 13b where the distance between the BP and the IP is greater than E_t. However, the vertical projection of the BP crosses the circle, therefore, Equation 3.3 is verified.

Once the relationship among BPs and IPs have been established, the algorithm decides which type of segmentation cut should be performed. The cut may be generated either straight away from the skeleton path, or orthogonal to it. The first case con-

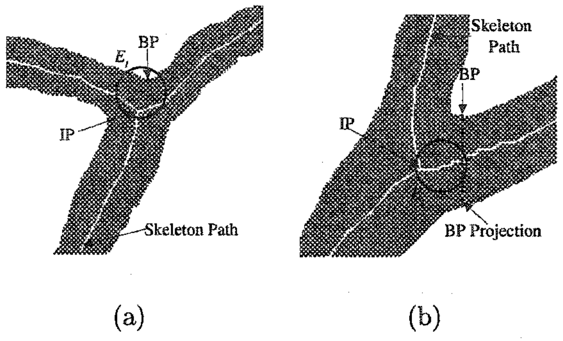


Figure 13 Distance verification between a BP and an IP (a) Distance verified by Equation 3.2 and (b) Distance verified by Equation 3.3.

templates the two classes of IPs described in the previous subsection (Figure 12). If the lower segment of a "Class 2" IP is linked to a upper segment of a "Class 1" IP (or vice-versa) and both IPs have one BP near to them, the skeleton path linking both IPs ($P_{skeleton}$) is used as part of the segmentation cut (see Figure 14b). Afterwards, complementary paths between the BPs and IPs (P_{comp}) are traced (see Figure 14c). Therefore, the segmentation path $P_{seg} = P_{skeleton} \bigcup P_{comp}$. This kind of cut is used very often to segment touching digits composed of circular strokes, such as "00", "06", "09", "89", and so forth.

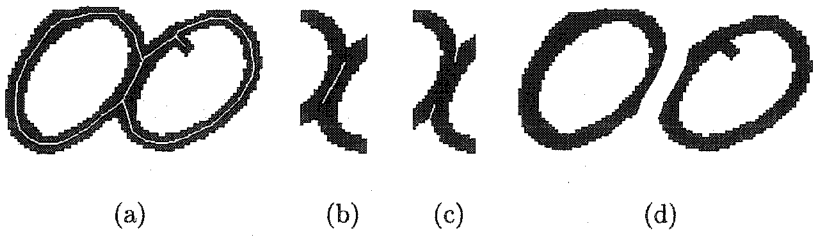


Figure 14 Segmentation path in the first case (a) Skeleton, (b) $P_{skeleton}$, (c) P_{comp}, and (d) Digits segmented.

When there is no connection between two IPs, but there is a BP related to an IP, the second case is employed. Here the segmentation path is traced orthogonally to the skeleton path. In order to find the best position to perform the cut, the algorithm evaluates $2 \times E_t$ different orthogonal cuts along the segment being analyzed and takes the shortest one. Figure 15 depicts the zone of segmentation around the IP (represented by a circle) and the shortest cut for each segment as well.

In order to reduce the number of segmentation cuts, the algorithm does not allow cuts on small segments. We define as small those segments where the number of pixels in the skeleton path between the IP and the TP is smaller than 20% of the total number of pixels in the skeleton path. An example of this is shown in Figure 15, where the segment "A" does not have a segmentation cut. This heuristic is worth of use, once the number of segmentation hypotheses to be assessed by the recognizer is reduced.

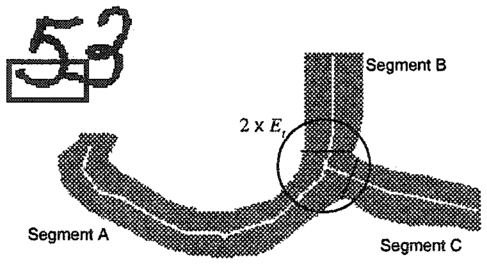


Figure 15 Example of orthogonal cuts.

In some cases, even if there is a connection between two characters, the skeleton path has no IP (see Figure 16a). To correctly segment this kind of images and avoid under-segmentation (the lack of segmentation point), the algorithm builds a segmentation path based on the vertical projection of the BP. Figure 16c shows the segmentation paths built based on the BP extracted from the profile of the image 16b.

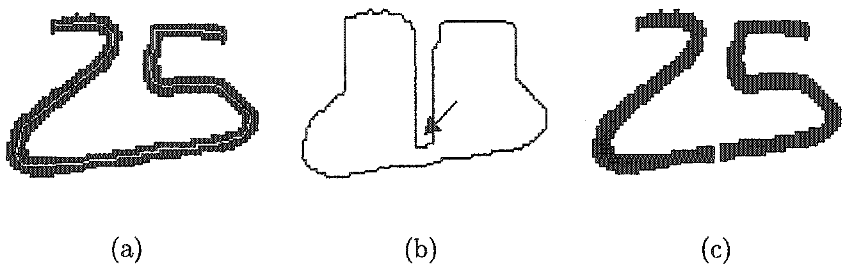


Figure 16 Segmentation based on BP only. (a) Original image, (b) BP from the profile, and (c) Image segmented.

Figure 17 shows us that with a limited number of rules, the algorithm is capable of providing the correct segmentation in most cases of connected characters depicted in Table VI.

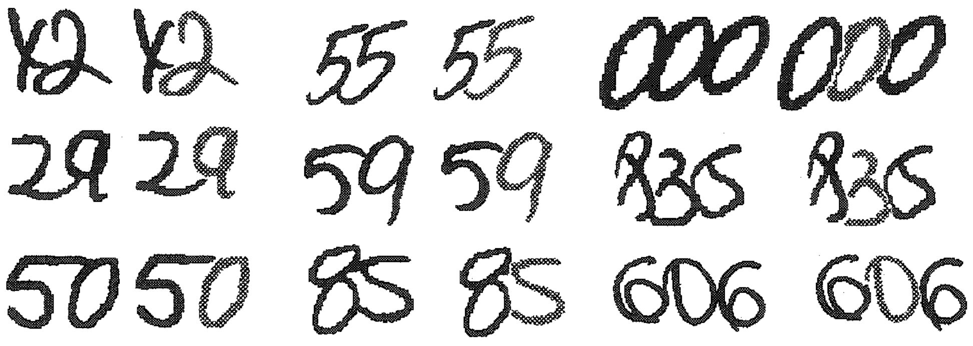


Figure 17 Examples of touching digits correctly segmented.

In spite of the fact that two different BPs (contour and profile) can be combined with IPs, sometimes no BP is found in the touching area. Some examples of this problem are shown in Figure 18 where the touching areas are signed by a circle. In such cases, the algorithm may provide either an under-segmentation (lack of segmentation path) or missegmentation.

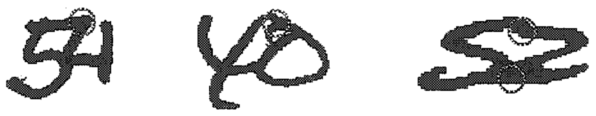


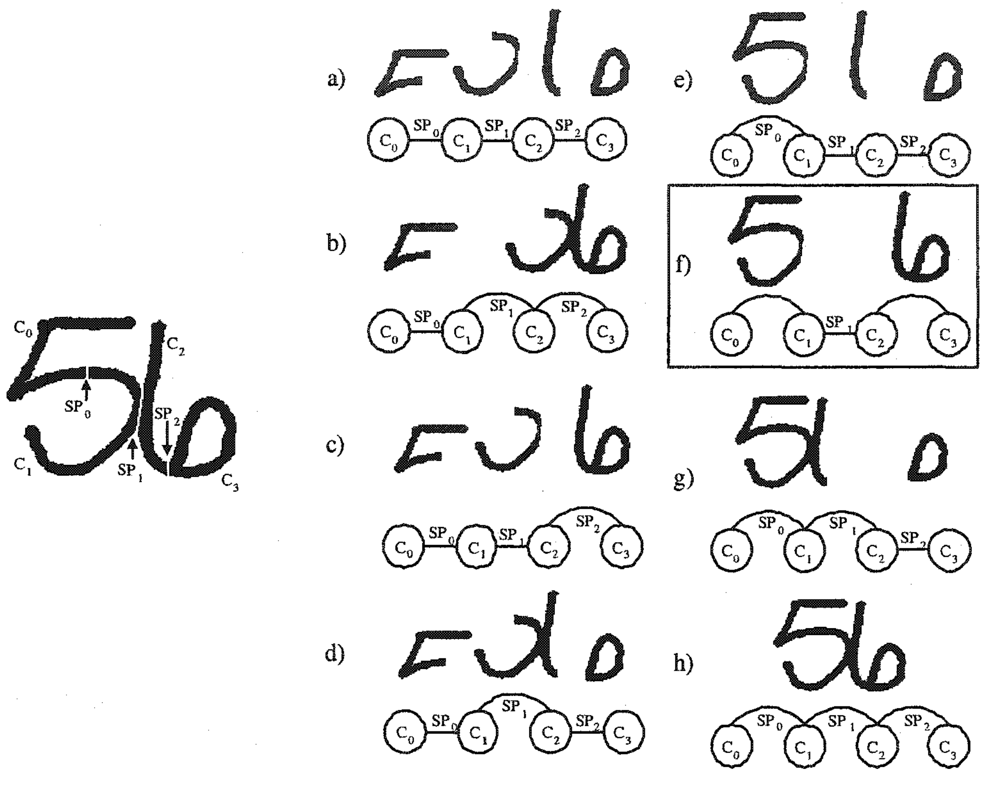
Figure 18 Lack of BPs.

3.2.3 Generating Hypotheses of Segmentation

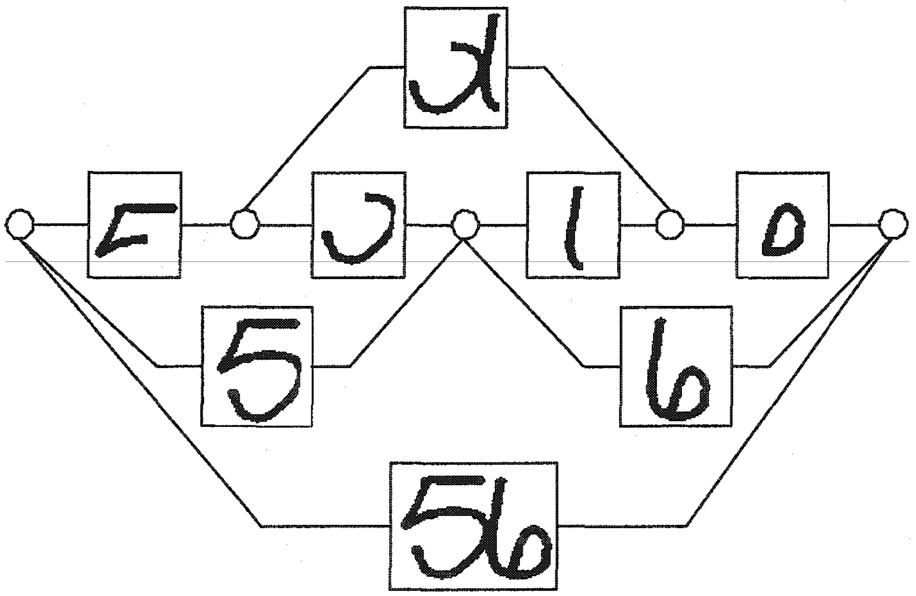
After generating all segmentation paths, the following task consists of determining all hypotheses of segmentation which will be assessed by the recognizer. In other words, we have to combine all sub-images generated in the previous step in a meaningful way, so that the number of hypotheses of segmentation may be naturally limited.

This is done through segmentation graphs where the pieces of character generated by the segmentation algorithm are represented by the states (C_i) and the segmentation paths (SP_i) by the transitions. In this way, we build all possible graphs by activating and deactivating SPs. Figure 19 exhibits the segmentation graphs for the touching pair "56" depicted on its left side. In such a case, the transitions represented by arcs stand for deactivated SPs while those represented by straight lines stand for activated SPs. The segmentation also can be represented through a single graph (Figure 19b) which can produce all segmentation hypothesis depicted in Figure 19a.

It can be observed that among all hypotheses depicted in this example, the optimal one is among them (hypothesis (f)). Thus, we would expect that the recognizer will assign the greatest score of recognition for this hypothesis. Nevertheless, since we have chosen an over-segmentation strategy, i.e., the algorithm produces several segmentation hypotheses to get the correct one among them, the system must deal with the over-segmentation problem. It happens when a not optimal hypothesis gets greater score than the optimal one. For example, the hypothesis (e) recognized as "510" could get a greater score than hypothesis (f), recognized as "56". Later in



(a)



(b)

Figure 19 Segmentation graphs: (a) Sub-graphs representing all segmentation hypotheses, and (b) segmentation represented through a single graph.

this thesis we will discuss how we address such a problem. The performance of this algorithm on digit string recognition will be discussed in Chapter 5.

3.3 Summary

In this chapter we have presented the first two modules of our system. We have seen that the component detection module takes a smoothing algorithm before detecting and grouping connected components. We also have described our segmentation algorithm, which combines three different features in order to generate the segmentation paths. In the next chapter we will discuss the Recognition and Verification strategy and how the problem of over-segmentation mentioned above has been addressed.

CHAPTER 4

RECOGNITION AND VERIFICATION STRATEGY

In this chapter we describe all modules defined previously in Section 2.1.4. In addition, we present the feature set used by each classifier and also describe how the low-level verifiers and general-purpose recognizer interact with each other.

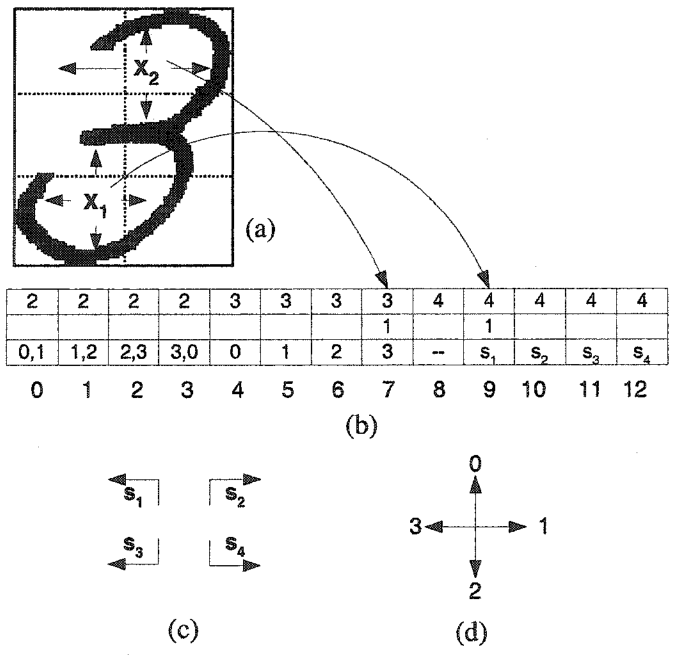
4.1 General-purpose recognizer

Our general-purpose recognizer is composed of 3 modules: e_{10}, e_3, and e_{13} (see Figure 5). e_{10} and e_3 are specialized in ten numerical and three non-numerical ("#", ",", and ".") classes respectively. Both classifiers use a mixture of concavity, contour based features and surface of the characters.

The basic idea of concavity measurements is the following: for each white pixel in the component, we search in 4-Freeman direction (Figure 20d), the number of black pixels that it can reach as well as which directions the black pixel is not reached. When black pixels are reached in all directions (e.g. point x_1 in Figure 20a), we branch out in four auxiliary directions (s_1 to s_4 in Figure 20c) in order to confirm if the current white pixel is really inside a closed contour. Those pixels that reach just one black pixel are discarded.

Thereafter, we increment the position in the feature vector that fits with results returned by the search (Figures 20a and b). In Figure 20b we represent the feature vector where each component has two labels. The superior label means the number of black pixels found during the search while the inferior label means the directions where the black pixels were not reached. For example, the pixel x_2 (Figure 20a) reaches the black pixel in all directions except in direction 3. Therefore, the position 7 of the feature vector is incremented. For pixel x_1, the position 9 is incremented because it reaches the black pixel in four directions. However, using the auxiliary direction s_1 we confirm that it is not inside a closed contour. When the pixel is inside a closed contour, the position incremented is the 8^{th}.

Since we are dividing the image into six zones, we consider six feature vectors of 13 components each. Therefore, in the example presented above, the pixel x_2 will update the second vector while the pixel x_1 will update the fifth vector. Finally, the overall concavity feature vector is composed of (13×6) 78 components which are normalized between 0 and 1 by summing up their values and then dividing each one by this summation. The remaining feature vectors considered in this work are normalized in the same way.



2	2	2	2	3	3	3	3	4	4	4	4	4
							1		1			
0,1	1,2	2,3	3,0	0	1	2	3	--	s_1	s_2	s_3	s_4
0	1	2	3	4	5	6	7	8	9	10	11	12

Figure 20 Concavities measurement: (a) Feature vector, (b) Concavities, (c) Auxiliary directions, and (d) 4-Freeman directions.

The contour information is extracted from a histogram of contour directions. For each zone, the contour line segments between neighboring pixels are grouped regarding 8-Freeman directions (Figure 21c). The number of line segments of each orientation is counted (Figure 21b). Therefore, the contour feature vector is composed of (8×6) 48 components normalized between 0 and 1. Finally, the last part of the feature vector is related to the character surface. We simply count the number of black pixels in each zone and normalize these values between 0 and 1. Thus, the final feature vector, which feeds e_{10} and e_3 has $(78 + 48 + 6)$ 132 components.

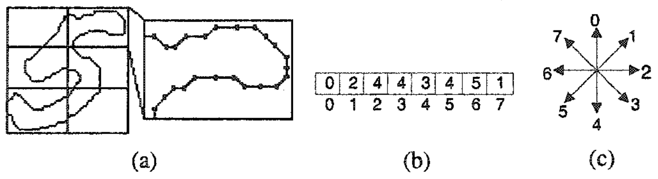


Figure 21 Contour measurement: (a) Contour image of the upper right corner zone, (b) Feature vector, and (c) 8-Freeman directions.

As depicted in Figure 5, the classifier e_{13} combines the e_{10} and e_3 outputs, one contextual feature of position, and four structural features. Therefore, e_{13} has 18 inputs. Such a scheme of combination has produced the best recognition rates in our experiments that consider the database of numerical amounts on Brazilian bank cheques. The contextual feature of position, which was detected at the component detection phase, is responsible for minimizing the confusion between the digit "1" and the symbol ",". As discussed in Section 3.1.2, we have observed that the symbol comma is always located entirely below the median line of the numeral string. In this way, when the component is located entirely below the median line, the 14^{th} position of the feature vector used by e_{13} receives 1, otherwise 0. As we can observe in Figure 22, the sole information capable of removing the confusion between the digit "1" and the symbol "," is the position of the component in the image context.

In order to reduce the confusion between the numerical classes "4" and "7" and the non-numerical symbol "#", we have used feature points of the skeleton. Four compo-

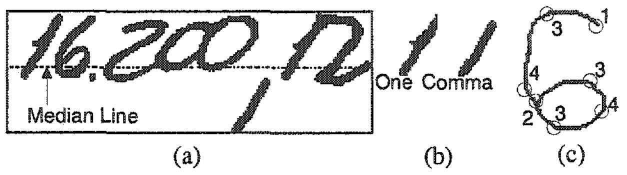
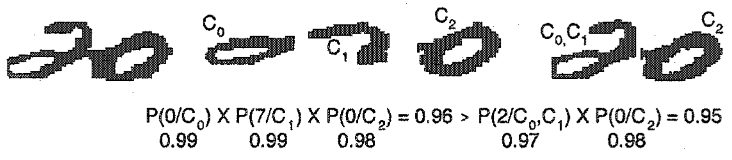


Figure 22 Contextual and structural features: (a) Numerical amount with the median line detected (b) Similarity between digit one and symbol comma, and (c) Structural features extracted of the skeleton.

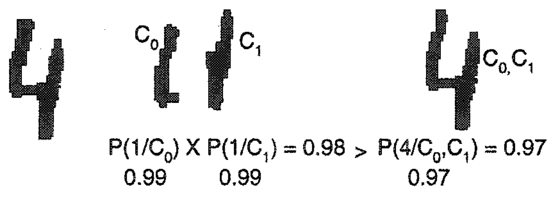
nents have been considered: end points, crossing points, and two directional points, which are detected when the skeletal path changes its direction in the horizontal or vertical axis. These points are represented in Figure 22c by the numbers 1, 2, 3, and 4 respectively. Once the skeletal points were detected, the points of each configuration are counted, normalized between 0 and 1 and the last four positions of the feature vector used by e_{13} updated.

We show in Chapter 5 that this classifier reaches very interesting recognition rates when dealing with isolated digits. However, when it is used within a complete system, it faces more complex problems such as over-segmentation and under-segmentation. In the first problem, the intra-character segmentation hypothesis provides better results than any of the inter-character segmentation hypotheses. Figures 23a, b, and c show some examples of this problem. In the second problem, the lack of segmentation cuts produces better results than the correct segmentation hypothesis (Figure 23d).

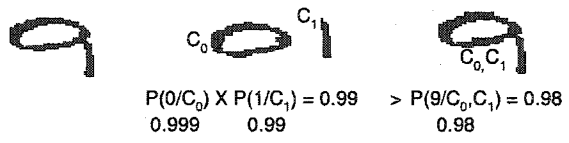
Instead of using heuristics to overcome these problems, we opted for the use of a strategy based on low-level verifiers, which is more robust and reliable. Initially, we developed a strategy based on an isolated verifier, which was responsible for detecting both over-segmentation and under-segmentation [124]. After some experiments, we realized that a more specialized verifier could produce better results than a generic


C_0
C_1
C_2
C_0,C_1
C_2
$P(0/C_0) \times P(7/C_1) \times P(0/C_2) = 0.96 > P(2/C_0,C_1) \times P(0/C_2) = 0.95$
0.99 0.99 0.98 0.97 0.98

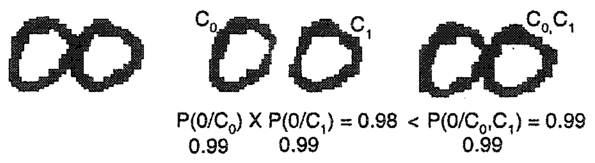
(a)



(b)



(c)



(d)

Figure 23 Misclassification caused by over-segmentation (a, b, and c) and under-segmentation (d)

one. In the following subsections we present both verifiers, their respective feature sets and how they interact with the general-purpose recognizer.

4.2 Over-segmentation Verifier

The main objective of this verifier is to improve the performance of the general-purpose recognizer by detecting over-segmented characters (Figures 23a, b, and c). Thus, the verifier takes into account two classes: isolated characters and over-segmentation. In order to discriminate these two classes, we developed a new feature set called Multi-level Concavity Analysis. First of all, we introduce the definitions of *Concavity Levels.* We define as *Initial Concavity Level* (ICL) for a background pixel, the number of black neighbors that it has in the 4-Freeman directions. We also consider one label to identify the background pixels located outside of a closed contour but with four black neighbors. Those pixels that have only one black neighbor are not considered. Figure 24 shows an image labelled with the four possible labels used in ICL.

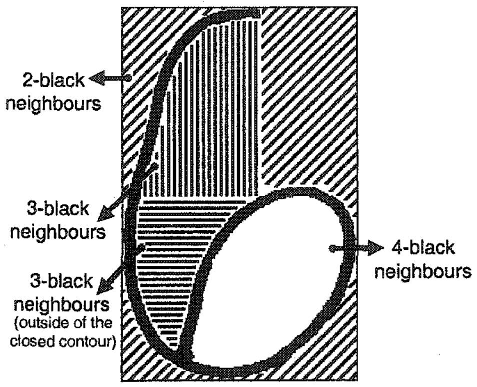


Figure 24 Image labelled with ICL where the four possible labels are represented.

The first step of this analysis consists of labelling with ICL the background pixels of the over-segmented piece (I_{seg}) and its corresponding original piece of handwriting (I_{orig}). Figure 25 exhibits two examples of this procedure. Afterwards, the following verification is carried out: for each background pixel found in both images (I_{orig} and I_{seg}), we assigned to the final image (represented by the MCA in Figure 25) a specific label (represented by the black area) when the ICLs are different, otherwise, we assigned the same ICL found in both images. The same verification procedure is carried out over the foreground pixels. However, in such a case we assigned a specific label (represented by the bold dot) to MCA when the pixels found in both images share the same label. The latter procedure aims at computing the relative area of the over-segmented part.

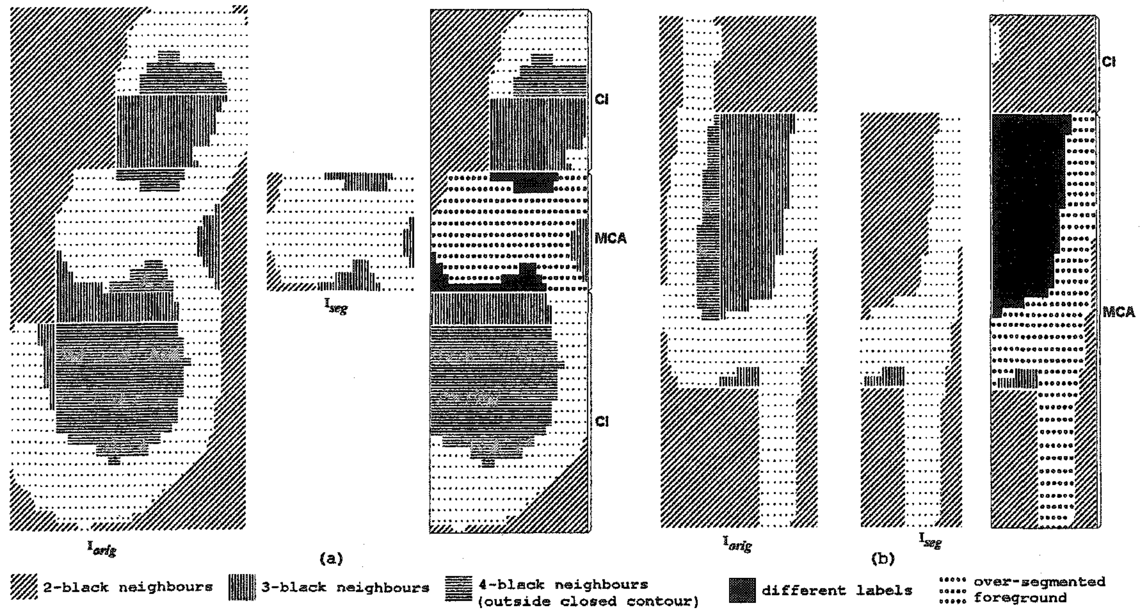


Figure 25 Two samples (a and b) of Multi-level Concavity Analysis (MCA) and Contextual Information (CI).

The second part of this analysis concerns with I_{seg} contextual information (represented by CI in Figure 25), which is extracted by taking into account the I_{seg} complement. As we can observe in Figure 25, the complement that we are using is limited to I_{seg} width and it is composed of the I_{orig} background, which was labelled with ICL, and the surface of the I_{orig} (represented by the dot in Figure 25). Figure 25 also shows the final labelling, which is composed of multi-level concavity analysis (MCA) and contextual information (CI) about the over-segmented piece.

Therefore, this feature set has 7 possible labels: the four labels of ICL, the label that represents the difference between concavity levels of I_{orig} and I_{seg}, the label that represents the surface of the over-segmented piece, and the label that represents the surface of the original image in the context of the segmentation. Finally, the multi-level concavity analysis with contextual information is divided into six regions (the same division shown in Figure 20b) and (7×6) 42 components normalized between 0 and 1 are considered.

4.3 Under-segmentation Verifier

To complement the previous verifier, this one is devoted to reduce the confusion between isolated and under-segmented characters. Thus, this verifier considers two classes: isolated characters and under-segmentation.

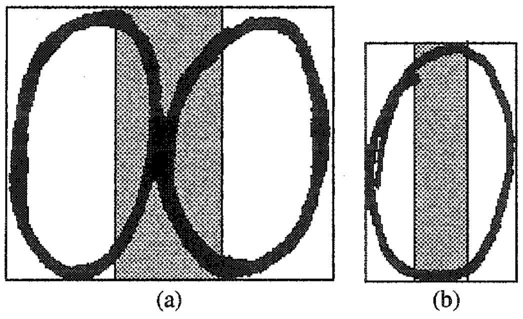


Figure 26 Zoning used by the under-segmentation verifier.

After analyzing the system errors, we observed that touching digits with strong concavities, e.g. "00", often are recognized as "0" with a high probability. After trying some features such as, horizontal transitions, profile distances, and structural information, we conclude that the same concavity analysis and information about the surface used by the general-purpose recognizer is a good feature set to discriminate isolated characters from under-segmented ones. However, a different way of zoning which divides the image into three vertical parts has been employed. Such a strategy aims at emphasizing the region of the image where the connections between characters occur often (Figure 26). As a result, $((13+1)\times 3)$ 42 components normalized between 0 and 1 are considered.

4.4 How the Classifier and Verifiers Interact With Each Other

According to the probabilistic model presented in Equation 2.10 (Section 2.1.1), the final probability for a hypothesis of segmentation-recognition is given through the product of the probabilities produced by its sub-components. The probability of a sub-component is given by the product of the probabilities produced by the general-purpose recognizer, over-segmentation verifier, and under-segmentation verifier.

In Figure 27 we present an example of how the verifiers interact with the general-purpose recognizer. In order to better illustrate this, we reproduced the problem depicted in Figure 23a, where the the over-segmented hypothesis got a better result than the correct one. The schema in Figure 27 is carried all the way through, from segmentation, feature extraction, to recognition and verification. In the column "outputs" we can see the outputs of the neural networks (represented by a black box) as well as the probability produced by them. According to the decision rules presented in Section 2.1.2, the verifiers always consider their first output, which contains a posterior probability of an input pattern of isolated character. Thus, when these probabilities are low, the output supplied by the general-purpose recognizer (e_{13}) is penalized, otherwise, it is confirmed.

We can visualize this penalty in the first two components of the second segmentation hypothesis. In such cases, the probabilities of these components of isolated character are very low (0.010) and hence the probability generated by the general-purpose recognizer is penalized. The opposite happens to both components of the first segmentation hypothesis as well as the last component of the second segmenta-

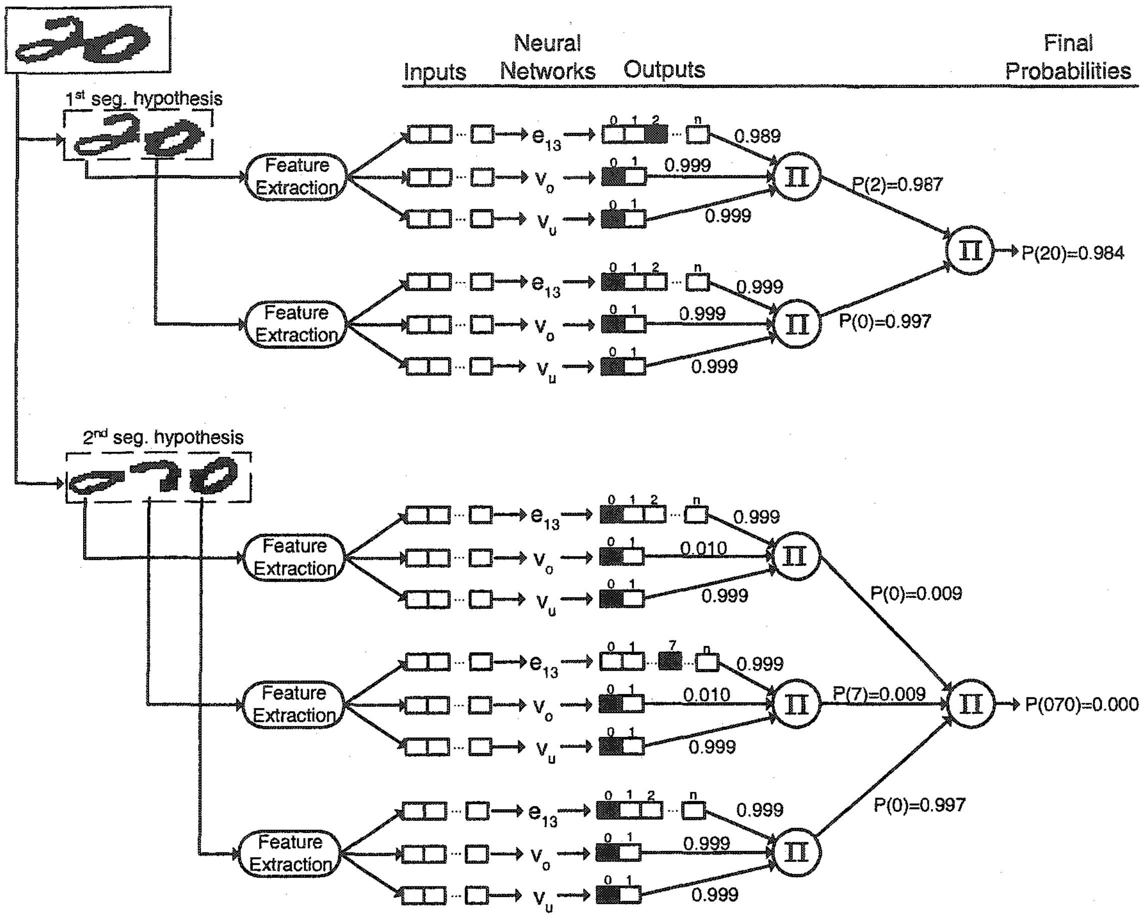


Figure 27 Interaction between the general-purpose recognizer and verifiers.

tion hypothesis. In these cases, both verifiers confirmed the output produced by the general-purpose recognizer.

In the above example, we have seen how the over-segmentation verifier eliminated the confusion presented in Figure 23a. Since the over-segmented pieces of the second segmentation hypothesis were penalized, the first segmentation hypothesis, which is the correct one, got a higher final probability.

4.5 Hypothesis Generation

A formal technique for finding a best single state sequence is the Viterbi algorithm [37]. The Viterbi algorithm is a form of dynamic programming, and a trellis structure efficiently implements the computation. However, in many cases, a considerable performance improvement can be obtained if the knowledge of the globally second best path, the third best path, etc., can be utilized in subsequent postprocessing. In order to produce an ordered list of the best L best paths, we have used a modified Viterbi algorithm, which was proposed in [21]. This algorithm was first applied in an HMM-context, therefore, some modifications must be made to cope with our problem. Since all a_{ij} (state transition probability) terms are equal, we can omit these terms as well as the initial state probability term π. The term $b_i(k)$ stands for our segmentation-recognition graph (e.g. Figure 28c), T is the number of connected components (CC) of the image, and N can be calculated as follows:

$$N = \max_{1 \leq i \leq T}[\#\text{seg. hypotheses of } \text{CC}_i \times \#\text{rec. hypotheses}]$$

As it can be observed from Figure 28b, we have considered just rank-2 hypotheses of recognition (i.e., #rec. hypothesis = 2) for each segmentation hypothesis, since the classifier very often produces the correct answer in rank-1 and 2. Thus, for the example depicted in Figure 28, $N = 4$.

In order to produce an ordered list of the best L state sequences, Chen et al [21] propose two main modifications on the Viterbi algorithm. The first one, is to extend the Ψ (survivor terminating in i_t) and δ (survivor score in i_t) to another dimension where the extra dimension represents the choice L. So for the recursion at time t, we need to consider all the possible $\delta_{t-1}(i, l)$ and record the L best path and its probability in $\Psi_t(j, l), \delta_t(j, l)$ for $l = 1, 2, \ldots, L$, respectively. At the end, the overall globally L best path probabilities must be in $\{\delta_T(i,l), 1 \leq i \leq N, 1 \leq l \leq L\}$ and can be trace back by following the corresponding $\Psi_t(i,l), t = T, T-1, \ldots, 1$. The second one consists in using a structure that, for each node of the graph, describes all the paths already computed going trough this node. Once the lth best path has been backtracked, for each of the nodes in this backtracked path, a counter (count(i,t)) is incremented by one. Then, starting from the first node ($t = 1$), we find the count(i,t)th maximum for the ith node at time t of this backtracked path. After this forward pass, the $(l+1)$th globally optimal path can be found by checking the score of the terminal nodes.

The modified Viterbi algorithm is described as follows:

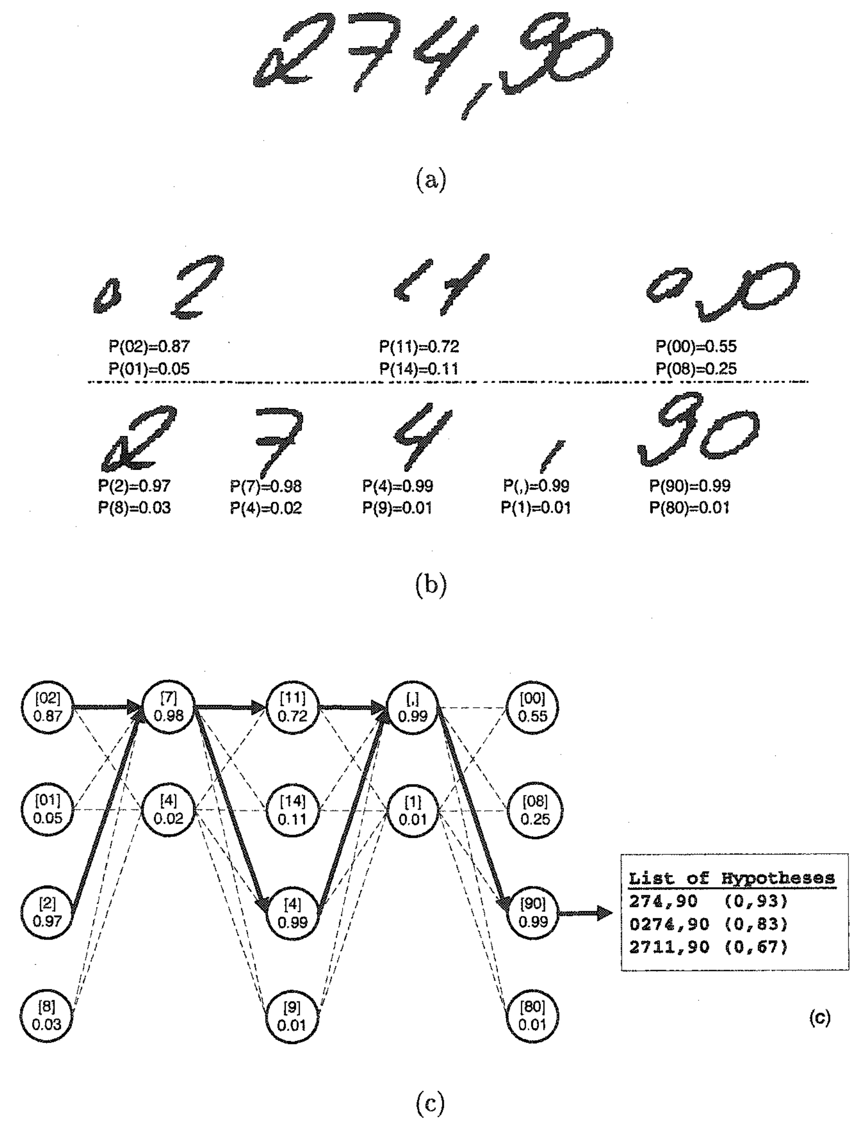
- Step 0) Storage:

t	time index
c	iteration index
$\Psi_t(i,l), 1 \leq t \leq T, 1 \leq i \leq N, 1 \leq l \leq L$	survivor terminating in i_t
$\delta_t(i,l), 1 \leq t \leq T, 1 \leq i \leq N, 1 \leq l \leq L$	survivor score in i_t
count$(i,t), 1 \leq i \leq N, 1 \leq t \leq T$	count of passes allowed at node i_t

- Step 1) Initialization:

$$\begin{aligned}
\delta_1(i,1) &= b_i(1) \quad \text{for} \quad 1 \leq i \leq N \\
\delta_1(i,l) &= 0 \quad \text{for} \quad 1 \leq i \leq N, 2 \leq l \leq L \\
\Psi_1(i,l) &= (0,0) \quad \text{for} \quad 1 \leq i \leq N, 1 \leq l \leq L \\
\text{count}(i,t) &= 1 \quad \text{for} \quad 1 \leq i \leq N, 1 \leq t \leq T \\
c &= 1
\end{aligned}$$

- Step 2) Pre-recursion: For $2 \leq t \leq T, 1 \leq j \leq N$



List of Hypotheses	
274,90	(0,93)
0274,90	(0,83)
2711,90	(0,67)

Figure 28 Generation of global hypotheses: (a) Original image, (b) Segmentation hypotheses with rank-2 hypotheses of recognition, and (c) Three best paths.

$$\delta_t(j,1) = \max_{1\leq i\leq N}[\delta_{t-1}(i,1)]b_j(t)$$

$$\Psi_t(j,1) = \arg\max_{1\leq i\leq N}[\delta_{t-1}(i,1)]$$

- Step 3) Backtracking:

$$P^* = (c-th) \max_{1\le i\le N, 1\le l\le count(i,T)} [\delta_T(i,l)]$$
$$(i_T^*, l_T^*) = \arg(c-th) \max_{1\le i\le N, 1\le l\le count(i,T)} [\delta_T(i,l)]$$
$$\text{count}(i_T^*, T) = \text{count}(i_T^*, T) + 1$$

where $(c-th)$ max$[\cdot]$ denotes the cth maximum.

If $t = T-1, T-2, \ldots, 1$

$$\begin{aligned} (i_t^*, l_t^*) &= \Psi_{t+1}(i_{t+1}^*, l_{t+1}^*) \\ \text{count}(i_t^*, t) &= \text{count}(i_t^*, t) + 1 \end{aligned}$$

If $I^* = \{i_1^* i_2^* \ldots i_T^*\}$, the cth optional state sequence satisfies the given criteria or the index c exceeds limit, exit. Otherwise, continue.

- Step 4) Forward-tracking: for $t = 2, 3, \ldots, T$

$$l = \text{count}(j_t^*, t)$$
$$\delta_t(j_t^*, l) = (l\text{th}) \max_{1\le i\le N, 1\le m\le count(i,t-1)} [\delta_{t-1}(i,m)] b_{j_t^*}(t)$$
$$\Psi_t(j_t^*, l) = \arg(l\text{th}) \max_{1\le i\le N, 1\le m\le count(i,t-1)} [\delta_{t-1}(i,m)]$$

increase c by 1;

repeat Step 3.

Figure 28 shows the three best hypotheses of segmentation-recognition. Since the $l$ best hypotheses are computed incrementally, we obtain a list of decreasing probabilities (Figure 28c). Once the list of hypotheses has been generated, it is submitted to the post-processor module which verifies whether it satisfies the application rules or not.

4.6 Post-Processor (PP)

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

In spite of the fact that numerical amounts produce a poor grammar, we can find in the literature various works that use some kind of syntactic analysis. Knerr et al. in [80] consider the segments below the baseline in order to obtain the final interpretation of the numerical amount. Dimauro et al in [32] use non-numerical symbols such as "#" in order to identify the beginning and the end of the numerical amount. Heutte et al in [61] present a post-processing module for French bank cheques, which takes into account specific non-numerical symbols such as the characters "F" and "C".

Usually in Brazil, two delimiters ("#") are affixed at the beginning and at the end of the numerical amount, a period "." is sometimes used to delimit a 3-tuple of digits and a comma "," is used to identify the cents portion in the numerical amount. In addition to these rules, the Central Bank of Brazil decided that the cents portion should be present in the numerical amount [7]. In order to deal with such rules we have developed a deterministic automaton which is associated with the term $P(M, V)$ of Equation 2.10. This automaton is depicted in Figure 29.

Once such an automaton is formed, we fit it to the probabilistic model in the following way: if the current hypothesis is verified by the automaton, then $P(M, V) = 0.99$, otherwise $P(M, V) = 0.01$. These values aim at re-ranking the list of hypotheses generated previously. Consider for example the list of hypotheses (a) presented in Table VII where the correct hypothesis is the second one (140,00). Since the

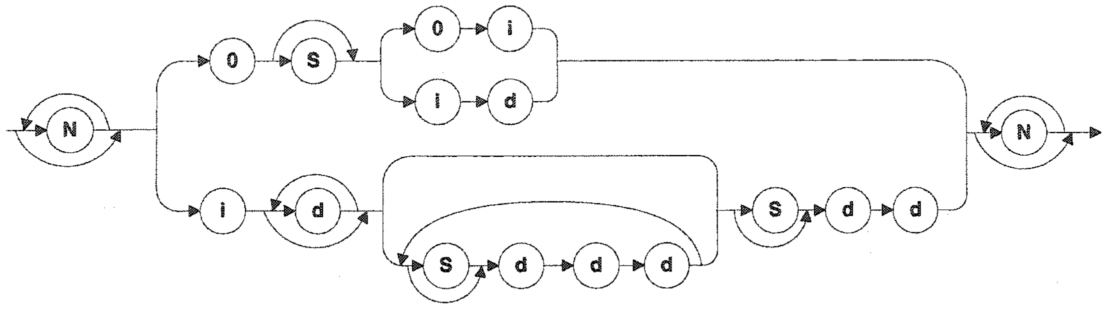


Figure 29 Syntactic graph used to carry out the syntactic analysis, where N stands for noise such as "#", 0 for the digit "0", S for separators "." and ",", i for the digits ranging from 1 to 9, and d for the digits ranging from 0 to 9.

automaton did not verify the first hypothesis of the original list, its probability will be multiplied by 0.01 while the probabilities of others will be multiplied by 0.99. List (b) of Table VII shows the decreasing probabilities after post-processing.

Table VII

(a) The original list of hypotheses and (b) the list after post-processing.

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

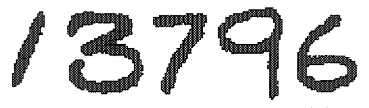
It is worthy of remark that we just choose a value near 1 (0.99 in this case, but it could be 1) to confirm the hypotheses verified by the automaton and a value near to 0 to penalize those not verified. We just do not use 0 because we opted to keep all hypotheses generated by the system in the final list. An efficient and very often used strategy for post-processing is to estimate $P(M, V)$ from some data. For example, Lethelier et al in [96] use an ergotic HMM model to express $P(M, V)$. It was possible because they had a real and huge database to train such a model. In our case, we are working with a laboratory database where the probability of occurrence of all amounts is the same. For example, the fact that very small amounts and very large amounts are less likely on cheques does not happen in our database. Moreover, we are dealing with a small database (about 1,300 images in the training set) and consequently it is impossible to model all variability of the numerical amount from such a small data.

4.7 Global Decision

The global decision module decides either to accept the recognition result or reject it. The goal of rejection is to minimize the number of recognition errors for a given number of rejects. A direct scheme of rejection is to reject the image that has a global probability less than a determined threshold. However, due to the probabilistic model used, a 10-digit string usually supplies a global probability smaller than a 2-digit string. Among the different strategies that we have tested, the one proposed by Fumera et al [42] provided a better error-reject trade-off for our system.

Basically, this technique suggests the use of multiple reject thresholds for the different data classes $(T_0, \ldots, T_n)$ to obtain the optimal decision and reject regions. In order to define such thresholds we have developed an iterative algorithm, which takes into account a decreasing function of the thresholds variables $R(T_0, \ldots, T_n)$ and a fixed error rate T_{error}. We start from all threshold values equal to 1, i.e., the error rate equal to 0 since all images are rejected. Then, at each step, the algorithm decreases the value of one of the thresholds in order to increase the accuracy until the error rate exceeds T_{error}. The error rate is defined in Equation 5.2.

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



					(a)
P(1)	P(3)	P(7)	P(9)	P(6)	
0.99	0.95	0.98	0.99	0.99	(b)
T_1	T_3	T_7	T_9	T_6	(c)

Figure 30 Rejection mechanism: (a) Digit string, (b) Probability of each component, and (c) Thresholds for the different classes.

Figure 30 exemplifies the global decision. In such a case, the string will be accepted if $P(1) > T_1$, $P(3) > T_3$ and so forth. We will see in the next section that this strategy of rejection produces interesting error-reject trade-offs. We will present experiments considering different T_{error}.

4.8 Summary

In this chapter, the Recognition and Verification strategy has been described in detail. We have seen that it is composed of three modules. A general-purpose recognizer that assigns a given input to one of the n existing classes of the system, and two verifiers which were especially developed to evaluate precisely the result of the recognizer in order to compensate for its weakness due to particular training, and consequently to make the whole system more reliable. Afterwards, the way that the general-purpose recognizer interact with the verifiers was discussed. Finally, we have presented how the results produced by the Recognition and Verification strategy are used in order to generate all possible segmentation-recognition hypotheses, how they are post processed and the mechanism used to make an accept/rejection decision. In the next chapter we shall see the experiments we have carried out in order to demonstrate the efficiency of the system described so far.

CHAPTER 5

EXPERIMENTAL RESULTS

In order to validate the concept of modular system as well as to show the robustness of our system, we ran experiments on two databases. The first database is composed of 2,000 images of numerical amounts and it aims at evaluating the performance of the system on recognition of numerical amounts on Brazilian bank cheques. The second database is the NIST SD19 (hsf_7 series) and it aims at validating the concept of modular system as well as to show the robustness of our system on a well-known database (see Appendix 1). For all reported results we used the following definitions of the recognition rate, error rate, rejection rate, and reliability rate. Let B be a test set with N_B string images. If the recognition system rejects N_{rej}, classifies correctly N_{rec}, and misclassifies the remaining N_{err}, then

$$\text{Recognition Rate} = \frac{N_{rec}}{N_B} \times 100 \tag{5.1}$$

$$\text{Error Rate} = \frac{N_{err}}{N_B} \times 100 \tag{5.2}$$

$$\text{Rejection Rate} = \frac{N_{rej}}{N_B} \times 100 \tag{5.3}$$

$$\text{Reliability} = \frac{\text{Recognition Rate}}{\text{Recognition Rate} \quad + \quad \text{Error Rate}} \times 100 \tag{5.4}$$

Therefore, the recognition rate, error rate and rejection rate sum up to 100%. Most of the works in the literature express their results in terms of recognition rate, however, for systems applied to real applications this rate is not so relevant. Since real applications request low error rates, it is much more interesting to show the recognition rate in relation to a specific error rate, which includes implicitly a corresponding reject rate. This rate also allows us to compute the reliability of the system for a given error rate (Equation 5.4).

5.1 Experiments on Numerical Amounts

For our experiments a database containing 2,000 images of numerical amounts was used (see Appendix 1). Most images of this database have a non-numerical symbol ("#") affixed at the beginning and at the end of the numerical amount, a period "." to delimit a 3-tuple of digits, and a comma "," to identify the cents portion in the numerical amount. This database was divided in the following way: 1,300, 200, and 500 images for training, validation, and testing respectively. From these three databases, we extracted 11,400, 2,000, and 4,000 isolated characters for training, validation, and testing respectively. For all three sets, about 80% of the database consists of digit images while the rest is composed of non-numerical images ("#", ",", and "."). Therefore, we have used the numeric part of the databases (80%) to train e_{10} and the rest (20%) to train e_3. The recognition rates achieved by e_{10}, e_3 on the test set were 99.2%, 99.0% respectively (zero-rejection level). Thereafter, we submitted the training set to e_{10} and e_3, and trained the classifier e_{13} with the outputs provided by e_{10} and e_3 plus the five components described in Section 4.1. e_{13} achieved a recognition rate of 98.9% on the test set.

As discussed in Section 4.2, the verifier v_o has two outputs: isolated characters and over-segmentation. In order to train this verifier, we have used the following data: 8,000 correctly segmented characters, 8,000 naturally isolated characters and 12,000 over-segmented parts, which were generated automatically by the segmentation algorithm through the segmentation of the isolated and touching characters. The first two parts are devoted to train the first class of the verifier, while the third one is devoted to train the second class. Therefore, the training set used by v_o is composed of 28,000 samples. The validation and test sets were built in the same manner, and they have 14,000 samples each. This verifier reached a recognition rate of 99.40% on the test set.

Finally, we have built the database for v_u. As described in Section 4.3, the task of such a verifier is to detect the under-segmentation. Thus, it considers two classes: isolated characters and under-segmentation. The database used in this case is composed of 9,000 samples, which are divided into 5,000 images of isolated characters and 4,000 images of touching characters. The validation and test sets were built considering the same distribution of samples and they are composed of 4,000 samples each. This verifier reached a recognition rate of 99.17% on the test set.

After training the classifier and verifiers, we carried out some experiments using the test set of numerical amounts (500 images). The number of characters per image in this database (average length) is about 9. Figure 31 shows some examples of numerical amounts on Brazilian bank cheques extracted from our database.

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

Basically, the post-processor differentiates the digits from symbols, e.g., 140,00 confused with 1#0,00. In such a case, the automaton does not accept a symbol between

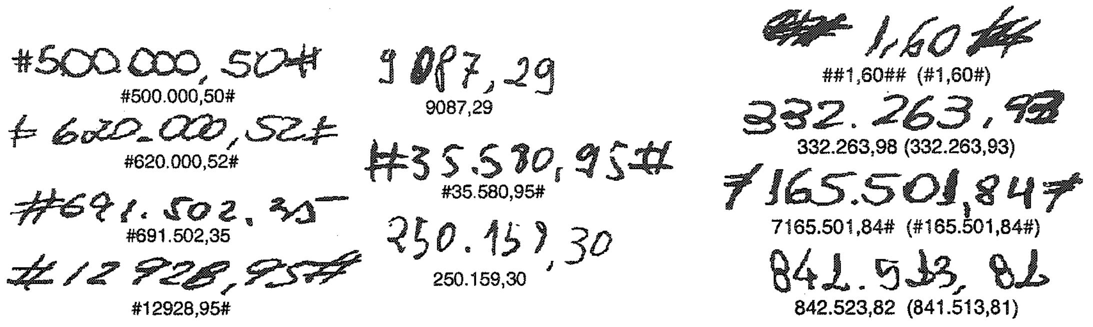


Figure 31 Examples of numerical amounts on Brazilian bank cheques: (a) Recognized and (b) Not recognized (the correct string is the one in parentheses).

Table VIII

Recognition rates (%) for the numerical amounts (zero-rejection level).

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

two digits. Considering the last experiment (e_{13}, v_o, v_u, pp), we divided the total error of the system into three classes: segmentation, recognition, and verification. The segmentation errors are caused by under-segmentation, which is due to a lack of basic points in the neighborhood to the connection stroke [122] (1.9%). The recognition errors are confusions of the general-purpose recognizer (51.7%), confusion generated by segmentation effects such as ligatures and noises produced by segmentation cuts (30.8%), and confusions generated by fragmentation (7.6%). The latter occurs when the system misclassifies a broken character, which usually are due to natural fragmentation caused by the handwriting style (Figure 34c) and noises acquired during scanning or pre-processing. (Figure 34d).

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

Table IX

Recognition rates (Rec.), Rejection rates (Rej.), and Reliability rates (Rel.) on the numerical amounts for different error rates.

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

Since bank cheque systems demand low error rates, we ran two experiments (based on the configuration "e_{13}, v_o, v_u, pp") where we fixed the error rates at 0.5 and 0.1% respectively. Table IX presents recognition, rejection, and reliability rates at these two error levels while Figure 32 shows the error-reject trade-off for the same experiment.

5.1.1 Experiments on NIST SD19

Since this database is composed of digit strings only, all Task Dependent Modules (grey boxes in Figure 5, Section 2.1.4, Chapter 2) related to the numerical amount task were removed, except the Post-Processor module, which was changed to verify the length of the string. Since we are dealing with different handwriting styles (Brazilian and North American), the general-purpose recognizer, in this case (e_{10}), over-segmentation verifier (v_o) and under-segmentation verifier (v_u) were re-trained.

In order to train e_{10}, we have used the NIST SD19 in the following way: the training and validation sets were composed of 195,000 and 28,000 samples from hsf_{0,1,2,3} respectively while the test set was composed of 60,089 samples from hsf_7. The

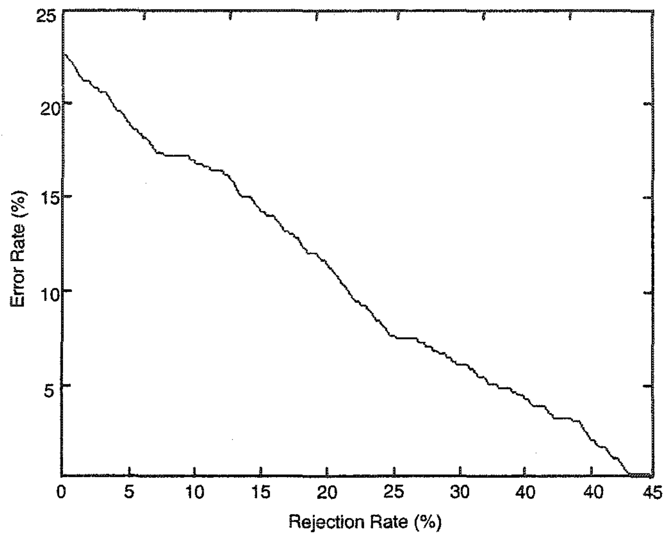


Figure 32 Error rate versus rejection rate for numerical amounts.

recognition rates (zero-rejection level) achieved by e_{10} were 99.66%, 99.65%, and 99.13% on the training, validation, and test sets respectively. The confusion matrices for both training and test sets are presented in Appendix 3. The verifiers were trained using the same methodology and number of samples described in the previous section. The recognition rates achieved by the over-segmentation and under-segmentation verifiers were 99.41% and 99.25% respectively. The experiments using numeral strings are based on 12,802 numeral strings extracted from the hsf_7 series and distributed into six classes: 2_digit (2,370), 3_digit (2,385) 4_digit (2,345), 5_digit (2,316), 6_digit (2,316) and 10_digit (1,217) strings respectively. These data exhibit different problems such as touching and fragmentation and they have also been used as a test set by Britto Jr. et al [14].

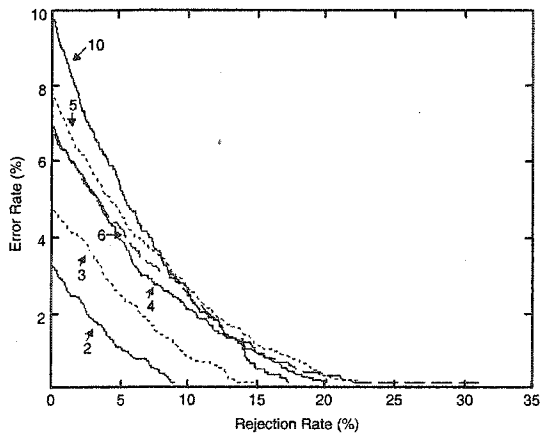
Table X summarizes the results for these experiments. In this table we can see the performance at the zero-rejection level for four different versions of the system. These results aim at showing the importance and contribution of each module to the global system. For the second experiment, which does not consider the post-processor module (number of digits) we present also the performance at three error

Table X

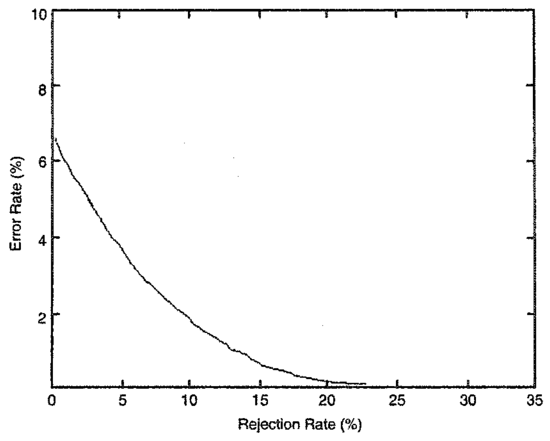
Recognition rates for the NIST experiment.

String	No. of	System Modules - (zero-rejection level)				Error Rate (e_{10}, v_o, v_u)		
Length	Strings	e_{10}	e_{10}, v_o, v_u	e_{10}, pp	All	2%	1%	0,5%
2	2370	91.56	**96.88**	96.41	97.21	96.08	94.93	93.88
3	2385	87.98	**95.38**	94.37	95.80	92.89	90.60	89.84
4	2345	84.91	**93.38**	91.96	94.11	88.18	85.78	84.36
5	2316	82.00	**92.40**	91.06	93.22	87.01	84.77	82.44
6	2169	85.66	**93.12**	94.12	95.90	88.16	85.68	84.03
10	1217	78.97	**90.24**	89.56	91.07	80.42	77.85	75.20

levels: 2, 1, and 0.5%. By comparing the two first experiments, we can observe the efficiency of the proposed verifiers on the NIST database, as well as conclude that such verifiers clearly improve the performance of the system. Figure 33a shows the behavior for strings of different lengths (from 2 to 10 digits) while Figure 33b exhibits the error-reject trade-off for all used fields in the test set.



(a)



(b)

Figure 33 Error rates versus rejection rates for the NIST database: (a) Error-reject trade-off for strings of different lengths and (b) Error-reject trade-off for all used fields in the test set.

By analyzing the system errors, we observed that they could be classified into four classes: confusions generated by e_{10} (67.8%), errors caused by segmentation (9%), errors caused by fragmentation (10.6%), and confusions generated by the low-level verifiers v_o and v_u (12.2%). Table XI summarizes these sources of errors as well as its frequency per string size. We can read this table in the following way. For 2-digit strings, we have detected 74 errors, which correspond to a global error rate of 3.12%. These errors are divided into the following: 43 due to the general-purpose recognizer, 15 to low-level verifiers, 9 to segmentation, and 7 to fragmentation.

Table XI

Distribution of the system errors.

String	G-P. Recognizer		Segmentation		Fragmentation		Verifiers		Total	
Length	Errors	%	Errors	%	Errors	%	Errors	%	Errors	%
2	43	1.81	9	0.38	7	0.29	15	0.64	74	3.12
3	78	3.26	9	0.38	8	0.33	15	0.64	110	4.61
4	104	4.46	17	0.73	13	0.56	20	0.86	154	6.61
5	126	5.43	14	0.60	25	1.08	11	0.48	176	7.59
6	98	4.49	13	0.59	8	0.37	31	1.42	150	6.87
10	89	6.73	9	0.68	23	1.74	8	0.60	129	9.75

The first class, which is the most frequent one, is related to the weakness of the general-purpose recognizer but also the difficult cases found in the test database as shown in Figure 34a. In this case, the digits 8 and 9 were classified as 6 and 7 respectively. The second class of errors is caused either by under-segmentation (20%), which is due to a lack of basic points in the neighborhood to the connection stroke, or effects generated by the segmentation algorithm such as ligatures (80%) which can be visualized in Figure 34b. The third class of errors is produced when the grouping algorithm fails. Usually it happens to images with poor quality (Figure 34d and sometimes images composed of two strokes (Figure 34c)).

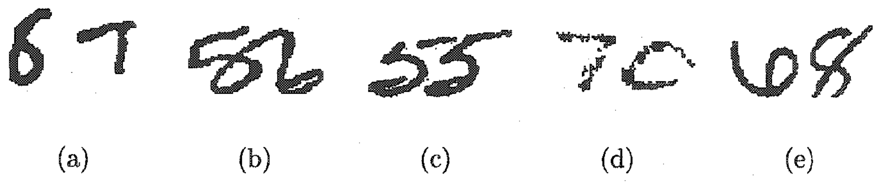


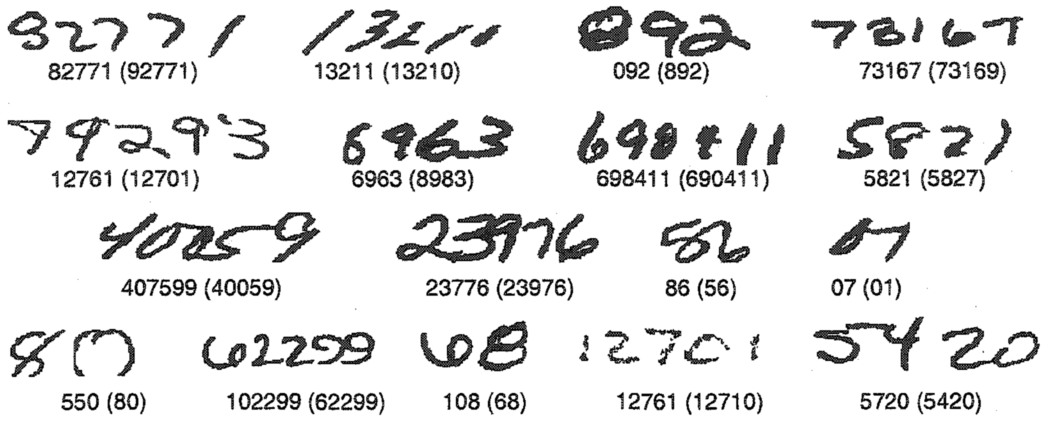
Figure 34 Different classes of errors generated by the system: (a) Recognition errors, (b) Ligature between two digits, (c) Natural fragmentation, (d) Fragmentation caused by noise, and (e) Isolated digits classified as under-segmentation by v_u.

The last class of errors is produced by the low-level verifiers, where v_u is responsible for 87% and v_o for 13% of the verification errors. The most frequent confusions generated by v_u are related to specific configurations of the digits "6", "0", and "8" (Figure 34e). In such cases, the digits have strong concavities and often they are damaged by pre-processing tools. Figure 35a shows some examples of misclassified images while Figure 35b shows examples of images containing touching or broken characters that were correctly recognized by the system.

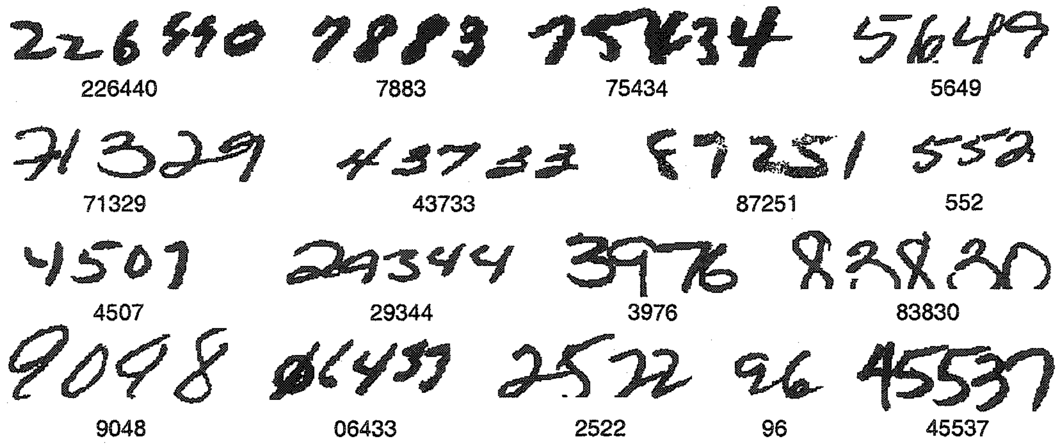
As we can notice, the part of the system that produces more errors is the general-purpose recognizer e_{10}. In order to reduce such errors and make the overall system more reliable, our next efforts will be focused on it.

5.2 Discussion and Comparison

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



(a)



(b)

Figure 35 Examples of digit strings (NIST SD19): (a) Not Recognized (the correct string is the one in parenthesis) and (b) Recognized.

optimization of the parameters for digit detection and grouping was carried out on different databases and the classifiers were trained on different databases as well.

As stated in Chapter 1, comparison with published methods is very delicate when we consider bank cheque recognition systems, since different databases and formats are used, different non-numerical classes are involved, and different sizes of databases are considered. For example, Lethelier et al [96] present a system for French bank

Table XII

Recognition rates on NIST databases reported by other authors

Authors	String	Number of	Zero-rejection	Error Rate		
	Length	tested strings	Level	2%	1%	0,5%
Ref. [55]	2	981	96.20	94.50	93.50	91.50
	3	986	92.70	86.00	79.50	70.50
	4	988	93.20	86.50	81.00	70.00
	5	988	91.10	81.00	77.50	70.50
	6	982	90.30	80.50	75.50	66.50
Ref. [95]	2	1000	95.23	-	95.20	-
	3	1000	88.01	-	87.90	-
	4	1000	80.69	-	80.50	-
	5	1000	78.61	-	78.40	-
	6	1000	70.49	-	70.20	-
Ref. [14]	2	2370	94.81	-	-	-
	3	2385	91.61	-	-	-
	4	2345	91.25	-	-	-
	5	2316	88.30	-	-	-
	6	2169	89.07	-	-	-
	10	1217	86.94	-	-	-

cheques, which copes with four non-numerical classes ("-", ".", ",", "F"). They claim a recognition rate of 60% (zero-rejection level) on a test set of 10,000 images of French bank cheques. Kaufmann and Bunke [74] propose a system for Swiss postal cheques that neither considers non-numerical classes nor the cents portion. The result achieved by this system is 79.3% (zero-rejection level) and 58.1% with an error rate of 0.2%.

Regarding the publications using the NIST database a more detailed comparison is possible. To facilitate this, Table XII reproduces the results of the most performative recognition systems described in Table III, Chapter 1. A direct comparison can be made with Ref. [14], once the same database has been used. By comparing the results reached by our system (Table X) with those reported by other authors (Table

XII), we can confirm that our system provides very good recognition rates at zero-recognition level and a very encouraging error-reject trade-off.

5.3 Summary

In this chapter we have presented comprehensive experiments on numerical amounts and NIST SD19 databases in order to support our ideas about modular system and verification. High recognition rates at zero-rejection level and a very encouraging error-reject trade-off have been obtained. Moreover, the results reached by our system compare favorably to other published methods. In spite of the fact we have got good results, we can observe from Table XI that such results can still be improved in some aspects. Since most of the errors of the system are generated from the general-purpose recognizer, we decide to dedicate some efforts to improve its performance. In the next chapter we discuss some attempts we have made in this direction.

CHAPTER 6

TOWARDS PERFORMANCE

So far, we have described in detail all modules of the system and how they interact with each other. We have demonstrated through experimentation on two different databases that the proposed strategy of low-level verification brings remarkable improvements to the recognition system. We also have seen that the modular framework proposed is suitable to process numerical fields in different applications. Thus, the results reported on numerical amounts and NIST database have been carried out using an identical system with the following exceptions: the optimization of the parameters for digit detection and grouping was carried out on different databases and the classifiers were trained on different databases as well.

In spite of the fact the system achieves compelling results, we have seen in the last chapter that the system still has different sources of errors, where the general-purpose classifier is the greatest one. In light of this, we decided to make some efforts towards the performance and reliability of the general-purpose recognizer. In this chapter we discuss three well known and established techniques we have investigated: high-level verification, feature selection, and ensemble of classifiers. All the experiments reported here consider the experts trained on NIST databases. This option was based on the fact that NIST is a bigger database and very well-known in the field of handwriting recognition.

6.1 High-level Verification

As mentioned elsewhere, the goal of the high-level verification is to confirm or deny the hypotheses produced by the general-purpose recognizer by recognizing them. In this section we discuss two different strategies of high-level verification: absolute and one-to-one. We will describe how such verifiers were implemented as well as to show their impacts on the numerical string recognition system. All verifiers discussed in this section are MLPs (as defined in Section 2.1.2).

In order to combine the outputs of the general-purpose classifier with the high-level verifiers we have tried different methods such as average and product. In the case of the one-to-one verifier we also have tried to use the output of the verifier without any combination. In all our experiments, we have got the best results by using either average or product, i.e., both combination rules produced very similar results. For simplicity, in the next subsections we show the results and examples by using the product rule.

6.1.1 Absolute High-Level Verifier

In this experiment ten absolute verifiers (one for each numerical class) were considered. Each verifier was trained with two classes: digit and $\overline{\text{digit}}$. For example, for the verifier of the digit class "0", we have used all zeros of the training set (19,500 samples) for the digit class and the same number of other digits for the $\overline{\text{digit}}$ class. We have tried different feature sets such as concavity analysis in 8-Freeman directions, and Moments [68], Edge Maps [25], and histograms. The feature set that produced better results in terms of recognition rates was the same one used by the general-purpose recognizer. Table XIII shows the recognition rates reached by each absolute high-level verifier.

The idea behind these verifiers is to confirm or deny the hypotheses yielded by the general-purpose recognizer. For instance, suppose that the main classifier provided the correct result in the second position of the list (top-2). In this case, if top-1 is denied by the verifier and top-2 is confirmed, the list could be re-ranked so that top-2 becomes top-1. This idea is exemplified in Figure 36, where "top-1:0(0.55)" means that the classifier recognized the input pattern as a "0" with a measurement level (or estimation of posteriori probability) of "0.55".

Table XIII

Recognition rates achieved by the absolute high-level verifiers.

Class	RR (%)	Class	RR (%)
0	99.66	5	99.66
1	99.08	6	99.50
2	99.58	7	99.84
3	99.20	8	99.28
4	99.80	9	99.10

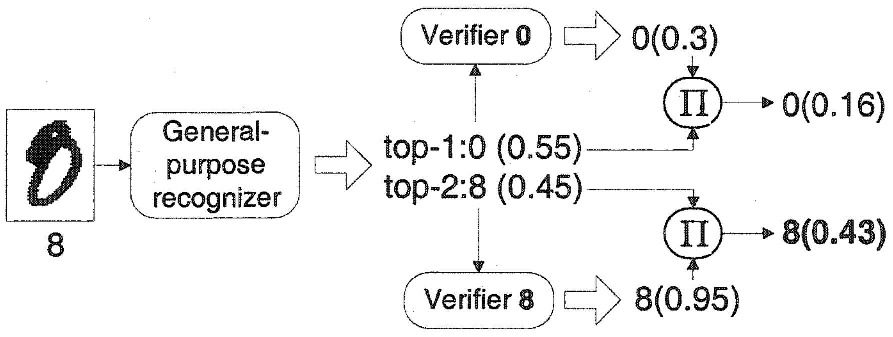


Figure 36 Absolute verifiers.

We have observed that this strategy of verification produces an improvement to naturally isolated digits, however, when the system faces problems such as touching and fragmentation, it does not seem very appropriate.

Table XIV presents the results on the different string lengths we are working with. As we can see, the overall performance achieved for numeral strings by the system that considers the absolute verifiers is almost the same than that reached by the original system for zero-rejection level. However, when dealing with error rates fixed at low level (e.g. 0.5%) this strategy is much worse. In spite of the fact that this scheme of verification succeeds in correcting some errors, it also decreases the probabilities since the final result is the product of probabilities. This makes the implementation of a rejection scheme more difficult, i.e., more correctly classified samples will be rejected in order to achieve a low error rate.

Table XIV

Performance of the system on strings of digits using high-level verifiers.

String Length	No. of Strings	Original System RR (%)	Original System Error at 0.5%	Absolute Verifier RR (%)	Absolute Verifier Error at 0.5%	One-to-One Verifier RR (%)	One-to-One Verifier Error 0.5%
2	2370	96.88	93.88	96.65	88.60	96.10	88.60
3	2385	95.38	89.84	95.03	84.90	94.98	83.79
4	2345	93.38	84.26	92.97	77.62	92.91	76.13
5	2316	92.40	82.44	92.01	75.50	91.03	75.29
6	2169	93.12	84.03	92.60	76.42	91.77	75.90
10	1215	90.24	75.20	89.51	63.19	89.00	62.80

6.1.2 One-to-One High-Level Verifier

The second strategy of high-level verification that we have implemented was the one-to-one verifier. Such a strategy is straightforward and makes it easy to concentrate on the local difference between two classes. In order to determine the main confusions of the general-purpose recognizer, we carried out an error analysis on the validation set of isolated digits and we observed 39 different confusions (theoretically, the number of possible confusing digit pairs is $10 \times 9/2 = 45$). Theoretically, we could solve 75.0% and 62.7% of all errors focusing on the top-39 and top-20 confusions, respectively. Therefore, it seems more cost effective focusing on top-20 confusions, since we have to deal with about 50% of the confusions produced by the system. Table XV presents top-20 confusions with their respective frequencies.

We trained each verifier with 39,000 samples (19,500 for each class of digit involved) using the same feature set that we have used for our general-purpose recognizer. For these verifiers we have tried the same feature sets we have experimented in the

Table XV

Top 20 digit confusion with frequencies

Confusion	Frequency	Confusion	Frequency
8-0	48	7-1	17
3-2	40	9-5	17
2-1	29	7-2	16
4-0	28	9-8	15
7-3	28	8-2	15
9-7	28	6-4	14
9-4	27	6-5	12
5-3	22	8-5	11
6-0	22	9-0	11
8-3	22	8-4	10

previous section, and again, the one that brought better results was the same one used by the general-purpose recognizer. The manner that such verifiers work is depicted in Figure 37. It can be observed that, when there is a verifier related to the top-2 outputs of the general-purpose recognizer, it will try to re-rank them in order to get the correct answer. If there is not such a verifier, the system assumes the output yielded by the general-purpose recognizer. In this example, the main recognizer misclassified an image, which belongs to the class "9", but the verifier succeed in re-ranking the output of the recognizer to get the right answer. The idea here is similar to the absolute verifier, but instead of using two different verifiers, in this case just one verifier is considered. The results achieved by this strategy for different string lengths are reported in Table XIV.

The performance here is slightly worse than that reached by the system with absolute verifiers. By analyzing some errors, we have noticed that this strategy did not succeed in correcting the main confusions, but caused some other misclassifications instead. In order to enhance the results supplied by this strategy, it will be necessary to

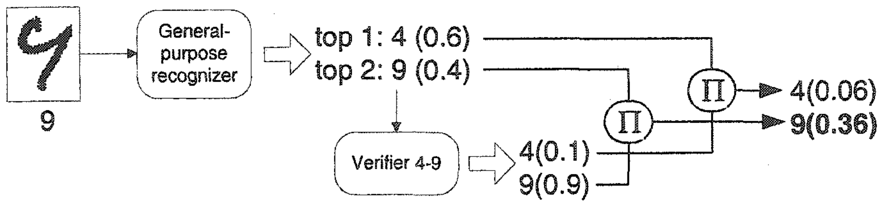


Figure 37 One-to-one verifier.

improve the verifier training set by including misrecognized samples. The difficulties of implementing such a solution lie in two points:

- Lack of samples to improve the database. If we consider our most frequent confusion (8-0), we have just 48 cases.
- If we include a few misrecognized samples in the training set of the verifier, probably we will introduce noise to our models. We can visualize this problem from Figure 38.

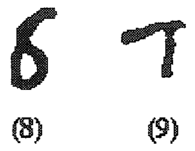


Figure 38 Misrecognized samples: 8 confused with 6 and 9 confused with 7.

6.1.3 Discussion

We have described two different strategies of high-level verification in order to improve the recognition rate of the system. As we can notice, both strategies (absolute and one-to-one) do not achieve satisfactory results on numeral strings. Nevertheless, they seem to be very effective when either there is a diversity of samples (confusions) to train the verifiers or when the system has a weak general-purpose recognizer, e.g., the system presented by Britto Jr. et al in [14].

One strategy could be the finding of different feature sets to feed the high-level verifiers. But in this case, the system should overcome the same kind of problems faced by multi-classifier systems, e.g., run time inefficiency and system complexity. Another solution could be the use of the samples rejected by the general-purpose recognizer to train the verifiers. In this way, they would be more specialized on the difficult cases. We also have noticed that when the recognition system requires very low error rates, in addition to the correct classification the verifiers must also provide high confidence levels, otherwise it becomes very difficult to implement an efficient rejection scheme.

6.2 Feature Selection

An important issue in constructing classifiers is the selection of the best discriminative features. In many applications, it is not unusual to find problems involving hundreds of features. However, it has been observed that beyond a certain point, the inclusion of additional features leads to a worse rather than better performance. Moreover, the choice of features to represent the patterns affects several aspects of the pattern recognition problem such as accuracy, required learning time, and the necessary number of samples.

This apparent paradox presents us with a feature selection problem in automatic design of pattern classifiers. Such a problem refers to the task of identifying and selecting an effective subset of features to represent patterns from a larger set of often mutually redundant or even irrelevant features. This is not a trivial problem since features are seldom entirely independent. There may be redundancy, where certain features are correlated so that it is not necessary to include all of them in modelling, and interdependence, where two or more features between them convey important information that is obscure if any of them is included on its own.

In the context of practical applications such as handwriting recognition, feature selection presents a multi-criterion optimization function, e.g. number of features and accuracy of classification. Genetic algorithms (GAs) offer a particularly attractive approach to solve this kind of problems since they are generally quite effective in rapid global search of large, non-linear and poorly understood spaces. It has been shown that GAs can be applied to general NP-complete problems. Moreover, simultaneous allocation of search effort to many regions of the search space contributes the power of GAs. In the last decade, GAs have been largely applied to the feature selection problem [83, 138, 146, 160]. The approach often combines different optimization objectives into a single objective function. The main drawback of this kind of strategy lies in the difficulty of exploring different possibilities of trade-off between classification accuracy and different subsets of selected features. In order to overcome this kind of problem, Emmanouilidis et al [36] propose the use of a multi-objective genetic algorithm (MOGA) to perform feature selection.

In this section we discuss the use of MOGA as a means to search for subsets of features, which contain discriminatory information to classify handwritten digit strings. The proposed strategy takes into account an efficient MOGA [147] to generate a set of alternative solutions and the use of a cross-validation method to indicate the best accuracy/complexity (number of features) trade-off. We demonstrate that the cross-validation is very important when working with a set of alternative solutions and it can not be neglected as in [36]. The classification accuracy is supplied by MLP in conjunction with the sensitivity analysis [111]. Such an approach makes it feasible to deal with huge databases in order to better represent the pattern recognition problem during the fitness evaluation. Some advantages of the proposed methodology include the ability to accommodate multiple criteria such as number of features and accuracy of the classifier, as well as the capacity to deal with huge databases in order to represent well the pattern recognition problem. Moreover, the use of MOGA help to avoid the phenomenon of premature convergence presented by simple GA.

6.2.1 Related Works

The preliminary works on feature selection started in the early 60s. The approaches at that time were based on probabilistic measures of class separability and on entropies. In most of the methods the independence of features was assumed and the features were selected on the basis of their individual merits. Since such methods ignore the interactions among features, they usually produce unsatisfactory subsets of features.

In order to perform feature selection taking into account the relationship between features, three approaches can be found in the literature [29]: Complete, Heuristic and Randomized searches. Since complete search over all possible subsets of a feature set (2^N where N is the number of features) is not computationally feasible in practice, several authors have explored the use of heuristics for feature subset selection, often in conjunction with branch and bound search. Forward selection and backward elimination are the most common sequential branch and bound search algorithms used in feature selection [73, 115]. Most of the current approaches assume monotonicity of some measure of classification performance. This ensures that adding features does not worsen the performance. However, many practical scenarios do not satisfy the monotonicity assumption. Moreover, this kind of search is not designed to handle multiple selection criteria.

Randomized algorithms make use of randomized or probabilistic steps or sampling processes. Several researchers have explored the use of such algorithms for feature selection [78, 101] while others have explored the use of randomized population-based heuristic search techniques such as genetic algorithms for feature selection for decision tree and nearest neighbor classifier [76, 140, 146] or neural networks [160, 161]. The advantage of feature selection techniques that employ GAs is that they do not require the restrictive monotonicity assumption. They can also deal with the use of multiple selection criteria, e.g., classification accuracy, feature measurement cost, etc. This makes them particularly attractive in the design of pattern classifiers in many practical scenarios such as signature verification [136], medical diagnosis [160], facial modelling [63] and handwriting recognition [76].

Due to the ability of GAs to deal with multi-objective optimization, several authors have explored them for feature selection for handwritten character recognition [142, 76, 102]. Feature selection using GA is often performed by aggregating different objectives into a single and parameterized objective, which is achieved through a linear combination of the objectives. The main drawback of this approach is that it is very difficult to explore different trade-offs between accuracy and different subsets of selected features.

In order to overcome this kind of problem, Emmanouilidis et al [36] propose the use of a MOGA to perform feature selection. Notwithstanding that only small databases were considered in this work, they achieved interesting results.

6.2.1.1 Filter and Wrapper Approaches to Feature Selection

Feature selection algorithms can also be classified into two categories based on whether or not feature selection is performed independently of the learning algorithm used to construct the classifier. If feature selection is done independently of the learning algorithm, the technique is said to follow a filter approach. Otherwise, it is said to follow a wrapper approach [73]. While the filter approach is generally computationally more efficient than the wrapper approach, its major drawback is that an optimal selection of features may not be independent of the inductive and representational biases of the learning algorithm that is used to construct the classifier. On the other hand, the wrapper approach involves the computational overhead of evaluating candidate feature subsets by executing a given learning algorithm on the database using each feature subset under consideration.

6.2.2 Multi-Objective Optimization using GAs

6.2.2.1 Definitions

A general multi-objective optimization problem consists of a number of objectives and it is associated with a number of inequality and equality constraints. Mathematically, the problem can be written as follows [137].

$$\text{Minimize (or Maximize)} \quad f_i(x) \quad i = 1, \ldots, N$$

$$\text{subject to:} \begin{cases} g_j(x) \leq 0 & j = 1, 2, \ldots, J \\ h_k(x) = 0 & k = 1, 2, \ldots, K \end{cases} \tag{6.1}$$

The parameter x is a p dimensional vector having p decision variables. Solutions to a multi-objective optimization problem can be expressed mathematically in terms of nondominated or superior points. In a minimization problem, a vector $x^{(1)}$ is partially less than another vector $x^{(2)}$, $(x^{(1)} \prec x^{(2)})$, when no value of $x^{(2)}$ is less than $x^{(1)}$ and at least one value of $x^{(2)}$ is strictly greater than $x^{(1)}$. If $x^{(1)}$ is partially less than $x^{(2)}$, we say that the solution $x^{(1)}$ *dominates* $x^{(2)}$. Any member of such vectors which is not dominated by any other member is said to be *nondominated.* The optimal solutions to a multi-objective optimization problem are nondominated solutions. They are also known as Pareto-optimal solutions.

For example, in the case of minimization for two criteria,

$$\begin{cases} \text{Minimize} & f(x) = (f_1(x), f_2(x)) \\ \text{such that} & x \in X(\text{the feasible region}) \end{cases}$$

a potential solution $x^{(1)}$ is said to dominate $x^{(2)}$ iff:

$$\begin{aligned} &\forall i \in \{1,2\} : f_i(x^{(1)}) \leq f_i(x^{(2)}) \quad \wedge \\ &\exists j \in \{1,2\} : f_j(x^{(1)}) < f_j(x^{(2)}) \end{aligned} \tag{6.2}$$

6.2.2.2 Classical Approach

A common difficulty with multi-objective optimization problem is the conflict between the objectives: in general, none of the feasible solutions allow simultaneous optimal solutions for all objectives. In other words, individual optimal solutions of each objective are usually different. Thus, mathematically the most favorable Pareto-optimum is that solution which offers the least objective conflict. One of the most classical methods is the weighted sum. In this strategy, multiple objective functions are combined into one overall objective $F(x)$ such that:

$$F(x) = \sum_{i=1}^{N} \omega_i f_i(x) \tag{6.3}$$

where $x \in X$, X is the objective space, the weights ω_i are fractional numbers ($0 \leq \omega_i \leq 1$), and $\sum_{i=1}^{N} \omega_i = 1$. However, setting up an appropriate weight vector also depends on the scaling of each objective function. It is likely that different objectives take different orders of magnitude. When such objectives are weighted to form a composite objective function, it would be better to scale them appropriately so that each has more or less the same order of magnitude.

In this method, the optimal solution is controlled by the weight vector ω. It is clear from the above equation that the preference of an objective can be changed by modifying the corresponding weight. In most cases, each objective is first optimized and all objective function values are computed at each individual optimum solution. Afterwards, depending on the importance of objectives, a suitable weight vector is chosen and the single-objective problem given in Equation 6.3 is used to find the desired solution.

Figure 39 shows an example of this approach where a two-objective problem is considered. Once the weight vector is defined we can calculate the composite function F. Its contour surface can then be visualized in the objective space, as shown by lines "a", "b", "c" and "d". Since F is a linear combination of both objectives f_1 and f_2 we would expect a straight line as the contour line of F on the objective space. This is because any solution on the contour line will have the same F value. If considered carefully, this contour line is not an arbitrary one. Its slope is related to the choice of the weight vector. In fact, for two objectives, its slope is $-w_1/w_2$. The location of the line depends on the value of F on any point of the line. The effect of lowering the contour line from "a" to "b" is in effect jumping from solutions of higher F values to a lower one.

In a minimization problem, the task is to find the contour line with the minimum F value. This happens with the contour line which is tangential to the search space and also lies in the bottom-left corner of this space. In Figure 39 this line is represented by "d". The tangent point "A" is the minimum solution of F, and is consequently the Pareto-optimal solution corresponding to the weight vector.

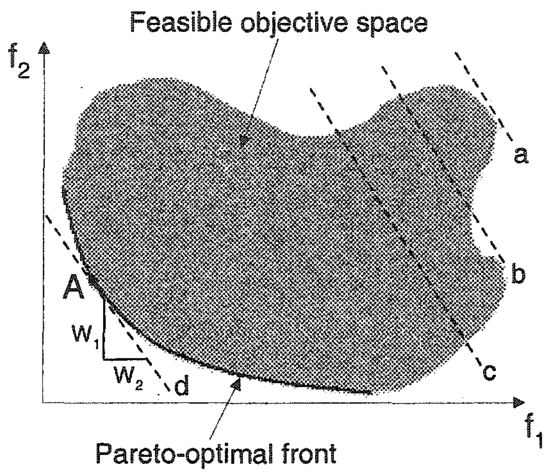


Figure 39 Illustration of the weighted sum approach on a convex Pareto-optimal front [30].

The only advantage of using this technique is the emphasis of one objective over the other. As we have seen, the optimization of the single objective may guarantee a Pareto-optimal solution but results in a single point solution. However, in real world situations we usually need different alternatives in order to make a decision. Moreover, if some of the objectives are noisy or have a discontinuous variable space, this method may not work properly. The main drawback of this approach is its sensitivity towards weights, i.e., the solution obtained largely depends on the underlying weight vector.

Deb in [30] mentions further potential problems with these approaches, i.e., application areas where their use is restricted. Moreover, this kind of method requires several optimization runs to obtain an approximation of the Pareto-optimal set. As the runs are performed independently from each other, usually synergies can not be exploited which, in turn, may create a high computation overhead.

6.2.2.3 Pareto-based Approach

In order to overcome the difficulties mentioned above, Pareto-based evolutionary optimization has become an alternative to classical techniques such as weighted sum method. This approach was first proposed by Goldberg in [48] and it explicitly uses Pareto dominance to determine the reproduction probability of each individual. Basically, it consists of assigning rank 1 to the nondominated individuals and removing them from contention, then finding a new set of nondominated individuals, ranked 2, and so forth. Figure 40 depicts the ranking by fronts.

Pareto-based ranking correctly assigns all nondominated individuals the same fitness, however, this does not guarantee that the Pareto set is uniformly sampled. When presented with multiple equivalent optima, finite populations tend to converge to only one of them, due to stochastic errors in the selection process. This phenomenon, known as genetic drift, has been observed in natural as well as in artificial evolution, and can also occur in Pareto-based evolutionary optimization.

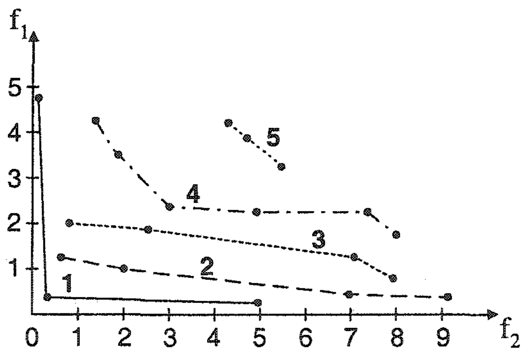


Figure 40 Ranking of a population by fronts.

In order to avoid such a problem, Goldberg and Richardson in [49] propose the additional use of fitness sharing. The main idea behind this is that individuals in a particular niche have to share the available resources. The more individuals are located in the neighborhood of a certain individual, the more its fitness value is degraded. In the following section we present the Pareto-based algorithm we have used as well as an implementation of the sharing function.

6.2.2.4 Nondominated Sorting Genetic Algorithm (NSGA)

Over the past decade, a number of multi-objective evolutionary algorithms have been proposed. Zitzler et al in [168] provide a systematic comparison of various evolutionary approaches to multi-objective optimization using six carefully chosen test functions. In this work, they found that NSGA (with elitism) proposed by Srinivas and Deb in [147] surpasses several other methods. Besides, such a method has been applied to solve various problems [110, 158]. For these reasons we opted to use such an algorithm in our study.

The idea behind the NSGA is that a ranking selection method is used to emphasize good points and a niche method is used to maintain stable subpopulations of good points. It differs from simple GA only in the way the selection operator works. The crossover and mutation remain as usual. Before the selection is performed, the population is ranked based on an individual's nondomination. The nondominated individuals present in the population are first identified from the current population. Then, all these individuals are assumed to constitute the first nondominated front in the population and assigned a large dummy fitness value. The same fitness value is assigned to give an equal reproductive potential to all these nondominated individuals. This is exemplified in Figure 41a. In such a case, a population of six individuals was classified into three nondominated fronts and each individual of the first front received a large dummy fitness (6.00 in this example).

In order to maintain the diversity in the population, these classified individuals are then shared with their dummy fitness values. Sharing is achieved by performing

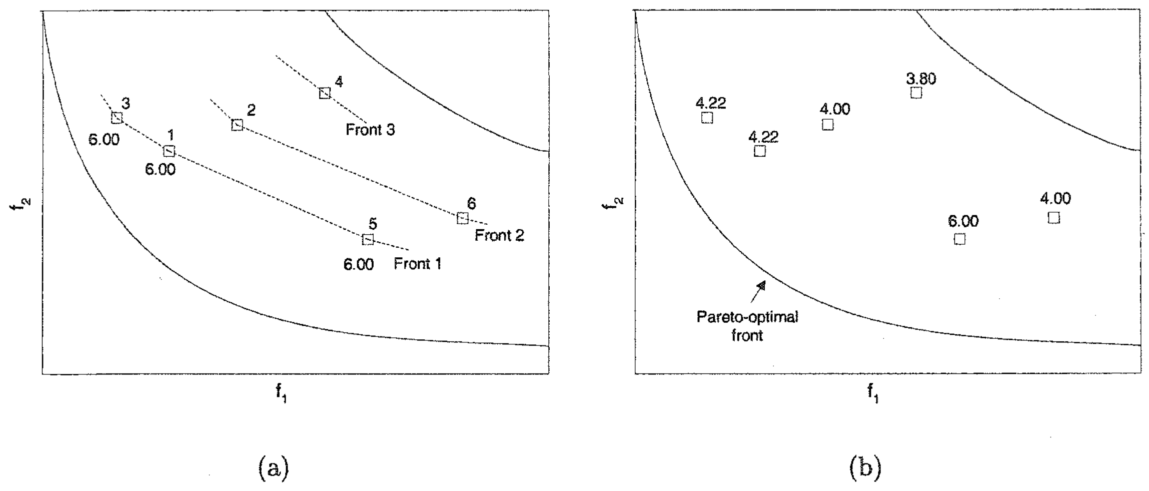


Figure 41 Sorting population: (a) The population is classified into three nondominated fronts and (b) Shared fitness values of six solutions.

selection operation using degraded fitness values obtained by dividing the original fitness value of an individual by a quantity proportional to the number of individuals around it. After sharing, these nondominated individuals are ignored temporarily to process the remaining population in the same way to identify individuals for the second nondominated front. These new sets of points are then assigned a new dummy fitness which is kept smaller than the minimum shared dummy fitness of the previous front. This process is continued until the entire population is classified into several fronts. This process is illustrate in Figure 41b. It can be observed from this Figure that the individuals "1" and "3" had their fitness shared because they are close to each other. In this case, their fitness were reduced from 6.00 to 4.22. Then, the dummy fitness is assigned to the individuals of the second front by multiplying the lowest value of the first front by a constant k (let us say $k = 0.95$ for this example). Therefore, the individuals of the second front will receive a dummy fitness of 4.00 (4.22×0.95). Since the two individuals of the second front are not close to each other, their dummy fitness is maintained and a dummy fitness is assigned to the individual of the last front ($4.00 \times 0.95 = 3.80$).

The population is then reproduced according to the dummy fitness values. Since individuals in the first front have the maximum fitness value, they get more copies than the rest of the population. This was intended to search for the nondominated regions of Pareto-optimal fronts. The efficiency of NSGA lies in the way multiple objectives are reduced to a dummy fitness function using nondominated sorting procedures.

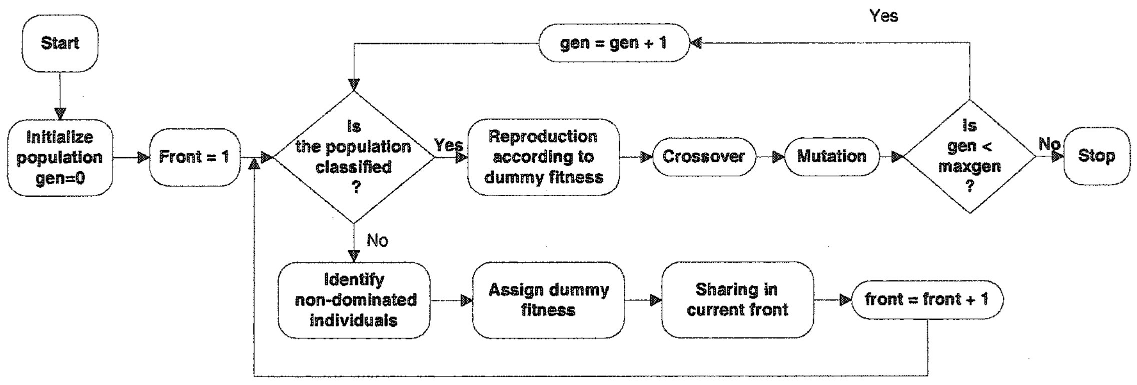


Figure 42 Flow chart of NSGA.

Figure 42 shows a flow chart of NSGA. The algorithm is similar to a simple GA except for the classification of nondominated fronts and the sharing operation. The sharing in each front is achieved by calculating a sharing function value between two individuals in the same front as:

$$Sh(d(i,j)) = \begin{cases} 1 - \left(\frac{d(i,j)}{\sigma_{share}}\right)^2 & \text{if} \quad d(i,j) < \sigma_{share} \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

where $d(i,j)$ is the distance between two individuals i and j in the current front and σ_{share} is the maximum distance allowed between any two individuals to become members of a niche. In any application of sharing, we can implement either genotypic sharing, since we always have a genotype (the encoding), or phenotypic sharing.

However, Deb and Goldberg in [31] indicate that in general, phenotypic sharing is superior to genotypic sharing. Thus, we have used a phenotypic sharing which is calculated from the normalized Euclidean distance between the objective functions.

The parameter σ_{share} can be calculated as follows [31]:

$$\sigma_{share} \approx \frac{0.5}{\sqrt[p]{q}} \tag{6.5}$$

where q is the desired number of distinct Pareto-optimal solutions and p is the number of decision variables. Although the calculation of σ_{share} depends on this parameter q, it has been shown [147] that the use of the above equation with $q \approx 10$ works in many test problems.

6.2.3 Proposed Methodology

Basically the proposed methodology works as follows: NSGA produces a set of solutions, called Pareto-optimal, which is validated through a second validation set not used during the learning of the classifier. Based on the validation curve, we pick one solution and thereafter train it to obtain a new optimized classifier. In the following subsection, we describe in details the proposed methodology, which is depicted in Figure 43. We can notice from this Figure that the strategy takes as input one model, which is associated to a feature set, and produces as output another model associated to an optimized feature set.

6.2.3.1 Implementation of NSGA

In our experiments, NSGA is based on bit representation, one-point crossover, bit-flip mutation and roulette wheel selection (with elitism). The following parameter settings were employed:

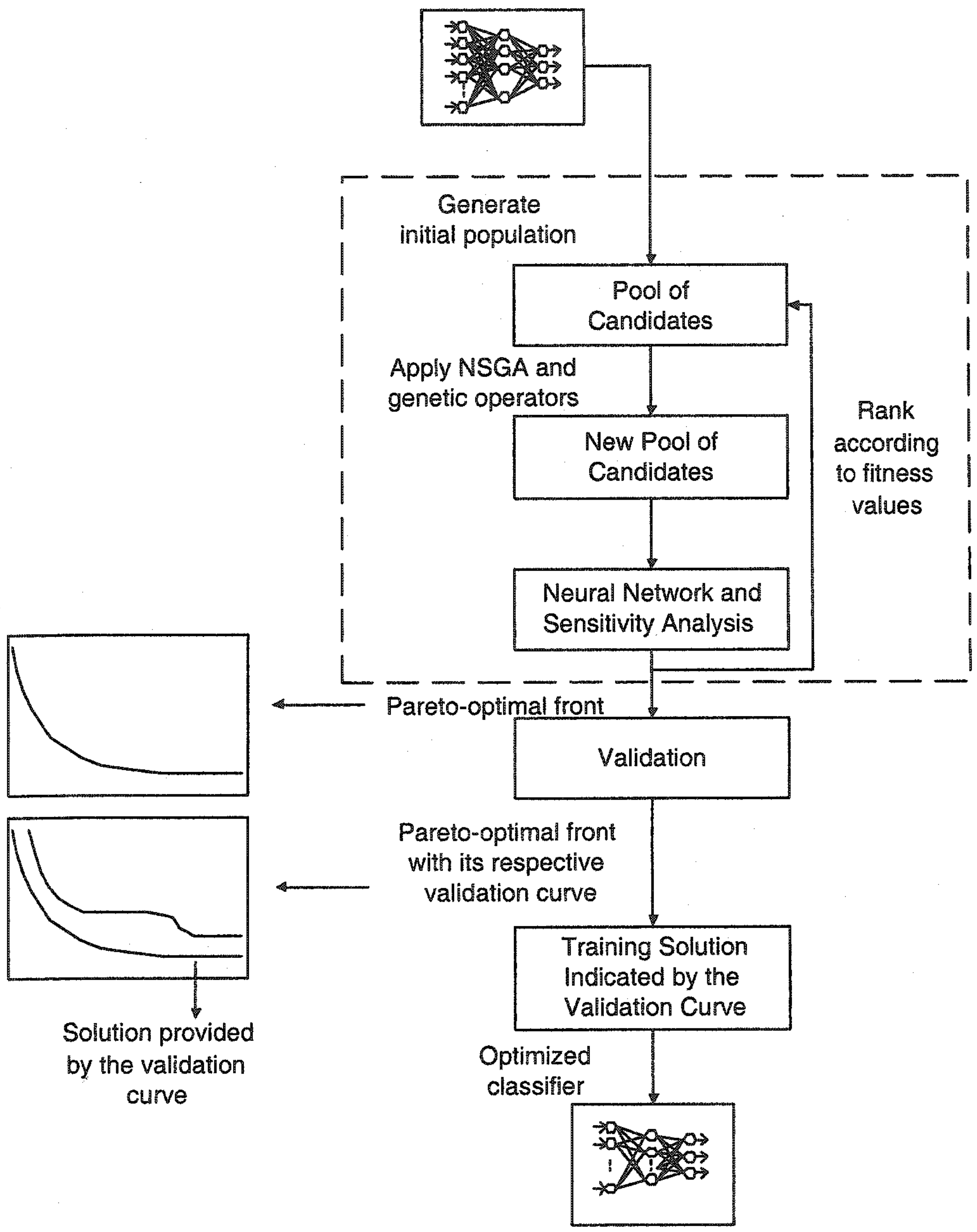

Generate
initial population
Pool of Candidates
Apply NSGA and genetic operators
New Pool of Candidates
Rank according to fitness values
Neural Network and Sensitivity Analysis
Pareto-optimal front
Validation
Pareto-optimal front with its respective validation curve
Training Solution Indicated by the Validation Curve
Solution provided by the validation curve
Optimized classifier

Figure 43 Flow chart of the proposed methodology.

- Population size: 128
- Number of generations: 1000
- Probability of crossover (P_c): 0.8

- Probability of mutation (P_m): 0.007
- Niche Distance (σ_{share}): 0.45

In order to define the probabilities of crossover and mutation, we have used the one-max problem, which is probably the most frequently-used test function in research on GAs because of its simplicity [19]. This function measures the fitness of an individual as the number of bits set to one on the chromosome. We have used a standard GA with a single-point crossover and the maximum generations of 1000. The fixed crossover and mutation rates are used in a run, and the combination of the crossover rates 0.0, 0.4, 0.6, 0.8 and 1.0 and the mutation rates of $0.1/L$, $1/L$ and $10/L$, where L is the length of the chromosome. The best results were achieved with $P_c = 0.8$ and $P_m = 1/L$. All these experiments are reported in [8]. Such results confirmed the values reported by Miki et al in [109]. The parameter σ_{share} was first defined using Equation 6.5, then it was tuned empirically.

As discussed elsewhere, our goal is to find the best accuracy/complexity trade-off for the classifier. This means that two objectives must be considered: minimization of the number of features and minimization of the error rate of the classifier on the validation set. Computing the first one is simple, i.e., the number of selected features (bit = 1). The problem lies in computing the second one, i.e., the error rate supplied by the classifier. Regarding a wrapper approach, in each generation, evaluation of a chromosome (a feature subset) requires training the corresponding neural network and computing its accuracy. This evaluation has to be performed for each of the chromosomes in the population. Since such a strategy is not feasible due to the limits imposed by the learning time of the huge training set considered in this work, we have adopted the strategy proposed by Moody and Utans in [111], whom use the sensitivity measure S_i to evaluate the change in training error that would result if input x_i were removed from the network. The sensitivity of the network model to variable is defined as:

$$S_i = \frac{1}{N} \sum_j S_{ij} \tag{6.6}$$

where S_{ij} is the sensitivity computed for exemplar x_j.

$$S_{ij} = SE(\bar{x}_i) - SE(x_{ij}) \tag{6.7}$$

with

$$\bar{x}_i = \frac{1}{N} \sum_{j=1}^{N} x_{ij} \tag{6.8}$$

S_i measures the effect on the training squared error (SE) of replacing the i^{th} input x_i by its average $\bar{x}_i$ for all N exemplars. Moody and Utans show that when variables with small sensitivity values (S_i) with respect to the network outputs are removed, they do not influence the final classification. So, in order to evaluate a given feature subset we replace the unselected features by their averages. In this way, we avoid training the neural network and hence turn the wrapper approach feasible for our problem. We call this strategy modified-wrapper. Such a kind of scheme has been employed also by Yuan et al in [161].

6.2.3.2 Validating the Pareto-optimal front

As depicted in Figure 43, the last step of the strategy consists of choosing the best solution from the Pareto-optimal front. After several experiments, we realized that the Pareto-optimal front by itself does not provide enough information to select the best solution. Often, the best solution found in the Pareto-optimal front does not have good generalization power on a different database. In order to overcome this kind of problem, we propose the use of a validation database, which is not used during the feature selection procedure, to verify the generalization power of the Pareto-optimal front.

6.2.4 Experiments

In the following section we report the results achieved by the foregoing methodology on isolated and strings of digits as well. All experiments in this work were based on a single-population master-slave MOGA. In this strategy, one master node executes the genetic operators (selection, crossover, and mutation), and the evaluation of fitness is distributed among several slave processors. We have used a Beowulf cluster with 17 (one master and 16 slaves) PCs (1.1Ghz CPU, 512Mb RAM) to execute our experiments.

6.2.4.1 Experiments on Isolated Digits

The database used to assess the impact of different subsets of inputs during the feature selection was the same we have employed for validation during of training of the classifier (28,000 samples from hsf_0123).

In order to show the limitations of the weighted-sum approach, we first tried to optimize the classifier using it. As expected, the results achieved by the weighted-sum approach presented a premature convergence to a specific region of the search space instead of maintaining a diverse population. This kind of behavior can be explained by the sensitivity towards weight presented by the weighted-sum approach. Since we have chosen weights to favor solutions with a small error rate rather than a small number of features, the selection pressure drove the search to the region where the error rates are smaller. Thus, after several trials using different weights we did not succeed in finding the Pareto-optimal front but rather an approximation of the Pareto-optimal solutions (Figure 44).

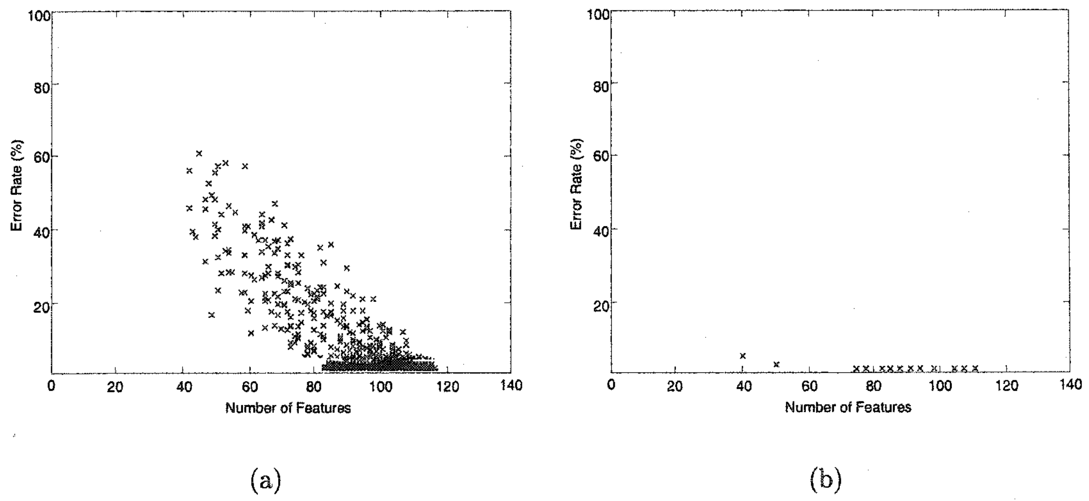
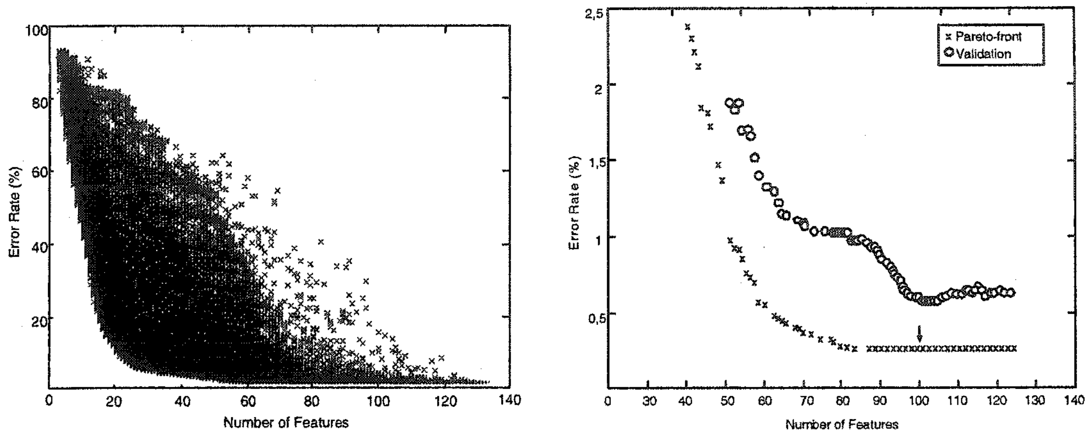


Figure 44 Feature selection using the weighted-sum approach (a) evolution of the population in the objectives plane (one trial) and (b) Pareto-optimal solutions found by the classical approach after several trials.

As we have discussed in Section 6.2.2.3, the Pareto-based approach was designed to overcome this kind of problem. Since NSGA uses a niching technique to preserve the diversity in the population, this algorithm is able to deal with the problem of converging prematurely to a specific region of the search space. Therefore, it can guide the search towards the Pareto-optimal set. Figure 45a depicts the evolution of the population in the objectives plane from the first generation to the last one. This plot demonstrates the efficacy of NSGA in converging close to the Pareto-optimal front with a wide variety of solutions.

As discussed in Section 6.2.3, after finding the Pareto-optimal front the next step is to find a solution. In order to perform this task we have used the second validation database (validation-2 – Appendix 1). Figures 45b shows the Pareto-optimal front as well as its correspondent validation curve, which depicts the performance of the



(a) (b)

Figure 45 Feature selection using a Pareto-based approach (a) Evolution of the population in the objective plane, (b) Pareto-optimal front found by NSGA and its correspondent validation curve.

entire Pareto-optimal front on this new validation set. After analyzing the validation curve, we selected a solution with 100 features (solution signed with an arrow in Figure 45b) and error rate on the new validation set smaller than 1% to retrain the general-purpose recognizer.

Thereafter we trained a new classifier using such a solution. The results are summarized in Table XVI. In order to validate the proposed strategy, we applied it to other two feature sets, namely, Directional Distance Distribution [118] (hereafter called Distances for simplicity) and Edge Maps [25]. It should be noted, though, that the original feature set of Distances proposed by Oh and Suen [118] contains 256 features. After carrying out some experiments with different strategies of zoning, we realized that using 96 features (6 zones: 3 horizontal and 2 vertical) we could achieve the same results as using 256 features (16 symmetrical zones). A description of both feature sets can be found in Appendix 2.

Table XVI presents a comparison between the original and the optimized classifiers for the three different feature sets. It shows that our strategy of feature selection succeeded in reducing the complexity of the feature set while keeping the recognition rates of the classifiers at the same level (for zero-rejection level and also for error rate fixed at low level – 0.5%).

Table XVI

Comparison between the original and optimized classifiers.

Feature Set	Original Classifiers			Optimized Classifiers		
	No. of Features	Rec. Rate (%)	Error at 0.5%	No. of Features	Rec. Rate (%)	Error at 0.5%
Concavities	132	99.13	98.50	100	99.16	98.54
Distances	96	98.17	92.80	90	98.21	92.98
Edge Maps	125	97.04	85.10	105	97.10	85.70

6.2.4.2 Experiments on Strings of Digits (1)

In the first experiment with strings of digits we just replaced the original classifier (e_{10}) by the optimized one presented in the previous section. The idea here is to find out whether the optimized classifier is good enough to recognize strings of digits. As discussed before, such a problem is much more complicated than the problem of naturally isolated digits since the system must face complications such as noise, fragmentation and touching digits. In order to answer the above question, we have applied the optimized classifier to recognize strings of digits. It is worthy of remark that both low-level verifiers remain the same. The results of this experiments are presented in Table XVII.

In a second time we carried out the feature selection for both verifiers and used them in the system afterwards. To perform this task, we have used the proposed methodology where the databases used to validate the Pareto-front for the over-segmentation

Table XVII

Performance of the system on strings of digits using the general-purpose classifier optimized with isolated digits.

String Length	Original Classifiers		Optimized Classifiers	
	Rec. Rate (%)	Error at 0.5%	Rec. Rate (%)	Error at 0.5%
2	96.88	93.88	97.21	94.00
3	95.38	89.84	94.62	89.72
4	93.38	84.36	93.34	84.38
5	92.40	82.44	92.36	82.67
6	93.12	84.03	92.60	83.70
10	90.24	75.20	89.19	74.77

and under-segmentation verifiers have 7,000 and 2,000 samples respectively. They were built in the same way we have described in Sections 4.2 and 4.3 . After running the feature selection we found the solution with 30 and 38 features for the over-segmentation and under-segmentation verifiers respectively. The original feature sets consist of 42 components each one.

After training these optimized verifiers, we applied them together with the optimized classifier to recognize the strings of digits. The recognition rates were kept at the same level as presented in Table XVII. Considering the three optimized classifiers in this experiment, the total features used by the system were reduced from 216 (132+42+42) to 168 (100+30+38), i.e., about 22% less features.

In spite of the fact that the optimized classifier succeed in keeping the rates at the same levels as the original system, such a classifier produced better recognition rates for strings composed of naturally isolated digits and worse performance for strings that show problems of touching digits, fragmentation and noise. Therefore, the improvement found in the former case compensates the problems found in the latter.

6.2.4.3 Experiments on Strings of Digits (2)

In order to improve the performance of the classifier globally, i.e. for naturally isolated digits as well as digits with problems of fragmentation, touching and so forth, we have performed a new series of experiments. The goal was to select a subset of features suitable for the problem of strings of digits rather than the problem of naturally isolated digits. Thus, instead of using a database of isolated digits to assess the fitness during the feature selection procedure, we used the 3-digit string database which is composed of 2,385 images. In this manner, problems such as overlapping, fragmentation and effects of segmentation are tackled during the optimization process and hence relevant features for those problems will not be discarded. The validation set used to find the best solution in the Pareto-optimal front is the 10-digit string database which contains 1,217 images.

After performing the feature selection for the general-purpose recognizer taking into account such databases, the best solution we found had 124 selected features, i.e. it is considerably greater than that presented in the previous section (100 features). After training such a classifier and using it to classify strings of digits we perform an error analysis in order to find out whether the optimized classifier got better results for strings with problems of touching digits, fragmentation and noise. We verified that it succeeded in improving the rates for strings with such problems, however, the rates for strings composed of naturally isolated digits was slightly lower than those shown in the previous section. Since most of the strings of the NIST database are composed of naturally isolated digits (about 75%), the slightly worse rates obtained in this case neutralized the improvement reached for strings with problems. Hence, the recognition rates of this experiment are almost the same as those of the previous experiment. Table XVIII reports the performance (0.5% error level) of the classifier optimized in the context of strings of digits. The results for 3- and 10-digit strings were omitted here since we have used such subsets during the optimization process.

Table XVIII

Performance of the system on strings of digits using the general-purpose classifier optimized with strings of digits.

String Length	Original Classifiers Rec. Rate (%)	Original Classifiers Error at 0.5%	Optimized Classifiers Rec. Rate (%)	Optimized Classifiers Error at 0.5%
2	96.88	93.88	96.91	93.90
4	93.38	84.36	93.33	84.36
5	92.40	82.44	92.40	82.44
6	93.12	84.03	92.35	83.95

6.2.5 Discussion

In spite of the fact that the Pareto-based approach offers several advantages over the classical one, we have seen through the experiments that both strategies found similar solutions (see Figures 44b and 45b). In our first experiment, we observed that the classical approach converged the search to the space where the most probable solutions are located due to the weights we have chosen. However, for problems where the solutions are located along the Pareto-front, the classical approach does not work properly. Moreover, to achieve part of the Pareto-front, the weighted-sum method was run several times with different weight vectors.

For the problem of feature selection for handwriting recognition we can observe that the main advantage of the Pareto-based approach is the ability of dealing with different databases without having to deal with problems such as scaling and finding the suitable values for the weight vector. Moreover, Pareto-based approaches have the ability of finding the Pareto-optimal front in the first run of the algorithm.

As we have seen, two different optimized classifiers were used to recognize strings of digits. The former was optimized considering a database composed of naturally isolated digits only while the latter took into account a database of strings of digits.

Through detailed experiments we have demonstrated that in both cases the optimized systems attained very similar reliability rates. In the first case, the classifier obtains slightly better rates for isolated digits and worse rates for strings with problems such as touching digits, fragmentation and noise while in the second case we observed the opposite. Since about 75% of the strings of digits of the NIST database consist of naturally isolated digits, the classifier optimized in this context, which has about 25% less features than the original classifier, provided the same reliability rates as those found in the original system.

Since there is an interdependence between features where two or more features between them convey important information, it is very difficult to analyze the unselected features independently. However, analyzing the unselected features of several runs of the algorithm we could notice the following aspects:

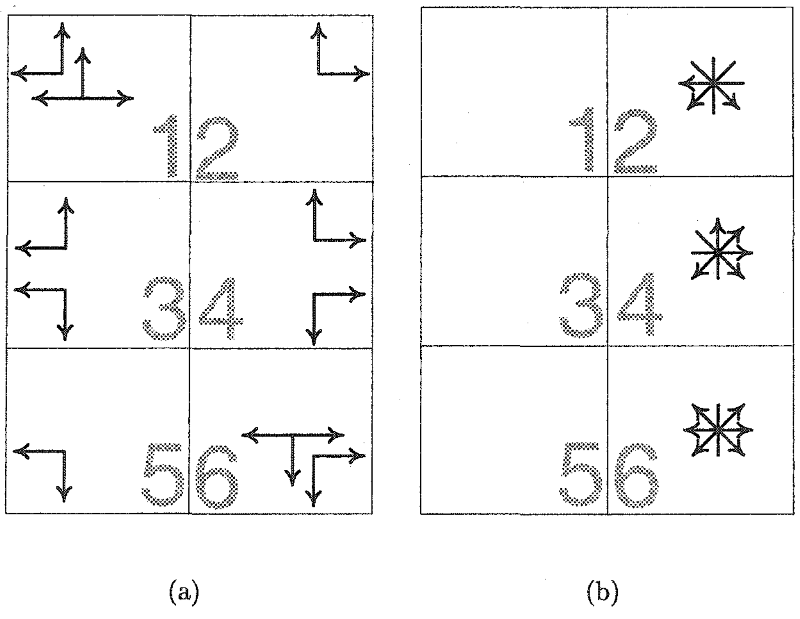


Figure 46 Unselected features for handwritten digit recognition: (a) Concavities and (b) Contour

1. In most of the solutions we observed that the unselected features of concavity have a certain symmetry among the zones, e.g. zones 1-6, 2-5 and 3-4 (Figure 46a). Moreover, we can conclude that for this kind of zoning there is a relationship between the geographic position of the zone and 4-Freeman directions. For example, in zone 2 (North-East) the unselected concavity configuration is the one that searches black pixels in directions North-East. We can notice the same behavior for the other zones.

2. All unselected features of contour are located just in the right side of the zoning (Figure 46b) in most solutions. In this case we can assume that such features are correlated so that it is not necessary to include them in the feature set.

3. The information related to the surface of the image was never unselected. This means that such an information is relevant to the feature vector we have used.

It is worth to remark that such an analysis is valid for the feature vector described in Section 4.1. However, it can be a very helpful tool to get a better insight for any kind of feature vector.

6.3 Ensemble of Classifiers

Our last efforts towards performance regard the ensemble of the classifiers. Such a strategy has been widely used to reduce model uncertainty and improve generalization performance. Developing techniques for generating candidate ensemble members is a very important direction of ensemble of classifiers research. Both theoretical [57, 81] and empirical [58, 130] research has demonstrated that a good ensemble is one where the individual classifiers in the ensemble are both accurate and make their errors on different parts of the input space (there is no gain in combining identical classifiers). In other words, an ideal ensemble consists of good classifiers (not necessarily excellent) that disagree as much as possible on difficult cases.

The literature has shown that varying the feature subsets used by each member of the ensemble should help to promote this necessary diversity. While traditional feature selection algorithms aim at finding the best trade-off between features and generalization, the task of ensemble feature selection has the additional goal of finding a set of feature sets that will promote disagreement among the component members of the ensemble. The random subspace method proposed by Ho in [64] was one early algorithm that constructs an ensemble by varying the subset of features. More recently some strategies based on GAs have been proposed [47, 130]. All these strategies claim better results than those produced by traditional methods for creating ensembles such as bagging and boosting. In spite of the good results brought by GA-based methods, they still can be improved in some aspects, e.g., avoiding classical methods such as the weighted sum to combine multiple objective functions. We have seen that when dealing with this kind of combination, one should deal with problems such as scaling and sensitivity towards the weights.

In this section we present a methodology for creating ensembles of classifiers which is able to cope with multiple ensembles simultaneously. Such a strategy is based on a hierarchical MOGA where the first level is devoted to generate a set of good classifiers while the second one combines these classifiers in order to find an ensemble. In the following sections we discuss the proposed methodology.

6.3.1 Related Works

Assuming the architecture of the ensemble as the main criterion, we can distinguish among serial, parallel, and hierarchical schemes, and if the classifiers of the ensemble are selected or not by the ensemble algorithm we can divide them into selection-oriented and combiner-oriented methods [70, 82]. Here we are more interested in the second class, which try to improve the overall accuracy of the ensemble by directly boosting the accuracy and the diversity of the experts of the ensemble. Basically, they can be divided into resampling methods and feature selection methods.

Resampling techniques can be used to generate different hypotheses. For instance, bootstrapping techniques [35] may be used to generate different training sets and a learning algorithm can be applied to the obtained subsets of data in order to produce multiple hypotheses. These techniques are effective especially with unstable learning algorithms, which are algorithms very sensitive to small changes in the training data. In bagging [11] the ensemble is formed by making bootstrap replicates of the training sets, and then multiple generated hypotheses are used to get an aggregated predictor. The aggregation can be performed by averaging the outputs in regression or by majority or weighted voting in classification problems.

While in bagging the samples are drawn with replacement using a uniform probability distribution, in boosting methods [40] the learning algorithm is called at each iteration using a different distribution or weighting over the training examples. This technique places the highest weight on the examples most often misclassified by the previous base learner: in this manner the classifiers of the ensemble focus their attention on the hardest examples. Then the boosting algorithm combines the base rules taking a weighted majority vote of the base rules.

The second class of methods regards those strategies based on feature selection. The concept behind these approaches consists in reducing the number of input features of the classifiers, a simple method to fight the effects of the classical curse of dimensionality problem. For instance, the random subspace method [64] relies on a pseudorandom procedure to select a small number of dimensions from a given feature space. In each pass, such a selection is made and a subspace is fixed where all points have a constant value (e.g. 0) in the unselected dimensions. All samples are projected to this subspace, and a classifier is constructed using the projected training samples. In the classification a sample of an unknown class is projected to the same subspace and classified using the corresponding classifier. In the same vein of the random subspace method lies the input decimation method [131], which reduces the correlation among the errors of the base classifiers, by decoupling the classifiers by training them with different subsets of the input features. It differs from the random subspace as for each class the correlation between each feature and the output of the class is explicitly computed, and the classifier is trained only on the most correlated subset of features.

Recently, several authors have been investigated GA to design ensemble of classifiers. Kuncheva and Jain [83] suggest two simple ways to use genetic algorithm to design an ensemble of classifiers. They present two versions of their algorithm. The former uses just disjoint feature subsets while the latter considers (possibly) overlapping feature subsets. The fitness function employed is the accuracy of the ensemble, however, no measure of diversity is considered. Gerra-Salcedo and Withley [47] used a simple GA to explore the space of all possible feature subsets, and then create an ensemble based on them. In their experiments, this GA-based approach outperformed classical methods such as Bagging and Boosting. In spite of the fact they achieved interesting results, they did not consider any measure of diversity. A more elaborate method, also based on GA, was proposed by Optiz [130]. In his work, he stresses the importance of a diversity measure by including it in the fitness calculation. The drawback of this method is that the objective functions are combined through the weighted sum. It is well known that when dealing with this kind of combination, one should deal with problems such as scaling and sensitivity towards the weights.

6.3.1.1 Measures of Diversity

Since a good ensemble is composed of diverse classifiers that bring a good generalization, a measure should be considered to quantify such a diversity. Kuncheva and Whitaker [85] study ten different measures of diversity in classifier ensemble and their relationship with the ensemble accuracy. They conclude that the measures of diversity should not be a replacement of the estimate of the accuracy of the ensemble but should stem from the intuitive concept of diversity. They also demonstrate that there is no agreement on a "best" measure of diversity in the literature.

Perhaps, the simplest metric for determining the diversity of the classification decision of a set of classifiers is "classification overlap" [15]. To compute the overlap among a set of classifiers requires counting the number of instances that were classified the same way by each of the classifiers, including instances that were classified incorrectly by all classifiers. To measure the diversity of a set of classifiers we can look at their classification overlap: a set of classifiers S_1 is more diverse than another set S_2 if overlap(S_1) < overlap(S_2).

Cunninghan and Carney [28] argue that the entropy is a good measure of diversity. For a test set of M samples where there are K classes a measures of entropy is:

$$E = \frac{1}{N}\frac{1}{M}\sum_{x=1}^{M}\sum_{k=1}^{K} -P_k^x log(P_k^x) \tag{6.9}$$

where P_k^x is the frequency of the k^{th} class for sample x and N is the number of classifiers . When the predictions of the classifiers are distributed evenly across the possible classes, the entropy is higher and the set of classifiers more diverse.

Krogh and Vedelsby [81] measure the diversity among the classifiers through the ambiguity, which is defined as follows:

$$a_i(x_k) = [V_i(x_k) - \overline{V}(x_k)]^2 \tag{6.10}$$

where a_i is the ambiguity of the i^{th} classifier on the example x_k, randomly drawn from an unknown distribution, while V_i and $\overline{V}$ are the i^{th} classifier and the ensemble predictions respectively. In other words, it is simply the variance of ensemble around the mean, and it measures the disagreement among the networks on input x. Thus the contribution to diversity of an ensemble member i as measured on a set of M samples is:

$$A_i = \frac{1}{M} \sum_{k=1}^{M} a_i(x_k) \tag{6.11}$$

and the ambiguity of the ensemble is

$$\overline{A} = \frac{1}{N} \sum A_i \tag{6.12}$$

where N is the number of classifiers. So, if the classifiers implement the same functions, the ambiguity $\overline{A}$ will be low, otherwise it will be high. In this scenario the error from the ensemble is

$$E = \overline{E} - \overline{A} \tag{6.13}$$

where $\overline{E}$ is the average errors of the single classifiers and $\overline{A}$ is the ambiguity of the ensemble. Equation 6.13 expresses the trade-off between bias and variance in the ensemble, but in a different way than the common bias-variance relation in which the averages are over possible training sets instead of ensemble averages. If the ensemble is strongly biased the ambiguity will be small, because the networks implement very similar functions and thus agree in inputs even outside the training set.

6.3.2 Proposed Methodology

In this section we describe the hierarchical approach proposed. As stated before, it is based on a two-level MOGA where the first level generates a set of good classifiers by conducting feature selection and the second one searches the best ensemble among such classifiers. In both cases, MOGAs are based on bit representation, one-point crossover, and bit-flip mutation. As depicted in Figure 47, the input of the second level is a byproduct of the first one, which was fully described in the previous section. Thus, next section focuses the second level of search.

6.3.2.1 2^{nd} Level: Finding the Best Ensemble

Let $W = e_1, e_2, \ldots, e_n$ be a set of n classifiers extracted from the Pareto-optimal and Q a chromosome of size n of the population. The relationship between W and Q is straightforward, i.e., the gene i of the chromosome Q is represented by the classifier e_i from W. Thus, if a chromosome has all bits selected, all classifiers of W will be included in the ensemble.

In order to find the best ensemble of classifiers, i.e., the most diverse set of classifiers that brings a good generalization, we must use two objective functions during this level of the search, namely, maximization of the recognition rate of the ensemble and maximization of the ambiguity as proposed in [81]. We have also tried the measures of diversity described in Section 6.3.1.1, but the choice of ambiguity yielded better results in our experiments.

At this level of the strategy we want to maximize the generalization of the ensemble, therefore, it will be necessary to use a way of combining the outputs of all classifiers to get a final decision. To do this, we have used the average, which is a simple and effective scheme of combining predictions of the neural networks [79]. Other combination rules such as product, min, and max have been tested but the simple

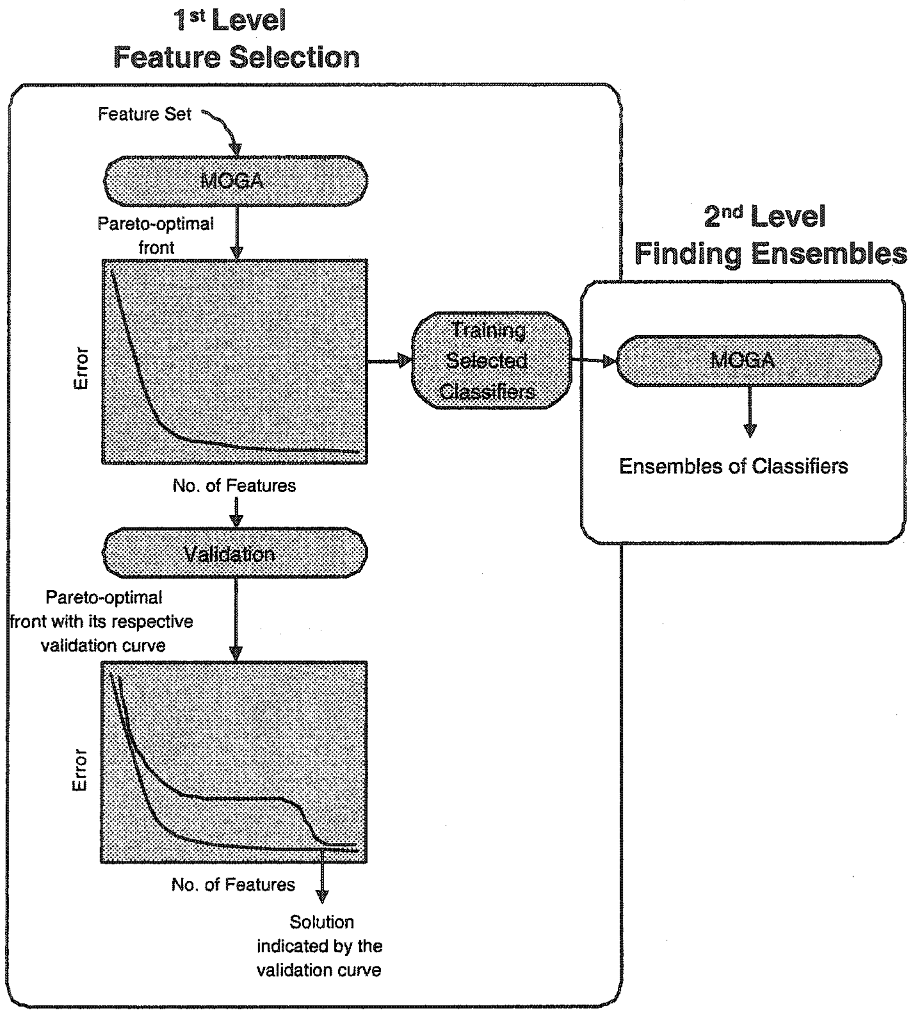


Figure 47 The flowchart of the proposed methodology.

average has produced better results. In order to evaluate the objective functions described above we have used the second validation set (see Appendix 1).

Different from other methodologies for ensemble creation based on feature selection where only one ensemble is considered, our approach considers w ensembles simultaneously, where w is the population size used by MOGA in the second level. This is due to the fact that each chromosome of the population represents a potential ensemble. Moreover, we will see in the experiments that this strategy produces more compact ensembles than other methods.

6.3.3 Experiments on Isolated Digits

The experiments we are going to describe here take into account the same parameters we have defined for the first level, i.e., feature selection. The difference is that the length of the chromosome in the first level is the number of components in the feature set, while here is the number of classifiers picked from the Pareto-optimal front in the previous level.

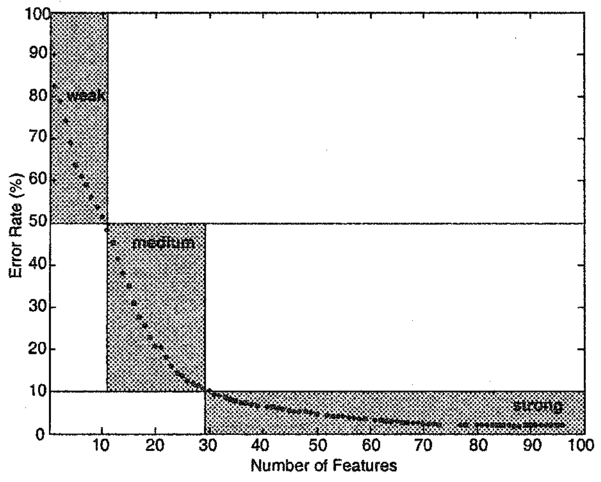


Figure 48 Different kinds of classifiers found in the Pareto-optimal front.

As depicted in Figure 48, the feature selection procedure produces quite a large number of classifiers, which should be trained for use in the second level. After some experiments, we found out that the second level always chooses "strong" classifiers to compose the ensemble. Thus, in order to speed up the training process and the second level of search as well, we decide to train and use in the second level just the "strong" classifiers. This decision was made after we realized that in our experiments the "weak" and "medium" classifiers did not cooperate with the ensemble at all.

Table XIX summarizes the "strong" classifiers produced by the first level for the three feature sets we have considered.

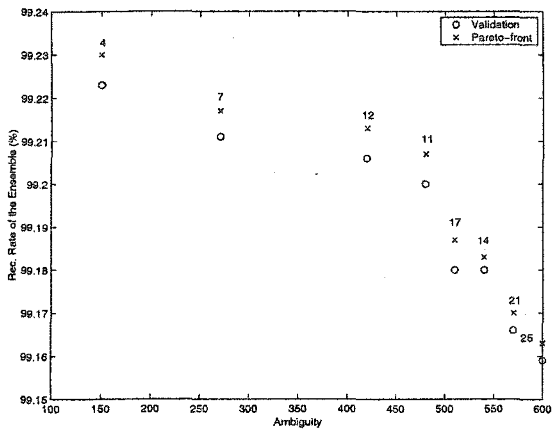
Table XIX

Summary of the classifiers produced by the first level.

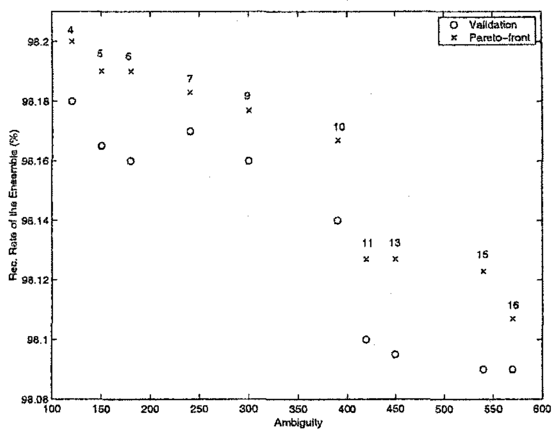
Feature Set	No. of Classifiers	Range of Features	Range of Rec. Rates (%)
Concavities	81	24-125	90.5 - 99.1
Distances	54	30-84	90.6 - 98.1
Edge Maps	78	35-113	90.5 - 97.0

Considering for example the feature set of concavities, the first level of the algorithm provided 81 "strong" classifiers which have the number of features ranging from 24 to 125 and recognition rates ranging from 90.5% to 99.1% on the test set. This shows the great diversity of the classifiers produced by the feature selection method. In order to assess the objective functions of the second-level MOGA (generalization of the ensemble and diversity) we have used the second validation set (validation-2 – Appendix 1).

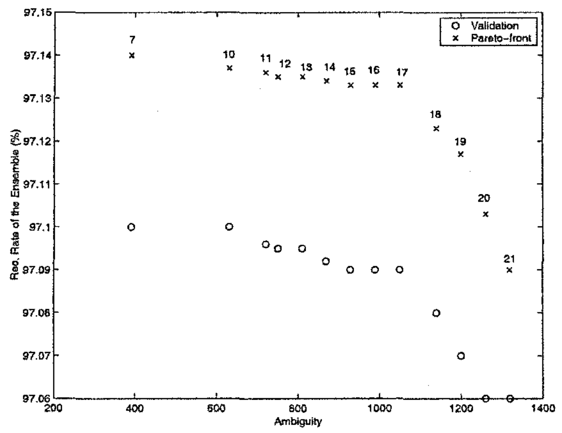
Like the first level, the second one also generates a set of possible solutions which are the trade-offs between the generalization of the ensemble and its diversity. Thus the problem now lies in choosing the most desirable ensemble among all. Figure 49 depicts the variety of ensembles yielded by the second-level MOGA for the three feature sets. The number over each point stands for the number of classifiers in the ensemble. Such information can be used to support the decision about which ensemble should be selected. Since we are aiming at performance, the direct choice will be the ensemble that provides better generalization. In our experiments, the ensemble that presents better performance has also the smallest number of classifiers.



(a)



(b)



(c)

Figure 49 The Pareto-optimal front produced by the second-level MOGA: (a) Concavities, (b) Distances, and (c) Edge Maps.

Table XX summarizes the best ensembles produced for the three different feature sets and their performance at zero-rejection level on the test set. For simplicity, we reproduce the results achieved by the classifiers optimized through feature selection, which were reported in Table XVI. As we can see, the results for zero-rejection level are very similar.

Table XX

Performance of the ensembles on the test set.

Feature Set	Number of Classifiers	Rec. Rate (%) zero-rejection level	Rec. Rate (%) Feature Selection
Concavities	4	99.22	99.16
Distances	4	98.18	98.21
Edge Maps	7	97.10	97.16

On the other hand, the ensembles respond better for error rates fixed at very low levels. This can be observed in Figures 50a, b, and, c. The most expressive result was achieved for the classifier trained with the Edge Map feature set, which attains a reasonable performance at zero-rejection level but performs very poorly at low error rates. In such a case, the ensemble of classifiers brought an improvement of about 8%. We have noticed that the ensemble reduces the high outputs of some outliers so that the threshold used for rejection can be reduced and consequently the number of samples rejected is reduced. Thus, aiming for a small error rate we have to consider the important role of the ensemble. For the sake of clarity, we do not plot the results of the classifiers optimized through feature selection since they produce results very close to those yielded by the original classifiers.

In order to complete these experiments we have combined the three ensembles. Again, several combination methods were tried, and here the average also yielded better results. However, the combination of the three ensembles was not good enough to surpass our best ensemble of classifiers (Figure 50d). After analyzing the errors produced by the three ensembles, we realized that they do not have enough complementarity in order to enhance the results. It is important to mention, though, that such ensembles were not optimized to compose a major ensemble. Figure 51 presents some examples of misclassification generated by the ensembles, where those signed with a square would represent mislabelling. We can observe that most of this errors

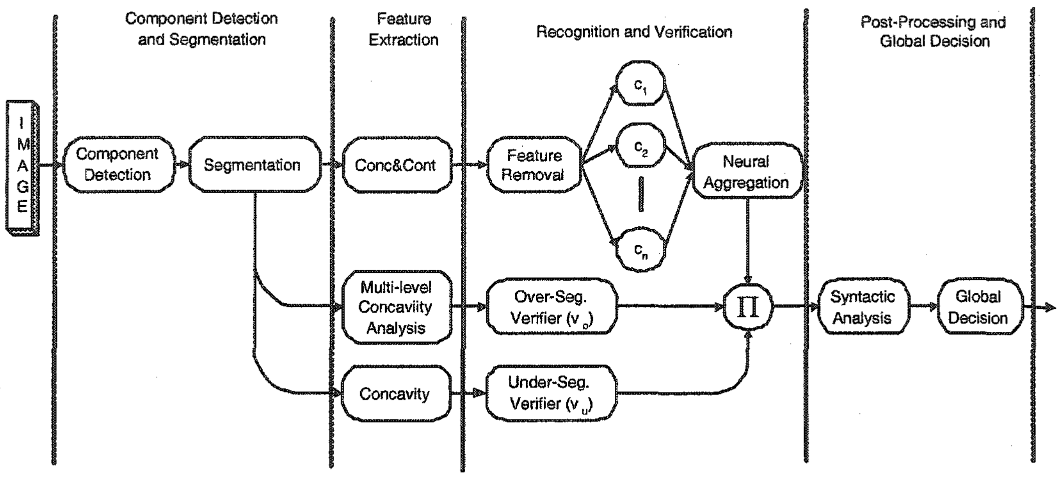


Figure 52 Ensemble of classifiers replacing the general-purpose recognizer.

Simple operations such as the product and average do not fit here since they can only estimate a posteriori probability if the members of the ensemble use mutually independent feature sets [2].

To surpass this problem, we have designed an MLP to combine the results of the ensemble. Thus, it has 40 inputs ($10 \times 4 = 40$), 30 hidden units, and 10 outputs and it was trained as described in Section 2.1.2. We have considered the same database we have used to train the general-purpose recognizer (e_{10}) described in Section 4.1. This classifier produced recognition rates at the same levels as those yielded by the ensemble discussed in the previous section, which uses the average to generate a final result.

Table XXI compares the results of the original system and the system that takes the ensemble of classifiers. Since the performance of the ensemble of classifier for both zero-rejection level and error rate fixed at 0.5% are very close to the original classifier, the results observed on strings of digits are very similar as well. In this manner, the error-reject curve in this case has the same shape as the one presented in Figure 33 (Section 5.1.1).

Table XXI

Performance of the system on strings of digits using the ensemble of classifiers.

String	Original Classifiers		Ensemble of Classifier	
Length	Rec. Rate (%)	Error at 0.5%	Rec. Rate (%)	Error at 0.5%
2	96.88	93.88	96.70	93.78
3	95.38	89.84	95.00	89.80
4	93.38	84.36	93.40	84.10
5	92.40	82.44	92.40	82.50
6	93.12	84.03	93.12	83.95
10	90.24	75.20	90.02	74.90

6.3.5 Discussion

In this section we have discussed a methodology for ensemble creation based on feature selection from a two-level MOGA. The experiments on three different feature sets have demonstrated the validity and efficiency of the proposed strategy by finding small ensembles, which succeed in improving the recognition rates for classifiers working with a very low error rates. On the other hand, the experiments also show that much more research must be done in order to get reliable classifiers, especially when dealing with strings of digits. Since the ensembles usually fail when it faces effects produced by fragmentation and the segmentation algorithm, an interesting direction to investigate could be either to reduce such effects or to learn more about them.

In order to better evaluate our methodology we have implemented the method proposed by Optiz in [130]. We have chosen Optiz's method because it seems to be more robust than the others we have found in the literature. Basically, our methodology brought slightly better results but with considerably smaller ensembles. Regarding Optiz's methodology, the best results were achieved with ensembles composed of 20 classifiers for all three feature sets, i.e., a population of 20 individuals, while in ours the ensembles were composed of 4, 4, and 7 classifiers as described in Table XX. It is worth of remark that Optiz achieves his best results using a very high mutation rate (50%). In our experiments, though, this configuration did not work at all. After trying some different rates, we realized that the same values we have used in our methodology yield better results for Optiz's strategy.

Finally, it is important to emphasize that the feature selection method we have applied is designed to tackle huge databases so that the pattern recognition problem can be better represented. On the other hand, our method is more time consuming, since a two-level optimization is involved.

6.3.6 Summary

In this chapter we have debated different strategies we have examined in order to improve the performance of the handwriting recognition system. First of all, the high-level verification was discussed, then feature selection and finally the ensemble of classifiers. In all attempts we have made, results on both isolated and string of digits were presented. We have seen that very often the results of isolated digits can be improved while the results of strings of digits stay at the same levels. This corroborates the statements made in the beginning of this work about the difficulties of recognizing numerical strings. The next chapter concludes this thesis.

CONCLUSION

The main focus of this thesis was the recognition of unconstrained handwritten numerical strings, where our efforts were concentrated towards the problems of under- and over-segmentation and how to improve the performance of such a system. We have seen that this is not a trivial problem since besides the inherent variability of the numerals, strings of digits feature problems such as touching digits, broken digits, overlapping, and unknown length of the string.

We have proposed a modular off-line system that can cope with different applications. It takes a segmentation-based recognition approach where an explicit segmentation is employed. Combination of different levels such as segmentation, recognition and post-processing is made within a multi-hypothesis approach and a probabilistic model, which allows a sound integration of all knowledge sources used to infer a plausible interpretation. We have shown a very efficient scheme of verification to deal with over-segmentation and under-segmentation problems. Such a scheme takes into account two low-level verifiers. The first verifier uses a new feature set, which is based on multi-level concavity analysis and contextual information, in order to reduce the confusion between isolated and over-segmented characters. The second one works on the opposite problem, i.e., eliminating the confusion between isolated and under-segmented characters.

In order to improve the overall performance of the system that deals with numerical amounts on Brazilian bank cheques, we have developed a simple and efficient post-processor which is based on a deterministic automaton. This module provided an improvement of about 6% in the recognition rate. Finally, the rejection mechanism minimizes the number of rejection errors for a given number of rejects. Comprehensive experiments on numerical amounts and NIST SD19 databases have been conducted. High recognition rates at zero-rejection level and a very encouraging error-reject trade-off have been obtained. The results reached by our system compare favorably to other published methods.

Thereafter we have tried some different strategies to improve the performance of the system by focusing on the general-purpose recognizer. The main contributions at this level were two-fold. Firstly, we have introduced a methodology for feature selection which uses a Pareto-based approach to generate the Pareto-optimal front where sensitivity analysis and neural network enable the use of a representative database to evaluate fitness. Secondly, we have proposed a methodology for ensemble creation based on feature selection from a two-level MOGA. The goal of the first level is to carry out the feature selection to yield a set of good classifiers.

Future Works

During the development of this thesis, we did not have the opportunity to address some issues due to time constraints. Here we outline some future directions we believe being worthy of investigation:

- Reduce the number of segmentation cuts so that the number of hypotheses of segmentation may be reduced. We think that this can be achieved by introducing some kind of intelligence into the segmentation algorithm, since our algorithm takes into account some heuristics that can be replaced by some learning mechanism.
- We believe that the high-level verification can be improved in some aspects:
  - Use just the samples rejected by the general-purpose recognizer to train the verifiers. In this way, they would be more specialized on the difficult cases.
  - Investigate different classifiers and feature sets. An interesting classifier to be investigated in this case could be SVM, since it is primarily a two-

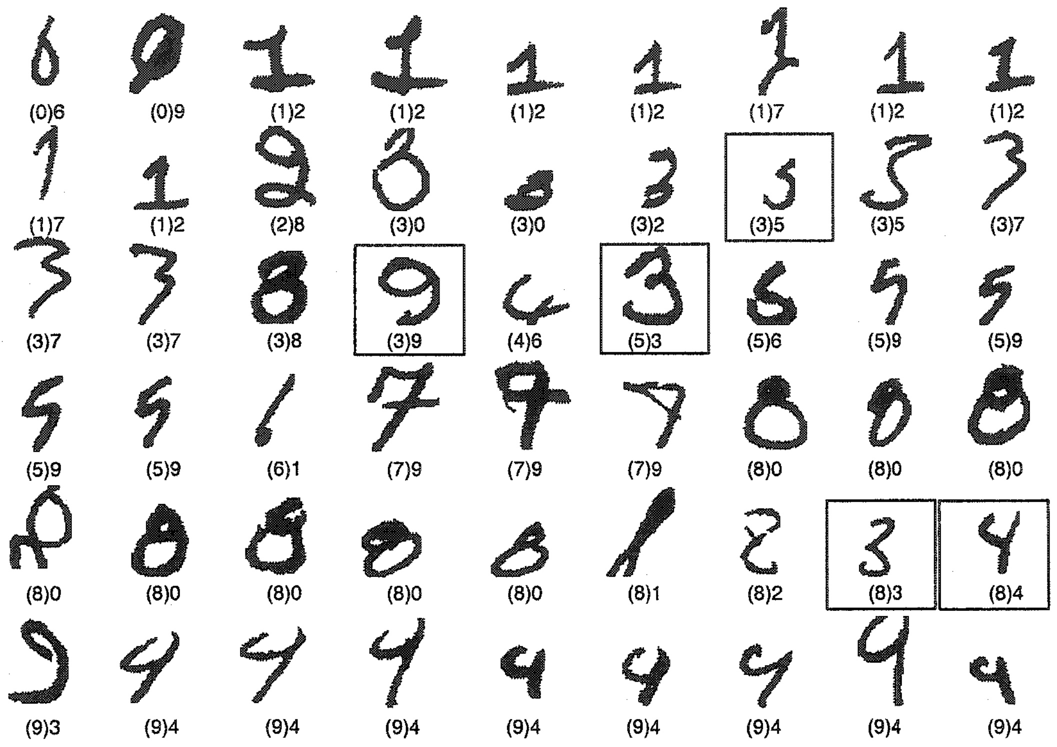


Figure 51 Examples of misclassification (the correct label is the one in parentheses).

6.3.4 Experiments on Strings of Digits

In this last experiment, we have replaced the general-purpose recognizer by the ensembles of concavities found in the previous section. Figure 52 depicts the system where the general-purpose recognizer is replaced by the ensemble of classifiers. Since all classifiers of the ensemble use parts of the same feature set, the complete feature set is extracted and then a module called "Feature Removal" feeds each classifier with their respective subset of features.

Since our system takes into account a probabilistic framework where the final probability is provided through the product of the general-purpose recognizer (the ensemble in this case) and two verifiers, the outputs of the ensemble must be combined in some way so that it could be interpreted as an estimation of a posteriori probability.

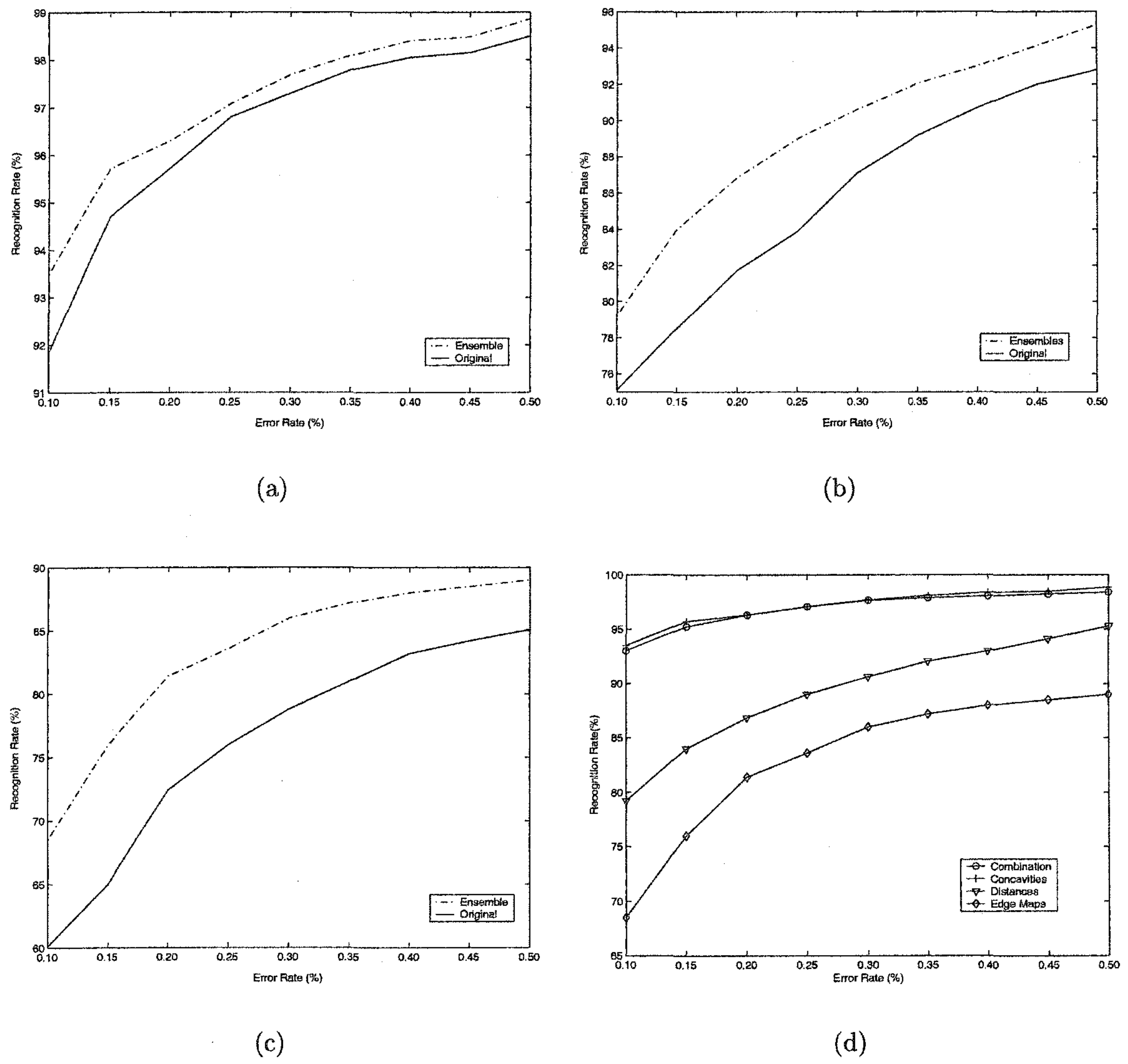


Figure 50 Improvements yielded by the ensembles for (a) Concavities, (b) Distances, (c) Edge Maps, and (d) Combination of the three ensembles.

are pretty difficult to recognize, with the exception of digits "1", which are not that difficult, but barely appear in the training set of NIST[1].

[1]This style of handwriting is very similar to that found in the Brazilian database (Figure 3b).

class classifier. However, in such a case, one must elaborate some way to combine the scores of the SVMs with the scores provided by the MLP, so that we can use the same rejection rule we have used so far.

  - Use some method of perturbation or transformation in order to generate difficult cases to train the high-level verifiers.
  - Make the verifiers produce high measurement levels for those samples correctly classified and low for those misclassified.

- Design a reliable mechanism to estimate the number of digits in the strings, so that this information could be used as another source of knowledge to improve the performance of the system.

- Optimize different feature sets at the same time by using the second level of search of the proposed methodology. In this way, we can get access to the complementarity of different feature sets.

- As suggested somewhere, try to reduce effects generated by the segmentation algorithm so that segmented digits would be more similar to isolated ones. Another direction would be to learn all those effects by generating some artificial data.

APPENDIX 1

Databases

1.1 Numerical Amount Database

This database was collected at the campus of the Pontifical Catholic University of Paraná (PUCPR) in Curitiba, Brazil. It is composed of 2,000 images of numerical amounts of Brazilian bank cheques where the writers were basically students on campus. Figure 53 shows some examples of such a database. The number of characters per image in this database (average length) is about 9.

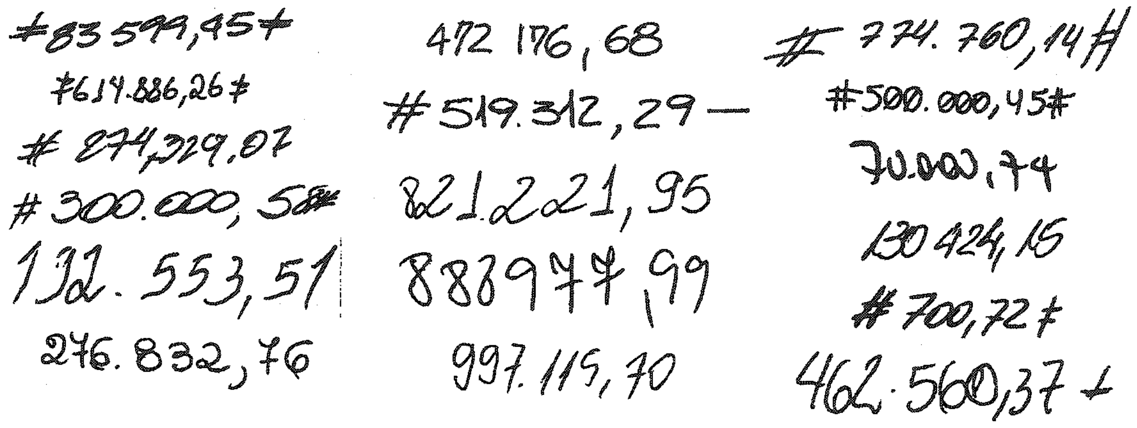


Figure 53 Examples of the numerical amount database.

The 2,000 images were divided into training, validation, and test sets as described in Table XXII.

Table XXII

Subset of the numerical amount database.

Sub-Database	Number of Images
Training Set	1,300
Validation Set	200
Test Set	500

Table XXIII

Isolated characters extracted from the numerical amounts.

Class	String Databases		
	Training	Validation	Test
0	1,190	210	309
1	908	160	289
2	1,043	184	336
3	942	166	316
4	897	158	316
5	925	163	317
6	905	159	302
7	929	163	310
8	808	142	313
9	858	151	303
Total (10 classes of digits)	9,405	1,656	3,111
#	437	77	242
.	924	163	434
,	624	109	242
Total (3 classes of symbols)	1,985	349	918
General total	11,390	2,005	4,029

From the foregoing subsets, 17,424 isolated characters divided into 13 classes were automatically extracted and manually revised. They were also divided into three subsets as shown in Table XXIII. We also extracted from this database 900 images of touching pairs.

Very often numerical amounts on cheques feature some particular behavior, e.g., certain amounts and certain configuration of touching digits such as "00" are more likely than others. However, it does not hold for this data, since it was designed so that the probability of occurrence of all amounts is the same. Therefore, the fact that very small amounts and very large amounts are less likely on cheques does not happen in this database.

1.2 NIST 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 labelled 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.

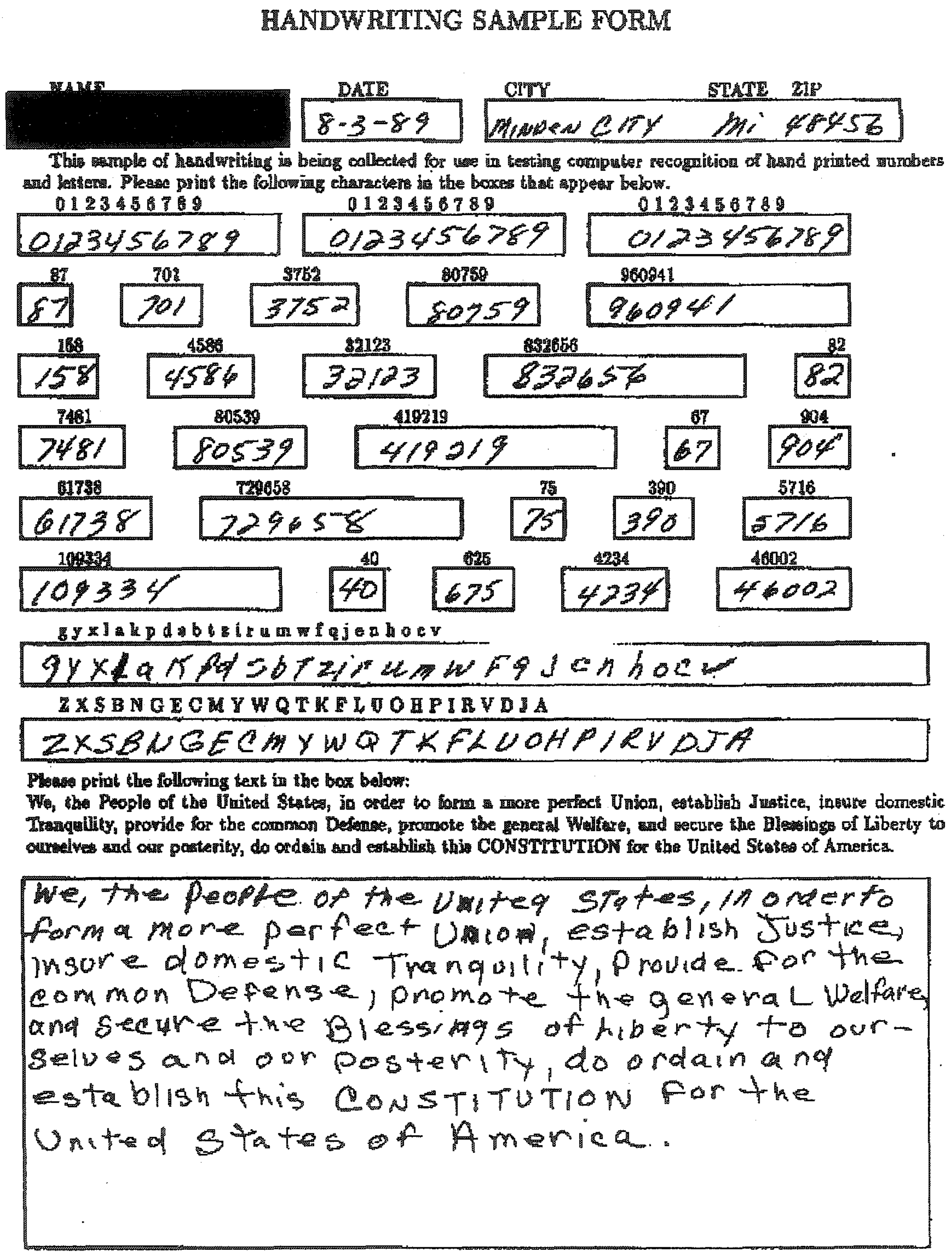
Table XXIV

HSF series distribution.

Sub-Database	Number of Images
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

An example of a full-page NIST form or HSF page is shown in Figure 54. 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 XXV.

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



HANDWRITING SAMPLE FORM

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

0123456789	0123456789	0123456789
0123456789	0123456789	0123456789

87	701	3752	80759	960941
87	701	3752	80759	960941

158	4586	32123	832656	82
158	4586	32123	832656	82

7481	80539	419219	67	904
7481	80539	419219	67	904

61738	729658	75	390	5716
61738	729658	75	390	5716

109334	40	625	4234	46002
109334	40	675	4234	46002

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

gyxlakpdsbtzirumwfqjenhocv

Z X S B N G E C M Y W Q T K F L U O H P I R V D J A

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 our-
selves and our posterity, do ordain and
establish this CONSTITUTION for the
United States of America..

Figure 54 Handwriting Sample Form (HSF full-page form)

Table XXV

Handwriting sample form (HSF) fields.

Field	Description
fld_0	Name
fld_1	Date
fld_2	City/State/ZIP
fld_3, ... fld_30	Numerical fields
fld_31	Lower case character box
fld_32	Upper case character box
fld_33	Free format text

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.

1.2.1 Isolated Digits

Table XXVI summarizes the distribution of the isolated digits we have used. They were extracted from hsf_0, hsf_1 hsf_2 hsf_3, and hsf_7. We have divided these data into training (195,000), validation (28.124), and test (60,089). For the experiments on feature selection, the test set is divided into validation-2 (30,000) and test (30,000). The training and validation sets use hsf_{0,1,2,3} while validation-2 and test sets use hsf_7.

1.2.2 Numerical Strings

Differently of the database of isolated digits, this one is provided in a raw format, i.e., the strings must be extracted from the pages (Figure 54). As part of his doctoral research, Britto Jr. [13] developed a system to perform this task. The test set he had proposed contains 12,802 distributed into six classes: 2-digit (2370), 3-digit (2385), 4-digit (2345), 5-digit (2316), 6-digit (2169), and 10-digit (1217) string, respectively. These data were extracted from hsf_7 series. Another contribution of Britto Jr's

Table XXVI

Number of samples by digit class - NIST SD19

Class	hsf_0123	hsf_7
0	22,971	5,893
1	24,772	6,567
2	22,131	5,967
3	23,172	6,036
4	21,549	5,873
5	19,545	5,684
6	22,128	5,900
7	23,208	6,254
8	22,029	5,889
9	21,619	6,026
Total	223,124	60,089

work lies in building a database for touching digits (pairs only). Such a database consists of 8,647 images.

APPENDIX 2

Feature Sets

2.1 Edge Maps

The idea behind this feature set is that handwritten numerals are essentially line drawings on a two-dimensional space. Thus, the input character is represented by its line segments, which may be horizontal, vertical, or diagonal. To facilitate the extraction of these segments (also called edges), the input image is first thinned. Then, a simple line detector is used to obtain the respective four-directional edge maps. The corresponding line detector masks used in our work are shown in Figure 55.

$$\underset{(a)}{\begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline 2 & 2 & 2 \\ \hline -1 & -1 & -1 \\ \hline \end{array}} \quad \underset{(b)}{\begin{array}{|c|c|c|}\hline -1 & 2 & -1 \\ \hline -1 & 2 & -1 \\ \hline -1 & 2 & -1 \\ \hline \end{array}} \quad \underset{(c)}{\begin{array}{|c|c|c|}\hline -1 & -1 & 2 \\ \hline -1 & 2 & -1 \\ \hline 2 & -1 & -1 \\ \hline \end{array}} \quad \underset{(d)}{\begin{array}{|c|c|c|}\hline 2 & -1 & -1 \\ \hline -1 & 2 & -1 \\ \hline -1 & -1 & 2 \\ \hline \end{array}}$$

Figure 55 Line detector masks used for extracting four-directional edge maps: (a) horizontal lines, (b) vertical lines, (c) 45-degree diagonal lines, and (d) -45-degree diagonal lines.

After extracting the respective line segments, the edge maps are compressed into an image of 5 × 5 pixels. This compression not only reduces the size of the feature vector used, thereby saving on computational overhead, but also retains the more dominant edge points in the edge map while weeding out isolated spurious edge points. In addition to these four-directional edge maps, the original image is also compressed and included as an additional feature in order to capture the global characteristics of the input image. At the end, we have a feature vector composed of 125 components normalized between 0 and 1. Figure 56 shows the overall edge map feature extraction procedure.

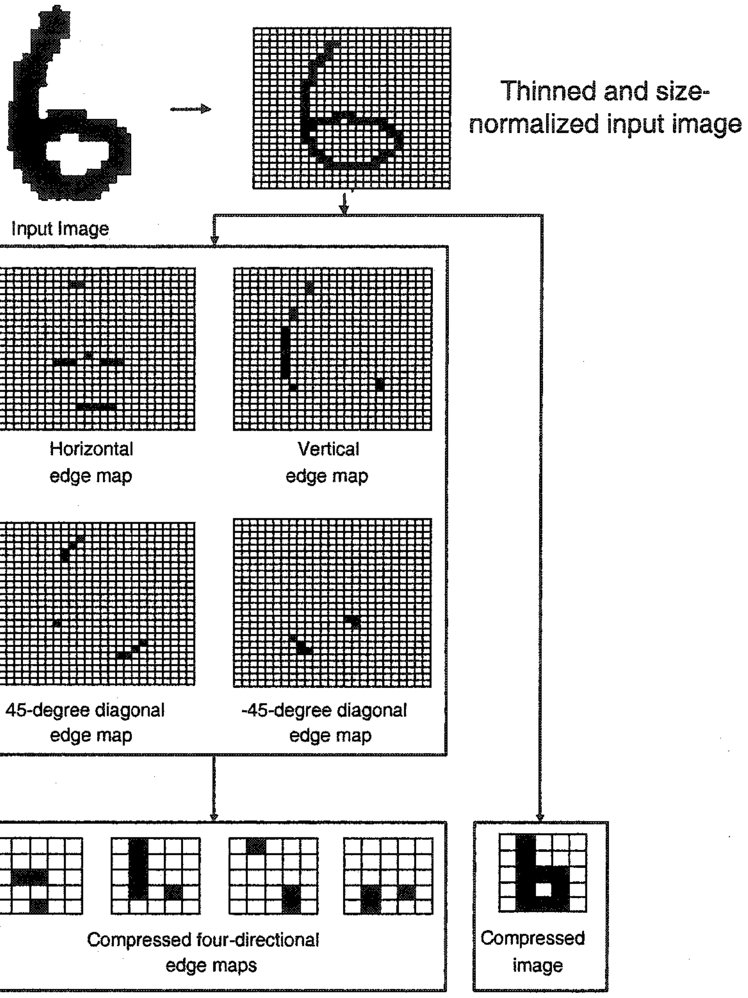

Input Image
Thinned and size-normalized input image
Horizontal edge map
Vertical edge map
45-degree diagonal edge map
-45-degree diagonal edge map
Compressed four-directional edge maps
Compressed image

Figure 56 Overall edge maps feature extraction. The compressed input image and compressed four-directional edge maps are used as features.

2.2 Directional Distance Distribution

To each of the pixels in the binary input pattern map (Figure 57), two sets of 8 bytes which we call the W (White) set and B (Black) set are allocated as shown in Figure 58. For a white pixel, the set W is used to encode the distances to the nearest black pixels in 8 directions. The set B is simply filled with value zero without computing the distances. Likewise, for a black pixel, the set B is used to encode the distances to the nearest white pixels in 8 directions. The set W is filled with value zero. (The 8-direction codes are 0(E), 1(NE), 2(N), 3(NW), 4(W), 5(SW), 6(S), 7(SE).)

```
 0123456789012345
0---********-----
1--**----*****---
2------**----***-
3----****-----**-
4---***--------**
5---**---------**
6--**----------**
7-***----------**
8-**----------**-
9***----------**-
0**----------**--
1**---------**---
2**-------***----
3***-----***-----
4-*********------
5---****---------
```

Figure 57   A sample pattern.

For the sample pattern in Figure 57, the pixel at coordinates (8,2) will have the WB encoding at the top of Figure 58. Because this pixel is white, the B set is filled with value zero. For the set W, to compute the distance value in all 8 directions, the pixel shoots a ray in each direction, which proceeds until it hits a black pixel. In case of hitting, the distance travelled is recorded into the byte corresponding to the ray direction. As an example, for the direction 0, the ray will travel the sequence, $(8,2)W \rightarrow (9,2)W \rightarrow (10,2)W \rightarrow (11,2)W \rightarrow (12,2)B$. So the travel distance 4 is recorded in the first byte of the W set.

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	
4	1	1	2	1	1	11	6	0	0	0	0	0	0	0	0	(8,2)
0	0	0	0	0	0	0	0	5	2	2	2	1	9	1	1	(8,1)

Figure 58   An example of WB encoding.

Figure 58 shows another example of the WB encoding for the black pixel located at (8,1). In this example, a case occurs when the ray arrives at the map boundary without hitting any white pixel. For example, the ray for the direction 3 travels the sequence, $(8,1)B \rightarrow (7,0)B \rightarrow$ boundary, and it meets the boundary before hitting a white pixel. In this case, the ray should stop at the boundary. So the following travel sequence will be followed; $(8,1)B \rightarrow (7,0)B \rightarrow$ boundary, and the travel distance is determined to be 2. This information is recorded in the fourth byte of B set which corresponds to the direction 3.

After computing WB codings for all the pixels in the map, we convert the map into a 3×2 block mesh as shown in Figure 20 and extract 16 averaged WB codings from each zone. This amounts to a 96-dimensional feature vector normalized between 0 and 1 by summing up their values and then dividing each one by this summation.

APPENDIX 3

Confusion Matrix NIST-SD19

Table XXVII

Confusion matrix (%) of general-purpose recognizer on the training set of 195,000. The first column shows the real identity of the sample while the top row is the result identity.

	0	1	2	3	4	5	6	7	8	9	Total
0	99.82	0.00	0.03	0.02	0.01	0.01	0.04	0.00	0.06	0.03	99.82
1	0.00	99.85	0.03	0.01	0.01	0.02	0.03	0.04	0.01	0.01	99.85
2	0.05	0.01	99.64	0.09	0.05	0.00	0.01	0.08	0.07	0.02	99.64
3	0.02	0.01	0.17	99.61	0.01	0.08	0.01	0.04	0.02	0.04	99.61
4	0.02	0.01	0.03	0.01	99.73	0.00	0.03	0.05	0.02	0.12	99.73
5	0.02	0.01	0.01	0.09	0.02	99.67	0.09	0.00	0.07	0.03	99.67
6	0.05	0.02	0.02	0.01	0.02	0.07	99.79	0.01	0.03	0.00	99.79
7	0.00	0.04	0.08	0.04	0.07	0.01	0.00	99.65	0.00	0.12	99.65
8	0.12	0.07	0.05	0.06	0.08	0.11	0.02	0.03	99.37	0.08	99.37
9	0.07	0.01	0.01	0.04	0.11	0.02	0.02	0.16	0.06	99.50	99.50
											99.66

Table XXVIII

Confusion matrix (%) of general-purpose recognizer on the test set of 60,089.

	0	1	2	3	4	5	6	7	8	9	Total
0	99.00	0.00	0.14	0.03	0.24	0.05	0.24	0.03	0.17	0.01	99.00
1	0.00	99.16	0.55	0.07	0.02	0.00	0.00	0.20	0.00	0.00	99.16
2	0.02	0.00	99.30	0.18	0.03	0.00	0.02	0.29	0.13	0.03	99.30
3	0.09	0.05	0.25	98.70	0.02	0.29	0.00	0.42	0.13	0.05	98.70
4	0.00	0.02	0.03	0.00	99.41	0.00	0.15	0.09	0.02	0.28	99.41
5	0.02	0.06	0.00	0.16	0.03	99.31	0.00	0.02	0.03	0.37	99.31
6	0.12	0.06	0.02	0.00	0.09	0.16	99.54	0.00	0.01	0.00	99.54
7	0.00	0.03	0.22	0.03	0.17	0.00	0.00	99.36	0.00	0.19	99.36
8	0.61	0.17	0.01	0.18	0.17	0.09	0.01	0.05	99.28	0.25	98.28
9	0.05	0.03	0.00	0.05	0.32	0.05	0.00	0.24	0.07	99.19	99.19
											99.13

BIBLIOGRAPHY

[1] I. S. I. Abuhaiba and P. Ahmed. A fuzzy graph theoretic approach to recognize the totally unconstrained handwritten numerals. *Pattern Recognition*, 26(9):1335–1350, 1993.

[2] F. M. Alkoot and J. Kittler. Experimental evaluation of expert fusion strategies. *Pattern Recognition Letters*, 20(11-13):1361–1369, 1999.

[3] A. Waibel amd T. Hanazawa, G. Hilton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Trans. on Acoustics, Speach, and Signal Processing*, 37:328–339, 1989.

[4] N. Arica and F. T. Y. Vural. An overview of character recognition focused on off-line handwriting. *IEEE Trans. on Systems, Man, and Cybernetics - Part C:Applications and Reviews*, 31(2):216–233, 2001.

[5] N. E. Ayat, M. Cheriet, and C. Y. Suen. Optimization of the svm kernels using an empirical error minimization scheme. In *Proc. of the International Workshop on Pattern Recognition with Support Vector Machine*, pages 354–369, Niagara Falls, Canada, 2002.

[6] R. R. Bailey and M. Srinath. Orthogonal moment features for use with parametric and non-parametric classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(4):389–399, 1996.

[7] Banco Central do Brasil. *Manual de Normas e Instruções - Circular nº 001825*, October 1990. http://www.bcb.gov.br/mPag.asp?codP=106&cod=112&perfil=1 (in portuguese).

[8] N. Benahmed. Optimisation de réseaux de neurones pour la reconnaissance de chiffres manuscrits isolés: Sélection et pondération des primitives par algorithmes génétiques. Master's thesis, Ecole de Technologie Superieure, Université du Quebec, Montreal - Canada, March 2002.

[9] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford Univ. Press, Oxford - U.K., 1995.

[10] H. Bourlard and N. Morgan. Merging multilayer perceptrons and hidden Markov models: Some experiments in continuous speech recognition. In E.Gelenbe, editor, *Neural Networks: Advances and Applications.* North Holland Press, 1991.

[11] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[12] J. S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *Proc. of Advances in Neural Information Processing Systems*, volume 2, pages 211–217. Morgan Kaufmann, 1990.

[13] A. Britto-Jr. *A Two-stage HMM-based method for recognizing handwritten numeral strings.* PhD thesis, Pontifícia Universidade Católica do Parana, Curitiba-Brazil, 2001.

[14] A. Britto-Jr., R. Sabourin, F. Bortolozzi, and C. Y. Suen. A string lenght predictor to control the level building of HMMs for handwritten numeral recognition. In *Proc. of 16th International Conference on Pattern Recognition (ICPR)*, volume 4, pages 31–34, Quebec City, Canada, 2002. IEEE Computer Society.

[15] C. E. Brodley. Adressing the selective superiority problem: Automatic algorithm/model class selection. In *Proc. of 10th International Conference on Machine Learning*, pages 17–24, 1993.

[16] T. M. Bruel. A system for the off-line recognition of handwritten text. In *Proc. of 12th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 129–133, Jerusalem, Israel, 1994.

[17] H. Byun and S. W. Lee. Applications of support vector machines for pattern recognition. In *Proc. of the International Workshop on Pattern Recognition with Support Vector Machine*, pages 213–236, Niagara Falls, Canada, 2002.

[18] J. Cai and Z. Q. Liu. Integration of structural and statistical information for unconstrained handwritten numeral recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(3):263–270, 1999.

[19] E. Cantu-Paz. *Efficient and Accurate Parallel Genetic Algorithms.* Kluwer Academic Publishers, 2000.

[20] R. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(7):690–706, 1996.

[21] M. Y. Chen, A. Kundu, and J. Zhou. Off-line handwritten word recognition using a hidden Markov model type stochastic network. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(5):481–496, 1994.

[22] Y. K. Chen and J. F. Wang. Segmentation of single- or multiple-touching

handwritten numeral string using background and foregound analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(11):1304–1317, 2000.

[23] D. Cheng and H. Yan. Recognition of handwritten digits based on contour information. *Pattern Recognition*, 31(3):235–255, 1998.

[24] K. W. Cheung, D. Y. Yeung, and R. T. Chin. A Bayesian framework for deformable pattern recognition with application to handwritten character recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(12):1382–1388, 1998.

[25] Y. C. Chim, A. A. Kassim, and Y. Ibrahim. Dual classifier system for hand-printed alphanumeric character recognition. *Pattern Analysis and Applications*, 1(3):155–162, 1998.

[26] S. J. Cho, J. Kim, and J. H. Kim. Verification of graphemes using neural networks in an HMM-based on-line Korean handwritting recognition system. In *Proc. of 7th International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, pages 219–228, Amsterdam, Netherlands, 2000.

[27] S. E. N. Correia and J. M. Carvalho. Optimizing the recognition rates of unconstrained handwritten numerals using biorthogonal spline wavelets. In *Proc. of 15th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 251–254, Barcelona, Spain, 2000. IEEE Computer Society.

[28] P. Cunningham and J. Carney. Diversity versus quality in classification ensembles based on feature selection. In *Proc. of the 11th European Conference on Machine Learning*, pages 109–116, 2000.

[29] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(3):131–156, 1997.

[30] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley and Sons Ltd, 2001.

[31] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function. In *Proc. of 3rd International Conference on Genetic Algorithms*, pages 42–50, 1989.

[32] G. Dimauro, S. Impedovo, G. Pirlo, and A. Salzo. Automatic bankcheck processing: A new engineered system. In S. Impedovo et al, editor, *International Journal of Pattern Recognition and Artificial Intelligence*, pages 467–503. World Scientific, 1997.

[33] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* John Wiley and Sons, 2^{nd} edition, 2001.

[34] G. Dzuba, A. Filatov, D. Gershuny, I. Kil, and V. Nikitin. Check amount recognition based on the cross validation of courtesy and legal amount fields. In S. Impedovo et al, editor, *International Journal of Pattern Recognition and Artificial Intelligence*, pages 639–655. World Scientific, 1997.

[35] B. Efron and Tibshirani R. *An introduction to the Bootstrap.* Chapman and Hall, 1993.

[36] C. Emmanouilidis, A. Hunter, and J. MacIntyre. A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator. In *Proc. of Congress on Evolutionary Computation*, volume 1, pages 309–316, 2000.

[37] G. D. Forney-Jr. The viterbi algorithm. *Procs. of IEEE*, 61(3):268–278, 1973.

[38] J. Freeman and D. Skapura. *Neural Networks - Algorithms, Applications and Programming Techniques.* Addison-Wesley, 1992.

[39] C. Freitas, M. Morita, L. S. Oliveira, E. Justino, A. Yacoubi, E. Lethelier, F. Bortolozzi, and R. Sabourin. Base de dados de cheques bancários Brasileiros. In *Proc. of XXVI Conferencia Latinoamericana de Informatica*, Atizapan de Zaragoza, Mexico, 2000. (in portuguese).

[40] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proc. of 13^{th} International Conference on Machine Learning*, pages 148–156, Bary, Italy, 1996.

[41] H. Fujisawa, Y. Nakano, and K. Kurino. Segmentation methods for character recognition: from segmentation to document structure analysis. *Proc. of IEEE*, 80:1079–1092, 1992.

[42] G. Fumera, F. Roli, and G. Giacinto. Reject option with multiple thresholds. *Pattern Recognition*, 33(12):2099–2101, 2000.

[43] P. D. Gader, B. Forester, M. Ganzberger, A. Billies, B. Mitchell, M. Whalen, and T.Youcum. Recognition of handwritten digits using template and model matching. *Pattern Recognition*, 5(24):421–431, 1991.

[44] P. D. Gader, J. M. Keller, and J. Cai. A fuzzy logic system for detection and recognition of street number fields on handwritten postal addresses. *IEEE Trans. on Fuzzy Systems*, 3(1):83–95, 1995.

[45] P. D. Gader and M. A. Khabou. Automatic feature generation for handwritten digit recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(12):1256–1261, 1996.

[46] P. D. Gader, M. A. Mohamed, and J. M. Keller. Fusion of handwritten word classifiers. *Pattern Recognition Letters*, 17(6):577–584, 1996.

[47] C. Gerra-Salcedo and D. Whitley. Genetic approach to feature selection for ensemble creatin. In *Proc. of Genetic and Evolutionary Computation Conference*, pages 236–243, Orlando, USA, 1999.

[48] D. Goldberg. *Genetic Algorithms in search, optimization and machine learning.* Reading, Mass., Addison-Wesley, 1989.

[49] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimisation. In *Proc. of 2^{nd} International Conference on Genetic Algorithms and Their Applications*, pages 41–49, 1987.

[50] N. Gorski. Practical combination of multiple classifiers. In *Progress in Handwritting Recognition*, pages 277–284. World Scientific, 1996.

[51] N. Gorski, V. Anisimov, E. Augustin, O. Baret, and S. Maximov. Industrial bank check processing: the A2iA CheckReader^{TM}. *International Journal on Document Analysis and Recognition*, 3:196–206, 2001.

[52] P. J. Grother. *NIST Special Database 19 - Handprinted forms and characters database.* National Institute of Standards and Technology (NIST), 1995.

[53] D. Guillevic and C. Y. Suen. Cursive script recognition applied to the processing of bank cheques. In *Proc. of 3^{th} International Conference on Document Analysis and Recognition (ICDAR)*, pages 11–14, Montreal, Canada, 1995.

[54] T. M. Ha and H. Bunke. Off-line, handwritten numeral recognition by perturbation method. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(5):535–539, 1997.

[55] T. M. Ha, M. Zimmermann, and H. Bunke. Off-line handwritten numeral string recognition by combining segmentation-based and segmentation-free methods. *Pattern Recognition*, 31(3):257–272, 1998.

[56] M. Hanmandlu, K. R. Mohan, S. Chakraborty, S. Goyal, and D. R. Choudhury. Unconstrained handwritten character recognition based on fuzzy logic. *Pattern Recognition*, 36(3):603–623, 2003.

[57] L. Hansen and O. Salomon. Neural network ensembles. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[58] S. Hashem. Optimal linear combinations of neural networks. *Neural Networks*, 10(4):599–614, 1997.

[59] T. Hastie and P. Y. Simard. Metrics and models for handwritten character recognition. *Statistical Science*, 13(1):54–65, 1998.

[60] L. Heutte, T. Paquet, J. V. Moreau, Y. Lecourtier, and C. Olivier. A structural/statistical feature based vector for handwritten character recognition. *Pattern Recognition Letters*, 19(7):629–641, 1998.

[61] L. Heutte, P. Pereira, O. Bougeois, J. Moreau, B. Plessis, and P. Courtellemont. Multi-bank check recognition system: Consideration on the numeral amount recognition module. In S.Impedovo et al, editor, *International Journal of Pattern Recognition and Artificial Intelligence*, pages 595–617. World Scientific, 1997.

[62] T. Hirano, Y. Okada, and F. Yoda. Structural character recognition using simulated annealing. In *Proc. of 4^{th} International Conference on Document Analysis and Recognition (ICDAR)*, pages 507–510, Ulm, Germany, 1997.

[63] S. Y. Ho and H. L. Huang. Facial modeling from an uncalibrated face image using a coarse-to-fine genetic algorithm. *Pattern Recognition*, 34(5):1015–1031, 2001.

[64] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[65] J. H. Holland. *Adaptation in natural artificial systems.* University of Michigan Press, Ann Arbor, 1975.

[66] G. F. Houle, D. B. Aragon, R. W. Smith, M. Shridhar, and F. Kimura. A multi-layered corroboration-based check reader. In *Proc. of IAPR Workshop on Document Analysis Systems*, pages 495–546, Malvern, USA, 1996.

[67] J. Hu and Y. Yan. Structural primitive extraction and coding for handwritten numeral recognition. *Pattern Recogniton*, 31:493–509, 1998.

[68] M. K. Hu. Visual pattern recognition by moment invariant. *IEEE Trans. on Information Theory*, 8:179–187, 1962.

[69] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptative

mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[70] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[71] A. K. Jain and D. Zongker. Representation and recognition of handwritten digits using deformable templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(12):1386–1391, 1997.

[72] B. K. Jang and R. T. Chin. One-pass parallel thinning: Analysis, properties, and quantitative evaluation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(11):1129–1140, 1992.

[73] G. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problems. In *Proc. of 11th International Conference on Machine Learning*, pages 121–129, 1994.

[74] G. Kaufmann and H. Bunke. Automated reading of cheque amounts. *Pattern Analysis and Applications*, 3(2):132–141, 2000.

[75] J. Keeler and D. E. Rumelhart. A self-organizing integrated segmentation and recogniton neural networks. In J. E. Moody, S. J. Hanson, and R. L. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 496–503. Morgan Kaufmann, 1992.

[76] G. Kim and S. Kim. Feature selection using genetic algorithms for handwritten character recognition. In *Proc. of 7th International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, pages 103–112, Amsterdam, Netherlands, 2000.

[77] F. Kimura and M. Shridhar. Segmentation-recognition algorithm for zip code field recognition. *Machine Vision and Applications*, 5:199–210, 1992.

[78] L. Kira and L. Rendell. A practical approach to feature selection. In *Proc. of 9th International Conference on Machine Learning*, pages 249–256. Morgan Kaufmann, 1992.

[79] J. Kittler, M. Hatef, R. Duin, and J. Matas. On combining classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[80] S. Knerr, V. Anisimov, O. Baret, N. Gorski, D. Price, and J. Simon. The A2iA intercheque system: Courtesy amount and legal amount recognition for french checks. In S.Impedovo et al, editor, *International Journal of Pattern*

Recognition and Artificial Intelligence, pages 505–547. World Scientific, 1997.

[81] A. Krogh and J. Vedelsby. Neural networks ensembles, cross validation, and active learning. In G.Tesauro et al, editor, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. MIT Press, 1995.

[82] L. Kuncheva, J. C. Bezdek, and R. P. W. Duin. Decision templates for multiple classifier fusion: An experimental comparison. *Pattern Recognition*, 34(2):299–314, 2001.

[83] L. Kuncheva and L. C. Jain. Designing classifier fusion systems by genetic algorithms. *IEEE Trans. on Evolutionary Computation*, 4(4):327–336, 2000.

[84] L. I. Kuncheva. A theoretical study on six classifier fusion strategies. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(2):281–286, 2002.

[85] L. I. Kuncheva and C. J. Whitaker. Ten measures of diversity in classifier ensembles:limits for two classifiers. In *Proc. of IEE Workshop on Intelligent Sensor Processing*, pages 1–10, 2001.

[86] L. Lam and C. Y. Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16(9):945–954, 1995.

[87] B. Lazzerini and F. Marcelloni. A linguistic fuzzy recognizer of off-line handwrriten characters. *Pattern Recognition Letters*, 21(4):319–327, 2000.

[88] E. Lecolinet and J. P. Crettez. A grapheme-based segmentation technique for cursive script recognition. In *Proc. of 1st International Conference on Document Analysis and Recognition (ICDAR)*, pages 740–749, St.Malo, France, 1991.

[89] Y. LeCun, B. Boser, J. S. Denker, B. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[90] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Procs of IEEE*, 86(11):2278–2324, 1998.

[91] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Muller. Efficient backprop. In G. Orr and K. Miller, editors, *Neural Networks: tricks of the trade.* Springer, 1998.

[92] D. S. Lee and S. N. Srihari. A theory of classifier combination:the neural network approach. In *Proc. of 3th International Conference on Document Analysis*

and Recognition (ICDAR), volume 1, pages 42–45, Montreal, Canada, 1995.

[93] S. W. Lee. Multilayer cluster neural network for totally unconstrained handwritten numeral recognition. *Neural Networks*, 8:783–792, 1995.

[94] S. W. Lee. Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural networks. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(6):648–652, 1996.

[95] S. W. Lee and S. Y. Kim. Integrated segmentation and recognition of handwritten numerals with cascade neural networks. *IEEE Trans. on Systems, Man, and Cybernetics, Part C:Applications and Reviews*, 29(2):285–290, 1999.

[96] E. Lethelier, M. Leroux, and M. Gilloux. An automatic reading system for handwritten numeral amounts on french checks. In *Proc. of 3rd International Conference on Document Analysis and Recognition (ICDAR)*, pages 92–97, Montreal, Canada, 1995. IEEE Computer Society.

[97] H. C. Leung. Speech recognition using stochastic explicit-segment moduling. In *Proc. of 2nd European Conference on Speech Communication and Technology*, pages 931–934, 1991.

[98] L. Ling, M. Lizaraga, N. Gomes, and A. Koerich. A prototype for brazilian bankcheck recognition. In S.Impedovo et al, editor, *International Journal of Pattern Recognition and Artificial Intelligence*, pages 549–569. World Scientific, 1997.

[99] R. P. Lippmann. Pattern classification using neural networks. *IEEE Communications Magazine*, 27(11):47–64, 1989.

[100] C. Liu and M. Nakagawa. Handwritten numeral recognition using neural networks:improving the accuracy by discriminative training. In *Proc. of 5th International Conference on Document Analysis and Recognition (ICDAR)*, pages 257–260, Bangalore, India, 1999.

[101] H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter approach. In *Proc. of 13th International Conference on Machine Learning*, pages 319–327. Morgan Kaufmann, 1996.

[102] W. Liu, M. Wang, and Y. Zhong. Selecting features with genetic algorithms in handwritten digits recognition. In *Proc. of International Conference on Evolutionary Computation*, pages 396–399, 1995.

[103] Z. Lu, Z. Chi, and W. C. Siu. Extraction and optimization of B-spline PBD

templates for recognition of connected handwritten digit strings. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(1):132–2002, 1996.

[104] G. Martin, M. Rashid, and J. Pittman. Integrated segmentation and recognition through exhaustive scans or learned saccadic jumps. In I. Guyon and P. S. P. Wang, editors, *Advances in pattern recognition systems using network technologies*, pages 187–203. World Scientific, 1993.

[105] O. Matan and J. C. Burges. Recognizing overlapping hand-printed characters by centered-objects integrated segmentation and recognition. In *Proc. of International Joint Conference on Neural Networks (IJCNN)*, pages 504–511, Seattle, USA, 1991.

[106] O. Matan, J. C. Burges, Y. LeCun, and J. S. Denker. Multi-digit recognition using a space displacement neural network. In J. E. Moody, S. J. Hanson, and R. L. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 488–495. Morgan Kaufmann, 1992.

[107] G. Mayraz and G. E. Hinton. Recognizing handwritten digits using hierarchical products of experts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(2):189–197, 2002.

[108] L. Micó and J. Oncina. Comparison of fast nearest neighbour classifier for handwritten character recogniton. *Pattern Recognition Letters*, 19(3-4):351–356, 1999.

[109] M. Miki, T. Hiroyasu, K. Kaneko, and K. Hatanaka. A parallel genetic algorithm with distributed environment scheme. In *Proc. of International Conference on System, Man, and Cybernetics*, volume 1, pages 695–700, 1999.

[110] K. Mitra, K. Deb, and S. K. Gupta. Multiobjective dynamic optimization of an industrial nylon 6 semibatch reactor using genetic algorithm. *Journal of Applied Polymer Science*, 69(1):69–87, 1998.

[111] J. Moody and J. Utans. Principled architecture selection for neural networks: Application to corporate bond rating prediction. In J. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan Kaufmann, 1991.

[112] S. Mori, C. Y. Suen, and K. Yamamoto. Historical review of OCR research and development. *Procs. of IEEE*, 80:1029–1057, 1992.

[113] M. Morita, L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. An

HMM-MLP hybrid system to recognize handwritten dates. In *Proc. of International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, Honolulu, USA, 2002. IEEE Computer Society.

[114] G. Nagy. State of the art in pattern recognition. *Procs of IEEE*, 56:336–362, 1992.

[115] P. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Trans. on Computers*, 26:917–922, 1977.

[116] H. Nishida. Curve description based on directional features and quasi-convexity/concavity. *Pattern Recognition*, 28(7):1045–1051, 1995.

[117] I-S. Oh, J-S. Lee, and C. Y. Suen. Analysis of class separation and combination of class-dependent features for handwriting recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(10):1089–1094, 1999.

[118] I-S. Oh and C. Y. Suen. Distance features for neural network-based recognition of handwritten characters. *International Journal on Document Analysis and Recognition*, 1(2):73–88, 1998.

[119] L. S. Oliveira, N. Benahmed, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Feature subset selection using genetic algorithms for handwritten digit recognition. In *Proc. of 14th Brazilian Symposium on Computer Graphics and Image Processing*, pages 362–369, Florianopolis, Brazil, 2001. IEEE Computer Society.

[120] L. S. Oliveira, E. Lethelier, F. Bortolozzi, and R. Sabourin. Handwritten digits segmentation based on structural approach. In *Proceedings of the 13th Brazilian Symposium on Computer Graphics and Imaging Processing*, pages 67–73, Gramado, Brazil, 2000. IEEE Computer Society.

[121] L. S. Oliveira, E. Lethelier, F. Bortolozzi, and R. Sabourin. A new approach to segment handwritten digits. In *Proc. of 7th International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, pages 577–582, Amsterdam, Netherlands, 2000.

[122] L. S. Oliveira, E. Lethelier, F. Bortolozzi, and R. Sabourin. A new segmentation approach for handwritten digits. In *Proc. of 15th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 323–326, Barcelona, Spain, 2000. IEEE Computer Society.

[123] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. High-level verification of handwritten numeral strings. In *Proc. of 14th Brazilian Symposium on*

Computer Graphics and Image Processing, pages 36–43, Florianopolis, Brazil, 2001. IEEE Computer Society.

[124] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. A modular system to recognize numerical amounts on Brazilian bank cheques. In *Proc. of 6th International Conference on Document Analysis and Recognition (ICDAR)*, pages 389–394, Seattle, USA, 2001. IEEE Computer Society.

[125] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Automatic recognition of handwritten numerical strings: A recognition and verification strategy. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(11):1438–1454, 2002.

[126] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Feature selection using multi-objective genetic algorithms for handwritten digit recognition. In *Proc. of 16th International Conference on Pattern Recognition*, volume 1, pages 568–571, Quebec City, Canada, 2002. IEEE Computer Society.

[127] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Feature selection for ensembles : A hierarchical multi-objective genetic algorithm approach. In *Proc. of 7th International Conference on Document Analysis and Recognition*, Edinburgh-Scotland, 2003. IEEE Computer Society.

[128] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Impacts of verification on a numeral string recognition system. *Pattern Recognition Letters*, 24(7):1023–1031, 2003.

[129] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. A methodology for feature selection using multi-objective genetic algorithms for handwritten digit string recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(6), 2003.

[130] D. W. Optiz. Feature selection for ensembles. In *Proc. of 16th International Conference on Artificial Intelligence*, pages 379–384, 1999.

[131] N. C. Oza and K. Tumer. Input decimation ensembles: Decorrelation through dimensionality reduction. In *Proc. of the 2nd International Workshop on Multiple Classifier Systems*, pages 238–247, Cambridge, UK, 2001.

[132] J. Park, V. Govindaraju, and S. N. Srihari. OCR in a hierarchical feature space. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(4):400–407, 1998.

[133] S. Pittner and S. V. Kamarthi. Feature extraction from wavelet coefficients for pattern recognition tasks. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(1):83–88, 1999.

[134] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[135] S. Procter, J. Illingworth, and A. J. Elms. The recognition of handwritten digit strings of unknown length using hidden Markov models. In *Proc. of 14th International Conference Pattern Recognition (ICPR)*, pages 1515–1517, 1998.

[136] V. E. Ramesh and N. Murty. Off-line signature verification using genetically optimized weighted features. *Pattern Recognition*, 32(2):217–233, 1999.

[137] S. Rao. *Optimization theory and application.* New Delhi:Wiley Eastern Limited, 1991.

[138] M. L. Raymer, W. F. Punch, E. D. Goodman, L. A. Kuhn, and L. C. Jain. Dimensionality reduction using genetic algorithms. *IEEE Trans. on Evolutionary Computation*, 4(2):164–171, 2000.

[139] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate bayesian a posteriori probabilities. *Neural Computation*, 3(4):461–483, 1991.

[140] M. Richeldi and P. Lanzi. Performing effective feature selection by investigating the deep structure of the data. In *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining*, pages 379–383, 1996.

[141] J. Schurmann. *Pattern Classification - A unified view of statistical and neural approaches.* Wiley interscience, 1996.

[142] D. Shi. Feature selection for handwritten Chinese character recognition based on genetic algorithm. In *International Conference on System, Man, and Cybernetics*, volume 5, pages 4201–4206, 1998.

[143] Z. Shi, N. Srihari, C. Y. Shin, and A. V. Ramanaprasad. A system for segmentation and recognition of totally unconstrained handwritten numeral strings. In *Proc. of 4th International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 455–458, 1997.

[144] M. Shridhar and A. Badreldin. High accuracy character recognition algorithm using fourier and topological descriptors. *Pattern Recognition*, 17(5):515–524, 1984.

[145] M. Shridhar and A. Badreldin. Recognition of isolated and simply connected handwritten numerals. *Pattern Recognition*, 19(1):1–12, 1986.

[146] W. Siedlecki and J. Sklansky. A note on genetic algorithms for large scale on feature selection. *Pattern Recognition Letters*, 10:335–347, 1989.

[147] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1995.

[148] N. W. Strathy. A method for segmentation of touching handwritten numerals. Master's thesis, Concordia University, Montreal - Canada, Setember 1993.

[149] C. Y. Suen, M. Berthod, and S. Mori. Automatic recognition of handprinted characters: The state of the art. *Procs. of IEEE*, 68:469–487, 1980.

[150] C. Y. Suen, K. Liu, and N. W. Strathy. Sorting and recognizing cheques and financial documents. In *Proc. of 3rd IAPR Workshop on Document Analysis Systems*, pages 1–18, Nagano, Japan, 1998.

[151] C. Y. Suen, C. Nadal, R. Legault, T. A. Mai, and L. Lam. Computer recognition of unconstrained handwritten numerals. *Procs of IEEE*, 80:1162–1180, 1992.

[152] H. Takahashi and T. Griffin. Recogniton enhancement by linear tournament verification. In *Proc. of 2nd International Conference on Document Analysis and Recognition (ICDAR)*, pages 585–588, Tsukuba, Japan, 1993. IEEE Computer Society.

[153] C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in on-line handwritting recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(8):787–808, 1990.

[154] D. Tax, M. Breukelen, R. Duin, and J. Kittler. Combining multiple classifiers by averaging or by multiplying? *Pattern Recognition*, 33(9):1475–1485, 2000.

[155] L. N. Teow and J.F. Loe. Robust vision-based features and classification schemes for off-line handwritten digit recognition. *Pattern Recognition*, 35(11):2355–2364, 2002.

[156] O. D. Trier, A. K. Jain, and T. Taxt. Feature extraction methods for character recognition: A survey. *Pattern Recognition*, 29(4):641–662, 1996.

[157] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

[158] D. S. Weile, E. Michielssen, and D. E. Goldberg. Genetic algorithm design of Pareto optimal broadband microwave absorbers. *IEEE Trans. on Electromagnetic Compatibility*, 38(3):518–525, 1996.

[159] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Trans. on Systems, Man, and Cybernetics*, 22(3):418–435, 1992.

[160] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13(1):44–49, 1998.

[161] H. Yuan, S. S. Tseng, W. Gangshan, and Z. Fuyan. A two-phase feature selection method using both filter and wrapper. In *Proc. of IEEE International Conference on Systems, Man, and Cybernetics*, volume 2, pages 132–136, 1999.

[162] B. Zhang, M. Fu, and H. Yan. A nonlinear neural network model of mixture of local principal component analysis: application to handwritten digits recogntion. *Pattern Recognition*, 34(2):203–214, 2001.

[163] B. Zhang, M. Fu, H. Yan, and M. A. Fabri. Handwritten digit recognition by adaptative-subspace self organizing map (ASSOM). *IEEE Trans. on Neural Networks*, 10:939–945, 1999.

[164] G. P. Zhang. Neural networks for classification: A survey. *IEEE Trans. on Systems, Man, and Cybernetics - Part C:Applications and Reviews*, 30(4):451–462, 2000.

[165] J. Zhou. *Recognition and Verification of Unconstrained Handwritten Numeral.* PhD thesis, Concordia University, Montreal-Canada, November 1999.

[166] J. Zhou, Q. Gan, A. Krzyzak, and C. Y. Suen. Recognition of handwritten numerals by quantun neural networks with fuzzy features. *International Journal on Document Analysis and Recognition*, 2(1):30–36, 1999.

[167] J. Zhou, Q. Gan, A. Krzyzak, and C. Y. Suen. Recognition and verification of touching handwritten numerals. In *Proc. of 7^{th} International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, pages 179–188, Amsterdam, Netherlands, 2000.

[168] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[169] J. J. Zou and H. Yan. Extracting strokes from static line images based on selective searching. *Pattern Recognition*, 32(6):935–946, 1999.