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
MARISA EMIKA MORITA

AUTOMATIC RECOGNITION OF HANDWRITTEN DATES ON BRAZILIAN BANK CHEQUES

MONTREAL, 26TH JUNE, 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. Pierre Dumouchel, President
Département de Génie Électrique, École de Technologie Supérieure

Dr. Jean Meunier, External Examiner
Departement d'Informatique et Recherche Opérationnelle, Univeristé de Montréal

Dr. Tony Wong, 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 28TH MAY, 2003

AT THE ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

RECONNAISSANCE AUTOMATIQUE DE DATES MANUSCRITES SUR LES CHÈQUES BRÉSILIENS

Marisa Emika Morita

SOMMAIRE

Cette thèse présente un système hybride HMM-MLP pour la segmentation et la reconnaissance de l'écriture manuscrite non-contrainte et plus particulièrement des dates présentes sur les chèques brésiliens. Le système développé doit tenir compte de nombreuses sources de variabilité, tel que les différents styles et types de données, les variations présentes dans le champ correspondant aux dates ou encore les cas de segmentation difficile. Tous ceci rend bien évidemment la tâche particulièrement complexe.

Le système présenté s'appuie sur une modélisation Markovienne (HMM) pour identifier et séparer la date en sous-champs. Le concept de méta-classes de chiffres est utilisé afin de réduire la dimension du lexique des jours et des années et permettre ainsi une segmentation plus précise. Ensuite, les trois sous-champs principaux (jour, mois et année) sont reconnus en utilisant le classifieur spécialisé relatif au type de données identifié. Dans ce cas, nous proposons d'utiliser un système HMM pour reconnaître et vérifier le mot correspondant au mois et un réseau MLP pour décoder les chiffres représentant le jour et l'année. La stratégie de reconnaissance des chaînes de chiffres utilise aussi des méta-classes de chiffres de manière à réduire la taille du lexique et à améliorer les taux de reconnaissance. De plus, afin de valider notre stratégie combinant la reconnaissance des chaînes de chiffres et la vérification au niveau mot, nous avons utilisé d'autre base de données que celle des dates. D'autre part, le système comprend un module de décision final intégrant la notion de rejet.

Enfin, nous proposons une approche de sélection de caractéristiques adaptée aux apprentissages non-supervisés. Pour ce faire, nous utilisons un algorithme génétique multi-critères qui génère un ensemble de solutions, correspondant aux meilleurs compromis entre les caractéristiques discriminantes et le nombre de regroupements pertinents. La stratégie proposée a été évaluée sur des problèmes synthétiques où les caractéristiques significatives et les regroupements appropriés sont connus pour tous les sous-espaces de caractéristiques possibles. Elle a ensuite été utilisée pour l'optimisation des HMM dans le contexte d'un apprentissage supervisé. Notre approche est alors évaluée en effectuant un certain nombre d'expériences sur la reconnaissance des mots isolés correspondant au mois. L'approche a aussi été utilisée

pour optimiser la vérification de mots du système de reconnaissance de date. Ainsi, nous avons démontré expérimentalement la faisabilité et l'efficacité de l'approche proposée.

RECONNAISSANCE AUTOMATIQUE DE DATES MANUSCRITES SUR LES CHÈQUES BRÉSILIENS

Marisa Emika Morita

RÉSUMÉ

L'écriture fut durant ce dernier siècle le mode de collecte, de stockage et de transmission de l'information le plus couramment utilisé. Aujourd'hui ce mode de communication ne se limite plus à l'échange entre les êtres humains mais il est aussi utilisé comme interface entre l'homme et la machine. Ainsi, la lecture automatique de l'écriture fut durant ces trente dernières années l'objet d'intenses recherches. Toutefois, les travaux initiateurs souffraient des limitations des ordinateurs de l'époque, en termes d'espace mémoire et de puissance de calcul. Mais avec l'explosion des technologies de l'information qui a lieu depuis le début des années 80, la quantité de recherche dans le domaine a énormément augmenté. D'autre part, l'intérêt suscité par ce champ de recherche n'est pas uniquement motivé par le fabuleux défi technologique, mais également par les nombreuses applications commerciales et les bénéfices que peuvent apporter un système de lecture automatique robuste.

En effet, chaque jour plus d'un milliard de documents commerciaux et financiers sont traités par ordinateur. La grande majorité d'entre eux sont traités manuellement par des opérateurs humains qui doivent lire puis retranscrire le contenu de chaque document ce qui représente une tâche fastidieuse et très coûteuse en temps. Il est donc naturel de chercher à automatiser ce processus en remplaçant l'opérateur humain par un système de reconnaissance capable d'effectuer le même travail. Les avancés récents dans le domaine de l'analyse et de la reconnaissance des documents, ainsi que la baisse des coûts de la puissance de calcul permettent aujourd'hui le développement de systèmes efficaces qui tentent de minimiser les interventions humaines. Cependant, la route est encore longue avant d'arriver à l'obtention d'une machine capable de simuler parfaitement le processus de la lecture chez l'être humain, tout particulièrement dans le cas de la reconnaissance hors-ligne d'écriture manuscrite non-contrainte.

Parmi les nombreuses études présentées dans la littérature la plupart portent sur la reconnaissance de sous-unités isolées d'écriture, tels que les caractères, les mots ou les chaînes de chiffres. Mais ce n'est que récemment que l'on a commencé à s'intéresser à la reconnaissance de "phrases" composées de séquences de mots. Les applications visées sont alors la lecture de pages de texte, des adresses postales ou encore du montant et de la date présents sur les chèques. Dans le cas des chèques bancaires, de nombreux systèmes ont déjà été développés pour extraire l'information

pertinente, vérifier la signature ou le montant. Mais il n'existe que très peu d'études concernant le traitement des dates présentes sur les chèques, une étape essentielle pour la lecture des chèques. En effet, dans de nombreux pays comme le Brésil ou le Canada, il est illégal de post-dater un chèque, la vérification de la date indiquée sur le chèque est alors indispensable. Ainsi, notre travail porte sur la lecture hors-ligne d'écriture manuscrite non-contrainte dans le cadre de la reconnaissance des dates présentes sur les chèques brésiliens.

Le développement d'un système efficace de reconnaissance de dates est un défi intéressant car ceci nécessite de gérer de nombreux niveaux de complexité. Le système doit fonctionner indépendamment du scripteur et tenir compte de différents types de données comme les chaînes de chiffres ou les mots manuscrits non-contraints (écriture bâton, cursive ou mixte). Bien que la dimension du lexique des noms de mois soit limitée, il existe certaines classes tel que "Setembro" (septembre), "Novembro" (novembre) et "Dezembro" (décembre) qui contiennent une sous-chaîne commune ("embro") ce qui complique fortement la tâche de reconnaissance. D'autre part, le système doit aussi tenir compte des variations présentes dans le champ correspondant à la date : Est-ce que le jour est représenté par un ou deux chiffres et l'année par deux ou quatre chiffres ? Y-a-t-il des séparateurs de champs ou un nom ville ? De plus, le système doit donc diviser le champ en sous-champs et gérer des cas où la segmentation est difficile étant donné que parfois les espaces entre les sous-champs et ceux localisés à l'intérieur d'un même sous-champ peuvent être similaires. Dans de tel cas de figure, il est très difficile de détecter correctement les espaces entre les sous-champs en utilisant une méthode de segmentation basée sur l'analyse des relations géométriques entre les différentes composantes connexes de l'image. Ceci peut s'expliquer par le fait que cette stratégie montre rapidement ses limites lorsque la segmentation correcte ne correspond pas avec la règle de segmentation prédéfinie.

Le but principal de notre recherche est donc de développer un système de reconnaissance de dates manuscrites non-contraintes. Afin d'évaluer un tel système, nous avons collecté une base de données d'environ 2.000 images des chèques bancaires brésiliens. Puisque séparer une date en sous-champs en utilisant une méthode à base de règles est une tâche difficile, nous avons préféré opter pour une stratégie basée sur une modélisation Markovienne (HMM) pour segmenter et identifier simultanément les dates en sous-champs. Ensuite, les trois sous-champs principaux (jour, mois et année) sont reconnus en utilisant le classifieur spécialisé pour ce type de données. Dans ce cas, nous proposons d'utiliser un système HMM pour reconnaître et vérifier le mot correspondant au mois et un système basé sur des réseaux de neurones de type MLP pour décoder les chiffres représentant le jour et l'année. En effet, il a été démontré que les réseaux de type MLP sont des classifieurs plus discriminants que les HMM pour effectuer la reconnaissance des chaînes numériques. Par contre, les approches basées sur la modélisation Markovienne sont plus efficaces pour effectuer

la segmentation des champs d'intérêts et la reconnaissance des mots manuscrits. D'autre part, le système comprend un module de décision final intégrant la notion de rejet.

Le second aspect de nos travaux porte sur l'optimisation du système de reconnaissance. Mais étant donné la complexité de ce type de système, nous avons préféré concentrer nos efforts sur l'optimisation du vérificateur de mots. Ainsi, nous proposons une approche de sélection de caractéristiques adaptée aux apprentissages non-supervisés. Pour ce faire, nous utilisons un algorithme génétique multi-critères qui génère un ensemble de solutions correspondant aux meilleurs compromis entre les caractéristiques discriminantes et le nombre de regroupements pertinents. La stratégie proposée a été évaluée sur des problèmes synthétiques où les caractéristiques significatives et les regroupements appropriés sont connus pour tous les sous-espaces de caractéristiques possibles. Elle a ensuite été utilisée pour l'optimisation de classifieurs dans le contexte d'un apprentissage supervisé. Notre approche est alors évaluée en effectuant un certain nombre d'expériences sur la reconnaissance des mots isolés correspondant au mois. Nous savons que le développement d'un système hybride utilisant respectivement HMM et MLP pour la reconnaissance de mots et de chaînes de chiffres n'est pas un concept nouveau. Toutefois, la contribution principale de ce travail de doctorat porte sur quatre aspects. Le premier repose sur l'utilisation d'une approche Markovienne pour segmenter les dates en sous-champs. Pour ce faire, nous utilisons le concept de méta-classes de chiffres de manière à réduire la dimension du lexique lors de la reconnaissance du jour et de l'année et améliorer les résultats de la segmentation. Un second point important de nos travaux de recherche concerne la méthode adoptée pour réduire la dimension du lexique utilisé pour la reconnaissance des chaînes de caractères et ainsi augmenter significativement les taux de reconnaissance. La stratégie utilise les méta-classes et la longueur des chaînes de chiffres, c'est à dire que le nombre de chiffres correspondant au jour et à l'année est obtenu en sortie des HMM. Le troisième aspect porte sur le système de vérification de mots. Les expériences apportent des résultats encourageant lors de la reconnaissance de dates, de mots et de chaînes de caractères et mettent en avant le rôle important du vérificateur de mots dans le système. En plus de la base de données de dates nous avons utilisé la base NIST SD19 et une base de montants de chèques brésiliens pour valider les stratégies respectivement employées pour la reconnaissance des chaînes de chiffres et la vérification des mots. Pour finir, la dernière contribution importante concerne l'approche de sélection de caractéristique non-supervisée qui a été développée pour la reconnaissance de mots manuscrits. Ainsi, nous avons démontré expérimentalement la faisabilité et l'efficacité de l'approche proposée.

ACKNOWLEDGEMENTS

First and foremost I would like to express my gratitude to Prof. Robert Sabourin and Prof. Ching Y. Suen for their support, guidance, encouragement, and assistance at so many levels throughout my study at École de Technologie Supérieure. I am also indebted to Prof. Flávio Bortolozzi who has also supported and encouraged my work.

I would like to thank everyone else in the Laboratoire de Vision, d'Imagerie et d'Intelligence Artificielle (LIVIA) and in the Centre for Pattern Recognition and Machine Intelligence (CENPARMI). Both laboratories have been an ideal environment, both socially and technically, in which to conduct research.

Thanks to Fundação Araucária, CENPARMI, and NSERC of Canada for supporting this work.

Special thanks go to Edouard Lethelier and Abdenaim El Yacoubi who have shared with their knowledge and research experience in the field.

I would also like to thank those people in Pontifical Catholic University of Paraná who help in building the database of Brazilian bank cheques, especially Cinthia O. A. Freitas.

Thanks to my friends and family, especially to my father Paulo Morita (in memoriam) and my mother Tieko Ito Morita, for their friendship, love, and encouragement.

Finally, I would like to thank my friend and husband Luiz S. Oliveira who has always been there for me. I want to dedicate this thesis to him and to our daughter Isabela.

CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Page

LIST OF ACRONYMS AND SYMBOLS

CA	Courtesy Amount
CN	Convolutional Network
CEDAR	Center of Excellence for Document Analysis and Recognition
CENPARMI	Centre for Pattern Recognition and Machine Intelligence
CUR	Cursive Handwriting
DB	Davies Bouldin
DDD	Distance Features
DP	Dynamic Programming
DPI	Dots Per Inch
ENG	English
FR	French
GA	Genetic Algorithm
GER	German
HMM	Hidden Markov Model
HNN	Hopfield Neural Network
HSF	Handwritten Sample Forms
k-NN	k-Nearest-Neighbor
L	Lowercase Letter
LA	Legal Amount
MLP	Multi-Layer Perceptron
MP	Upper Contour Minimum

NN	Neural Network
NIST	National Institute of Standards and Technology
NSGA	Nondominated Sorting Genetic Algorithm
OMNI	Omni-writer
POR	Portuguese
QNN	Quantum Neural Network
RBF	Radial Basis Function
RR	Recognition Rate
RJ	Rejection Rate
RL	Reliability Rate
SDNN	Space Displacement Neural Network
SOM	Self-Organized Maps
SP	Segmentation Point
SSF	Segmentation into Sub-Fields
SVM	Support Vector Machine
TDNN	Time Delay Neural Network
TR	Training Set
TS	Testing Set
U	Uppercase Letter
UNC	Unconstrained Handwriting
VL	Validation Set
WD	Writer-Dependent
WR	Word Recognition

WV	Word Verification
0U	Probability of beginning a word by an uppercase letter
0L	Probability of beginning a word by a lowercase letter
1D	Probability of being 1-digit day
2D	Probability of being 2-digit day
$A = \{a_{ij}\}$	Probability of going from state s_i at time t to state s_j at time $(t+1)$, and at the same time producing a real observation o_t at time t
$A' = \{a'_{ij}\}$	Probability of null transition from state s_i at time t to state s_j at time t, producing null observation symbol Φ
$B_p = \{b^p_{ij}(k)\}$	Output pdf of observing the k^{th} symbol in the p^{th} feature set when a transition from state s_i at time t to state s_j at time $(t+1)$ is taken
$\alpha_t(i)$	Probability of the partial observation sequence $(o_0, o_1 \ldots, o_{t-1})$ (until time $t-1$) and the state s_i reached at time t given the model λ
$\beta_t(i)$	Probability of the partial observation sequence from the time t to the end, given state s_i reached at time t and the model λ
$\delta_t(i)$	Probability of the best path that accounts for the first t observations and ends at state s_i at time t
C_i	Cluster i
$C_{0,1,2,3}$	Meta-class of digits 0, 1, 2, and 3
C_{0-9}	10 numerical classes
$C_{1,2}$	Meta-class of digits 1 and 2
$C_{0,9}$	Meta-class of digits 0 and 9
$C_{0,1,2,9}$	Meta-class of digits 0, 1, 2, and 9

D	Number of selected features
D_i	Average distortion in cell C_i
$D_{overall}$	Overall average distortion
Δ	Limit for jumps
$d(i,j)$	Distance between two individuals i and j
d_{ij}	Euclidean distance between the centroids Z_i and Z_j
e	Classifier of expert
$e_{0,1,2,3}$	Classifier trained with meta-class $C_{0,1,2,3}$
e_{0-9}	Classifier trained with 10 numerical classes
$e_{1,2}$	Classifier trained with meta-class $C_{1,2}$
$e_{0,9}$	Classifier trained with meta-class $C_{0,9}$
$e_{0,1,2,9}$	Classifier trained with meta-class $C_{0,1,2,9}$
F	Final state
F_1	Global feature set
F_2	Concavity feature set
$\gamma_t(i)$	Probability of being in state s_i at time t, given λ and O
I	Initial state
I_{DB}	DB index
K	Number of selected clusters
LL	Probability of being two lowercase letters
LU	Probability of being one lowercase letter followed by one uppercase letter
Λ	Set of specified patterns

λ	Complete parameter set of the model
M_a	Model of class "a"
M_A	Model of class "A"
$M_{a'}$	Model of class "a"
$M_{A'}$	Model of class "A"
M_{city}	City model
M_{day}	1- or 2-digit day model
M_{de}	"De" separator model
M_{month}	Month model
M_p	Number of possible observation symbols for the p^{th} feature set
M_{space}	Inter-sub-field space model
M_{space}^{digit}	Intra-digit space model
M_{space}^{word}	Intra-word space model
M_v	Model of class "v"
M_V	Model of class "V"
$M_{v'}$	Model of class "v"
$M_{V'}$	Model of class "V"
M_{year}	2- or 4-digit year model
$M_{0,1,2,3}$	Model of meta-class $C_{0,1,2,3}$
M_{0-9}	Model of 10 numerical classes
$M_{1,2}$	Model of meta-class $C_{1,2}$
$M_{0,9}$	Model of meta-class $C_{0,9}$
$M_{0,1,2,9}$	Model of meta-class $C_{0,1,2,9}$

N	Number of states in the model
N_S	Number of mutually exclusive sets $\Omega = \omega_1 \cup \ldots \cup \omega_{N_S}$
N_{TS}	Number of string images in a test set
N_{rej}	Number of rejected string images
N_{rec}	Number of classified string images
N_{err}	Number of misclassified string images
σ_{share}	Maximum distance allowed between any two individuals
O	Observation sequence
o_t^p	Observation at time t being drawn from the p^{th} finite feature set
ω_i	Mutually exclusive set of the pattern space
Ω	Pattern space
p	Decision variable
p^{th}	Finite feature set
P	Number of feature sets
$P(\cdot)$	Probability
$P(O\|\lambda)$	Probability of O given λ
Φ	Null observation symbol
$\pi = \{\pi_i\}$	Initial state distribution
$\Psi_t(i)$	Function used to recover the best state sequence
q	Desired number of distinct Pareto-optimal solutions
q_t	State at time t
$S = \{s_0, s_1, \ldots, s_{N-1}\}$	Set of possible states of the model
Sep1	First separator

Sep2	Second separator
Sep3	Third separator
S_i	Average Euclidean distance of the vectors X in cluster C_i with respect to its centroid Z_i
T	Length of the observation sequence $O = \{o_0, o_1, \ldots, o_{T-1}\}$
$(T_0, \ldots, T_n)$	Thresholds used for rejection
T_{error}	Fixed error rate
θ	Global slant
u	Index that indicates the observation sequence $O(u)$
U	Size of the set of training sequences $O(0), O(1), \ldots, O(U-1)$
UU	Probability of being two uppercase letters
UL	Probability of being one uppercase letter followed by one lowercase letter
V_p	Discrete set of possible observation symbols corresponding to the p^{th} feature set
x	Input pattern
X	Vector
$\xi_t^1(i,j)$	Probability of being in state s_i at time t and in state s_j at time $(t+1)$, producing a real observation O_t, given λ and O
$\xi_t^2(i,j)$	Probability of being in state s_i at time t and in state s_j at time t, producing the null observation Φ, given λ and O
Z_i	Centroid of the cluster C_i

INTRODUCTION

Writing, which has been the most natural mode of collecting, storing, and transmitting information through the centuries, now serves not only for communication among humans but also serves for communication of humans and machines. Machine simulation of human reading has been the subject of intensive research for the last three decades. However, the early investigations were limited by the memory and power of the computer available at that time. With the explosion of information technology, there has been a dramatic increase of research in this field since the beginning of 1980s. The interest devoted to this field is not explained only by the exciting challenges involved, but also the huge benefits that a system, designed in the context of a commercial application, could bring.

According to the way handwriting data is generated, two different approaches can be distinguished: on-line and off-line. In the former, the data are captured during the writing process by a special pen on an electronic surface. In the latter, the data are acquired by a scanner after the writing process is over. In this case, the recognition of off-line handwriting is more complex than the on-line case due to the presence of noise in the image acquisition process and the loss of temporal information such as the writing sequence and the velocity. This information is very helpful in a recognition process. Off-line and on-line recognition systems are also discriminated by the applications they are devoted to. The off-line recognition is dedicated to bank check processing, mail sorting, reading of commercial forms, etc, while the on-line recognition is mainly dedicated to pen computing industry and security domains such as signature verification and author authentication. In this thesis, we are concerned with the off-line recognition.

Each day, billions of business and financial documents have to be processed by computer. 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 recognition system that is able to do the operator's job. Recent advances in the field of document analysis and recognition and the availability of relatively inexpensive computer power can now allow the development of effective systems, which have attempted to minimize the manual effort involved. Nevertheless, 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.

Problem Statement

Many studies on the recognition of isolated units of writing such as characters, words or strings of digits can be found in the literature. Only recently the recognition of a sentence composed of a sequence of words or different data types has been investigated. Some typical applications on sentence recognition are reading texts from pages [72], street names from postal addresses [123], processing of legal amounts [48] and dates [109] on cheques.

With respect to bank processing, many systems have been developed for information extraction, courtesy amount and legal amount recognition, and signature verification. Comparatively, there is very limited published work on the processing of date information on cheques even though this is a necessity in some application environments, e.g., in Brazil and Canada, it is illegal to process post-dated cheques. The focus of this work is the off-line recognition of unconstrained handwritten dates written on Brazilian bank cheques.

In this application, the date from left to right can consist of the following sub-fields: city name, first separator (Sep1), day, second separator (Sep2), month, third separator (Sep3) and year. Figure 1 details the lexicon of each date sub-field present in the laboratory database and Figure 2 shows some samples of handwritten dates.

In both images, the grey color represents the obligatory date sub-fields (day, month and year).

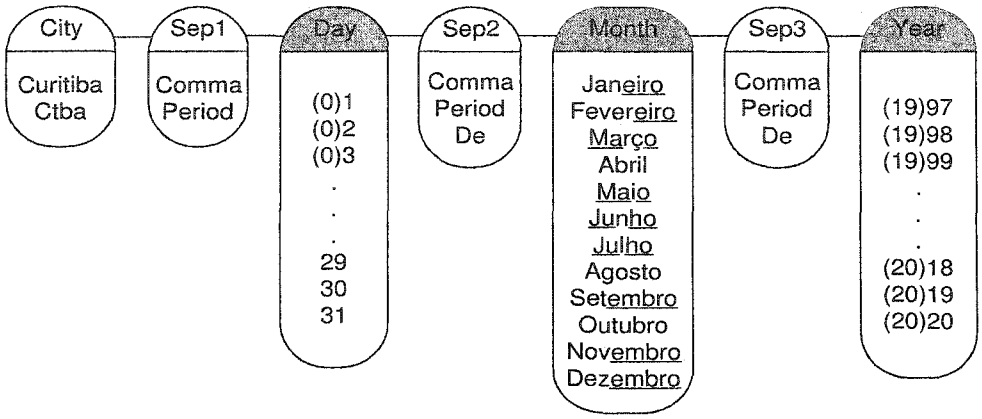


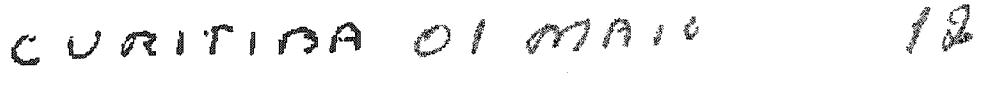
Figure 1 Lexicon of each date sub-field (the common sub-strings are highlighted by underlines)

The development of an effective off-line date processing system is very challenging because it has to tackle many levels of complexity. Firstly, the off-line approach is more complex than the on-line approach. Besides, the system is not writer-dependent (multiple writer or omni-writer) and it must consider different data types such as strings of digits and unconstrained handwritten words (handprinted, cursive, or mixed). As shown in Figure 1, although the lexicon size of month words is limited, there are some classes such as "Setembro" (September), "Novembro" (November) and "Dezembro" (December) that contain a common sub-string ("embro") and can affect the performance of the recognizer. The system must also take into account the variations present in the date field such as 1- or 2-digit day, 2- or 4-digit year, the presence or absence of the city name and separators.

In addition, the system must segment a sentence into its constituent parts and deal with difficult cases of segmentation since there are handwritten dates where the spaces between sub-fields (inter-sub-field) and within a sub-field (intra-sub-field) are

(a)

(b)

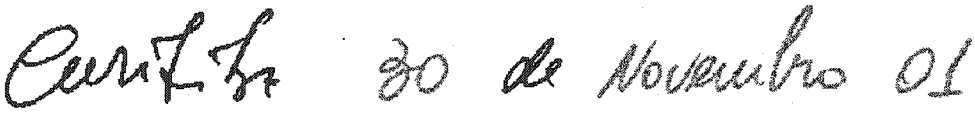


(c)

(d)

(e)

(f)



(g)

(h)

Figure 2 Samples of handwritten dates on Brazilian bank cheques

similar as shown in Figures 2(b) and 2(c). For example, in Figure 2(b) the intra-sub-field space between "1" and "0" is almost the same as the inter-sub-field spaces

between "Curitiba" and "3" or "Fevereiro" and "10". In such a case, it will be very difficult to detect the correct inter-sub-field spaces using a segmentation method based on an analysis of the geometric relationship of adjacent components in an image, which is perhaps the most frequently used method. This can be explained by the fact that this strategy shows its limits rapidly when the correct segmentation does not fit with the pre-defined rules of the segmenter.

Goals of the Research

The primary goal of our research is to develop a recognition system for unconstrained handwritten dates. Due to the complexity of processing the whole date at the same time, the baseline system discussed in this thesis first segments the date sub-fields and then recognizes the three obligatory ones (day, month, and year) using specialized classifiers. The segmentation of a date image into sub-fields is performed through the Hidden Markov Models (HMMs). We use the Multi-Layer Perceptron (MLP) neural networks to deal with strings of digits (day and year) and the HMMs to recognize and verify words (month).

The second aspect of our research lies in the optimization of the baseline system. But due to the magnitude of this kind of system, our focus will be the optimization of the word verifier. In order to perform this task, we propose a methodology for feature selection in unsupervised learning.

Contributions

The starting point of our research was the recognition of isolated month words. For this reason, our first contributions were made on word recognition and afterwards on date recognition. They are summarized as follows.

We 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 [25] and Appendix 1.

A combination strategy involving both holistic and analytical HMM approaches was presented in [83] to recognize isolated month words. We achieved satisfactory results given the small size of our date database and the limitations of our feature set based on the detection of loops, ascenders, and descenders, namely global features. Other results using different word recognition models were reported in [82]. In [88], we improved the foregoing results by combining the global and concavity features through HMMs considering only the analytical approach. Besides, we used the legal amount database in order to increase the frequency of characters in training.

An HMM-MLP hybrid system was presented in [84] to process complex date images written on Brazilian bank cheques. Firstly, the system splits the date sub-fields considering multiple-hypotheses of segmentation. After that, a recognition and verification strategy is used to recognize the three obligatory date sub-fields (day, month, and year) considering different classifiers. Since separating a date into sub-fields using a rule-based segmentation method is a difficult task, the HMMs were employed to identify and segment such sub-fields through the recognition process. MLPs and HMMs were adopted to deal with strings of digits (day and year) and words (month) respectively. This is justified by the fact that MLPs have been widely used for digit recognition and the literature shows better results using this kind of classifier [36, 91] than HMMs [46]. On the other hand, HMMs have been successfully applied to handwritten word recognition more recently [79, 121]. In [85] we improved the results presented in [84] by using a simpler approach that does not consider multiple hypotheses of segmentation and the digit string verification as well. Furthermore, in [86] we validated the strategy we developed for word recognition and verification by conducting some experiments using the legal amount database and comparing the

results achieved with an other study which makes use of the same database. This was done since it is very difficult to compare our work with others due to its special application, i.e., date recognition on Brazilian bank cheques. Besides, comparison in the same context with other approaches is very delicate when different databases and formats are used, different word classes are involved, and different sizes of databases are considered.

A methodology for feature selection in unsupervised learning for handwritten word recognition was proposed in [87]. It makes use of an efficient multi-objective genetic algorithm to generate a set of solutions, which contain the more discriminant features and the more pertinent number of clusters. The proposed strategy is assessed using two synthetic data sets where the significant features and the appropriate clusters in any given feature subspace are known. Afterwards, it is applied to optimize classifiers in a supervised learning context, i.e., handwritten word recognition. In this scenario, our approach is evaluated by conducting some experiments on isolated month word recognition. In this thesis, it is also used to optimize the word verifier of the date recognition system.

We have knowledge that the development of a hybrid system using HMMs and MLPs is not a new concept for word and digit string recognition respectively. However, the main contributions of our research focus on four aspects. The first one lies in the HMM-based approach for separating date sub-fields. It makes use of the concept of meta-classes of digits in order to reduce the lexicon size of the day and year and produce a more precise segmentation. The second important feature of our research is the scheme adopted to reduce the lexicon size on digit string recognition to improve the recognition results. Such a strategy uses the meta-classes of digits and the output of the HMMs in the form of digit string length, i.e., the information on the number of digits present in a string (day or year). The third aspect is the word verification scheme. Experiments show encouraging results on date recognition

and the important role of the word verifier in the system. In addition to the date database, we have used the NIST (National Institute for Standards and Technology) SD19 and legal amount databases to validate the strategies employed in digit string recognition and word verification respectively. Finally, the last main contribution is the unsupervised feature selection methodology we have developed for handwritten word recognition. Comprehensive experiments demonstrate the feasibility and efficiency of the proposed methodology.

Outline of the Thesis

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

In Chapter 2 we provide a brief overview of the date recognition system and we introduce all definitions related to the system. Firstly, the concept of meta-classes of digits is defined. Then, the Markovian and Neural classifiers used in the system are described. Finally, the definitions about levels of verification are presented.

Chapter 3 is devoted to the first module of the system, namely preprocessing. This module consists of several preprocessing steps which aim at producing images that are easy for the system to operate more accurately.

Chapter 4 addresses the HMM-based strategy we have developed for separating a date image into sub-fields. All steps necessary to perform segmentation are detailed in this chapter. The segmentation steps include the description of the feature sets that feed our Markovian classifiers, the training mechanism, and the way segmentation is done.

The module that recognizes the strings of digits is reported in Chapter 5. In this chapter, we also describe the word recognizer and verifier and how they interact with each other. Additionally, we present the final decision module which makes an accept/rejection decision.

In order to support the ideas proposed in this thesis, Chapter 6 presents comprehensive experiments, which are conducted on three different databases: dates, legal amounts, and NIST SD19. The specifications of such databases are reported in Appendix 1.

In Chapter 7 we discuss the technique we have investigated in order to optimize the word verifier of the data recognition system: feature selection in unsupervised learning. Finally, the final chapter concludes this thesis and gives some directions for future work.

As mentioned before, this thesis contains five appendixes. The first one, Appendix 1, describes the databases used by the system, which are the date, legal mount, and NIST SD19 databases. The second one, Appendix 2, reports the theory of HMMs. In Appendixes 3 and 4, we describe the weighted least square approach and the process of vector quantization respectively used in the system. The last one, Appendix 5, details the distance features, which are employed in the optimization of the word verifier. Nevertheless, the theory of neural networks and genetic algorithms is not discussed in detail in this thesis because we have assumed that the reader is familiar with such subjects. For those who do not feel comfortable in these fields we would suggest the reading of the following references: [5] for neural networks and [31] for genetic algorithms.

CHAPTER 1

STATE OF THE ART

Automatic processing of unconstrained handwritten dates on bank cheques represents a typical application in the field of off-line handwriting recognition. In this chapter, we address the main topics related to our research. First of all, we review some recognition techniques for handwriting recognition. After that, we discuss the problem of handwritten digit string recognition, handwritten word recognition, and handwritten sentence recognition since processing of dates on cheques is related to such fields. Furthermore, we show the performance of some date recognition systems on cheques.

1.1 Handwriting Recognition Techniques

Numerous techniques for handwriting recognition have been investigated based on four general approaches of pattern recognition, as suggested by [42]: template matching, statistical techniques, structural techniques, and neural networks. Such approaches are neither necessary independent nor disjointed from each other. Occasionally, a technique in one approach can also be considered to be a member of other approaches.

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 three classes: direct matching [27], deformable templates and elastic matching [17], and relaxation matching [116].

Statistical techniques is concerned with statistical decision functions and a set of optimally criteria, which determine the probability of the observed pattern belonging

to a certain class. Several popular handwriting recognition approaches belong to this domain:

- The k-Nearest-Neighbor (k-NN) rule is a popular non-parametric recognition method, where the *a 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 [34]. 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. A comparison of fast nearest neighbor classifiers for handwriting recognition is given in [76].

- The Bayesian classifier assigns a pattern to a class with the maximum *a posteriori* probability. Class prototypes are used in the training stage to estimate the class-conditional probability density function for a feature vector [18].

- 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 [99].

- Hidden Markov Model (HMM) is a doubly stochastic process, with an underlying stochastic process that is not observable (hence the word hidden), but can be observed through another stochastic process that produces the sequence of observations [96]. An HMM is called discrete if the observations are naturally discrete or quantized vectors from a codebook or continuous if these observations are continuous. HMMs have been proven to be one of the most powerful tools for modeling speech and later on a wide variety of other real-world signals. These probabilistic models offer many desirable properties for modeling characters or words. One of the most important properties is the existence

of efficient algorithms to automatically train the models without any need of labeling presegmented data. The algorithms for training and recognition the HMMs we have used in this thesis are discussed in Appendix 2. HMMs have been extensively applied to handwritten word recognition [33, 59, 101, 121] and their applications to handwritten digit recognition [47, 8] have been growing. The literature presents two basic approaches for handwriting recognition using HMM: Model-Discriminant HMM and Path-Discriminant HMM. In the former, a model is constructed for each class (word, character, or segmentation unit) in the training phase. In the latter, a single HMM is constructed for the whole language or context. The performances of these two approaches are compared in various experiments by utilizing different lexicon sizes [59].

- Fuzzy set reasoning is a technique that employs fuzzy set elements to describe the similarities between the features of the characters. Fuzzy set elements give more realistic result when there is not *a priori* knowledge about the data, and therefore, the probabilities cannot be calculated. The literature reports different approaches based on this technique such as fuzzy graphs [1], fuzzy rules [28], and linguistic fuzzy [60].

- Support Vector Machine (SVM) is based on the statistical learning theory [112] and quadratic programming optimization. An SVM is basically a binary classifier and multiple SVMs can be combined to form a system for multi-class classification. In the past few years, SVM has received increasing attention in the community of machine learning due to its excellent generalization performance. More recently, some SVM classification systems have been developed for handwriting digit recognition, and some promising results have been reported [3, 7].

In structural techniques 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 categorized into two classes: grammatical methods [103] and graphical methods [51].

A Neural Network (NN) is defined as a computing structure consisting of a massively parallel interconnection of adaptative "neural" processors. The main advantages of neural networks are: the ability to be trained automatically from examples, good performance with noisy data, possible parallel implementation, and efficient tools for learning large databases. NNs have been widely used in this field and promising results have been achieved, especially in handwriting digit recognition. The most widely studied and used neural network is the Multi-Layer Perceptron (MLP) [5]. Such an architecture trained with back-propagation [63] is among the most popular and versatile forms of neural network classifiers and is also among the most frequently used traditional classifiers for handwriting recognition. See [125] for a review. Other architectures include Convolutional Network (CN) [62], Self-Organized Maps (SOM) [124], Radial Basis Function (RBF) [5], Space Displacement Neural Network (SDNN) [75], Time Delay Neural Network (TDNN) [67], Quantum Neural Network (QNN) [127], and Hopfield Neural Network (HNN) [68].

The above review indicates that there are many recognition techniques available for handwriting recognition systems. All of them have their own superiorities and weaknesses. In recent years, many researchers have combined such techniques in order to improve the recognition results. The idea is not rely on a single decision making scheme. Various classifier combination schemes have been devised and it has been experimentally demonstrated that some of them consistently outperform a single best classifier [54, 117].

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 [46, 56, 91].

1.2 Handwritten Digit String Recognition

Intensive research on the recognition of isolated digits in the past decade has led to recognition rates close to 99% (zero-rejection level) [36, 91]. Many experiments have been conducted on the CENPARMI (Centre for Pattern Recognition and Machine Intelligence), CEDAR (Center of Excellence for Document Analysis and Recognition), and NIST (National Institute for Standards and Technology) databases, which are well-known databases used by researchers in this domain. The recent efforts have focussed on digit string recognition due to its large number of potential applications. The recognition of numerical strings differs from that of isolated digits by including the classification of digit groups and segmentation of touching digits. It is also different from the problem of recognizing handwritten words 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 digit can be broken into several parts.

Strategies for handwritten numerical string recognition can be divided into segmentation-then-recognition [102, 103] and segmentation-based recognition [37, 74]. In the former, 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.

The segmentation-based recognition 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 [10, 67, 91] or implicit when each pixel column [47] or slide window [49, 64, 73] is a potential cut location. A good review about segmentation can be found in [9].

In the remaining of this section we discuss some important works on handwritten digit string recognition. The results claimed by authors are reported in Table I on the NIST database (this database is described in Appendix 1). In this Table, the results were presented for 2-digit and 4-digit strings without the knowledge of its length.

Ha et al in [37] 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.

Oliveira et al in [91] propose a segmentation-based recognition where the segmentation is explicitly performed. They present a recognition and verification strategy

based on MLP classifiers. The verification scheme contains two verifiers in order to deal with the problems of over-segmentation and under-segmentation. They use 12,802 strings of the NIST SD19 (hsf_7 series).

In opposite, Lee and Kim in [64] make use of an implicit segmentation. 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 use 5,000 strings of digits but they did not specify the used data.

In the same vein, Britto Jr. et al in [47] 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 a implicit segmentation-based strategy and another considering just the recognition aspects of isolated digits. They use 12,802 strings of the NIST SD19 (hsf_7 series).

As stated before, Table I summarizes the recognition rates (RRs) with zero-rejection level. Although the publications shown in this table use the same database, comparison is possible only between Oliveira et al [91] and Britto Jr et al [47] due to they consider the same subset and samples of the NIST database. In such a case, Oliveira et al achieved better results based on their proposed explicit segmentation-based recognition strategy using neural classifiers.

In real applications, the reliability needs to be very high, especially for cheque or other financial document processing systems. It has been estimated that a system becomes commercially efficient only when the error rate is 1% or lower. However,

Table I

Recognition rates on NIST database reported in the literature

Authors	Classifier	String Length	RR (%)	Tested Strings	Database	Year
Ha et al [37]	MLP	2	96.2	981	NIST SD3	1998
		4	93.2	988		
Lee and Kim [64]	MLP	2	95.2	1,000	NIST	1999
		4	80.6	1,000		
Oliveira et al [91]	MLP	2	96.8	2,370	NIST SD19	2002
		4	93.3	2,345	(hsf_7 series)	
Britto Jr. et al [47]	HMM	2	94.8	2,370	NIST SD19	2002
		4	91.2	2,345	(hsf_7 series)	

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 system since the number of digits is fixed and known a priori. The performance of some bank check processing systems found in the literature are reported in Table II.

Table II

Performance on courtesy amount recognition

System	Classifier	RR (%)	Error (%)	Test Set	Database	Year
Lethelier et al [67]	RBF-TDNN	60.0	40.0	10,000	French cheques	1995
Lee et al [68]	HNN	72.7	27.3	121	Brazilian cheques	1997
Parascript [20]	Matching	53.0	1.0	5,000	American cheques	1997
Kaufmann et al [48]	MLP	79.3	20.7	1,500	Swiss cheques	2000
Oliveira et al [91]	MLP	57.1	0.5	500	Brazilian cheques	2002
Zhang et al [126]	MLP	69.8	1.0	400	Canadian cheques	2002

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 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% or lower) vary from 50 to 70%.

1.3 Handwritten Word Recognition

The majority of research in handwritten word recognition has integrated the lexicon as constraint to build lexicon-driven strategies in opposite to handwritten digit string recognition. The lexicon is a list of possible words that could possibly occur in an image. This lexicon is usually determined by the application at hand. This aims at decreasing the complexity of the problem since the ambiguity makes many characters unidentifiable without referring to context. Indeed, the same representation may lead to several interpretations, in the absence of the context which can be a lexicon or grammatical constraints. In Figure 3, for instance, the group of letters in the word "junho" shown inside the circle could be interpreted as 'june'", "fune", "fine", etc.

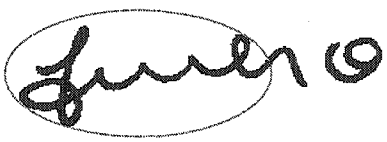


Figure 3 Ambiguity in handwritten words

Handwritten word techniques use either analytic or holistic approaches for training and recognition. In an analytic approach, the segmentation of words into segments that relate to characters is required. Nevertheless, this is not a trivial task due to problems such as touching, overlapping, or broken characters as stated before. Moreover, this operation is made more difficult because of the ambiguity encountered in handwritten words. Therefore, most successful analytical methods employ

segmentation-based recognition strategies where the segmentation can be explicit [2, 59, 121] or implicit [11, 30, 79].

In an holistic approach, word recognition is performed considering the whole word. In such a case, there is no attempt to split the word image into segments. Still, it is possible that the image would be segmented in order to produce a sequence of observations. Unlike analytical methods, holistic methods are constrained to applications with a small lexicon size as in bank cheque processing systems [17, 23, 38].

Most works in handwritten word recognition assume that the word has been already segmented by an algorithm appropriate to the application domain prior to being processed by word recognizer. Some of them are described as follows. The location and segmentation of handwritten words from their surroundings is a complex task for most real applications and it is discussed in Section 1.4.

Mohamed and Gader in [79] combine implicit segmentation-based continuous HMM and explicit segmentation-based dynamic programming (DP) techniques for unconstrained handwritten word recognition. The combination module uses differences in classifier capabilities to achieve significantly better performance. They report results on the BD test set of the CEDAR database that contains 317 city name images.

El Yacoubi et al in [121] design a system to recognize unconstrained handwritten words for large vocabularies. After preprocessing, a word image is divided explicitly into a sequence of segments and then two feature sets are extracted from the sequence of segments. The word models are made up of the concatenation of appropriate letter models and an HMM-based interpolation technique is used to optimally combine the two feature sets. They consider two rejection mechanisms depending on whether or not the word image is guaranteed to belong to the lexicon. The experiments are carried out on 4,313 French city name images manually localized on real mail envelopes.

Kim et al in [52] take an HMM-MLP hybrid system for recognizing cursive script words. They have designed explicit segmentation-based HMM and an holistic approach for the MLP. The main contributions of this work lie in the HMM-based approach and a new multiplication method of combining two distinct classifiers. Experiments are carried out on the handwritten cursive legal words of the CENPARMI database (English set). They use 2,482 word images for testing.

Arika et al in [2] propose an analytic scheme, which makes use of a sequence of segmentation and recognition algorithms, for cursive handwriting recognition. First, some global parameters, such as slant angle, baselines, and stroke width and height are estimated. Second, a segmentation method finds character segmentation paths by combining gray scale and binary information. Third, HMM is employed for shape recognition to label and rank the character candidates. The estimation of feature space information and HMM ranks are combined in a graph optimization problem for word-level recognition. The performance of the system is tested using 2,000 words of the database of Lancester-Oslo/Bergen corpus, which contains single author cursive handwriting.

The performance of some recent applications found in the literature, including those discussed above, are reported in Table III. The recognition rates are shown for different lexicon sizes at zero-rejection level. In such a case, it is very difficult to compare the results because of the experiments were conducted on different databases, different classes of words, and different number of testing samples. Generally, the larger the lexicon is, the results get less satisfactory. Besides the lexicon size, the results also depend on the ambiguity among the classes of words being considered and whether a system is writer-dependent or not (omni-writer). However, the results are helpful in order to illustrate the current state of art on word recognition.

Table III

Recent results on word recognition

Authors	Classifier	Lexicon Size	RR (%)	Test Set	Database	Comments
Oliveira Jr. et al [14]	MLP	12	87.2	1,200	Month words (POR)	UNC, OMNI
Xu et al [118]	HMM-MLP	29	85.3	2,063	Month words (FR/ENG)	UNC, OMNI
Knerr et al [55]	HMM-NN	30	92.9	40,000	LA words (FR)	CUR, OMNI
Guillevic et al [35]	HMM-k-NN	30	86.7	4,500	LA words (FR)	UNC, OMNI
Kim et al [52]	HMM-MLP	32	92.2	2,482	LA words (ENG)	CUR, OMNI
Freitas et al [24]	HMM	39	70.6	2,387	LA words (POR)	UNC, OMNI
Mohamed et al [79]	HMM-DP	100	89.3	317	City names (ENG)	UNC, OMNI
Kundu et al [59]	HMM	100	88.2	3,000	Postal words (ENG)	UNC, OMNI
El Yacoubi et al [121]	HMM	100	96.3	4,313	City names (FR)	UNC, OMNI
Bunke et al [6]	HMM	150	98.4	3,000	Words (ENG)	CUR, WD
El Yacoubi et al [121]	HMM	1,000	88.9	4,313	City names (FR)	UNC, OMNI
Arika et al [2]	HMM	1,000	90.8	2,000	Words (ENG)	CUR, WD

LA:Legal Amount, FR:French, POR:Portuguese, ENG:English
UNC:Unconstrained,CUR:Cursive, OMNI:Omni-writer, WD:Writer-Dependent

1.4 Handwritten Sentence Recognition

Recognition of numeral strings and words has been extensively studied in the literature. Only recently the recognition of a sentence composed of a sequence of words or different data types has been investigated. Some typical applications on sentence recognition are reading texts from pages [22, 71, 72], street names from postal addresses [50, 94, 95, 105, 123], processing of legal amounts [32, 48] and dates [109, 119] on cheques. In such applications, sentences are segmented into its constituent parts and make use of word and numeral string recognition techniques. In the literature two main different strategies of segmentation can be observed: segmentation-based and segmentation-free methods.

In the segmentation-based strategy, the most frequently used method splits explicitly a sentence into its parts usually based on spatial distance clues, i.e., the distances (gaps) between adjacent components [22, 70, 71, 100]. Notwithstanding, the segmentation based on gap metrics shows its limits rapidly when handwritten sentences do not have a uniform spacing. Besides, the gaps between components can not be easily estimated by a 1-dimensional metric. In order to overcome the foregoing problem, Kim et al in [50] propose an intelligent word segmentation method applied to street name images. This method incorporates the author's writing style in terms of spacing using an NN. In the same vein, Xu et al in [119] make use of different kinds of knowledge at different segmentation stages. The knowledge includes information on the writing style by using an NN to differentiate between numeric and alphabetic data, and syntactic and semantic constraints.

In this strategy, multiple hypotheses of segmentation are often considered in order to improve the performance of segmentation. The idea is to generate a list of segmentation hypotheses where each word candidate of each sentence hypothesis is submitted to recognition. The result of segmentation and linguistic analysis can be used to rank

each sentence hypothesis. In [50], an NN has been designed to split a sentence into segments and then an exhaustive combination of word segments is submitted to a word recognizer with given a lexicon. Statistical characteristics of character segmentation are used to limit the number of combinations. Further improvements of this methods was given in [95]. Favata et al in [22] take a two step approach to sentence recognition. After a sentence has been divided into segments, they group segments as a function of their spatial inter-relationship and then as a function of their response to recognizer. The output of the recognizer is a directed graph which contains multiple interpretations of the sentence. The second step searches this graph with a linguistic processor which ranks each path according to statistics of the language and certain measured statistical characteristics of the recognizer.

On the other hand, the segmentation-free strategy treats complete handwritten sentences as single units. Scagliola in [98] proposes a system for recognizing legal amounts on Italian cheques without word segmentation. The author demonstrates that when the *a priori* knowledge is described by a regular grammar, the phrase recognition can be performed by a simple extension of the dynamic programming algorithms used for single word recognition.

Marti et al in [72] present a system for reading unconstrained handwritten text. The kernel of the system is an HMM for handwriting recognition. This HMM is enhanced by a statistical language model incorporating linguistic information beyond the word level. The HMM has a hierarchical structure with character models at the lowest level. These models are concatenated to words and to whole sentences. Under such an architecture, the segmentation of a text line into individual words is not required, instead, it is obtained during the recognition process.

In the same manner, El Yacoubi et al in [123] build an embedded HMM network connecting street name HMMs with extraneous HMMs in order to conjointly locate

and recognize street name without the need of an *a priori* segmentation of the street line.

In Table IV, we show the performance of some applications on sentence recognition found in the literature. In such cases, part of the recognitions errors can come from the segmentation stage.

Table IV

Performance on sentence recognition

Authors	Application	Sentence Level			Word Level (zero-rejection level)		
		RR (%)	Error (%)	Test Set	RR (%)	Lexicon Size	Test Set
Marti et al [71]	Text lines (ENG)	-	-	-	73.4	412	3,899
Marti et al [72]	Text lines (ENG)	-	-	-	61.8	2,703	11,000
Kaufmann et al [48]	Legal amount (GER)	71.9	8.1	1,500	-	-	-
El Yacoubi et al [123]	Street names (FR)	45.6	1.5	2,856	-	-	-

FR:French, ENG:English, GER:German

1.4.1 Date Recognition on Cheques

With respect to cheque processing, a strong demand exists elsewhere. Millions of bank cheques issued from thousands of banks and financial institutions are daily used over the world for monetary transactions. Thus, a machine capable of reading bank cheques will have wide applications in banks and those companies where huge quantities of cheques have to be processed since most of the cheques are still processed manually by human operators. Usually, bank cheque processing is performed in big centers or at branch agencies which are equipped with fast scanners/sorters, archiving

systems, and videocoding terminals for operators who make data entry. Operators look at cheque images one by one and enter the cheque amounts.

Nowadays, there exist various works related to the processing of courtesy [20, 48, 67, 91, 108] and legal [32, 48] amount recognition on cheques. Nevertheless, very few studies have been made on the processing of date information on cheques [21, 117] even though the ability to automatically process handwritten dates is important in application environment where cheques cannot be cashed prior to the dates shown.

Table V shows the performance of two date recognition systems written on Canadian bank checks. In both cases, each date image can appear in any one of two patterns: $MM\ S\ DD\ S\ 19\ YY$ and $DD\ S\ MM\ S\ 19\ YY$ where MM, S, DD, and YY refer to month, separator, day and year sub-fields respectively. The month sub-field can be written in different data types (digits or words in English or French), while the day and year sub-fields only in digits. In these systems, the date image is first segmented into day, month, and year sub-fields and then specialized classifiers are used to recognize such sub-fields and their different data types. In such a case, the work presented by Xu in [117] is an extension of the work presented by Fan et al in [21].

Table V

Performance on date recognition

Authors	RR (%)	Error (%)	Test Set	Database	Year
Fan et al [21]	21.8	78.2	4,564	CENPARMI	1998
Xu [117]	44.5	5.2	1,197	CENPARMI (English set)	2002
Xu [117]	36.4	5.0	2,088	CENPARMI (French set)	2002

1.5 Discussion

By analyzing the state-of-the-art systems, we can observe that NNs have been successfully applied to handwritten numeral string recognition and HMMs widely used in word recognition.

Standard databases such as NIST, CENPARMI, and CEDAR are being extensively used for the evaluation of a method. By using standard databases, meaningful comparison of performances between recognition algorithms has become possible. Nevertheless, we have seen that it is difficult to carry out a deeper analysis since different databases, different number of testing samples, and different classes are used. Sometimes the authors use just one part of the database even when the entire set is available.

The literature indicates recognition rates close to 99% for handwritten isolated digits. When the topic comes to numerical string recognition, the performances go dramatically down due to problems such as touching digits, overlapping, and unknown number of digits. In this context, almost no contextual information is available in opposite to handwritten word recognition systems.

We have seen that holistic methods have usually been used for the recognition of handwritten words and applications with a small lexicon size. However, an example of an holistic approach for the recognition of touching-digit pairs can be found in [114]. Although analytical methods requires the segmentation of a word into segments, they have advantages over global ones. The first is that for a given learning set, is more reliable to train a small set of units such as characters than whole words. Indeed, the frequency of each word is far lower than the frequency of its letters, which are shared by all the words of the lexicon. Furthermore, global methods do not satisfy the portability criterion, since for each new application, the set of words of the associated lexicon must be trained.

Regarding segmentation-based recognition strategies, explicit methods have to face touching characters usually through heuristic-based algorithms, which usually are time-consuming and difficult to define. For this reason, several researchers have been investigated how to avoid it by using implicit segmentation. But due to the bi-dimensional nature of images and the overlap between characters, implicit methods are less efficient here than in on-line handwriting recognition. Indeed, vertical sampling loses the sequential aspect of the strokes, which is better represented by explicit methods. For instance, we can cite [37, 91] that surpass all methods based on implicit segmentation we have found in the literature [47, 49, 64, 73]. Nevertheless, implicit methods complement explicit ones and are particulary efficient in dealing with touching characters, for which it is hard to find regularly explicit segmentation points.

The recognition of handwritten words and strings of digits are present in almost every application involving handwriting recognition, for instance, such as postal address reading, processing of tax and census forms, and amounts and dates on cheques. Considering bank check systems, the literature indicates various studies on processing of amounts. Notwithstanding, we have observed that very few papers have been published on date recognition. Date recognition systems is a typical application on sentence recognition. This kind of application requires the segmentation of a sentence into parts, which is an important stage since it affects the performance on sentence recognition. Most of segmentation methods are based on an analysis of the geometric relationship of adjacent components in an image. However, the main drawback of this strategy is when the correct segmentation does not fit with the pre-defined rules of the segmenter, e.g., handwritten sentences that do not have a uniform spacing. In order to overcome this problem, some researchers have used segmentation-free methods where the segmentation is performed through the recognition process.

Although various studies have been done in the last decade and a variety of recognition techniques have emerged, 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. Nevertheless, we have seen that important contributions on this subject have been made.

1.6 Summary

In this chapter we have presented a state of the art of the main topics related to our research: handwritten numerical string recognition, handwritten word recognition, and handwritten sentence recognition. We have seen that several and important contributions have been made in these fields. Some recent works have been briefly described in terms of types of classifiers, test databases, and results. Different approaches for recognizing strings of digits, words, and sentences have also been presented and discussed. In addition, the performance of some date recognition systems has been reported. In the next chapter we will introduce our system with a brief overview and we will present the definitions related to such a system.

CHAPTER 2

SYSTEM OVERVIEW AND DEFINITIONS

The purpose of this chapter is to provide a brief overview of the date recognition system and present all the definitions related to the system.

2.1 System Overview

The date recognition system discussed in this thesis is designed to deal with unconstrained handwritten dates written on Brazilian bank cheques by multiple-writers. The system receives a 256-grey level date image as input. It takes an HMM-based strategy for separating the date sub-fields since we have seen that this is a difficult task without resorting to recognition. After the date sub-fields have been identified through the HMMs we can recognize the three obligatory ones (day, month and year) using specialized classifiers according to their respective data types which are known. In such cases, an MLP approach has been used to deal with strings of digits (day and year) and an HMM strategy to recognize and verify words (month). The system also contains a final decision module which makes an accept/rejection decision.

Figure 4 shows the modules of the system which are: preprocessing, segmentation into sub-fields, digit string recognition, word recognition and verification, and final decision. In the following chapters we give a more in-depth description of these modules composing our date recognition system. In Chapters 3 and 4, we describe the parts of preprocessing and segmentation that correspond the modules of preprocessing and segmentation into sub-fields respectively. In Chapter 5, we report the part of recognition which includes the modules of digit string recognition, word recognition and verification, and final decision. Finally, in Chapter 6, we show the results of experiments carried out.

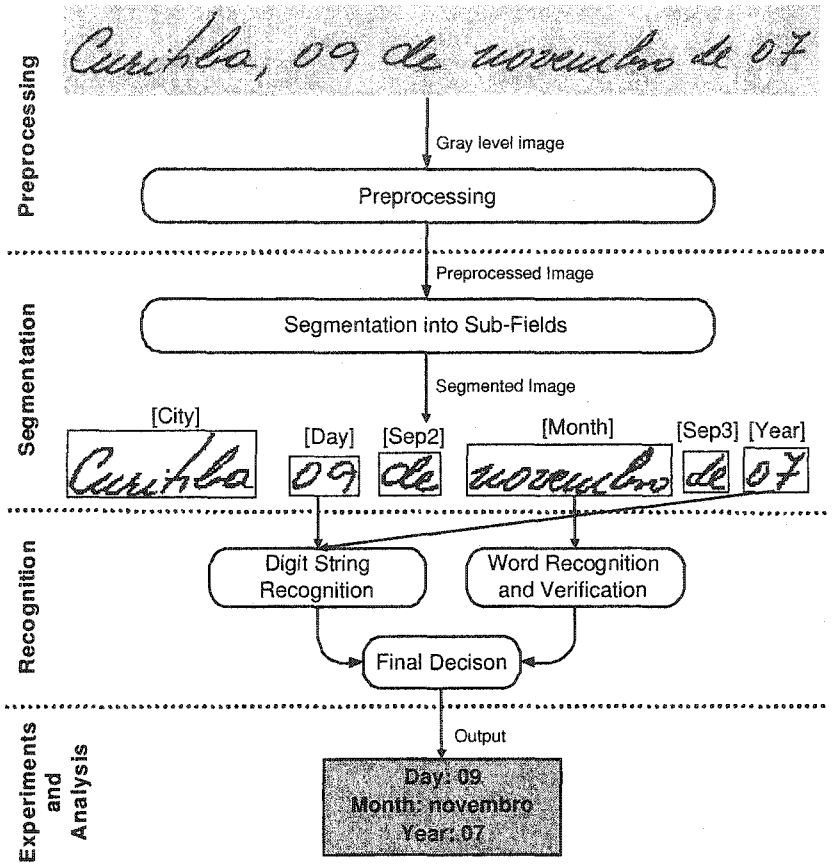


Figure 4 Block diagram of the date recognition system

2.2 Definitions

In this section the definitions related to the date recognition system are introduced. Section 2.2.1 presents the concept of meta-classes of digits. Sections 2.2.2 and 2.2.3 describe the HMMs and MLPs used in the HMM-MLP hybrid system respectively. In Section 2.2.4 we discuss the verification and its different levels as well.

2.2.1 Meta-Classes of Digits

We have defined four meta-classes of digits ($C_{0,1,2,3}$, $C_{1,2}$, $C_{0,9}$ and $C_{0,1,2,9}$) based on the classes of digits present in each position of 1- or 2-digit day and 2- or 4-digit year as shown in Figure 5. A meta-class of digits is formed by the union of two or more

of the original classes in order to break down the complexity of their segmentation and the following recognition processes [29, 58]. This is possible because the lexicon of the day and year is known and limited. While the class of digits C_{0-9} deals with the 10 numerical classes, the meta-classes of digits represented by the shaded boxes in Figure 5 work with specific classes of digits. The objective is to build HMMs based on these meta-classes in order to reduce the lexicon size of the day and year and improve the precision of their segmentation. Besides, it can be applied to digit string recognition to improve the recognition results since very often confusions between some classes of digits can be avoided (e.g., 4 and 9, 8 and 0). We can observe the efficiency of this strategy in Section 6.2.

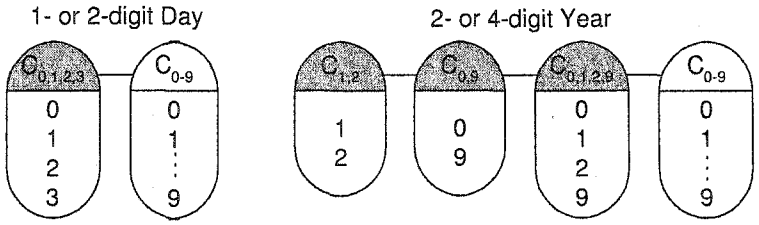


Figure 5 Classes of digits present in each position of 1- or 2-digit day and 2- or 4-digit year

2.2.2 Hidden Markov Models

HMMs are finite stochastic processes that have been proven to be one of the most powerful tools for modeling speech and later on a wide variety of other real-world signals. These probabilistic models offer many desirable properties for modeling characters, words, or sentences. One of the most important properties is the existence of efficient algorithms to automatically train the models without any need of labeling presegmented data. This constitutes a key feature of the approach we developed for segmentation into sub-fields, word recognition and verification. The algorithms for training and recognition the HMMs we have used in our system are reported in

Appendix 2. In our study, we are dealing with the discrete HMMs. Furthermore, we are segmenting explicitly each image into segments (graphemes) based on cut rules for feature extraction.

As pointed out earlier, a date image can consist of city name, separators (Sep1, Sep2, and Sep3), day, month and year sub-fields. Due to the huge size of the vocabulary of dates, our elementary HMMs are built at city, space, and character levels. Thus, the word, date, and its sub-field models are formed by the concatenation of appropriate elementary HMMs. In the following sections we explain the justification behind the proposed models. In Section 2.2.2.1 we present the elementary HMMs used in the system and in Section 2.2.2.2 we describe the word models employed in word recognition and verification. Section 2.2.2.3 reports the date and its sub-field models used in segmentation. Finally, the decision rules we have adopted in segmentation and word recognition are addressed in Section 2.2.2.4.

2.2.2.1 Elementary Models

In this section, we introduce the topologies of the elementary HMMs used in the system. Such models are built at city, space and character levels and they are employed in the following modules of the system: segmentation into sub-fields (SSF), word recognition (WR) and verification (WV).

The model depicted in Figure 6(a) was adopted to represent all the city names and noise (e.g., Sep1). The objective of this model is to identify them in the sentence.

A left-right topology has been used to model spaces and characters such as digits and letters. The topology of the space models is very simple as shown in Figure 6(b). It consists of two states linked by two transitions that encode a space (transition t_{01}) or no space (transition $t_{01} = \Phi$). Besides the inter-sub-field model, we have defined two more HMMs that model the intra-digit and intra-word spaces.

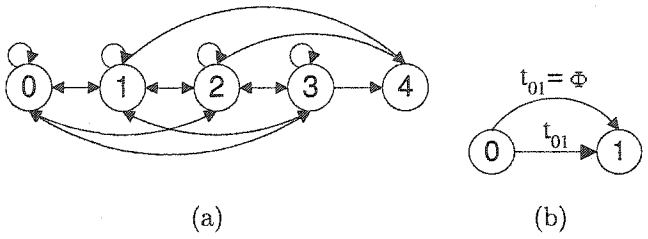


Figure 6 (a) Topology of the city model and (b) Topology of the space models

Two topologies of character models were chosen based on the output of our segmentation algorithm that may produce a correct segmentation of a character, a character under-segmentation or a character over-segmentation into two, three, or four graphemes depending on each character (a detailed description of the segmentation algorithm is given in Section 4.2.2). Only the characters "M", "m", "E", "F", and "2" may produce an over-segmentation into four graphemes. In order to cope with these configurations of segmentations, we have designed topologies with three different paths leading from the initial state to the final state. Figures 7(a) and 7(b) show examples of both topologies.

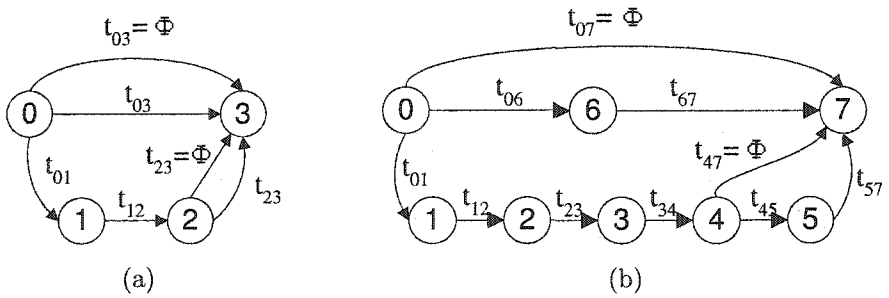


Figure 7 (a) and (b) Topologies of the character models

The model in Figure 7(a) is employed in the segmentation and word recognition modules. In such a topology, the transition t_{03} either (a) models under-segmentation and emits the null symbol Φ, or (b) models a character correctly and emits a symbol. The transitions t_{01}, t_{12} and t_{23} model character segmentation into two or three

graphemes. The model in Figure 7(b) is used in the word verification module and it is based on the previous one, but in this case the model has transitions (t_{12}, t_{34}, t_{57} and t_{67} of Figure 7(b)) that encode the way that the graphemes are linked together, e.g., if there is a segmentation point we can encode whether its vertical position is closer to the upper or lower base lines. Considering uppercase and lowercase letters, we need 40 models since the month alphabet is reduced to 20 letter classes and we are not considering the unused ones. Thus, regarding the two topologies, we have 80 HMMs. For the digit case, we have defined five HMMs based on the topology of the character model shown in Figure 7(a). One model considers the 10 numerical classes (M_{0-9}) and the other ones are defined based on the meta-classes of digits (e.g., the $M_{1,2}$ model copes with the class of digits $C_{1,2}$ and so forth) (see Figure 5).

Therefore, considering one city, three space, 80 letter and five digit models, the system takes into consideration 89 elementary HMMs which were trained using the Baum-Welch algorithm with the Cross-Validation procedure (see Appendix 2). Table VI describes those models as well as where they are used in the system.

2.2.2.2 Word Models

Basically, the word models consist of the concatenation of the foregoing topologies. Since no information on word recognition is available on the handwritten style (uppercase, lowercase), we propose to use the architecture of the word model shown in Figure 8(a) which can deal with the problem of mixed handwritten words. This architecture consists of an initial state (I) and a final state (F), two elementary letter HMMs in parallel and four elementary intra-word space HMMs linked by four transitions: two uppercase letters (UU), two lowercase letters (LL), one uppercase letter followed by one lowercase letter (UL), and one lowercase letter followed by one uppercase-letter (LU). The probabilities of these transitions are estimated by their occurrence frequency in the training set. In the same manner, the probabilities

Table VI

Description of the elementary HMMs used in the system

Model	Topology	Classes	Usage
M_{city}	Figure 6(a)	Curitiba and Ctba	SSF
M_{space}	Figure 6(b)	Inter-sub-field space	SSF
M_{space}^{digit}	Figure 6(b)	Intra-digit space	SSF
M_{space}^{word}	Figure 6(b)	Intra-word space	SSF and WR
M_a	Figure 7(a)	a	SSF and WR
M_A	Figure 7(a)	A	SSF and WR
⋮	⋮	⋮	⋱
M_v	Figure 7(a)	v	SSF and WR
M_V	Figure 7(a)	V	SSF and WR
$M_{a'}$	Figure 7(b)	a	WV
$M_{A'}$	Figure 7(b)	A	WV
⋮	⋮	⋮	⋱
$M_{v'}$	Figure 7(b)	v	WV
$M_{V'}$	Figure 7(b)	V	WV
$M_{0,1,2,3}$	Figure 7(a)	0,1,2, and 3 ($C_{0,1,2,3}$)	SSF
M_{0-9}	Figure 7(a)	0-9 (C_{0-9})	SSF
$M_{1,2}$	Figure 7(a)	1 and 2 ($C_{1,2}$)	SSF
$M_{0,9}$	Figure 7(a)	0 and 9 ($C_{0,9}$)	SSF
$M_{0,1,2,9}$	Figure 7(a)	0,1,2, and 9 ($C_{0,1,2,9}$)	SSF

of beginning a word by an uppercase-letter (0U) or a lowercase letter (0L) are also estimated in the training set.

Figure 8(a) shows the architecture of the word models adopted on word recognition, while Figure 8(b) illustrates the architecture used for word verification. We can observe that the architecture shown in Figure 8(b) diverges only in two aspects: it does not consider the space models and its elementary HMMs contains transitions that encode the way that the graphemes are linked together.

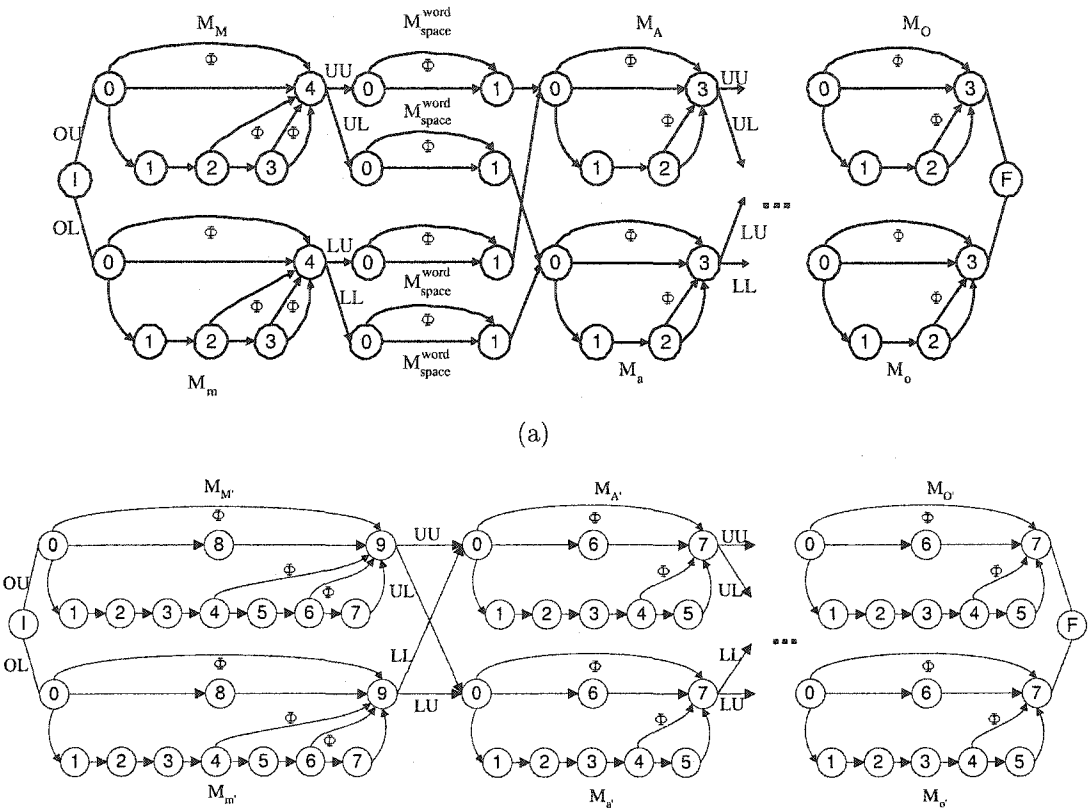


Figure 8 Word models of class "Maio" (May): (a) Recognition and (b) Verification

2.2.2.3 Date and its Sub-Field Models

Based on the fact that during the process of segmenting the image into sub-fields, we do not want to recognize the content of each sub-field, i.e., their identification is sufficient, the date model has been simplified by reducing the date lexicon size to improve the segmentation results.

This is performed by considering one model to represent all the city names and the first separator (Sep1), and the same "De" model to represent the second and third separators (Sep2 and Sep3). Besides, the concept of meta-classes of digits described in Section 2.2.1 has been used to build the day and year models. Figures 9(a) and

9(b) respectively show the date models without and with lexicon size reduction of an image with all the sub-fields.

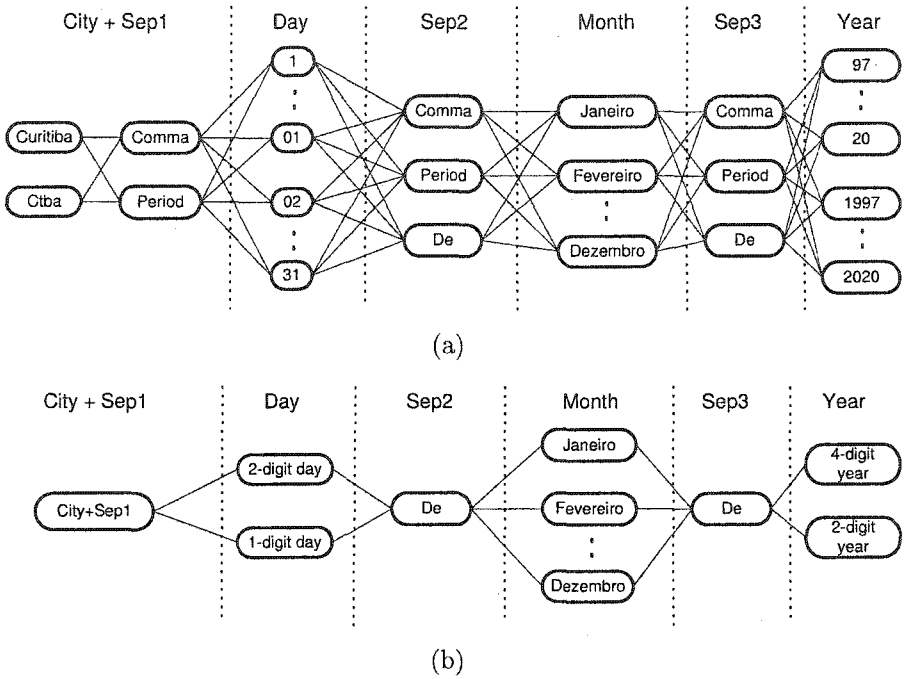


Figure 9 Date models: (a) Without lexicon size reduction and (b) With lexicon size reduction

We can observe from Figure 9(b) that the lexicon size of month words has not been reduced since it can help to identify the other sub-fields. Besides, the date model shown in Figure 9(b) represents an image with all the sub-fields. Considering that some sub-fields are optional and there is one model for each sub-field, we have eight possible date models which are created by concatenating appropriate sub-field models and by considering the inter-sub-field space as a model. Such date models act as a segmentation engine of the date into sub-fields. The date sub-field models are described in Table VII and the eight date models are presented in Table VIII where M_{space} stands for the inter-sub-field space model defined in Table VI.

Table VII

Description of the date sub-field models

Model	Description
M_{city}	City model (plus Sep1)
M_{day}	1- or 2-digit day model
M_{de}	"De" separator model (Sep2 and Sep3)
M_{month}	Month model (12 word classes)
M_{year}	2- or 4-digit year model

Table VIII

Date models

No.	Date Model										
1	M_{city}	M_{space}	M_{day}	M_{space}	M_{de}	M_{space}	M_{month}	M_{space}	M_{de}	M_{space}	M_{year}
2			M_{day}	M_{space}	M_{de}	M_{space}	M_{month}	M_{space}	M_{de}	M_{space}	M_{year}
3	M_{city}	M_{space}	M_{day}	M_{space}	M_{de}	M_{space}	M_{month}	M_{space}			M_{year}
4			M_{day}	M_{space}	M_{de}	M_{space}	M_{month}	M_{space}			M_{year}
5	M_{city}	M_{space}	M_{day}	M_{space}			M_{month}	M_{space}	M_{de}	M_{space}	M_{year}
6			M_{day}	M_{space}			M_{month}	M_{space}	M_{de}	M_{space}	M_{year}
7	M_{city}	M_{space}	M_{day}	M_{space}			M_{month}	M_{space}			M_{year}
8			M_{day}	M_{space}			M_{month}	M_{space}			M_{year}

Basically, the month model consists of an initial state (I), a final state (F), and 12 models in parallel that represent the 12 word classes. The architecture of each word model and the "De" separator model are the same shown in Figure 8 (a). The day model consists of an initial state (I), a final state (F), and the 2-digit day model in parallel with the 1-digit day model as shown in Figure 10. The 2-digit day model is formed by concatenating the following elementary models shown in Table VI: $M_{0,1,2,3}$, M_{space}^{digit}, and M_{0-9}. The 1-digit day model is related to the M_{0-9} model. The probabilities of being 1- (1D) or 2-digit (2D) day are estimated in the training set of the date database. The year model is built in the same manner.

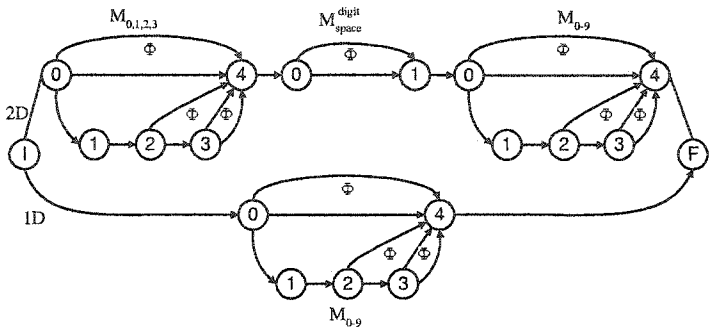


Figure 10 Day model for 1- or 2-digit strings

2.2.2.4 Decision Rules

The decision rules used in the segmentation into-sub-fields and word recognition are based on the maximization of the *a posteriori* probability that a pattern x has generated an unknown observation sequence O, such as:

$$P(\overline{x}|O) = \max_{x} P(x|O) \tag{2.1}$$

Applying Bayes rule, we obtain the fundamental equation of pattern recognition:

$$P(x|O) = \frac{P(O|x)P(x)}{P(O)} \tag{2.2}$$

Since $P(O)$ does not depend on x, recognition becomes equivalent to maximizing the joint probability:

$$P(x, O) = P(O|x)P(x) \tag{2.3}$$

where $P(x)$ is the *a priori* probability of the pattern x (class distribution in the training set). In this case, we prefer to consider $P(x)$ as equiprobable due to the small date training set. Besides, the distribution of each word class is very similar to the other ones. The estimation of $P(O|x)$ requires a probabilistic model that accounts for the shape variation of x. We assume that such a model consists of a global Markov created by concatenating elementary HMMs.

2.2.3 Neural Networks

Although many types of neural networks can be used for classification purposes [69], we opted for an MLP which is the most widely studied and used neural network classifier. Moreover, MLPs are efficient tools for learning large databases [61]. Therefore, all classifiers presented in this work are MLPs trained with the gradient descent applied to a sum-of-squares error function [5]. The transfer function employed is the familiar sigmoid function.

In order to monitor the generalization performance during learning and terminate the algorithm when the improvement levels off, we have used the method of cross-validation. Such a method takes into account a validation set, which is not used for learning, to measure the generalization performance of the network. During learning, the performance of the network on the training set will continue to improve, but its performance on the validation set will only improve to the point where the network starts to overfit the training set, that the learning algorithm is terminated.

Let Ω be a pattern space which consists of N_S mutually exclusive sets $\Omega = \omega_1 \cup \ldots \cup \omega_{N_S}$, each of ω_i, $i \in \Lambda = 1, \ldots, N_S$ representing a set of specified patterns called a class. Let x be an input pattern that should be assigned to one of the N_S existing classes. e means the classifier and $e(x) = m^i(x)|\forall i(1 \leq i \leq N_S)$ 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 neural classifiers of the system.

We have adopted one 10-numerical-class classifier (e_{0-9}) and four MLPs specialized in the lexicon of the four meta-classes of digits (e.g., the $e_{1,2}$ classifier works with the lexicon of the class of digits $C_{1,2}$ and so on) (see Figure 5). In this case, the digit string recognition module will determine which of these classifiers will be used according to the sub-field (day or year) and its number of digits obtained by the segmentation module. This scheme aims at reducing the lexicon size on digit string recognition to increase the recognition results (see Chapter 6). Table IX details these classifiers which have one hidden layer where the units of input and output are fully connected with the units of the hidden layer. The number of hidden units used, which were determined empirically, are also described in this table. The learning rate and the momentum term were set at high values in the beginning to make the weights quickly fit the long ravines in the weight space, then these parameters were reduced several times according to the number of iterations to enable the weights fit the sharp curvatures.

Table IX

Description of the MLPs used in the system

Classifier	Hidden Units	Classes
$e_{0,1,2,3}$	70	0,1,2 and 3 ($C_{0,1,2,3}$)
e_{0-9}	80	0-9 (C_{0-9})
$e_{0,1,2,9}$	70	0,1,2 and 9 ($C_{0,1,2,9}$)
$e_{0,9}$	70	0 and 9 ($C_{0,9}$)
$e_{1,2}$	70	1 and 2 ($C_{1,2}$)

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 all classifiers is defined as:

$$e(x) = \max_{i \in \Lambda} m^i(x) \tag{2.4}$$

2.2.4 Levels of Verification

Takahashi and Griffin in [110] 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, Oliveira et al in [91] introduce the concepts of high and low-level verifications. The idea of the high-level verification is to confirm or deny the hypotheses produced by the classifier by recognizing them. On the other hand, the low-level verification does not recognize a hypothesis, but rather determines whether a hypothesis generated by the classifier is valid or not.

Based on these concepts, we propose to use an absolute high-level word verifier in order to improve the recognition rate and reliability of the system. The objective of an absolute high-level word verifier is to re-rank the N best hypotheses of month word recognition using a specialized word classifier. The word recognizer takes into account both segmentation and recognition aspects, while the verifier considers just the recognition aspects. This verifier deals with the loss in terms of recognition performance brought by the word recognition module. A similar strategy was adopted by Britto Jr. et al in [46]. Section 5.2 presents more details on this verifier.

2.3 Summary

In this chapter all definitions used by the system were presented. We have seen that different classifiers were used to cope with the different data types. A neural approach has been adopted to decipher strings of digits (day and year) and a Markovian strategy to recognize and verify words (month). In addition, the concepts about meta-classes of digits, which is an underpinning idea in the strategy we have devel-

oped for date segmentation and digit string recognition, were introduced. We also discuss the verification and its different levels as well. In the next chapter the first module depicted in Figure 4 is described in detail.

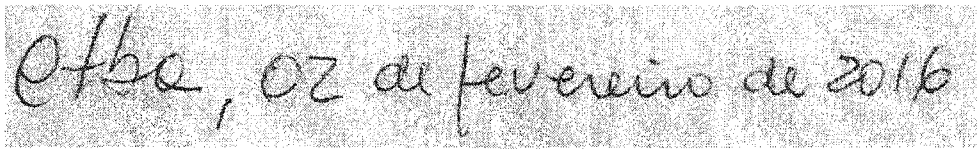
CHAPTER 3

PREPROCESSING

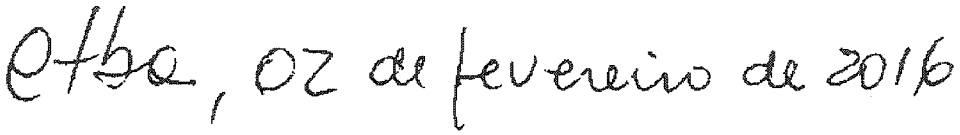
In this chapter we describe the first part of the system, which involves the preprocessing module. The goal of preprocessing in handwriting recognition systems is to reduce irrelevant information such as noise and intra-class variability (e.g., slant correction) that increase the task complexity in a writer-independent recognizer. In our system, the following preprocessing steps have been implemented and applied to all date images, which were acquired at 300 DPI (Dots Per Inch) and 256 gray levels (a detailed description of such database is given in Appendix 1): binarization, slant correction, and smoothing. In such cases, we have used well known techniques which are described throughout this chapter. For this reason, we do not devote much attention to such subjects. Besides, although the preprocessing is an important part of a recognition system, our main contributions lie in the segmentation and recognition aspects as mentioned before as well as the optimization of the system.

3.1 Binarization

In order to reduce data storage and increase processing speed, it is often desirable to represent gray-scale or color images as binary images. In order to accomplish this task, we have used Otsu's method [93] which is based on a global thresholding technique. In such a case, one threshold value is picked for the entire image. This is a well-known method that has been widely used in the literature [6, 79]. It is based solely on the statistical distribution of gray values. For each possible threshold value, Otsu's method divides the histogram into two classes and calculates ratio of between class variance to the variance of the total image. The optimum threshold is the one that maximizes the variance between the two classes. Figure 11 shows the result of Otsu's technique.



(a)



(b)

Figure 11 Result of Otsu's technique: (a) Gray level image and (b) Image after binarization

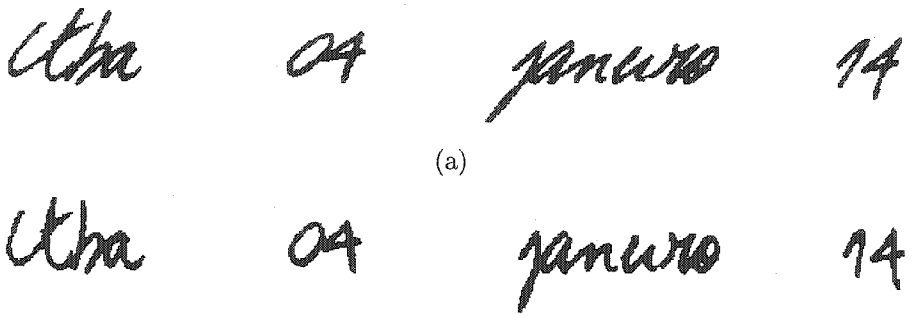
3.2 Slant Correction

The technique we have used is based on primary contours of the connected components detected in a date image (the contour detection is described in Section 3.2.1). Such a technique is based on [120] and it works as follows. The primary contour is divided into segments with the same length as the stroke thickness (see Section 3.2.2). The global slant (θ) is obtained by the median slant of all segments and its correction is made using a shear transformation defined in Equation 3.1. Figure 12 illustrates the result of such a slant correction algorithm.

$$\begin{cases} x' = x - (y \times \tan(\theta)) \\ y' = y \end{cases} \tag{3.1}$$

3.2.1 Contour Detection

Firstly, the connected components of an image are detected and then their contours are determined by following the 8-Freeman direction. In our system, the contours can be classified as primary or secondary [66].



(a)

(b)

Figure 12 Result of the slant correction algorithm: (a) Original image and (b) Image after slant correction

The primary contours correspond to the external contours of the connect components that are completely or partially located in the median region between the upper and lower baselines of a date image. Such lines are determined from the median line, which is the ordinate that corresponds to the highest peak of the horizontal transition histogram as shown in Figure 13. If the histogram contains two or more equal peaks we keep the one with the highest ordinate, taking into consideration that the reading of an image starts in the upper left corner and continues downwards column by column. To find the upper (lower) baseline we search for the first ordinate of the histogram above (below) the median line that has 50% of the density of the highest peak (see Figure 13(b)). In both cases, the threshold was chosen based on experimentation.

The primary contours are composed of upper and lower contours as shown in Figures 14(c) and 14(d) respectively. The upper contour of a connect component begins (ends) in the leftmost (rightmost) point in the median region. If there are two or more points located in the same abscissa we maintain the one with the highest ordinate. The lower contour is the remaining part of the primary contour. The objective of detecting the foregoing points in the median region is to obtain more precise upper and lower contours (see Figure 15) since such contours are relevant information used in segmentation (Chapter 4).

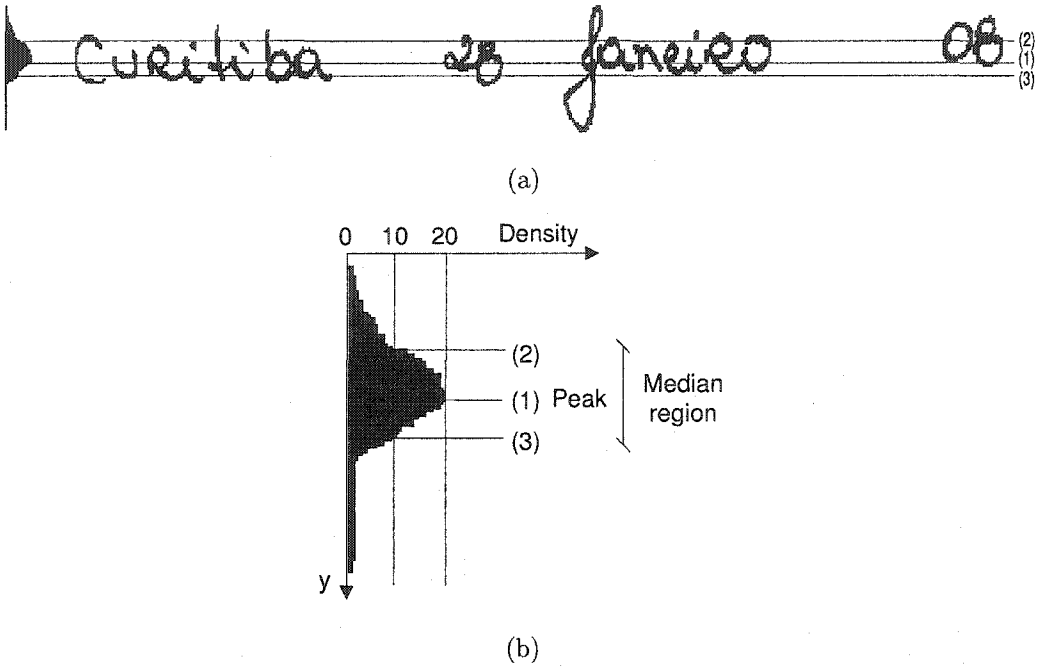


Figure 13 (a) and (b) Horizontal transition histogram ((1) median line, (2) upper baseline, and (3) lower baseline)

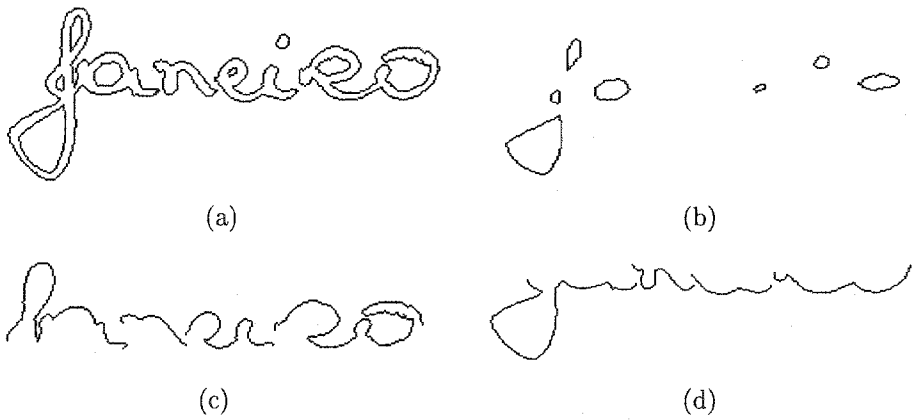


Figure 14 (a) Contour detection of word "Janeiro", (b) Secondary contours, (c) Upper primary contours, and (d) Lower primary contours

The secondary contours are situated outside of the median region or they are the internal contours of the primary ones (loops) (see Figure 14(b)). The loops are an important piece of information used throughout the system.

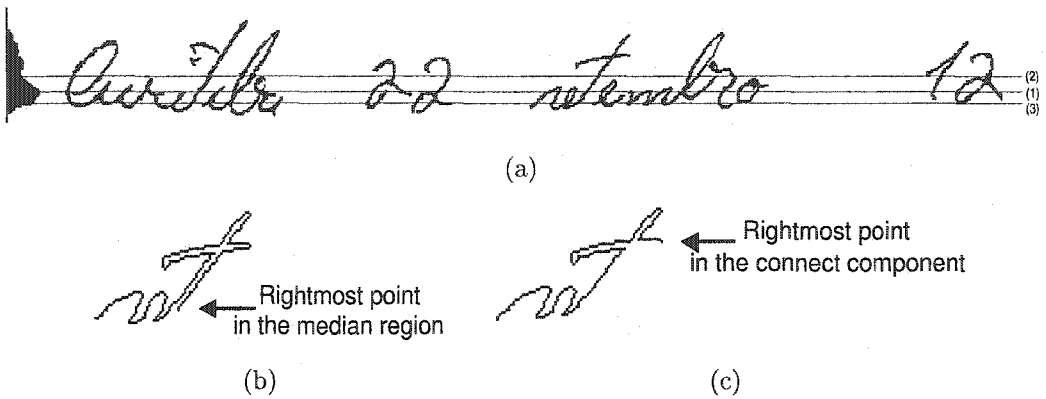


Figure 15 (a) Horizontal transition histogram, (b) A more precise detection of the upper contour of "set", and (c) A less precise detection of the upper contour of "set"

3.2.2 Stroke Thickness

The stroke thickness of an image is obtained from the vertical histogram. For each column we analysis the stroke thicknesses and their frequency. Thus, after evaluating all the columns of an image we pick the thickness which had the highest occurrence. Figure 16 illustrates an image which contains the "a" character and its histogram. In such a case, the stroke thickness chosen was five.

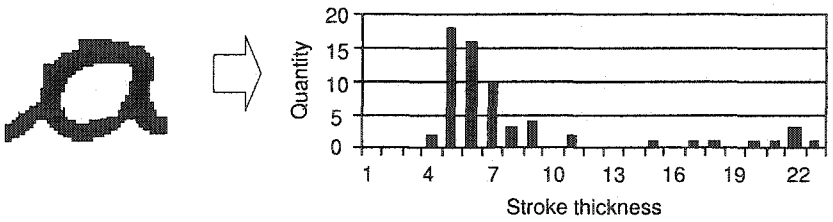


Figure 16 Vertical histogram of stroke thickness

3.3 Smoothing

After slant correction, a smoothing operation is done to regularize the edges in the image and to remove small bits of noise. This technique is based on [107] and it works as follows. A 3×3 mask (see Figure 17) 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 center 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 four times for each target pixel, once for each possible rotation of the mask.

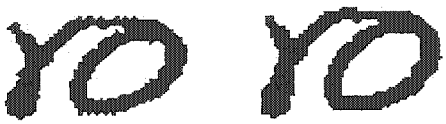
X	X	X
=	T	=
=	=	=

Figure 17 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 completely eroded. 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. Figure 17 shows the result of this smoothing algorithm.

3.4 Summary

In this chapter we have presented the first module of our system, which is composed of the following preprocessing techniques: binarization, slant correction, and smoothing. In such cases, we have used well known techniques found in the literature. This



(a) (b)

Figure 18 Result of the smoothing operation: (a) Original image and (b) Image after smoothing

module aims at producing data that are easy for the system to operate accurately. In the next chapter we will discuss the strategy we have used to segment a date image into sub-fields.

CHAPTER 4

SEGMENTATION

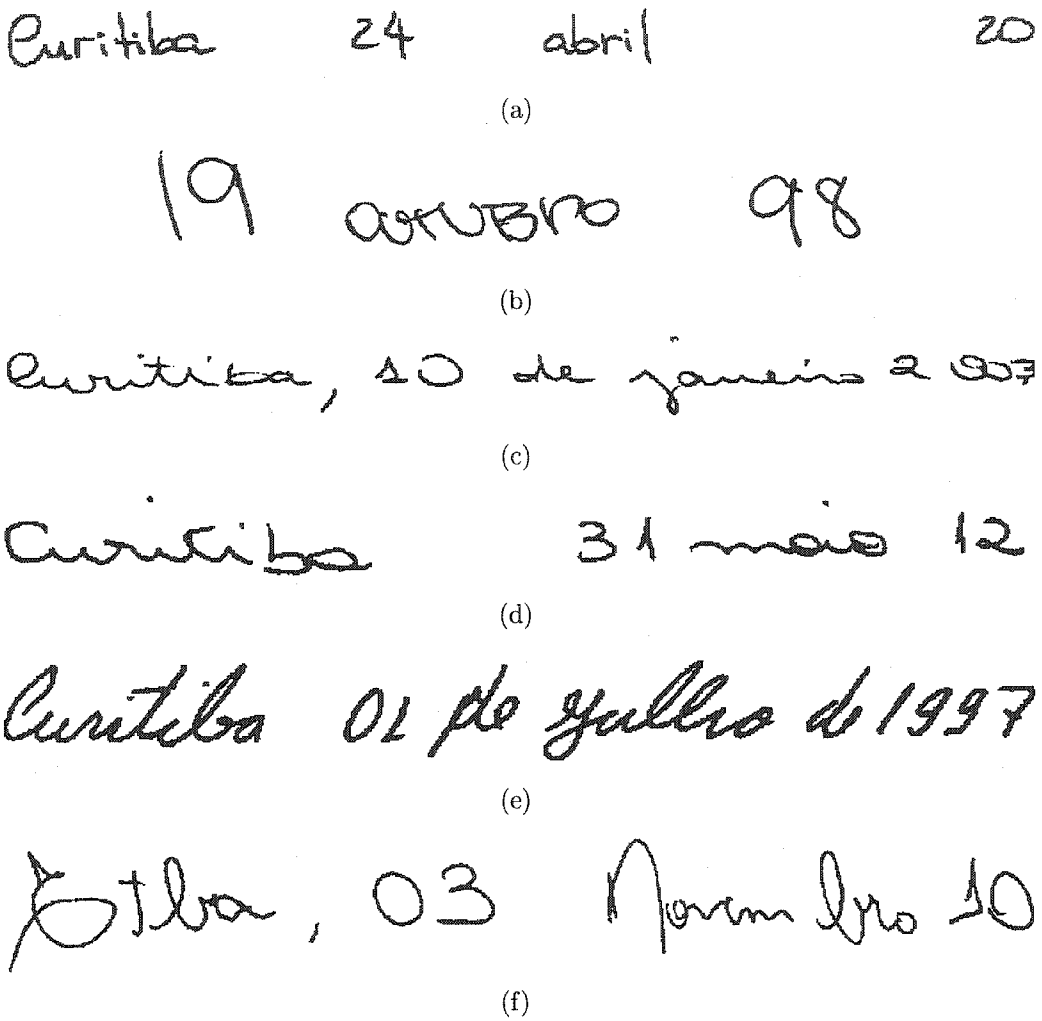
After preprocessing, the next step is the segmentation. The purpose of this module is to divide the whole date image into sub-fields and at the same time identify each sub-field. Thus, the day, month, and year sub-fields can be processed by the system using specialized classifiers. In this chapter we present the strategy we have developed for segmentation and the steps necessary to perform this task.

4.1 Segmentation Strategy

We have seen that the most frequently used segmentation strategy splits explicitly a sentence into its parts usually based on spatial distance clues. However, we have seen that the main drawback of this strategy is when the correct segmentation does not fit with the pre-defined rules of the segmenter, e.g., handwritten sentences that do not have a uniform spacing.

In our application, the handwritten dates present a great variability in terms of spaces as shown in Figure 30. In the case of Figures 19(a) and 19(b), the date inter-sub-field spaces are well defined and easy to detect. But it will be very difficult to correctly detect all of them in Figures 19(c), 19(d), 19(e), and 19(f) without resorting to recognition due to the similarities among some inter-sub-field and intra-sub-field spaces. For example, in Figure 19(c) the intra-sub-field space between "2" and "003" is almost the same as the inter-sub-field spaces between "de" and "janeiro" or "janeiro" and "2003".

Hence, we opted by using a segmentation-free strategy. We are considering an HMM-based approach to perform the date segmentation into sub-fields. Since we are dealing with the discrete HMMs, each preprocessed date image must be transformed as a



(a)

(b)

(c)

(d)

(e)

(f)

Figure 19 Samples of handwritten dates on Brazilian bank cheques

whole into a sequence of observations. This is done by performing three steps: reference line detection, segmentation, and feature extraction. The first step is required to carry out the other ones. The segmentation step splits explicitly a date image into a sequence of segments (graphemes), each of which consists of a full character, a piece of a character, or more than a character. Then, two feature sets are extracted from the sequence of graphemes and combined using a multiple-codebook-based HMM.

4.2 Description of the Segmentation Sub-Modules

In Sections 4.2.1, 4.2.2, and 4.2.3 we describe how we detect the reference lines, the grapheme segmentation algorithm, and the two feature sets respectively. Besides, we report in Section 4.2.4 the mechanism we have used to train the elementary models employed at this stage. Finally, in Section 4.2.5 we describe how the segmentation is performed.

4.2.1 Reference Line Detection

Considering a cursive word, we can define five reference lines (e.g., the word "Junho" in Figure 20(c)): (1) median line, (2) upper baseline, (3) lower baseline, (4) upper line, and (5) lower line. We can observe from this figure that such lines delimit the upper, median, and lower regions. Usually, the median region contains the lowercase letters and the upper and lower regions possess the ascenders and descenders.

Since we are dealing with a sentence composed of different data types such as strings of digits and words, we split a date image into parts in order to detect the reference lines in each sub-image and improve the precision of their estimation. At this level, there is no problem if a sub-image does not correspond to a date sub-field, e.g., the second sub-image of Figure 20(b) contains the day ("05") and the "de" separator. The steps to determine the reference lines are shown in Figure 20 and they are described as follows.

The sub-images are located by looking for the more relevant spaces present in the date, e.g., the three spaces (Space 1, 2, and 3) shown in Figure 20(b). In order to perform this search, we first detect the spaces between the connected components that are completely or partially situated in the median region between the upper and lower baselines of a date image. In such a case, the median region is determined through the horizontal transition histogram (see Figure 20(a)). Basically, this is

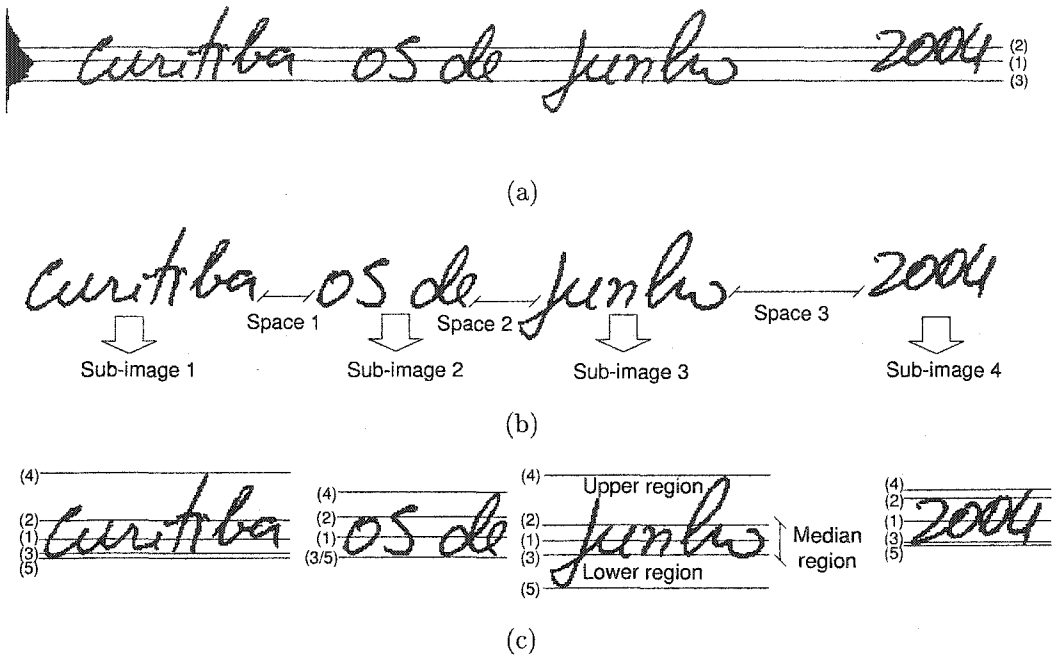


Figure 20 (a) Horizontal transition histogram, (b) Location of the date sub-images, and (c) Detection of the reference lines in each sub-image ((1) median line, (2) upper baseline, (3) lower baseline, (4) upper line, and (5) lower line)

performed in the same manner as we did to identify the primary contours (the contour detection is described in Section 3.2.1). After that, we compute the average space from all detected spaces and those ones greater than it are considered as the more relevant spaces.

Thus, for each sub-image the upper (lower) baseline is determined based on the upper contour maxima (lower contour minima) and it corresponds to their average height. In such a case, we try to eliminate the undesirable maxima (minima) (e.g., the second minimum of Figure 21(b)) by using an approach that is based on the weighted least square technique (see Appendix 3). This approach yields a more precise estimation of the upper and lower baselines than the horizontal transition histogram. The upper (lower) line corresponds to the highest (lowest) point of the upper (lower) contour. The median line is the midway between the upper and lower baselines.

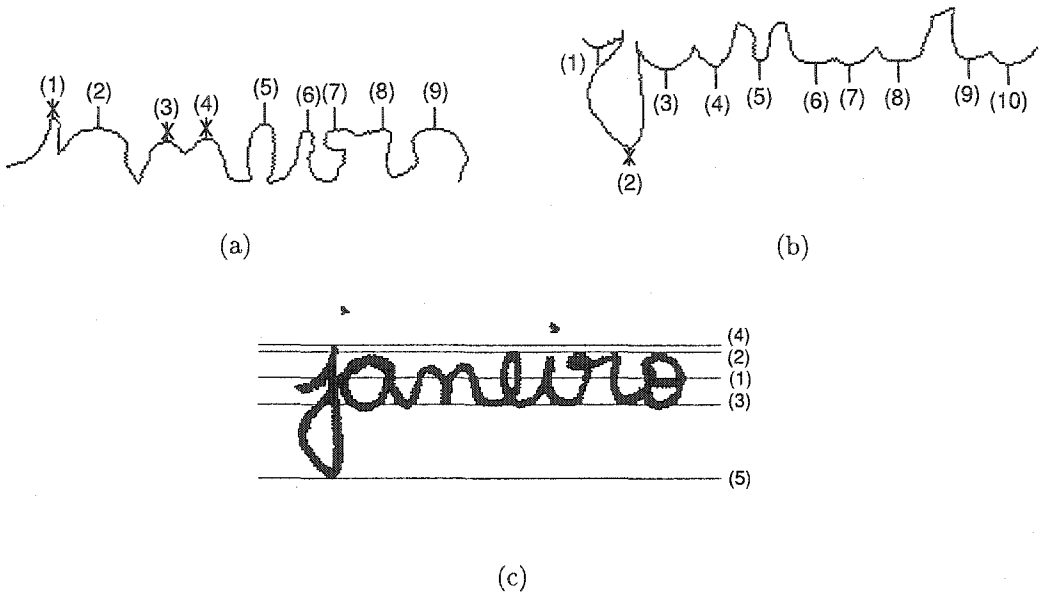


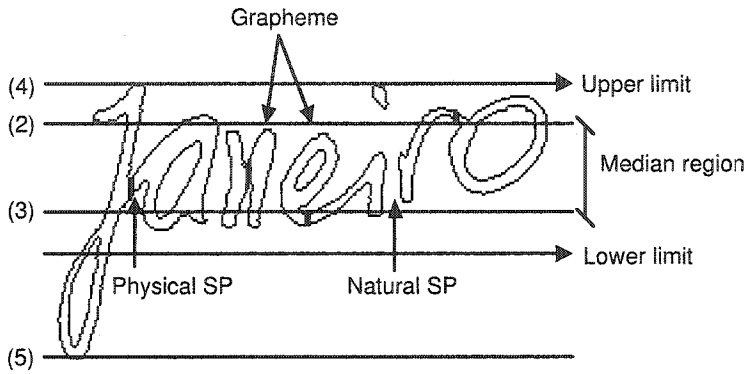
Figure 21 (a) and (b) Filtering of maxima and minima, and (c) Reference lines ((1) median line, (2) upper baseline, (3) lower baseline, (4) upper line, and (5) lower line)

4.2.2 Segmentation into Graphemes

Our segmentation strategy is based on explicit methods. Since we have seen that it is very difficult to correctly segment a word or a digit string image into characters, our concern is to design a segmentation process that supplies several segmentation points (SPs) where the optimal ones are determined during recognition. Our algorithm makes use of upper and lower contours, upper contour minima, loops, reference lines, spaces, and some heuristics to identify SPs. All the thresholds we have used were chosen after carrying out several experiments on the validation set. As output, such an algorithm provides a sequence of graphemes where each one consists of a correctly segmented, an under-segmented, or an over-segmented character.

This algorithm takes into account two types of SPs. The former corresponds to a natural SP, while the latter one is a physical SP which is located at the upper contour minimum (MP) or its neighborhood (see Figure 22). However, in order for an MP

to give rise to an SP, it must satisfy some empirical rules which are described as follows.



(a)

Figure 22 Types of SPs: natural and physical ((1) median line, (2) upper baseline, (3) lower baseline, (4) upper line, and (5) lower line)

Usually, the SPs are identified next or inside the median region of a handwritten word as we can observe in Figure 22. Based on this hypothesis, we consider the upper contour minima (MPs) situated between the upper and lower limits. The lower limit is computed by summing the lower baseline and 40% of the median region height, only if this limit does not exceed the lower line. Otherwise, the lower limit will be the same as the lower line. Basically, the upper limit is determined in the same way.

Furthermore, we take into account the MPs that permit a vertical projection from the upper contour to the lower contour. In particular cases, we look in the neighborhood of an MP:

- If the vertical projection crosses a loop before reaching the lower contour (loop configuration) (e.g., Figure 23(a))
- If the vertical projection is tangent to the lower contour (tangency configuration) (e.g., the grey segment in Figure 23(b))

- If the length of the vertical projection from the upper contour to the lower contour is greater than *stroke_thickness* × 2.0 (length configuration) (e.g., Figure 23(c))

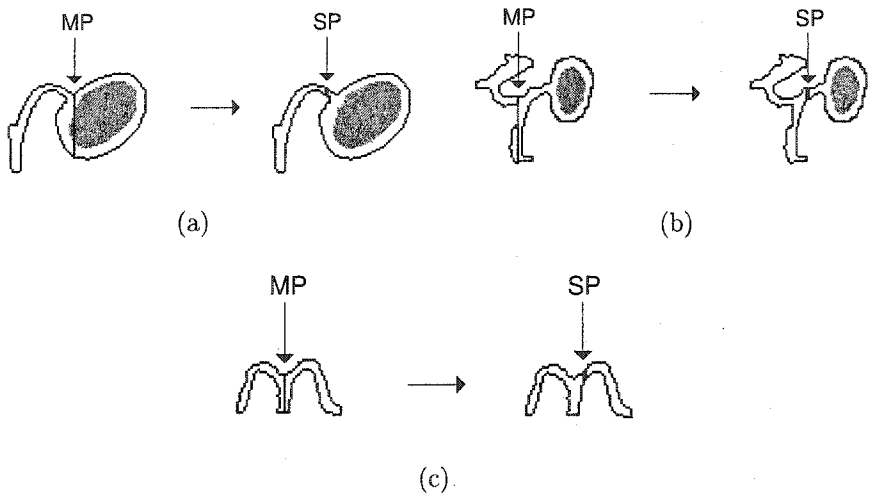
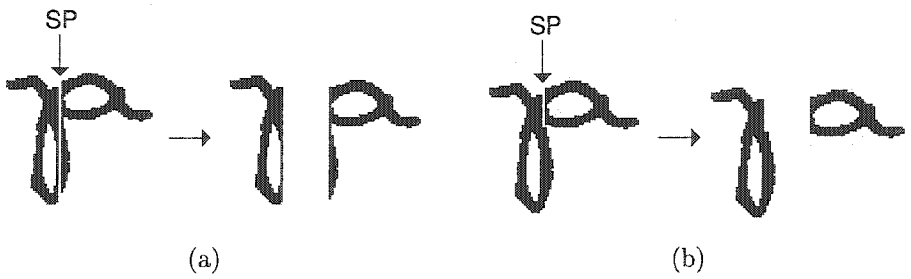


Figure 23 Configuration of MPs: (a) Loop configuration, (b) Tangency configuration, and (c) Length configuration

For the foregoing cases, our algorithm tries to find an SP in the upper contour to the right neighborhood of an MP. It starts from the MP and follows the upper contour until *stroke_thickness* × 2.0 (the description of stroke thickness is given in Section 3.2.2). The upper contour point that minimizes the vertical projection from the upper contour to the lower contour without crossing a loop or touching the tangent of the lower contour will correspond to an SP. If the vertical projection of the SP passes beside a loop (Figure 24(a)), we permit this SP to be cut in the vertical direction followed by the horizontal direction (Figure 24(b)). Notwithstanding, if the algorithm does not detect an SP in the right neighborhood of the MP, this same search is performed to the left neighborhood of the MP.

After the identification of SPs in an image, we evaluate them and discard those ones which have the following characteristics:

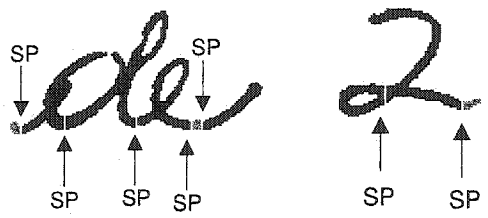


(a) (b)

Figure 24 (a) MP that minimizes the vertical projection, (b) SP cut in the vertical and horizontal directions

- If the geodesic distance between two SPs belonging to the same upper contour is lower than $stroke\_thickness \times 2.0$, we remove the one with the lowest ordinate
- If the geodesic distance between an SP and the initial or final point of upper contour is lower than $stroke\_thickness \times 2.0$

The objective of eliminating SPs in such cases is to avoid small segments (e.g., the grey segments in Figure 25). Although we have used a smoothing algorithm (see Section 3.3) that regulates the edge of an image and as a consequence decreases the detection of SPs, this preprocessing is not sufficient.



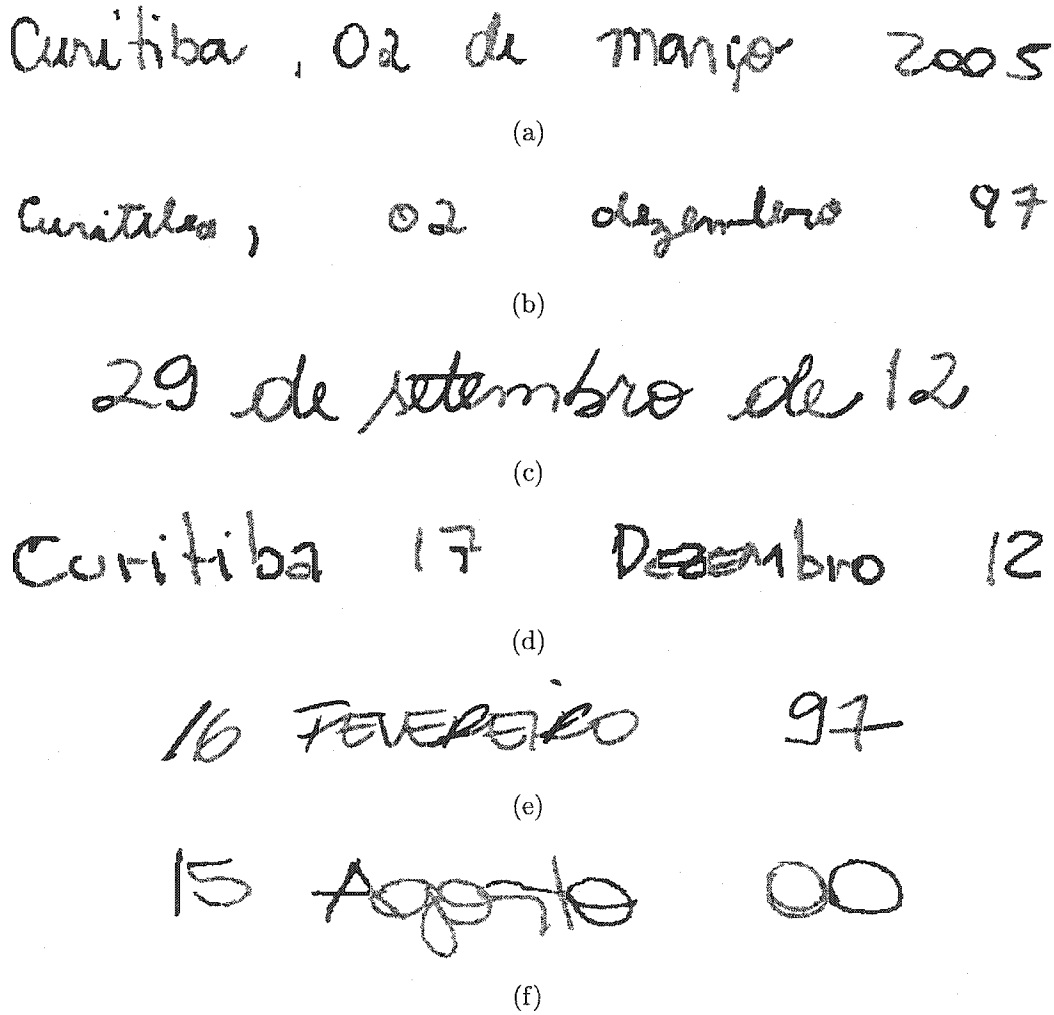
(a) (b)

Figure 25 Examples of small segments (the grey ones)

As a result, our segmentation algorithm may produce a correct segmentation of a character, a character under-segmentation or a character over-segmentation into two, three, or four graphemes depending on each character.

4.2.2.1 Discussion

In this section, an analysis of the proposed segmentation algorithm is carried out. It tries to segment the ligatures within a word by seeking for SPs located at the MPs or their neighborhoods. Therefore, such an algorithm is more suitable for cursive words (e.g., "Curitiba" and "março" in Figure 26(a)) than uppercase words (e.g., "FEVEREIRO" in Figure 26(e)) or strings of digits (e.g., "17" in Figure 26(d)).



(a)

(b)

(c)

(d)

(e)

(f)

Figure 26 Examples of images segmented into graphemes

Moreover, it attempts to avoid a character under-segmentation by looking for an SP in the neighborhood of an MP, which has a loop, tangency or length configuration as discussed before. Nevertheless, a character under-segmentation can happen in the following cases:

- Absence of an MP (e.g, the string "EI" in "FEVEREIRO" in Figure 26(e))
- Occurrence of loop configuration (e.g., the strings "00" in "2005", "ete" in "setembro", and "go" in "Agosto" in Figures 26(a), 26(c), and 26(f) respectively)
- An MP situated outside the region between the upper and lower limits (e.g., the string "ze" in "dezembro" in Figure26(b))

Finally, this algorithm is not invariant to the slant correction since we cut an SP in its vertical projection. In our system, we correct the slant in the preprocessing stage as stated in Section 3.2.

4.2.3 Feature Extraction

As we pointed out earlier, HMMs are fed by two feature sets. The first set targets the recognition of cursive handwriting, it is based on global features such as loops, ascenders, and descenders. The second one, targets the discrimination of both letters and digits, is based on concavity measurements. Both feature sets are combined with the space primitives. In Figure 27, we show an example of a segmented date image represented by these two feature sets. While F_1 corresponds to a mixture of global and space features, F_2 is related to a mixture of concavity and space features.

4.2.3.1 Global Features

Ascenders, descenders, and loops are detected through the upper contour maxima, lower contour minima, and secondary contours respectively.

F1: h- h	s4	-h s1 o s0 - - - - - s1 - - -	s6	- 0 - s0h
F2: 60 0	s4	28 s1 0 s0 1 2 6 7 8 9 s1 2 4 3	s6	3 4 9 s07

Figure 27 Image segmented into a sequence of graphemes where each one is represented by two feature sets (F_1 and F_2)

Ascenders and descenders are usually located in the upper and lower regions respectively. Unfortunately, because of the variability of handwriting, it is very difficult to precisely detect such regions. Consequently, this can lead to the detection of false ascenders and descenders. For example, in Figure 28 the letter "b" in "dezembro" is considered as a descender due to it exceeds the lower baseline. This explains why we are identifying ascenders (descenders) in the region between the upper (lower) line and upper (lower) limit (the upper and lower limits are defined in Section 4.2.2).

Furthermore, we are classing them into big or small primitives according to their position (maxima and minima) in such regions. The region of small ascenders (descenders) is situated between the upper (lower) limit and this limit decreased (increased) by 40% of the median region height. The remaining of the upper (lower) region is considered the area of big ascenders (descenders). In both cases, the threshold we have used was chosen based on experimentation carried out on the validation set. Therefore, if an upper contour maximum (lower contour minima) is located in the small or big region, it corresponds to a small or big ascender (descender) respectively. An example of the detection of these regions is illustrated in Figure 28.

Loops can be identified in any of the three regions (upper, median, and lower) depending on the position of their gravity centers. Loops belonging to the median region are encoded in two ways (big or small). The big loops are those ones greater than the half of the median region height.

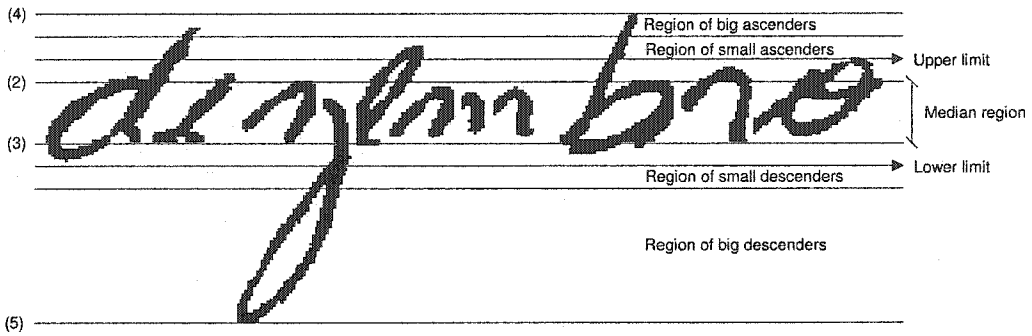


Figure 28 Detection of small and big regions ((1) median line, (2) upper baseline, (3) lower baseline, (4) upper line, and (5) lower line)

Thus, the combination of the foregoing primitives plus a primitive that determines if a grapheme does not contain ascender, descender, and loop produces a 20-symbol alphabet (see Table X). Each symbol has been evaluated in the training set in order to avoid that either it does not occur or has few samples. In this case, we are not taking into consideration the horizontal order of ascender, descender, and loop within a grapheme.

4.2.3.2 Concavity Features

The basic idea of concavity measurements [39] is the following: for each white pixel in the grapheme, we search in 4-Freeman directions (Figure 29(c)), to find out which directions reach black pixels, as well as which directions do not reach any black pixels. When black pixels are reached in all directions (e.g., point x_1 in Figure 29(a)), we branch out in four auxiliary directions (s_1 to s_4 in Figure 29(b)) 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 according to the results returned by the search (Figure 29(a)). In this figure we represent the feature vector where each component has two labels. The superior label means the number of directions which reached black pixels during the search, while the inferior label means

Table X

Description of the 20-symbol alphabet

Symbol	Description
O	Big loop in the median region
o	Small loop in the median region
H	Big ascender
h	Small ascender
B	Big descender
b	Small descender
L	Big ascender and a loop in the upper region
l	Small ascender and a loop in the upper region
Z	Big or small descender and a loop in the superior region
-	Absence of ascender, descender, and loops
D	Big ascender and a big or small loop in the median region
d	Small ascender and a big loop in the median region
s	Small ascender and a small loop in the median region
Q	Big descender and a big or small loop in the median region
q	Small descender and a big or small loop in the median region
K	Big or small ascender, a loop in the upper region, and a big or small loop in the median region
G	Big or small descender, a loop in the upper region, and a big or small loop in the median region
M	Big or small ascender and descender
F	Big or small descender and a loop in the upper or lower region
x	Big or small descender and a big or small loop in the median region

the directions where black pixels were not reached. For example, the pixel x_2 in Figure 29(a) reaches the black pixel in directions 1 and 2. Therefore, the position 3 of the feature vector is incremented. For the pixel x_1, the position 11 is incremented because it reaches the black pixel in all four directions. However, using the auxiliary direction s_3 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 each grapheme into two equal zones (horizontal) as shown in Figure 29(a), we consider two feature vectors of 13 components each. Therefore, in

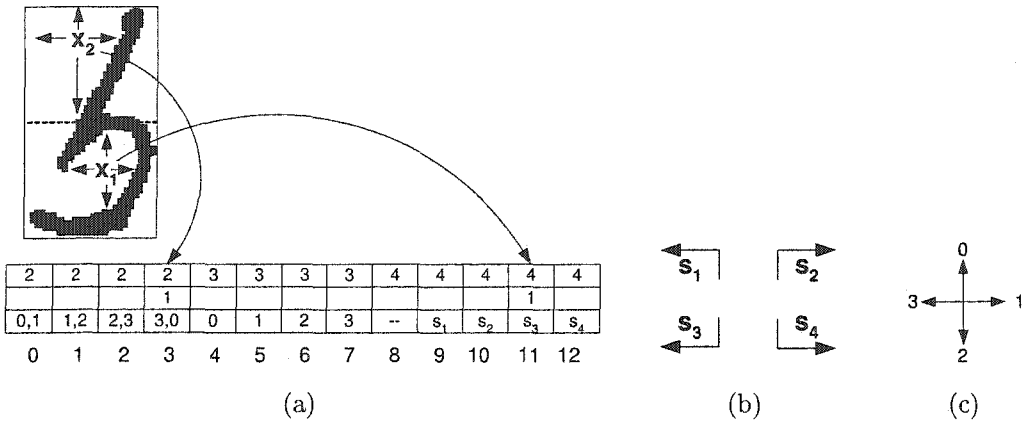


Figure 29 Concavity measurements: (a) Concavities and feature vector, (b) Auxiliary directions, and (c) 4-Freeman directions

this figure the pixel x_2 will update the first vector while the pixel x_1 will update the second one. Finally, the overall concavity feature vector is composed of 26 (13×2) components which are normalized between 0 and 1 by summing up their values and then dividing each one by this summation. The remaining concavity feature vectors considered in this work are normalized in the same way. Since we are working with the discrete HMMs, the concavity vectors are clustered into symbols by a vector quantization algorithm (see Appendix 4). In this case, the codebook size we have adopted was 100. This number was chosen after several tests carried out on the validation set.

4.2.3.3 Space features

The spaces in the median region between two connected components have been extracted from a date image and then they are combined with the global and concavity features. The space values are also clustered into symbols by a vector quantization algorithm (see Appendix 4). The codebook size we have used was eight, which was chosen based on experimentation carried out on the validation set.

4.2.4 Training Mechanism

The mechanism we have used to train the elementary models consists of two steps of training.

In the first step, the city model is trained using images of city names extracted from the date database. The number of states we have chosen was five which was determined after several tests considering different numbers of states. We have used about 980 and 370 images of city names for training and validation respectively.

In the second step, besides the date database we have also considered the legal amount database, which is composed of isolated words, in order to increase the training and validation sets. Indeed, the performance of a system strongly depends on the size of the training set. This was possible because our system takes into consideration one model for each letter. The use of meta-classes of digits also allows the sharing of data during training and consequently it increases the training and validation sets. In this case, the parameters of the city model are initialized based on the parameters obtained in the first step of training. Then, the other elementary models presented in the date and word images are trained systematically by concatenating them. We have used about 1,200 and 400 date images as well as 8,300 and 1,900 word images for training and validation respectively (a detailed description of the database is given in Appendix 1).

4.2.5 How the Segmentation is Performed

The segmentation of a date into sub-fields is delivered by the HMMs as a byproduct of the recognition process. This is made by backtracking the best path produced by the Viterbi algorithm (see Appendix 2). In this case, the system takes into account the result of the segmentation of the best date model (among the eight possibilities presented in Section 2.2.2.3) that represents a date image. In [84] we have

experimented two hypotheses of segmentation instead of one. But using this strategy, the complexity of the system increased, in opposite to the date recognition results. For this reason, we have considered in our system one hypothesis of segmentation. Figure 30 illustrates the entire process and in Section 6.1.1 we show the efficiency of our segmentation approach.

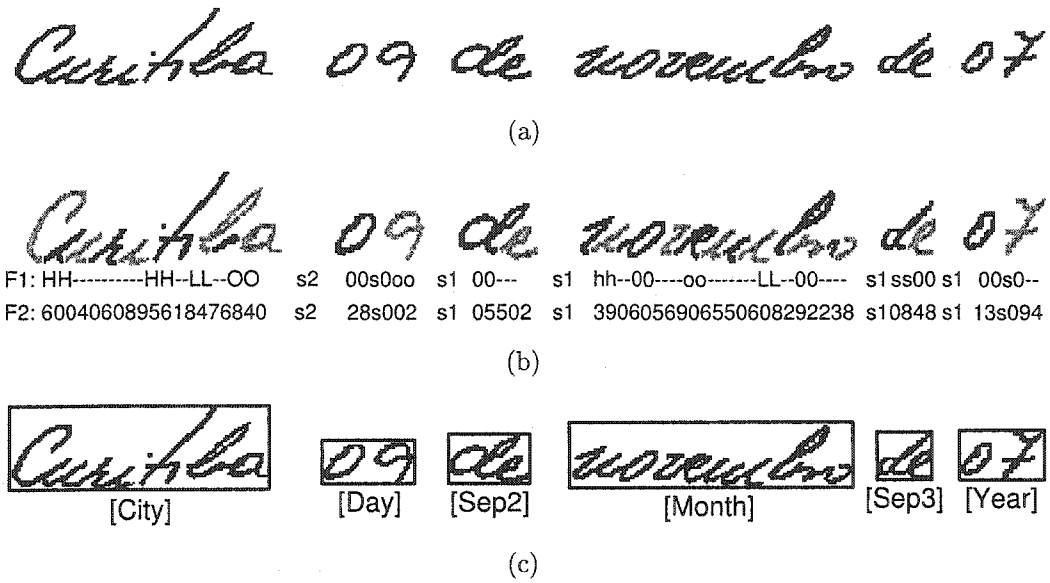


Figure 30 (a) Original image, (b) Image segmented into a sequence of graphemes where each one is represented by two feature sets (F_1 and F_2), and (c) Result of the segmentation into sub-fields

4.3 Summary

In this chapter we have presented the module of segmentation into sub-fields. It takes an HMM-based strategy for separating the date sub-fields since we have seen that this is a difficult task without resorting to recognition. The main strength of such an approach lies in the modeling phase. We have built HMMs based on the concept of the meta-classes of digits in order to reduce the lexicon size of the day and year and improve the precision of their segmentation.

Given an approach based on the discrete HMMs, each date image must be represented by a sequence of observations. To fulfill this requirement, three steps were carried out: reference line detection, segmentation, and feature extraction. Each step has been described in this chapter. The first one is essential for the segmentation and feature extraction steps that come next. In order to improve the estimation of the reference lines, we have detected them in each sub-image, each of which consists of a sub-field, part of a sub-field, or more than a sub-field. In the second step, we have detailed our segmentation algorithm which aims at providing a sequence of graphemes from a given date image, where each grapheme comprises of a correctly segmented, an under-segmented, or an over-segmented character. Finally, the last step extracts two feature sets at the grapheme level. Both feature sets are combined with the space primitives.

After that, the two feature sets are used as input to HMMs, which are devoted to segment the date sub-fields. In our system, we have considered only one hypothesis of segmentation.

The mechanism we have used to train the elementary HMMs employed in segmentation is also been reported in this chapter. In the next chapter, we will discuss the recognition part of our system, which involves three modules: digit string recognition, word recognition and verification, and final decision.

CHAPTER 5

RECOGNITION

In this chapter we describe the following three modules of the system: digit string recognition, word recognition and verification, and final decision. Section 5.1 describes the digit string recognition module and Section 5.2 presents the word recognition and verification scheme we have developed. Finally, in Section 5.3 we discuss the final decision module which makes an accept/rejection decision.

5.1 Digit String Recognition

After segmenting a date image into its constituent parts, the sub-images of the day and year are used as input to the digit string recognizer. Moreover, the number of digits supplied by the HMMs is used as information *a priori* on digit string recognition to determine which classifiers will be employed depending on the sub-field (day or year). As discussed in Section 2.2.3, we have defined five neural classifiers. e_{0-9} copes with the 10 numerical classes and the other classifiers $e_{0,1,2,3}$, $e_{0,1,2,9}$, $e_{0,9}$, and $e_{1,2}$ are specialized in the lexicon of the meta-classes of digits $C_{0,1,2,3}$, $C_{0,1,2,9}$, $C_{0,9}$, and $C_{1,2}$ respectively. This strategy aims at reducing the lexicon size on digit string recognition to improve the recognition results. Figure 31 shows the block diagram of the digit string recognition system to be described below. It is based on the system developed by Oliveira et al in [91] that recognizes strings of digits. Nevertheless, their system makes use of one classifier that works with 10 numerical classes, i.e., without the concept of meta-classes.

The digit segmentation module shown in Figure 31 is based on the relationship of three complementary sets of structural features, namely, contour, profile, and skeletal points. The segmentation hypotheses are generated through a segmentation

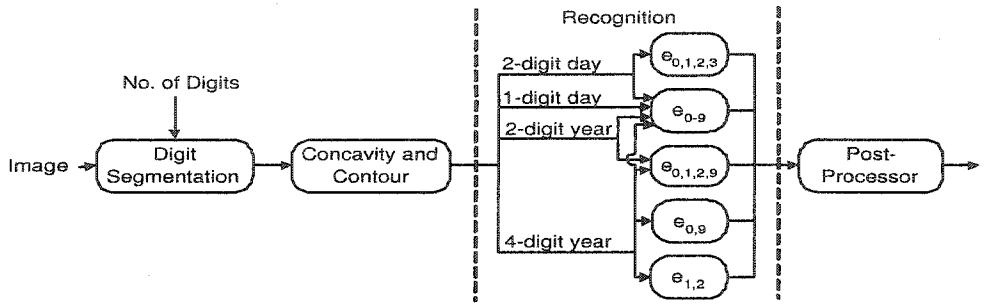


Figure 31 Block diagram of the digit string recognition system

graph, which is decomposed into linear sub-graphs representing the segmentation hypotheses. More details on such an algorithm can be found in [90].

For each segmentation hypothesis a mixture of concavity / contour is extracted. We have used six concavity feature vectors of 13 components each since we are dividing the image into six zones. In this way, the overall concavity feature vector is composed of 78 (13×6) components normalized between 0 and 1. Basically, the employed concavity features in digit string recognition are very similar to the concavity features described in Section 4.2.3.2, they differ in the size of concavity vector and the zoning used.

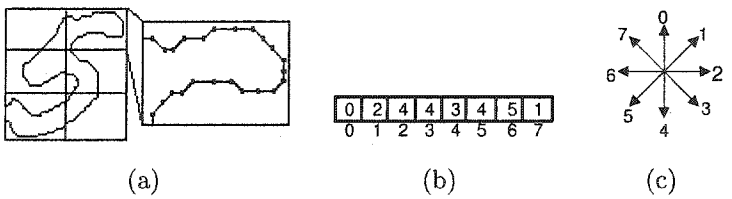


Figure 32 Contour measurement: (a) Contour image of the upper right corner zone, (b) Feature vector, and (c) 8-Freeman directions

The contour information is extracted from a histogram of contour directions. For each zone, the contour line segments between neighboring pixels are grouped into 8-Freeman directions (Figure 32(c)). The number of line segments of each orientation is counted (Figure 32(b)). Therefore, the contour feature vector is composed of 48

(8×6) 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 this value between 0 and 1. Thus, the final feature vector, which feeds our classifiers, has 132 $(78 + 48 + 6)$ components.

In order to train those classifiers, we have used images of isolated digits extracted from the courtesy amount and date databases. Table XI describes the databases used for training (TR), validation (VL) and testing (TS) as well as the recognition rates achieved on validation (RR VL) and test (RR TS) sets (a detailed description of the database is given in Appendix 1).

Table XI

Description of the classifiers

Classifier	Classes	TR	VL	TS	RR VL (%)	RR TS (%)
$e_{0,1,2,3}$	0,1,2 and 3 ($C_{0,1,2,3}$)	7,302	1,774	2,305	99.7	99.4
e_{0-9}	0-9 (C_{0-9})	14,211	3,217	4,710	99.0	98.9
$e_{0,1,2,9}$	0,1,2 and 9 ($C_{0,1,2,9}$)	7,377	1,787	2,345	99.7	99.4
$e_{0,9}$	0 and 9 ($C_{0,9}$)	3,594	845	1,111	99.9	99.8
$e_{1,2}$	1 and 2 ($C_{1,2}$)	3,783	942	1,234	99.8	99.5

Since we are dealing with multi-hypotheses of segmentation and recognition the generation of k best hypotheses of a string of digits is carried out by means of a Modified Viterbi algorithm, which ensures the calculation of the k best paths of segmentation-recognition graph [91]. Thus, the final probability for a hypothesis of segmentation-recognition is given by the product of the probabilities produced by the classifiers as shown in Figure 33. For simplicity, this figure presents just one hypothesis of segmentation. Afterwards, each hypothesis is submitted to the post-processor module depicted in Figure 31, which verifies whether it belongs to the lexicon of the day or year depending on the sub-field.

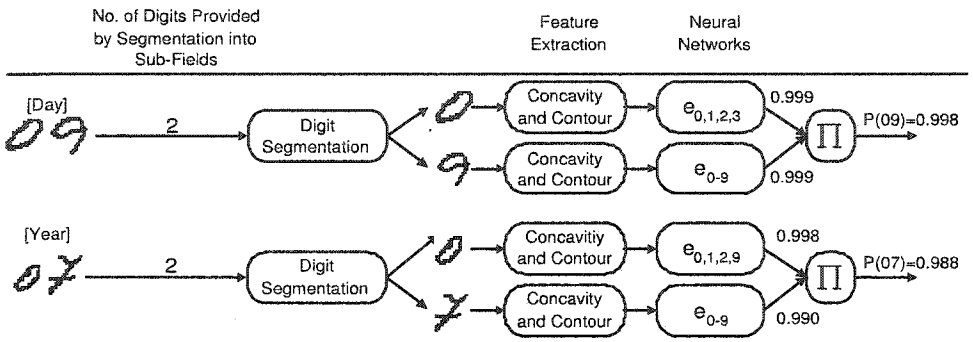


Figure 33 Digit string recognition through an example

5.2 Word Recognition and Verification

In this section, we describe the HMM-based word recognition and verification approach. The recognizer is devoted to the word segmentation and recognition aspects once it uses the same models employed in segmentation into sub-fields, while the proposed verifier is specialized in the word recognition problem. This verifier deals with the loss in terms of recognition performance brought by the word recognition module and it aims at improving the word recognition results and reliability of the system. The word models employed in word recognition and verification are built by concatenating elementary letter models as discussed in Section 2.2.2.2. The word recognition and verification scheme takes an analytical approach based on an explicit segmentation.

5.2.1 Word Recognizer

The word recognizer receives as input the two sequences of observations related to the word image identified in the sentence, that were extracted in the segmentation into sub-field module. Then, the word recognizer computes the word probabilities for the 12 word models using the Forward procedure (see Appendix 2). However, only the two best hypotheses generated by the word recognizer will be confirmed by the word verifier.

5.2.2 Word Verifier

The word verifier takes a word image as input which is transformed into a sequence of observations. This is done by segmenting the image into graphemes and then two feature sets are extracted from the sequence of graphemes. The first set is based on global features, while the second one is a mixture of concavity and contour features. Both feature sets are combined with the segmentation primitives. Since the segmentation algorithm and the global features are the same described throughout this thesis, here we explain the concavity and contour features, which differ in the size of concavity and contour vector, and the segmentation primitives.

Since we are dividing a grapheme into two equal zones (horizontal) and for the concavity features we are considering only those white pixels that reach three or more black pixels, we have two concavity vectors of 9 components each instead of 13 components as employed in date segmentation. For each vector, we have introduced eight more components related to the information about the contour image. They have been used to increase the discrimination between some pairs of letters (e.g., "L" (JULHO) and "N" (JUNHO)). In this manner, the final feature vector has 34 $(2 \times (9+8))$ components. The feature vectors are clustered into symbols by a vector quantization algorithm (see Appendix 4). We have used a codebook with the size of 100 chosen after carrying out several tests on the validation set.

The segmentation features have been used to reduce confusions such as "n" (junho) and "l" (julho) and "i" (maio) and "r" (marco) since they try to reflect the way that the graphemes are linked together. For connected graphemes, we encode the nature of segmentation points in two ways depending on whether its vertical position is closer to the upper or lower baselines. We have also defined a primitive to indicate no segmentation point between two graphemes.

Therefore, given an input word image, the output of the feature extraction process is a pair of symbolic descriptions of equal length, each consisting of an alternating sequence of grapheme shapes and associated segmentation point symbols. In order to train the elementary models employed in word verification, we have used about 9,500 and 2,300 word images for training and validation respectively extracted from the date and legal amount databases (see Appendix 1).

5.2.3 How the Word Verifier interacts with the Word Recognizer

The objective of the absolute high-level word verifier is to re-rank the output of the word recognizer. Basically, the word verifier computes the probabilities for two word models that correspond to the two best hypotheses (Top1 and Top2) generated by the word recognizer using the Forward procedure. Then, we multiply the probabilities produced by the word recognizer and verifier. In Figure 34, we present an example of how the word verifier interacts with the word recognizer. We can see in this figure that the word recognizer generates the list of hypotheses which contain the correct one ("Novembro"), but it is not in the top of the list. On the other hand, the word verifier succeeds in re-ranking the correct hypothesis to the top of the list ($0.48 \times 0.90 > 0.50 \times 0.10$). In such an example, we have used fictitious probabilities in order to better illustrate the problem. In Section 6.1.2 we will see the improvements produced by this scheme of verification.

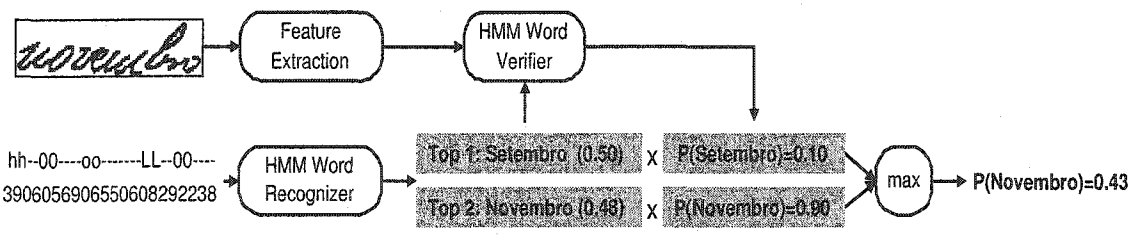


Figure 34 Example of how the word verifier interacts with the word recognizer

5.3 Final Decision

The final decision module accepts the recognition result when all three obligatory date sub-fields are correctly classified, otherwise it rejects it. The goal of rejection is to minimize the number of recognition errors for a given number of rejects. The scheme of rejection we have used is based on Fumera et al [26].

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 such a case, we have considered one threshold for each class of digits and for each month word. In order to define such thresholds we have developed an iterative algorithm, which takes into account a decreasing function of the threshold 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 6.2 (see Chapter 6).

Thereafter, the rejection of an image is straightforward. We just compare the probability of its components (month and digits) with their corresponding thresholds. If any of the components has the probability less than its corresponding threshold, the entire string is rejected, otherwise it is accepted.

We will see in the next chapter that this strategy of rejection produces interesting error-reject trade-offs. We will present experimental results considering different values of T_{error}.

5.4 Summary

In this chapter we have presented the part of recognition which includes the modules of digit string recognition, word recognition and verification, and final decision.

Besides the date segmentation into sub-fields, the scheme we have proposed to reduce the lexicon size on digit string recognition is an important contribution of our work and it aims at improving the recognition results. This strategy makes use of the output of the HMMs in the form of digit string length, i.e., the information on the number of digits present in a string (day or year) and the meta-classes of digits.

The word verification scheme we have developed is another important feature presented in this chapter. In the next chapter we show the results of experiments carried out and the importance of the role of the word verifier is in the system.

CHAPTER 6

EXPERIMENTS AND ANALYSIS

This chapter is devoted to the experiments conducted on three databases to assess our approach. The first database contains about 2,000 images of handwritten dates and it aims at evaluating the performance of the system on the recognition of dates written on Brazilian bank cheques. The second database is the NIST SD19 (hsf_7 series) and it aims at validating the concept of meta-classes of digits on digit string recognition on a well-known database. Finally, the last one corresponds to the legal amount database which have about 10,500 isolated images of handwritten words. The purpose is to validate the word recognition and verification scheme since it is very difficult to compare our work with others due to its special application, i.e., date recognition on Brazilian bank cheques. Besides, comparison in the same context with other approaches is very delicate when different databases and formats are used, different word classes are involved, and different sizes of databases are considered. In order to assess the recognition and verification scheme, we carried out some experiments on word recognition using a legal amount database and then we compare the results achieved with other study which makes use of the same database. A detailed description of the foregoing databases is given in Appendix 1.

For all reported results we used the following definitions of the recognition rate, error rate, rejection rate and reliability rate. Let TS be a test set with N_{TS} string images. If the recognition system rejects N_{rej}, classifies correctly N_{rec} and misclassifies the remaining N_{err}, then:

$$\text{Recognition Rate} = \frac{N_{rec}}{N_{TS}} \times 100 \tag{6.1}$$

$$\text{Error Rate} = \frac{N_{err}}{N_{TS}} \times 100 \tag{6.2}$$

$$\text{Rejection Rate} = \frac{N_{rej}}{N_{TS}} \times 100 \tag{6.3}$$

$$\text{Reliability} = \frac{\text{Recognition Rate}}{\text{Recognition Rate} \quad + \quad \text{Error Rate}} \times 100 \tag{6.4}$$

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

6.1 Experiments on Date

This section reports the experiments carried out on the date database which contains about 2,000 handwritten date images. It was divided into three sets:

- Training: 1,182 images
- Validation: 396 images
- Test: 401 images

6.1.1 Date Segmentation Results

The first step of our system is to find the best date model (among the eight possibilities presented in Section 2.2.2.3) that represents a date image. At this level the system reached 93.6% and 95.5% on the validation and test sets respectively. Table XII details the segmentation rate of each date sub-field on the validation (VL) and test (TS) sets as well as the result when the number of digits present in a string (day or year) is correctly estimated by the HMMs. The results shown in this table were

evaluated automatically by the system. This was possible because we have a labelled database which also contains information about the position of each date sub-field.

Table XII

Segmentation results

Set	City	Day	Sep2	Month	Sep3	Year	No. of Digits	
							Day	Year
VL	94.9%	94.1%	94.6%	97.9%	98.9%	100.0%	91.1%	100.0%
TS	95.7%	96.2%	95.5%	99.5%	100.0%	100.0%	92.2%	100.0%

We can notice in Table XII that the results on year segmentation is higher than the results reached on day segmentation. In this application, the year segmentation is less complex than the day due to the low frequency of the "De" separator before the year and its location (i.e., the year is the last sub-field present in the date field).

Figure 35 shows examples where the date sub-fields are missegmented and Figure 36 demonstrates difficult cases of segmentation, where the spaces between sub-fields and within sub-fields are very similar. However in such cases, our approach succeeded in segmenting the date sub-fields correctly.

By analyzing the errors on date segmentation on the validation set, we observed that they are generalized by the confusions between the following two sub-fields: city and day (Figures 35(a) and 35(b)), day and Sep2 (Figure 35(b)), Sep1 and day (Figure 35(c)), month and Sep2 or Sep3 (Figure 35(d)), and day and month (Figure 35(e)). These errors can be visualized in Table XIII. From this table we can note that the main confusions come from City-Day and Day-Sep2 sub-fields. Consequently, the segmentation rates of the city, day, and Sep2 sub-fields are lower than the other ones as shown in Table XII.

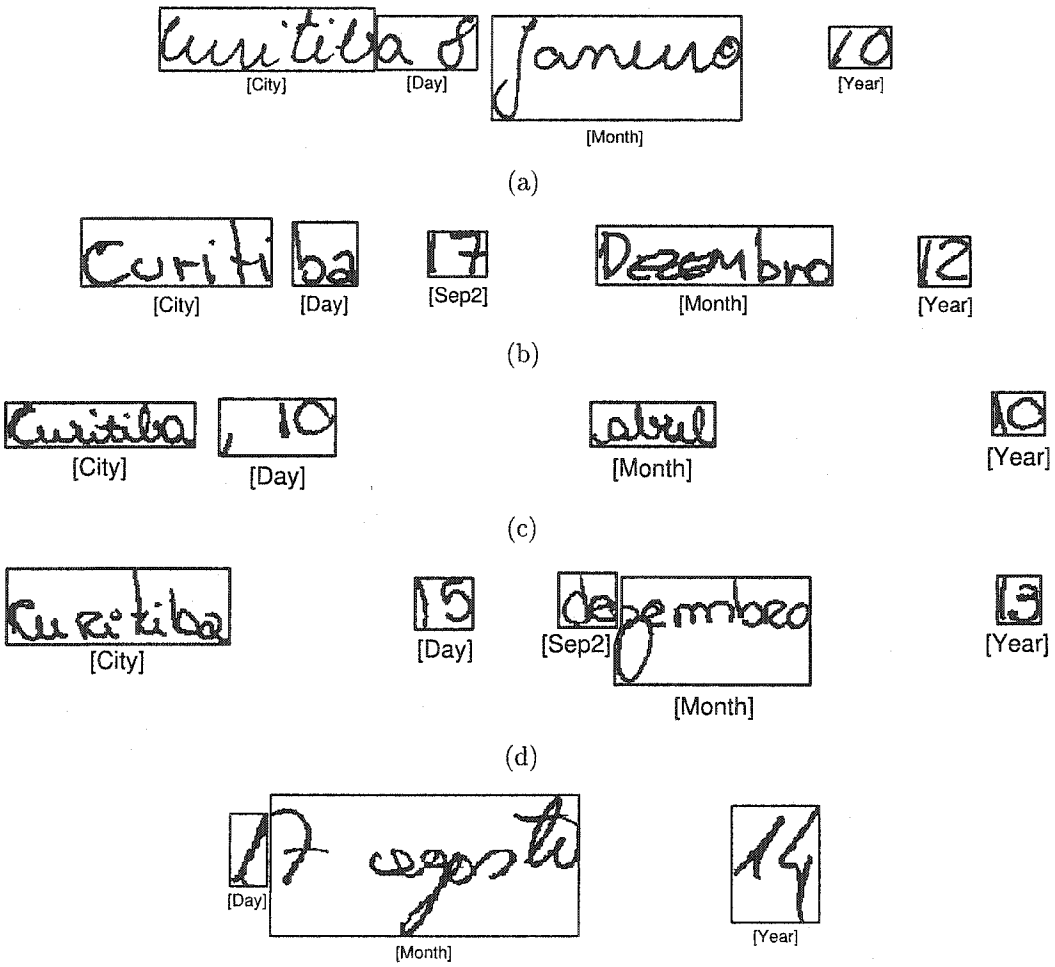


Figure 35 Examples of missegmented date images

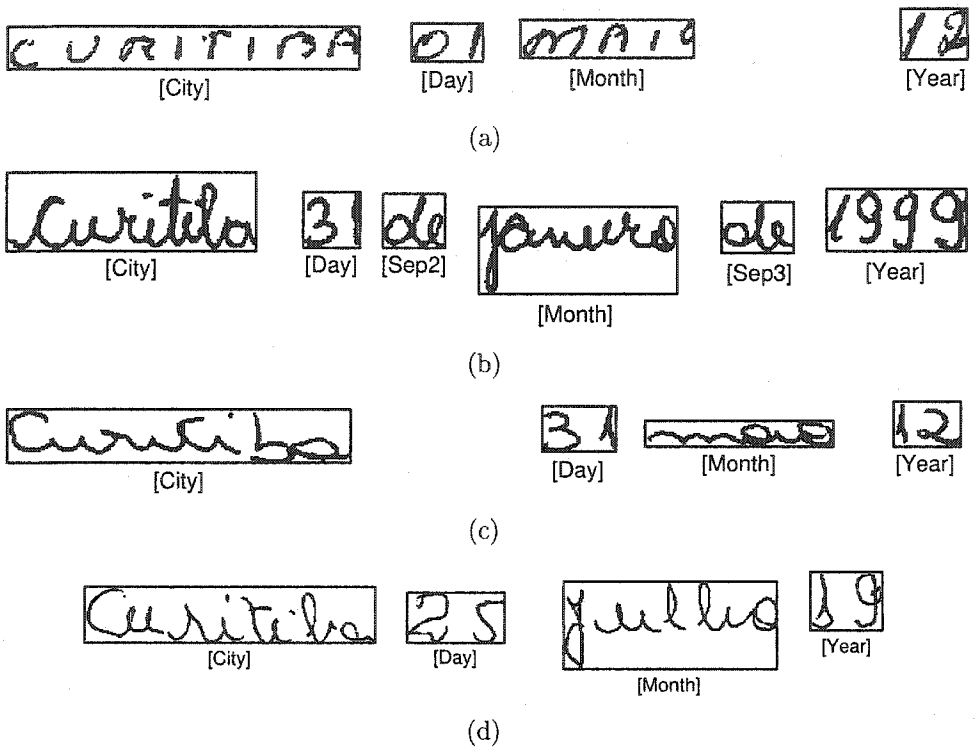


Figure 36 Examples of well-segmented date images

Table XIII

Confusions on date sub-field segmentation on the validation set

Confusion	Error Rate (%)
City-Day	38
Day-Sep2	36
Sep1-Day	10
Month-Sep2/Sep3	14
Day-Month	2

6.1.2 Date Recognition Results

Table XIV reports the improvements on date recognition using the word verifier on the validation (VL) and test (TS) sets (zero-rejection level). A date image is counted as correctly classified if the three obligatory sub-fields are correctly classified. Besides, this table presents the results on digit string recognition and word recognition with verification.

Table XIV

Performance of the system ($-$: results without verification and $\surd$: results with verification

Set	Word Verifier	Date	Month	1-digit Day	2-digit Day	2-digit Year	4-digit Year
VL	$-$	79.5%	89.6%	60.0%	93.2%	97.2%	-
	$\surd$	82.5%	93.1%	60.0%	93.2%	97.2%	-
TS	$-$	80.7%	89.5%	71.4%	92.6%	97.7%	100.0%
	$\surd$	82.5%	91.5%	71.4%	92.6%	97.7%	100.0%

From Table XIV we can notice that the verification brings an increase in the recognition rates on date by 3.0% and 1.8% on the validation and test sets respectively. Furthermore, we were able to increase the word recognition rates by 3.5% and 2.0% on the validation and test sets respectively. We observed on the validation set that the presence of common sub-strings among some word classes can affect the performance on month word recognition such as:

- The termination in "eiro" for "Janeiro" and "Fevereiro";
- The termination in "embro" for "Setembro", "Novembro" and "Dezembro";
- Almost all characters between "Junho" and "Julho" and between "Maio" and "Março".

Indeed, the foregoing confusions can be visualized in Table XV, which shows the confusion matrix on month word recognition and verification carried out on the test set. The last column of this table details the recognition rate (RR) of each word class ("1" for "Janeiro", "2" for "Fevereiro", "3" for "Março", "4" for "Abril", "5" for "Maio", "6" for "Junho", "7" for "Julho", "8" for "Agosto", "9" for "Setembro", "10" for "Outubro", "11" for "Novembro", and "12" for "Dezembro"). Figures 37(e) and 37(f) show some misclassified date images because the month words were not correctly recognized. In such cases, the errors correspond to under-segmentation problems due to the lack of local minima and confusion of the word classifiers respectively.

Table XV

Confusion matrix on month word recognition and verification on the test set

Class	1	2	3	4	5	6	7	8	9	10	11	12	No. of Images	RR (%)
1	38	0	0	0	0	0	0	0	0	0	1	0	39	97.4
2	0	29	0	0	0	0	0	0	1	1	1	0	32	90.6
3	1	0	30	0	5	0	0	0	0	0	0	0	36	83.3
4	0	0	0	35	4	0	0	0	0	0	0	0	39	89.7
5	1	0	3	0	34	0	0	0	0	0	0	0	38	89.4
6	0	0	0	0	1	28	0	0	0	0	0	0	29	96.5
7	2	0	0	0	0	4	25	0	0	1	0	0	32	78.1
8	0	0	0	0	0	0	0	28	0	0	0	0	28	100.0
9	0	0	0	0	0	0	0	0	30	1	0	0	31	96.7
10	0	0	0	0	0	0	0	0	0	30	0	0	30	100.0
11	0	0	0	0	0	0	0	0	1	0	32	1	34	94.1
12	1	0	0	0	0	1	0	1	1	0	1	28	33	84.8

Furthermore, we can notice in Table XIV that the results on year recognition are higher for 2-digit strings than the results achieved on day recognition for 2-digit strings. This can be explained by the fact that in our application the year segmentation is less complex than the day as discussed before. The low recognition rates for 1-digit day on the test and validation sets and the difference of about 11% between

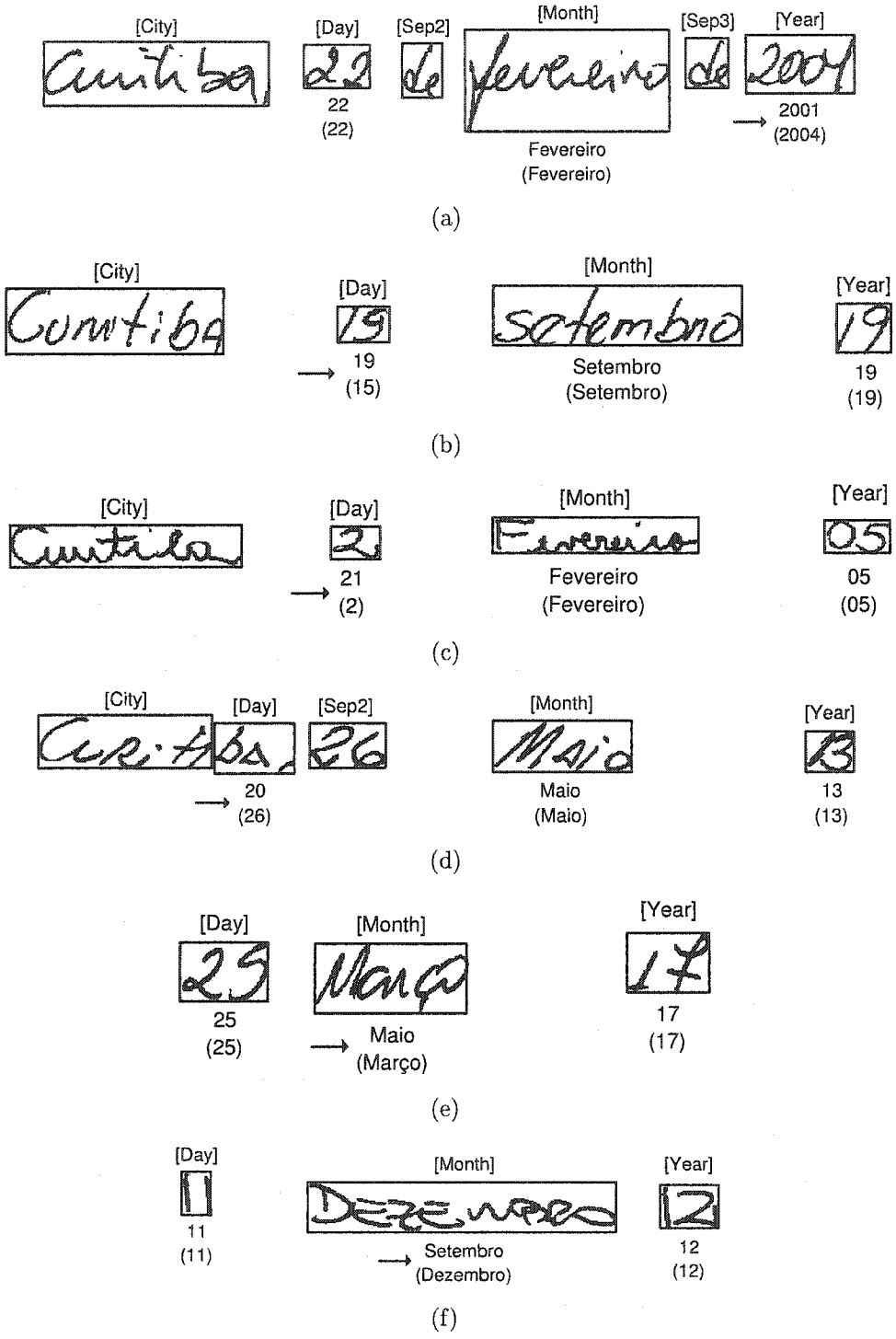


Figure 37 Examples of misclassified date images (the correct string is the one in parentheses): errors that come from (a) Digit segmentation, (b) Digit string recognition, (c) Number of digits, (d) Day segmentation, (e) Grapheme segmentation, and (f) Word recognition and verification

them can be explained by the fact that we have few images for 1-digit day on the test (21 images) and validation (25 images) sets. In the validation set, for example, one misclassified image means 4% of error.

By analyzing the errors on digit string recognition on the validation set, we noted that they can be classified into 5 classes: errors caused by recognition (Rec.), fragmentation (Frag.), digit segmentation (Digit Seg.), and date segmentation when our HMM-based approach does not correctly segment the day and year sub-fields (Seg. into Sub-Fields) or it does not properly identify the number of digits present in a string (day or year) (No. of Digits), e.g., two digits were identified in Figure 37(c) instead of one. The confusions produced by fragmentation are found basically when the digit segmentation algorithm groups the fragmented part with the wrong neighbor. Usually, it fails for images that have neighbors (left and right) with similar distances to the fragmented part and for images with poor quality. The digit segmentation errors can be caused either by under-segmentation (Figure 37(a)), which is due to a lack of basic points in the neighborhood of the connection stroke, or wrong segmentation. These errors can be visualized in Table XVI while some examples are illustrated in Figure 37(a), 37(b), 37(c), and 37(d). In respect of day recognition, this table shows that the day missegmentation is the main source of errors.

Table XVI

Error analysis on digit string recognition on the validation set

Sub-Field	Seg. into Sub-Fields	Rec.	Frag.	Digit Seg.	No. of Digits
Day	57.1%	17.2%	8.5%	-	17.2%
Year	-	58.3%	16.7%	25.0%	-

As mentioned before, Figure 37 show some examples of misrecognized date images, while Figure 38 illustrates some well-classified date images. Although the day in

Figure 38(e) was not correctly segmented, the digit string recognizer was capable of classifying it properly. In the same manner, the word recognition and verification scheme was able to classify appropriately the month word in Figure 38(f).

Since bank cheque systems demand low error rates, two experiments were carried out on date recognition using the test set where the error rates were fixed at 1.2% and 2.0% respectively. Table XVII presents recognition (RR), rejection (RJ), and reliability (RL) rates at these two error levels, while Figure 39 shows the evolution of the recognition rate, error rate and reliability as a function of the rejection rate. We can observe from this figure that the strategy of rejection which considers multiple reject thresholds produces interesting error-reject trade-offs.

Table XVII

Recognition rates (RR), Rejection rates (RJ) and Reliability rates (RL) for different error rates on date recognition

Error = 1.2%			Error = 2.0%		
RR(%)	RJ(%)	RL(%)	RR(%)	RJ(%)	RL(%)
62.1	36.6	98.1	65.3	32.6	97.0

6.2 Experiments on NIST SD19

In order to validate the concept of meta-classes of digits on digit string recognition, we performed two experiments using a subset of the validation set of the NIST SD19 database (hsf_7 series). In such experiments, we have considered images of 2-digit strings related to the lexicon of 2-digit day. Table XVIII summarizes the recognition rates (RR) achieved on these experiments. The former (Exp. I) considers the concept of meta-classes of digits using the classifiers $e_{0,1,2,3}$ and e_{0-9}, while the latter (Exp. II) makes use of the e_{0-9} classifier, i.e., without the concept of meta-classes. The use of this concept on digit string recognition seems to be a good strategy when the

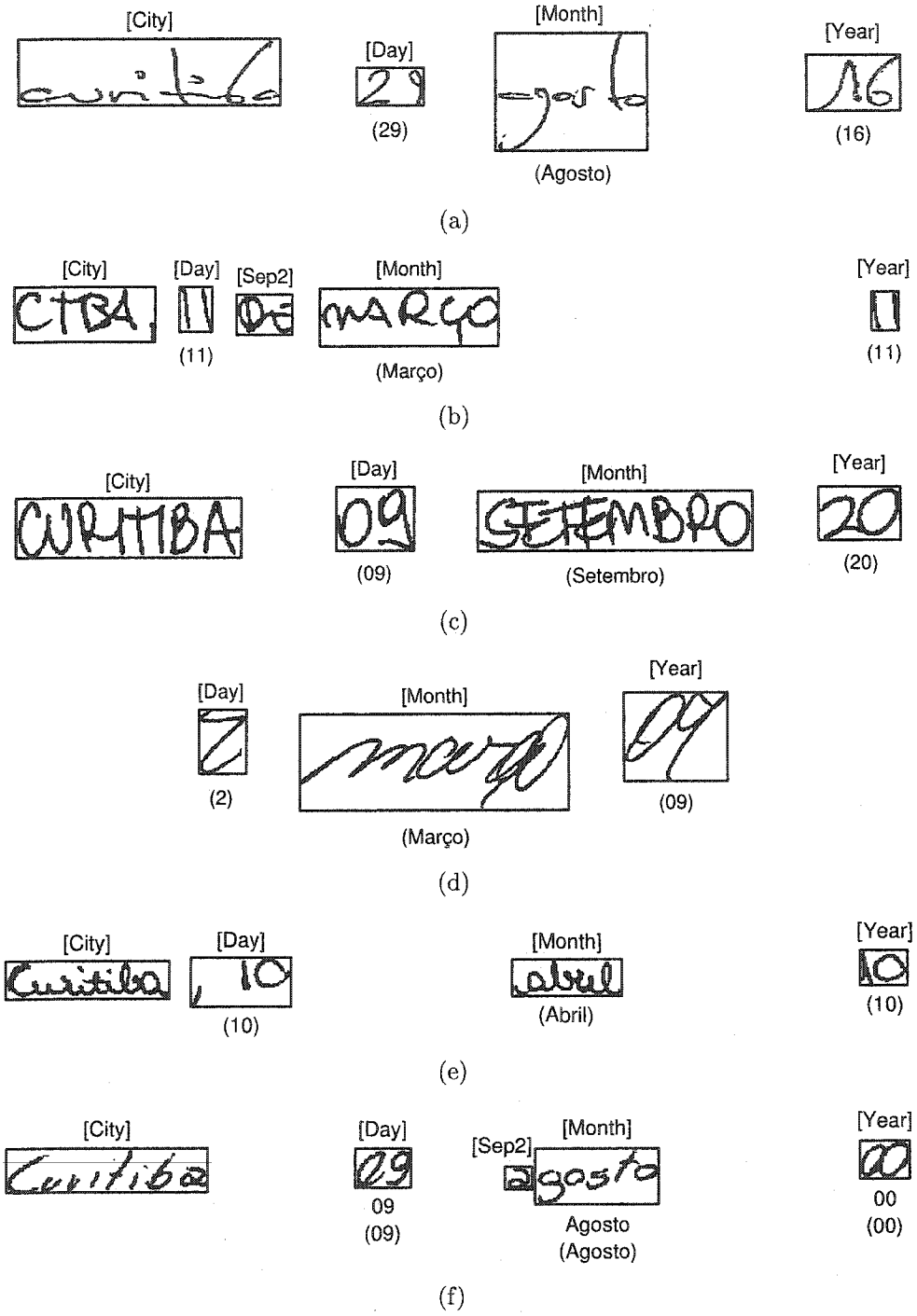


Figure 38 Examples of classified date images

lexicon is known and limited since it enhanced the recognition results from 97.1%

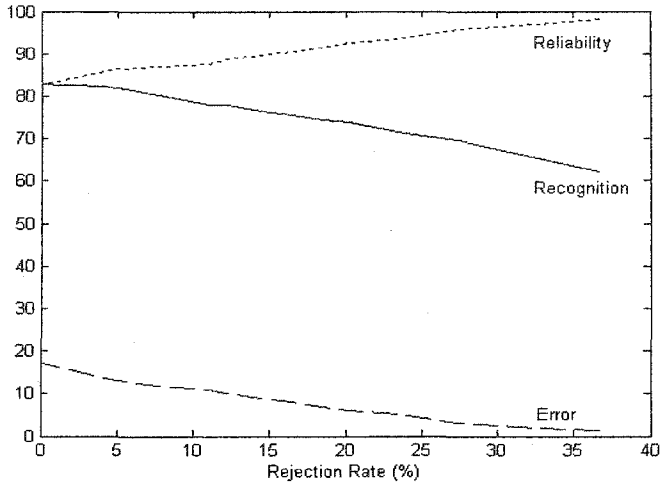


Figure 39 Error rate versus rejection rate for date recognition

to 99.2% as shown in Table XVIII. To train the classifiers $e_{0,1,2,3}$ and e_{0-9} we have used 78,000 and 195,000 images of isolated digits respectively from hsf_{0,1,2,3}.

Table XVIII

Recognition rates on NIST database for 2-digit strings

Experiment	No. of Images	RR (%)
Exp. I	986	99.2
Exp. II	986	97.1

6.3 Experiments on Isolated Words of Legal Amount

This section addresses the experiments conducted on legal amount database, which contains about 10,500 isolated images of handwritten words from a vocabulary of 40 words. It was divided into three sets: 6,264, 2,092, and 2,074 images for training, validation, and testing respectively.

As mentioned before, the idea of using such a database is to assess the word recognition and verification scheme. In order to accomplish this task, we have performed some experiments. Table XIX reports the two best recognition rates (Top1 and Top2) on the test set without considering the word verifier. In this case, the objective is to compare the performance of our word recognizer with the one developed by Freitas et al in [24]. This was possible since we have used the same database. Their work considers one global Markov model for each class of words and makes use of global, concavity, and convexity features. However, in this case modeling characters is better than modeling words due to the small number of images for training some classes of words which do not have a uniform distribution. Thus, by considering character models we can increase the training set and improve the performance on word recognition. We can observe from Table XIX the improvements on word recognition using our approach. The recognition rates for Top1 and Top2 were increased by 15.2% and 9.9% respectively.

Table XIX

The two best recognition rates on word recognition on the test set

Work	Word Recognizer		Word Recognizer and Verifier
	Top1 (%)	Top2 (%)	Top1 (%)
Current	85.8	92.3	88.1
Freitas et al [24]	70.6	82.4	-

In the top of it, the word verifier brings an improvement of the recognition rate on word recognition on the test set from 85.8% to 88.1% (Top1). The results show the efficiency of the strategy we have developed for word recognition and verification.

6.4 Discussion

It is very difficult to compare with other sentence recognition engines due to the special application of our work. Comparison in the same context with other approaches is very delicate when we consider bank check recognition systems since different databases and formats are used, different word classes are involved, and different sizes of databases are considered. Besides, the literature indicates few studies on this area. For example, Suen et al in [109] present a system for segmenting date images into sub-fields written on Canadian bank checks. In this application each date image can appear in any one of two patterns: $MM\ S\ DD\ S\ 19\ YY$ and $DD\ S\ MM\ S\ 19\ YY$ where MM, S, DD, and YY refer to month, separator, day and year sub-fields respectively. The month sub-field can be written in different data types (digits or words in English or French), while the day and year sub-fields only in digits. They claim a segmentation rate of 83.1% on the test set of 310 French and 499 English cheques from CENPARMI_IRIS database. Xu et al in [119] present an extension of the previous work. It focuses on the more difficult task of separating the day and month sub-fields. The result achieved was 89.9% using 1,000 English date images from the CENPARMI_IRIS database. Considering our approach, we reached an interesting segmentation rate of 95.5% on the test set.

More recently, Xu in [117] presents a complete date processing system. In this work, the author achieved recognition rates of 44.5% and 36.4% on date recognition using the CENPARMI database where the error rates were fixed at 5.2% and 5.0% respectively. While the first experiment was carried out on the English set composed of 1,197 date images, the second one was performed on the French set which consists of 2,088 date images. At this level, our system reached a recognition rate of 65.3% on the test set with a fixed error rate of 2.0%.

In respect of month word recognition, Xu et al in [118] report a modified product rule to combine two MLP classifiers and an HMM classifier. Their goal is to recognize month words in English and French. The recognition rate got was 85.36% using a test set of 2,063 samples from a separate set of CENPARMI_IRIS database. Oliveira Jr. et al in [14] consider the same classes of month words we have used, but they make use of a different database. Their best result on month word recognition obtained was 87.2% by combining (average) two MLP classifiers on the test set composed of 1,200 isolated images of month words. In [13], Oliveira Jr. et al improved their results by combining (multiplication) HMM and two NNs. They achieved a recognition rate of 90.4% on month word recognition on the test set. Concerning our system, we achieved a recognition rate of 91.5% on month word recognition on the test set considering the word verifier. We believe that this result is quite promising since we must take into consideration the segmentation aspects.

Since a direct comparison of our work with others is not possible due to its special application, i.e., date recognition on Brazilian bank cheques, we carried out some experiments on legal amount and NIST SD19 databases in order to assess the proposed digit string recognition and word recognition and verification schemes. We have seen encouraging results using such schemes. This confirms the efficiency of our strategies we have developed for digit string recognition and word recognition and verification.

Lastly, we consider that our system has achieved good performance taking into account its complexity. In spite of the fact the system has reached comprehensive results, we have seen that it has various sources of errors, for instance, errors generated by date segmentation, digit segmentation, digit string recognition, word recognition and verification, etc. Therefore, the system still can be improved. To accomplish this, we will optimize one part of the system due to the its magnitude. Our focus

will be the optimization of the word verifier. We will address this issue in the next chapter.

CHAPTER 7

OPTIMIZATION OF THE WORD VERIFIER

So far, we have described in detail all modules of the date recognition system and how they interact with each other. Besides, we have reported the results we achieved on date recognition using the laboratory date database. We have also demonstrated through experimentation using the NIST and legal amount databases, the feasibility and efficiency of the strategies we developed for digit string and word recognition. Although the system has achieved compelling results, we have seen in the last chapter that the system still has different sources of errors that can come from date segmentation, digit segmentation, digit string recognition, grapheme segmentation, word recognition and verification, etc. In this light, we decided to make some efforts towards the performance of the system. But due to the magnitude of the date recognition system, our focus will be the optimization of the word verifier since the performance on word recognition is lower than digit string recognition. Regarding the word recognition problem, the word recognizer takes into consideration both segmentation and recognition aspects, while the word verifier considers just the recognition ones. Thus, as starting point it seems simpler to optimize the word verifier. In this chapter we discuss the technique we have investigated: feature selection in unsupervised learning.

7.1 Unsupervised Feature Selection

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. In this way, the selection of the best discriminative features plays an important role when constructing classifiers. Nevertheless, this is not a trivial task especially when dealing with a lot of features. In order to choose a

subset of the original features by reducing irrelevant and redundant ones automated feature selection algorithms have been used. The literature contains several studies on feature selection for supervised learning [92, 104]. But only recently, the feature selection for unsupervised learning has been investigated [19, 53].

The objective in unsupervised feature selection is to search for a subset of features that best uncovers "natural" groupings (clusters) from data according to some criterion. This is a difficult task because to find the subset of features that maximizes the performance criterion, the clusters have to be defined. The problem is made more difficult when the number of clusters is unknown beforehand which happens in most real-life situations. Hence, it is necessary to explore different numbers of clusters using traditional clustering methods such as the K-Means algorithm [41] and its variants. Thus, clustering can become a trial-and-error work. Besides, its result may not be very promising especially when the number of clusters is large and not easy to estimate.

In this context, feature selection presents a multi-criterion optimization function, e.g., the number of features and a validity index to measure the quality of the clusters. Genetic algorithm (GA) offers 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. In the last decade, GA has been largely applied to the feature selection problem. 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-offs among objectives. In order to overcome this kind of problem, some authors [53] propose the use of a multi-objective genetic algorithm to perform feature selection.

A methodology for feature selection in unsupervised learning for handwritten word recognition is presented in the following sections. It makes use of the Nondominated

Sorting Genetic Algorithm (NSGA) proposed by Srinivas and Deb in [106] which deals with multi-objective optimization. The objective is to find a set of nondominant solutions which contain the more discriminant features and the more pertinent number of clusters. In order to guide this search we have used two criteria: minimization of the number of features and minimization of a validity index that measures the quality of clusters. A standard K-Means algorithm is applied to form the given number of clusters based on the selected features. The proposed strategy is assessed using two synthetic data sets where the significant features and the appropriate clusters in any given feature subspace are known. Afterwards, it is applied to optimize classifiers in a supervised learning context, i.e., handwritten word recognition. In this practical scenario, our approach is first evaluated by conducting some experiments on isolated handwritten month word recognition and then we try to optimize the word verifier of the date recognition system.

This section is organized as follows. In Section 7.1.1, we review some related works in unsupervised feature selection. Section 7.1.2 presents the method of multi-objective optimization we have adopted. In Section 7.1.3, we describe the proposed methodology and in Section 7.1.4, we assess our approach using two synthetic data sets. Section 7.1.5 reports the preliminary results we achieved on isolated handwritten month word recognition. Finally, the experiments carried out on date recognition are presented in Section 7.1.6.

7.1.1 Related Works

Most of works concerning feature selection have been carried out under the supervised learning paradigm, paying little attention to unsupervised learning tasks. Supervised feature selection algorithms are used when class labels of the data are available, otherwise unsupervised feature selection algorithms are employed.

Conventional methods of feature selection involve evaluating different feature subsets using some index and selecting the best among them. The index usually measures the capability of the respective subsets in classification or clustering depending on whether the selection process is supervised or unsupervised. A problem of these methods, when applied to large data sets, is the high computational complexity involved in searching. 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 [44]. Most of the current approaches assume monotonicity of some measure of classification or clustering 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. Recently, several researchers have explored the use of genetic algorithms to perform feature selection. The advantage of feature selection techniques that employ GAs is that they do not require the restrictive monotonicity assumption since they can deal with the use of multiple selection criteria. Comparison and discussion of some of the above methods may be found in [57].

In a supervised learning context, feature selection algorithms can 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 [44].

In order to maintain the definition of wrapper/filter approaches used in supervised feature selection, Dy and Brodley in [19] define a wrapper approach in unsupervised

learning when applying a clustering algorithm to each feature subset in the search space and then evaluating the feature subset by criterion function that utilizes the clustering result. On the other hand, a filter approach makes use of some intrinsic property of the data to select features without utilizing the clustering algorithm. Since our interest lies with unsupervised feature selection, we discuss here some related works in this domain.

Devaney and Ram [16] combine a sequential forward and backward search. In this study, they evaluate each feature subset by measuring the category utility of clusters, which were found by applying COWEB (a hierarchical clustering algorithm). Talavera in [111] applies blind (similar to the filter) and feedback (analogous to the wrapper) approaches to COWEB, and uses a feature dependence measure to select features. Dy and Brodley in [19] propose a wrapper approach that uses an expectation-maximization clustering algorithm with order identification. Feature subsets are evaluated in terms of clustering quality based on scatter separability or maximum likelihood.

Mitra et al in [78] present an algorithm for unsupervised feature selection that uses feature dependency/similarity for redundancy reduction and does not require any search process. Besides, a new feature similarity measure, called maximum information compression index, is introduced. Kim et al in [53] make use of an evolutionary local selection algorithm to search for possible combination of features and number of clusters, with the guidance of the K-Means algorithm. In this study, they consider as fitness criteria the within-cluster scatter, between-cluster separation, number of selected features, and number of selected clusters.

Following the definition proposed by Dy and Brodley in [19], our unsupervised feature selection method can be classified as a wrapper approach. Basically, it works in the same vein as Kim et al [53]. Notwithstanding, our approach makes use of a different

search algorithm and different fitness criteria. Besides, this approach is the only one we have knowledge that is applied to optimize classifiers in a supervised learning context.

7.1.2 Multi-Objective Optimization using Genetic Algorithms

7.1.2.1 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 [97].

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

$$\text{subject to:} \begin{cases} g_j(x) & \leq & 0 & j = 1, 2, \ldots, J \\ h_k(x) & = & 0 & k = 1, 2, \ldots, K \end{cases} \tag{7.1}$$

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 smaller than another vector $x^{(2)}$, $(x^{(1)} \prec x^{(2)})$, when no value of $x^{(2)}$ is smaller than $x^{(1)}$ and at least one value of $x^{(2)}$ is strictly greater than $x^{(1)}$. If $x^{(1)}$ is partially smaller 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 $x^{(2)}$ if:

$$\begin{aligned} &\forall i \in \{1,2\} : f_i(x^{(1)}) \leq f_i(x^{(2)}) \quad \wedge \\ &\exists j \in \{1,2\} : f_j(x^{(1)}) < f_j(x^{(2)}) \end{aligned} \tag{7.2}$$

7.1.2.2 Nondominated Sorting Genetic Algorithm (NSGA)

Over the past decade, a number of multi-objective evolutionary algorithms have been proposed. Zitzler et al in [128] 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 [106] surpasses several other methods. Besides, such a method has been applied to solve various problems [77, 115]. 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 40a. 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).

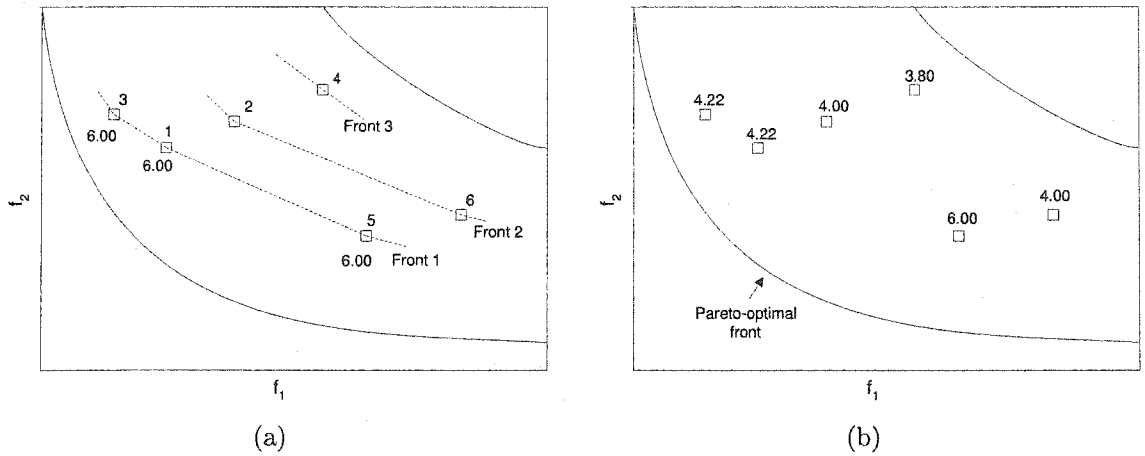


Figure 40 Sorting population: (a) The population is classified into three nondominated fronts and (b) Shared fitness values of six solutions

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 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 40b. 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 3.80 (4.00×0.95).

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.

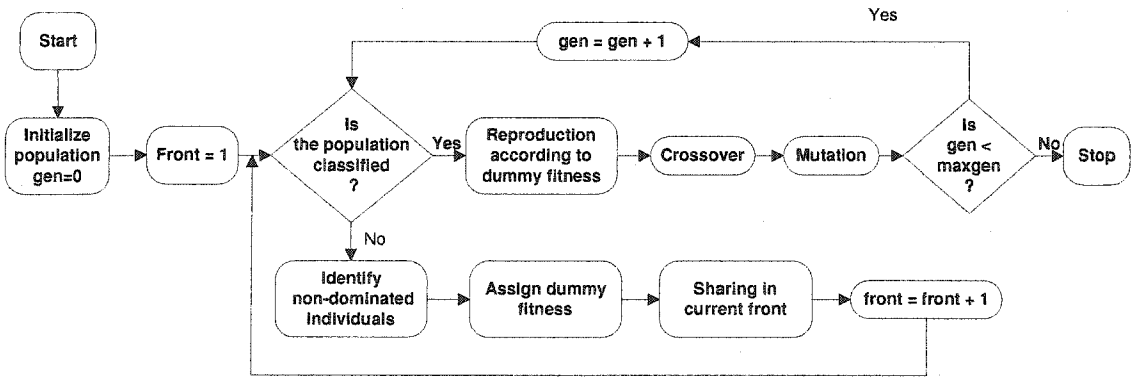


Figure 41 Flow chart of NSGA

Figure 41 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:

$$Sh(d(i,j)) = \begin{cases} 1 - \left(\frac{d(i,j)}{\sigma_{share}}\right)^2 & \text{if} \quad d(i,j) < \sigma_{share} \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

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 [15] 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 [15]:

$$\sigma_{share} \approx \frac{0.5}{\sqrt[p]{q}} \tag{7.4}$$

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 [106] that the use of the above equation with $q \approx 10$ works in many test problems.

7.1.3 Proposed Methodology

7.1.3.1 Objective Functions

As stated before, we have used two criteria: minimization of a validity index and minimization of the number of features.

In order to measure the quality of clusters, the within-cluster scatter and between-cluster separation have been widely used by various researchers. Kim et al in [53] make use of two objective functions to compute these measurements independently. Vesanto et al in [113] and Bandyopadhyay et al in [4] combine them in one objective function using the Davies-Bouldin (DB) index proposed by Davies et al in [12]. To make such indices suitable for our problem, they must be normalized by the number

of selected features. This is due to the fact that they are based on geometric distance metrics and are therefore not directly applicable here because they are biased by the dimensionality of the space, which is variable in feature selection problems. In our experiments, we have considered the normalized DB index. Both criteria are described as follows.

The DB index is a function of the ratio of the sum of within-cluster scatter to between-cluster separation. The scatter within the i_{th} cluster is computed as follows:

$$S_i = \frac{1}{|C_i|} \sum_{X \in C_i} \{||X - Z_i||\} \tag{7.5}$$

and the distance between clusters C_i and C_j is defined as:

$$d_{ij} = \{||Z_i - Z_j||\} \tag{7.6}$$

S_i is the average Euclidean distance of the vectors X in cluster C_i with respect to its centroid Z_i. d_{ij} is the Euclidean distance between the centroids Z_i and Z_j of the clusters C_i and C_j respectively. Subsequently, we compute:

$$R_i = \max_{j, j \neq i} \{\frac{S_i + S_j}{d_{ij}}\} \tag{7.7}$$

The DB index is then defined as:

$$I_{DB} = \frac{1}{D} \frac{1}{K} \sum_{i=1}^{K} R_i \quad 0 \leq I_{DB} \leq 1 \tag{7.8}$$

where K corresponds to the number of selected clusters and D is the number of selected features. The objective is to achieve proper clustering by minimizing the DB index.

We have observed through experimentation (see Section 7.1.4.2) that the value of DB index decreases as the number of features increases by normalizing such an index by D. In order to compensate this effect we have also considered as objective the minimization of the number of features. In this case, one feature must be set at least.

7.1.3.2 Implementation of NSGA

For all experiments, NSGA is based on bit representation (binary codification), one point crossover, bit-flip mutation and roulette wheel selection (with elitism). In such experiments, we have used the Equation 7.4 with $q = 10$ in order to have some insight about the parameter of NSGA σ_{share}. However, this parameter and the other ones were tuned based on experimentation.

Since the proposed strategy tries to find a set of solutions with the more discriminant features and a proper value of the number of clusters, each chromosome in the population encodes these two types of information. While the first positions encode the features, the remaining is devoted to the number of clusters. In order to find high-quality solutions, we have considered two objectives: minimization of the number of features and minimization of the DB index.

Computing the first objective is simple, i.e., the bits equal to 1 in the first part of the chromosome provide the number of selected features. The second one is evaluated after performing clustering. In this case, a standard K-Means algorithm (see Appendix 4) is applied to form the clusters based on the selected features and

the number of selected clusters, which is obtained by computing the bits equal to 1 in the second part of the chromosome.

7.1.4 Evaluation of the Methodology on Two Synthetic Data Sets

It is not easy to evaluate the quality of an unsupervised learning algorithm especially performing feature selection at the same time due to the fact that the clusters depend on the dimensionality of the selected features. In order to assess the proposed methodology and improve our insight about it, we carried out two experiments using two synthetic data sets, where the distributions of their points, the significant features, and the appropriate clusters in any given feature subspace are known. In such cases, we can evaluate the solutions found in the Pareto-optimal front by examining the selected features and the number of selected clusters as well.

7.1.4.1 Experiment I

The first synthetic data set has 300 points, two significant features in which the points form three well defined clusters. All clusters are formed by generating points from a pseudo-Gaussian distribution with standard deviation equal to 0.04. Figure 42 illustrates this data set.

In this experiment, the chromosomes are composed of 12 bits, the first two bits encode the features, while the remaining (position 3 to 12) encode the number of clusters that can vary from 2 to 10. The cases of zero or one cluster are meaningless in this application. The NSGA parameters are: population size=20, number of generations=100, probability of crossover=0.8, probability of mutation=1/12, and the niche distance=0.4.

Table XX shows the set of nondominant solutions found by NSGA. We can notice in this table that both solutions describe the data set depicted in Figure 42 very

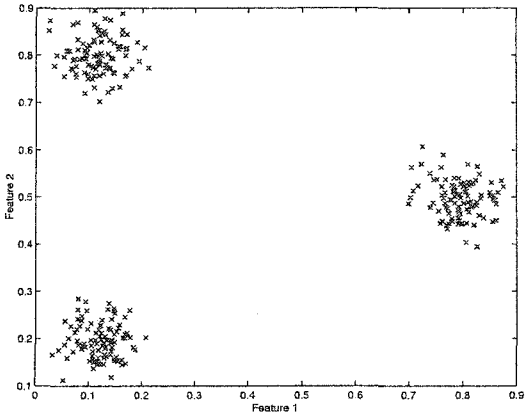


Figure 42 Data set of Experiment I

well. The solutions Sol_1 and Sol_2 are extremely good along one of the two criteria. However, the solution with three clusters and one feature (feature 2) was not in the final population because it was dominated by the solution Sol_2 along DB index criterion.

Table XX

Solutions for Experiment I

Solution	No. of Clusters	DB Index	No. of Features	Features
Sol_1	3	0.074246	2	1 and 2
Sol_2	2	0.086910	1	1

7.1.4.2 Experiment II

The second synthetic data set has 300 points and ten features and it is constructed as follows. Three clusters are formed along features 1 and 2 in the same way as the first data set. Features 3, 4, and 5 are similar to feature 2. Finally, for features 6, 7, 8, 9, and 10 the points are distributed uniformly. All the clusters are formed by generating points from a pseudo-Gaussian distribution with standard deviation

equal to 0.04. Figure 43 illustrates this data set by projecting the points onto some of the feature subspaces with two dimensions.

The chromosomes are represented by 20 bits, the first ten bits encode the features, while the remaining (position 11 to 20) encode the number of clusters that can vary from 2 to 10. The NSGA parameters are: population size=20, number of generations=100, probability of crossover=0.8, probability of mutation=1/20, and the niche distance=0.4.

Table XXI shows the set of nondominant solutions found by NSGA. We can observe from this table that the features 6 through 10 which have no significance were not selected. Besides, the solutions describe the data set depicted in Figure 43 very well. In this experiment, the interaction between our two optimization criteria can be visualized as discussed before in Section 7.1.3.1.

Table XXI

Solutions for Experiment II

Solution	No. of Clusters	DB Index	No. of Features	Features
Sol_1	3	0.037460	5	1, 2, 3, 4 and 5
Sol_2	3	0.043207	4	1, 2, 3 and 4
Sol_3	3	0.052015	3	1, 2, and 4
Sol_4	3	0.073910	2	1 and 4
Sol_5	2	0.090448	1	1

Furthermore, although the selected features and the number of selected clusters of Sol_2 (Exp. I) and Sol_5 (Exp. II) shown in Tables XX and XXI respectively are basically the same, they have different indices. It is possible that for the same feature subspace and number of selected clusters, there exist different DB index since the K-Means results are highly dependent on the initialization procedure.

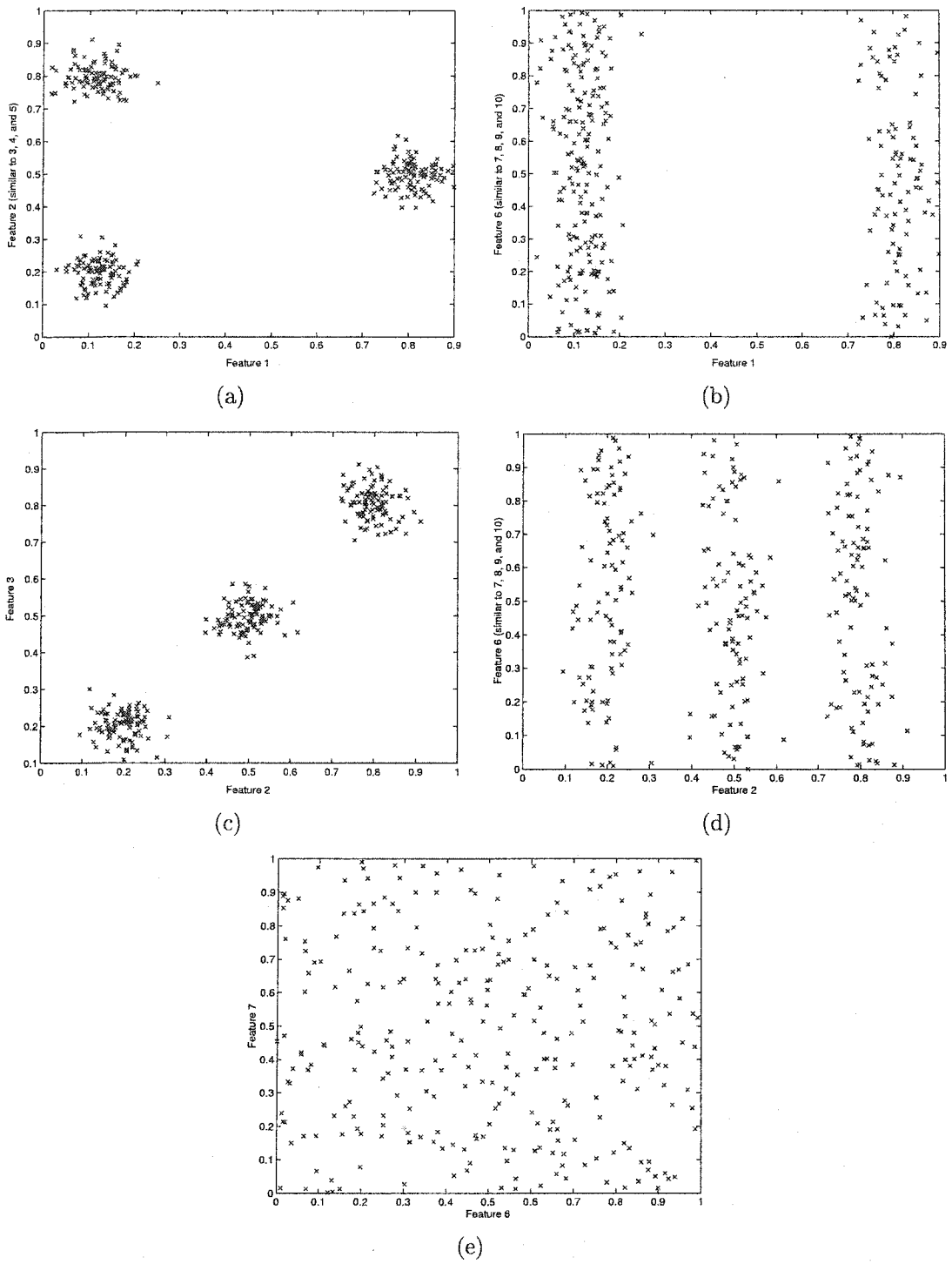


Figure 43 Some 2-dimensional projections of the data set of Experiment II

7.1.5 Methodology Applied to Isolated Month Word Recognition

Regarding the results achieved in the experiments using the synthetic data sets, it can be concluded that the proposed methodology is capable of identifying the more significant features and the more relevant number of clusters. The next step lies in evaluating its effect on isolated month word recognition.

Basically, our methodology works as follows. NSGA produces automatically a set of nondominant solutions called Pareto-optimal, which corresponds to the best trade-offs between the number of features and quality of clusters. However, when applying such a strategy to a supervised learning context, one solution from Pareto-optimal front must be chosen to be used in the system. In order to perform this task, firstly we train each solution of the Pareto-optimal front to validate the best solutions found by NSGA. In this case, we have not considered those solutions with few features (number of features lower or equal to 10) since we have knowledge that using few ones are not sufficient for classification. Thereafter, such classifiers are used in the system and the solution that supplies the best word recognition result on the validation set is chosen.

In this section we describe the three experiments carried out on month word recognition. The word classifier used in such experiments is based on the word verifier of the date recognition system we have presented so far (see Section 5.2.2). The word verification module splits a word image into graphemes, each of which consists of a correctly segmented, under-segmented, or and over-segmented character. Then, two feature sets are extracted from the sequence of graphemes to feed the Markovian classifier. However, in order to better assess our approach, in these experiments we have considered only one feature set, which is combined with segmentation primitives.

The feature set we have used in the first experiment (Exp. I) is based on a mixture of concavity and contour features, which were employed in word verification. As

discussed before, each grapheme is divided into two zones where for each region a concavity and contour feature vector of 17 components is extracted. In this manner, the final feature vector has 34 components. The other two experiments make use of a different feature set which is based on distance features, namely DDD features (see Appendix 5). In the second experiment (Exp. II), we have adopted the same zoning used in the first one. But in this case, for each region a vector of 16 components is extracted. This leads to a final feature vector of 32 components which are normalized between 0 and 1 by summing up their values and then dividing each one by this summation. For the third experiment (Exp. III), we have tried a different zoning. We are dividing each grapheme into four zones using the reference baselines (see Figure 44), and therefore we have four feature vectors of 16 components each. The overall feature vector is composed of 64 components which are normalized between 0 and 1 in the same way. Such experiments are summarized in Table XXII.

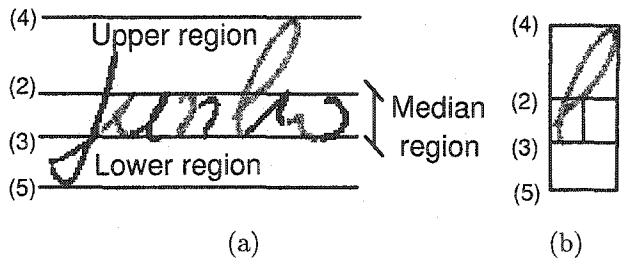


Figure 44 Zoning for Exp. III: (a) Reference line detection and (b) Zoning ((2) upper baseline, (3) lower baseline, (4) upper line, and (5) lower line)

Since the word classifier is based on discrete HMMs and the preceding feature vectors contain real values (low-level features), we must convert them into symbols (high-level features) by using a clustering technique. Instead of using a traditional strategy as we did in word verification, which considers the entire feature set and tries exhaustively various number of clusters, we propose to use the foregoing methodology to find automatically the most representative concepts (clusters) and the more discrim-

Table XXII

Description of the experiments on isolated month word recognition

Exp.	Features	No. of Features
I	Conc&Contour	34
II	DDD	32
III	DDD	64

inant features. However, in order to evaluate the results achieved by our approach, we compare them with the results obtained from the traditional strategy.

7.1.5.1 Experimental Results and Analysis

This section is devoted to the three experiments conducted on a database in which we do not have knowledge about the clusters and relevant features. This database contains about 2,000 isolated images of handwritten Brazilian month words and it was divided into three sets: 1,182, 396, and 401 images for training, validation, and testing respectively. In order to increase the training and validation sets, we have also considered 8,325 and 1,906 word images respectively extracted from the legal amount database. For clustering we have used about 80,000 feature vectors extracted from the training set of 9,507 words.

In the first experiment (Exp. I), the chromosomes are represented by 184 bits. The first thirty four bits encode the concavity and contour features, while the remaining (position 35 to 184) encode the number of clusters that can vary from 2 to 150. The NSGA parameters are: population size=96, number of generations=1,000, probability of crossover=0.8, probability of mutation=1/184, and the niche distance=0.3.

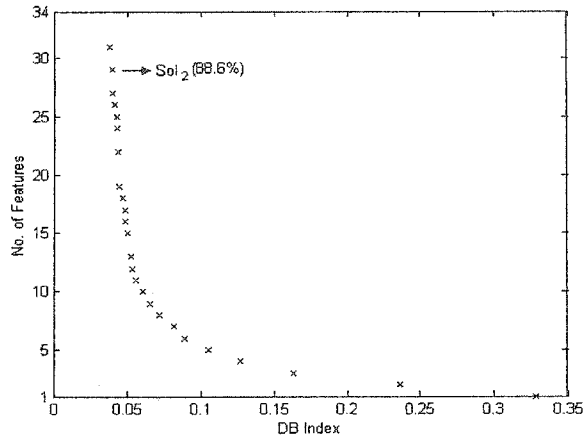
For the second experiment (Exp. II), the chromosomes contain 182 bits. While the first thirty two bits encode the DDD features, the remaining (position 33 to 182)

encode the number of clusters that can vary from 2 to 150. The parameters of NSGA are: population size=98, number of generations=1,000, probability of crossover=0.8, probability of mutation=1/182, and the niche distance=0.3.

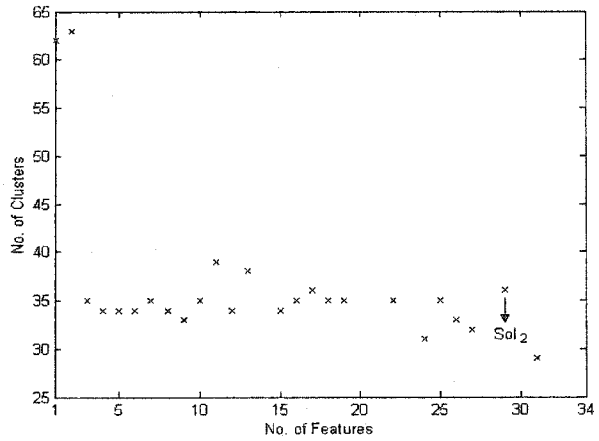
Finally, the chromosomes in the last experiment (Exp. III) are composed of 214 bits. The first sixty four bits encode the DDD features, while the remaining (position 65 to 214) encode the number of clusters that can vary from 2 to 150. The NSGA parameters are: population size=98, number of generations=1,000, probability of crossover=0.8, probability of mutation=1/214, and the niche distance=0.3.

Figures 45, 46, and 47 illustrate the Pareto-optimal front found by NSGA in such experiments respectively. Furthermore, such figures show the relationship between the number of clusters $\times$ number of features and the recognition rate (RR) on the validation set $\times$ number of features. The selected solution in the first experiment, which supplied the best word recognition result on the validation set, was the solution Sol_2 (29 concavity and contour features and 36 clusters). In the second experiment, the selected solution was Sol_9 (23 DDD features and 24 clusters), while in the last was Sol_{11} (53 DDD features and 63 clusters). Regarding the selected solutions Sol_2 (Exp. I), Sol_9 (Exp. II), and Sol_{11} (Exp. III), the recognition rates (RRs) (with zero-rejection level) on the validation (VL) and test (TS) sets are reported in Table XXIII. From Figures 45(a), 46(a), and 47(a) we can visualize that the value of the DB index decreases as the number of features increases. Moreover, as we expected, the number of nondominant solutions augments as the number of features gets bigger, e.g., in Exp. III . In such a case, the number of solutions to be trained is also increased as a consequence.

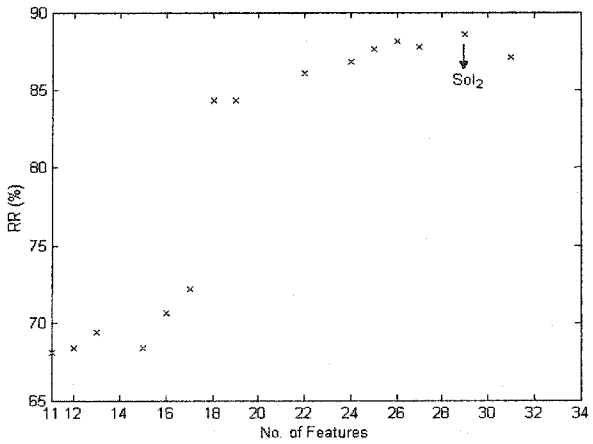
Table XXIV reports the recognition rates (with zero-rejection level) achieved on month word recognition on the validation set by using the traditional scheme for each experiment where the number of clusters was fixed at 20, 40, 60, 80, 100, 120,



(a)

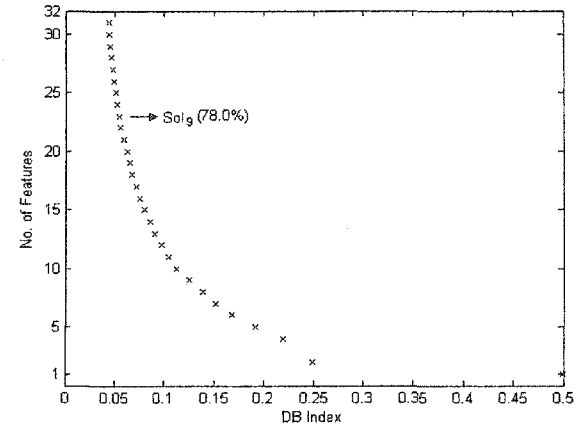


(b)

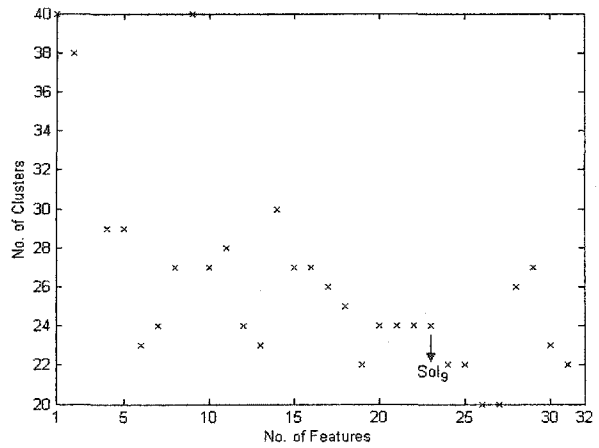


(c)

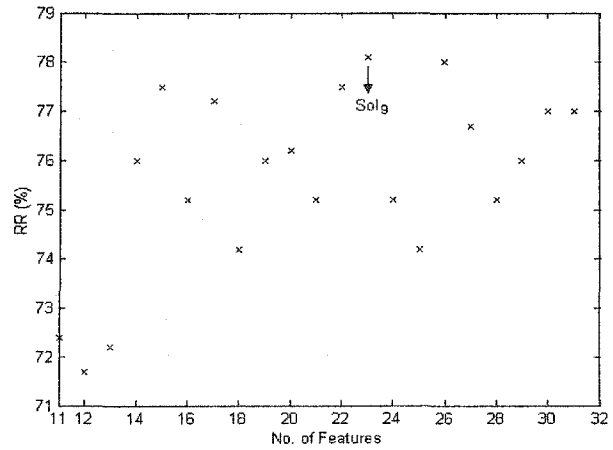
Figure 45 Exp. I: (a) Pareto-optimal front, (b) Number of clusters × number of features, and (c) RR × number of features



(a)

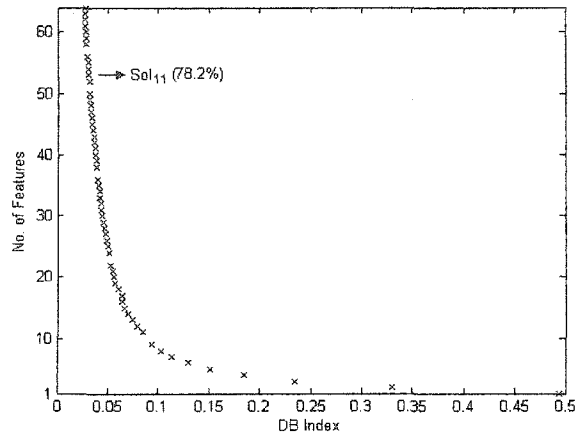


(b)

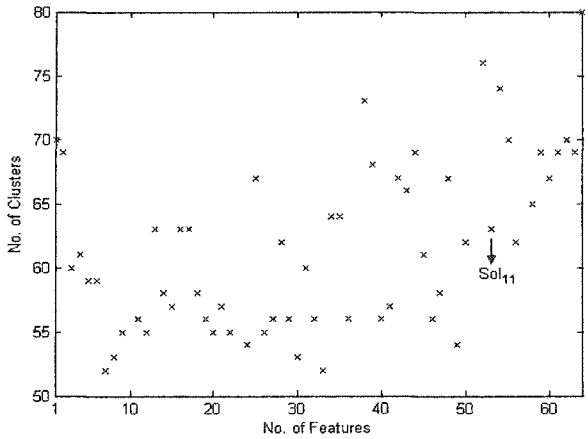


(c)

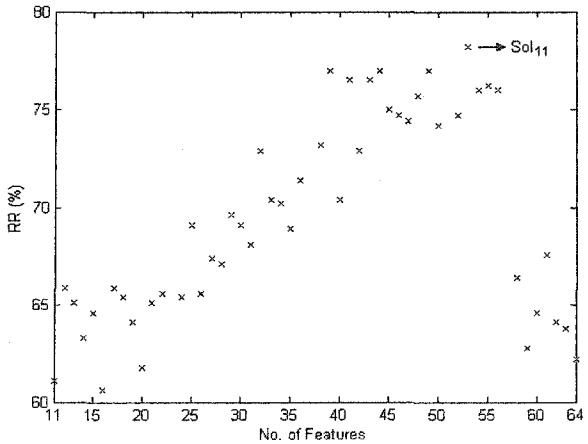
Figure 46 Exp. II: (a) Pareto-optimal front, (b) Number of clusters × number of features, and (c) RR × number of features



(a)



(b)



(c)

Figure 47 Exp III: (a) Pareto-optimal front, (b) Number of clusters × number of features, and (c) RR × number of features

Table XXIII

The best solutions obtained by NSGA and their RRs on month word recognition

Exp.	Solution	No. of Clusters	DB Index	No. of Features	RR VL (%)	RR TS (%)
I	Sol_2	36	0.039582	29	88.6	86.0
II	Sol_9	24	0.054439	23	78.0	71.0
III	Sol_{11}	63	0.030852	53	78.2	77.2

Table XXIV

The best solution obtained by the traditional strategy and their RRs on month word recognition

Exp.	Solution	No. of Clusters	No. of Features	RR (VL) (%)	RR (TS) (%)
I	Sol_4	80	34	88.3	86.2
II	Sol_2	40	32	79.0	73.0
III	Sol_3	60	64	64.3	64.5

and 140. The solution that brought better results in Exp. I was Sol_4 (34 features and 80 clusters), while in Exp. II and III were solutions Sol_2 (32 features and 40 clusters) and Sol_3 (64 features and 60 clusters), respectively. Table XXIV also details the word recognition results reached on the test set. It can be observed that the best word recognition results achieved using our approach are similar or higher to the results reached using the traditional strategy, which makes use of the entire feature vector and tries empirically various number of clusters without performing feature selection. For instance, in Exp. I and II, the results are very closed taking into consideration the small validation and test sets we have used.

The foregoing experiments confirm the efficiency of the proposed methodology in selecting a powerful subset of features and a proper value of the number of clusters.

Besides, it reduced the number of features (Exp. I: from 34 to 29, Exp. II: from 32 to 23, and Exp. III: from 64 to 53) and found proper number of clusters while keeping the recognition rates at the same level or higher as the traditional strategy. We can notice that using our approach the improvements on word recognition were expressive in Exp. III. In this case, it is easier to get better results when the original feature set performs poorly. On the other hand, the main drawback of this methodology is the processing time since for each solution found by NSGA, the K-Means algorithm is executed to form clusters based on the selected features and the selected number of clusters.

7.1.6 Optimization of the Word Verifier of the Date Recognition System

We have seen so far that the proposed strategy was assessed on two synthetic data sets where the number of clusters is known. Then, it was applied to a more complex problem with high-dimension space and where we do not have knowledge about the number of clusters, i.e., the recognition of isolated handwritten month words. Since comprehensive experiments have demonstrated the feasibility and efficiency of the proposed methodology, our next step concerns in optimizing the word verifier of the baseline date recognition system. To accomplish this, we have conducted three experiments using the optimized features obtained in the previous experiments on isolated month word recognition. After that, we compare the results achieved with those obtained from the baseline system.

7.1.6.1 Experimental Results

The three experiments described in this section were carried out on the test set of 401 date images, the same one used by the baseline system. In order to train the word verifier, we have used about 9,500 and 2,300 images of handwritten words for training and validation respectively.

As mentioned before, the word verifier of the baseline system considers two feature sets. The former is global features (20-symbol alphabet), while the latter is a mixture of concavity and contour features (34 features and 100 clusters). Both feature sets are combined with the segmentation primitives.

For the three experiments, the word verifier also makes use of two feature sets. The first experiment (Exp. IV) employs global and the optimized concavity and contour features (29 features and 36 clusters). For the other ones (Exp. V and VI), we consider the optimized DDD and concavity and contour features as well. Table XXV reports the performance of the system using the optimized features and the baseline system. This table details recognition rates (with zero-rejection level) achieved on the validation (VL) and test (TS) sets for date and month word recognition.

Table XXV

Performance on date and word recognition

System	Feature Set	No. of Features	No. of Clusters	Date		Month	
				VL	TS	VL	TS
Baseline	Global	20	20	82.5%	82.5%	93.1%	91.5%
	Conc&Contour	34	100				
Exp. IV	Global	20	20	82.3%	82.7%	92.4%	91.0%
	Conc&Contour (optimized)	29	36				
Exp. V	DDD (optimized)	23	24	82.5%	82.5%	92.9%	91.0%
	Conc&Contour (optimized)	29	36				
Exp. VI	DDD (optimized)	53	63	82.3%	81.2%	93.1%	89.7%
	Conc&Contour (optimized)	29	36				

From Table XXV, we can note that in Exp. IV, by reducing the number of concavity and contour features from 34 to 29 and the number of clusters from 100 to 36, the recognition rates on date and month word recognition were very similar to the baseline system. The same happens when employing the optimized DDD and concavity and contour features. For the next experiments, we fixed the error rates around 2.0% on the test set. Table XXVI presents the recognition (RR), rejection (RJ), and reliability (RL) rates on date recognition. In such cases, we have also reached similar results by comparing with the baseline system.

Table XXVI

Recognition Rates (RR), Rejection Rates (RJ), and Reliability Rates (RL) on date recognition

System	Error $\equiv$ 2.0		
	RR (%)	RJ (%)	RL (%)
Baseline	65.3	32.6	97.0
Exp. IV	66.0	31.6	96.7
Exp. V	65.8	31.9	96.7
Exp. VI	67.3	30.4	96.8

7.1.7 Discussion

In this chapter a methodology for feature selection in unsupervised learning based on multi-objective optimization has been presented. It generates a set of nondominant solutions called Pareto-optimal which corresponds to the best trade-offs between the number of features and quality of clusters. The proposed strategy was evaluated using two synthetic data sets and then applied to handwritten month word recognition in order to optimize the word classifiers as well as the word verifiers of the date recognition system. Even though the main weakness of our approach is related to its processing time, the results achieved show the efficiency of the proposed methodology where the number of features was reduced, a proper number of clusters was found,

and the recognition rates were kept at the same level or higher as the traditional strategy. Therefore, it can be successfully applied to the problem of feature selection in unsupervised learning and extended for use in a supervised learning context as we have demonstrated through experimentation.

CONCLUSION

In this thesis, we have presented an HMM-MLP hybrid system for segmenting and recognizing date images written on Brazilian bank cheques. The system evolves by dealing with many sources of variability, such as heterogeneous data types and styles, variations present in the date field, and difficult cases of segmentation that make the recognizer task particular hard to do.

The proposed system takes an HMM-based strategy for separating the date into sub-fields since we have seen that this is a difficult task without resorting to recognition. Thus, it is not necessary to perform *a priori* segmentation and premature errors can be avoided. Besides, by identifying each date sub-field through the HMMs we can recognize the three obligatory ones using specialized classifiers according to their respective data types which are known. However, the main strength of such an approach lies in the modeling phase. We have built HMMs based on the concept of the meta-classes of digits in order to reduce the lexicon size of the day and year and improve the precision of their segmentation. We have shown difficult cases of segmentation in which our strategy works well.

In order to deal with the heterogeneous data types, we propose to use HMMs and MLPs to work with words and strings of digits respectively. This is justified by the fact that HMMs have been successfully applied to handwritten word recognition and MLPs to digit recognition. Moreover, the literature has shown better results on digit recognition using MLPs [36, 91] than HMMs [46]. Although such methods are very well known, this work presents an interesting strategy for lexicon size reduction that makes use of the output of the HMMs (the number of digits present in a string) and the concept of meta-classes of digits, which are beneficial to the digit string component. We have seen that the use of meta-classes of digits on digit string recognition for 2-digit strings reached encouraging results on NIST SD19 database.

We also have demonstrated that the proposed word verification scheme brought an increase in the overall recognition rate of the system. In order to assess such an approach we have conducted experiments on legal amount database and compared the results reached with an other system that makes use of the same database. The results we achieved show the efficiency of the strategy we developed for word recognition and verification.

Even though the system has achieved good performance taking into account its complexity, we have seen that the system still has different sources of errors that can come from date segmentation, digit segmentation, digit string recognition, grapheme segmentation, word recognition and verification, etc. In order to improve the performance of the system, we have focused on the optimization of the word verifier. We have proposed a wrapper approach for feature selection in unsupervised learning based on multi-objective optimization. It generates a set of nondominant solutions which contain the more discriminant features and the more pertinent number of clusters. Firstly, this strategy was assessed using two synthetic data sets and then it was applied to handwritten month word recognition in order to optimize the word classifiers as well as the word verifiers of the date recognition system. The results achieved show the efficiency of the proposed methodology where the number of features was reduced while the recognition rates were kept at the same level as the traditional strategy. Our approach is the only one we have knowledge that makes use of unsupervised learning in the context of the supervised learning [40].

Future Work

In order to improve the performance of the system, we believe that the following directions can be investigated as future work:

- The algorithm that splits an image into graphemes attempts to avoid a character under-segmentation. Nevertheless, we have seen that in the presence of loops in an image a character under-segmentation can still occur. In order to deal with such cases, some rules to segment loops can be introduced.

- The date segmentation module makes use of two feature sets that feed the discrete HMMs. The former is based on high-level global features, while the second one is based on low-level concavity measurements that are converted into a sequence of symbols by a vector quantization algorithm. Instead of using the traditional strategy for searching a proper number of clusters, the proposed methodology for unsupervised feature selection can be used to find the number of clusters and the more discriminant features. Besides, we can investigate other optimized feature sets to be employed in date segmentation.

- Investigate other optimized feature sets to be used by the word verifier.

- Design a digit verification scheme in order to deal with the main confusions generated by our classifier. It could be interesting to investigate different classifiers and feature sets.

- Regarding the proposed methodology for unsupervised feature selection, we can explore other validity indices that measure the quality of clusters in order to guide the search towards the more discriminant features and the best number of clusters. In addition, we can investigate a mechanism that performs the same search but for different types of zoning instead of one.

APPENDIX 1

Databases

Laboratory Database of Brazilian Bank Cheques

This database was collected at the campus of the Pontifical Catholic University of Paraná (PUCPR) in Curitiba, Brazil. It contains isolated handwritten images of date, courtesy and legal amounts where writers were basically students of the campus. They were written in a separate blank sheet of paper which was put under a form of a Brazilian bank cheque. Figure 48 illustrates an example of a Brazilian bank cheque. In this way, we do not consider any extraction process of these fields from the background of cheques. There were no constraints on the writing style. However, we consider only blue or black pen to fill in the information on the paper. At the moment the database consists of a total of almost 2,000 laboratory cheques. There are about 2,000 handwritten images of date, and the same value for the amounts. All images were acquired in 300 DPI and 256 gray levels. Besides, they were binarized using the Otsu's method [93] described in Chapter 3. A detailed description of this database is reported in [25].

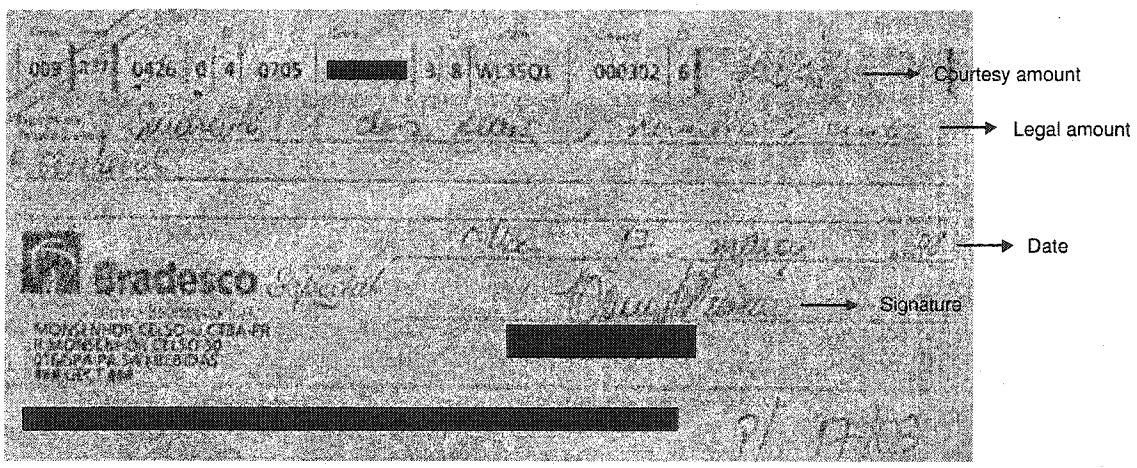


Figure 48 Example of a Brazilian bank cheque

The date database contains about 2,000 date images and it was divided into three sets: 1,182, 396, and 401 images for training (TR), validation (VL), and testing (TS)

respectively. As stated before, the date from left to right can consist of the following sub-fields: city name, first separator (Sep1), day, second separator (Sep2), month, third separator (Sep3) and year. Figure 49 illustrates some samples of handwritten dates. In such images, the grey color represents the obligatory date sub-fields (day, month and year). We can observe from this figure that some writers put their own "De" separators. However, the "De" separators are usually printed on real cheques (see Figure 48). Some statistics of the date database are shown in Table XXVII.

In order to increase the training and validation sets of digits and letters, we have considered the legal (LA) and courtesy (CA) amount databases. For the legal amount database, we have used 8,325 and 1,906 word images for training and validation respectively. Table XXVIII summarizes the distribution of each class of uppercase (U) and lowercase (L) letters presented in the date and legal amount databases. The courtesy amount database was divided into three subsets as shown in Table XXIX. From these subsets we have extracted 14,172 images of isolated digits.

In order to assess the word recognition and verification scheme, we have conducted experiments using 10,430 images of isolated words extracted from the legal amount database. For such experiments, we have divided the legal amount database into three different sets: 6,264, 2,092, and 2,074 images for training, validation, and testing respectively. Table XXX shows the number of each class of words extracted from these subsets and Table XXXI summarizes the distribution of the isolated letters extracted from the training and validation sets.

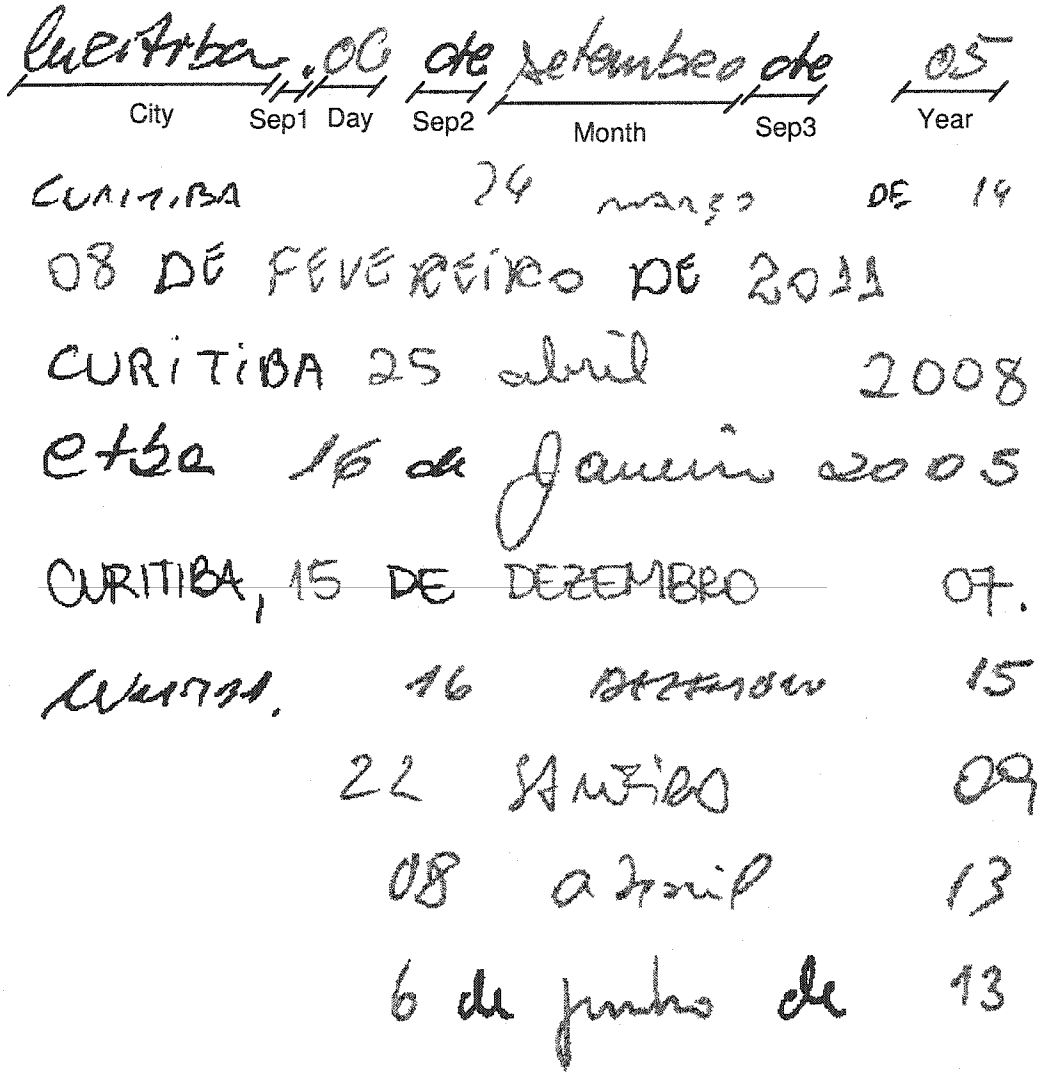


Figure 49 Samples of handwritten date images

Table XXVII

Distribution of each class presented in the date database

Class	TR	VL	TS
Curitiba	848	321	300
Ctba	128	44	55
Total (2 classes of city names)	976	365	355
.	22	7	3
,	419	146	126
Total (2 classes of Sep1)	441	153	129
1-digit day	116	25	21
2-digit day	1,066	371	380
Total (classes of 1- and 2-digit day)	1,182	396	401
De	75	9	18
.	16	2	4
,	3	0	4
Total (3 classes of Sep2)	94	11	26
Janeiro	115	39	39
Fevereiro	94	32	32
Março	105	35	36
Abril	115	38	39
Maio	111	37	38
Junho	86	28	29
Julho	95	33	32
Agosto	84	28	28
Setembro	91	31	31
Outubro	87	29	30
Novembro	99	33	34
Dezembro	100	33	33
Total (12 classes of month words)	1,182	396	401
De	32	1	2
.	2	1	2
,	1	0	0
Total (3 classes of Sep3)	35	2	4
2-digit year	1,085	395	393
4-digit year	97	1	8
Total (classes of 2- or 4-digit year)	1,182	396	401

Table XXVIII

Distribution of each letter presented in the date and legal amount databases

Class	Database					
	Date		LA		Date + LA	
	TR	VL	TR	VL	TR	VL
a (A)	394 (136)	136 (41)	3,276 (683)	761 (163)	3,670 (819)	897 (204)
b (B)	425 (67)	146 (18)	0 (0)	0 (0)	425 (67)	146 (18)
c (C)	86 (19)	25 (10)	1,864 (486)	407 (110)	1,950 (505)	432 (120)
d (D)	72 (28)	24 (9)	437 (161)	92 (39)	509 (189)	116 (48)
e (E)	746 (132)	261 (35)	6,449 (1,248)	1,450 (270)	7,195 (1,380)	1,711 (305)
f (F)	55 (39)	19 (13)	0 (0)	0 (0)	55 (39)	19 (13)
g (G)	73 (11)	22 (6)	0 (0)	0 (0)	73 (11)	22 (6)
h (H)	156 (25)	51 (10)	136 (38)	31 (2)	292 (63)	82 (12)
i (I)	365 (70)	130 (16)	3,680 (636)	868 (150)	4,045 (706)	998 (166)
j (J)	193 (103)	45 (55)	0 (0)	0 (0)	193 (103)	45 (55)
l (L)	173 (37)	63 (8)	711 (167)	182 (39)	884 (204)	245 (47)
m (M)	416 (90)	132 (37)	908 (197)	239 (48)	1,324 (287)	371 (85)
n (N)	241 (59)	75 (25)	4,493 (1,000)	1,024 (200)	4,734 (1,059)	1,099 (225)
o (O)	1,116 (221)	364 (84)	3,992 (888)	842 (191)	5,108 (1,109)	1,206 (275)
q (Q)	0 (0)	0 (0)	722 (316)	173 (81)	722 (316)	173 (81)
r (R)	744 (156)	253 (49)	1,792 (446)	420 (109)	2,536 (602)	673 (158)
s (S)	150 (25)	48 (11)	4,432 (1,145)	1,037 (248)	4,582 (1,170)	1,085 (259)
t (T)	230 (32)	74 (14)	5,585 (1,020)	1,255 (242)	5,815 (1,052)	1,329 (256)
u (U)	297 (58)	100 (19)	1,192 (226)	292 (70)	1,489 (284)	392 (89)
v (V)	161 (32)	57 (8)	1,354 (328)	258 (55)	1,515 (360)	315 (63)
z (Z)	82 (18)	27 (6)	495 (128)	95 (26)	577 (146)	122 (32)
Total L	6,175	2,052	41,518	9,426	47,693	11,478
Total U	1,358	474	9,113	2,043	10,471	2,517

Table XXIX

Isolated digits extracted from the date and courtesy amount databases

Class	Database								
	Date			CA			Date + CA		
	TR	VL	TS	TR	VL	TS	TR	VL	TS
0	1,120	353	350	1,190	210	309	2,310	563	659
1	1,102	385	369	908	160	289	2,010	545	658
2	730	213	240	1,043	184	336	1,773	397	576
3	267	103	96	942	166	316	1,209	269	412
4	218	66	69	897	158	316	1,115	224	385
5	231	66	67	925	163	317	1,156	229	384
6	205	67	74	905	159	302	1,110	226	376
7	258	86	86	929	163	310	1,187	249	396
8	249	91	99	808	142	313	1,057	233	412
9	426	131	149	858	151	303	1,284	282	452
Total	4,806	1,561	1,599	9,405	1,656	3,111	14,211	3,217	4,710

Table XXX

Classes extracted from the legal amount database

Class	TR	VL	TS
Cem	3	1	1
Cento	119	40	40
Cinquenta	177	59	58
Dezenove	23	8	8
Dezessete	20	7	6
Dois	203	68	68
Duzentos	121	41	40
Nove	178	60	60
Noventa	171	57	56
Oito	170	57	54
Onze	28	9	10
Quatorze	17	6	6
Quatrocentos	108	36	35
Quinze	21	7	8
Seis	174	58	58
Sessenta	178	59	60
Seiscentos	111	37	37
Tres	186	62	61
Trezentos	127	42	42
Um	181	61	60
Centavos	535	179	177
Setecentos	121	40	40
Cinco	177	59	59
Dez	27	9	10
Dezesseis	13	4	5
Dezoito	18	6	7
Doze	16	5	6
Mil	671	224	223
Novecentos	112	37	37
Oitenta	187	62	63
Oitocentos	129	43	41
Quarenta	182	61	61
Quatro	164	55	54
Quinhentos	120	40	39
Reais	670	223	222
Sete	169	57	52
Setenta	200	67	66
Treze	20	7	7
Trinta	213	71	69
Vinte	204	68	68
Total (40 classes of words)	6,264	2,092	2,074

Table XXXI

Distribution of each letter presented in the legal amount database

Class	TR	VL
a (A)	2,491 (493)	799 (197)
c (C)	1,406 (351)	460 (125)
d (D)	331 (110)	96 (52)
e (E)	4,899 (878)	1,553 (375)
h (H)	104 (27)	30 (10)
i (I)	2,813 (451)	893 (193)
l (L)	556 (117)	170 (54)
m (M)	713 (142)	220 (66)
n (N)	3,426 (696)	1,075 (300)
o (O)	3,010 (611)	955 (254)
q (Q)	537 (252)	184 (80)
r (R)	1,354 (333)	439 (124)
s (S)	3,371 (838)	1,081 (326)
t (T)	4,241 (722)	1,353 (304)
u (U)	924 (167)	295 (71)
v (V)	1,009 (214)	311 (98)
z (Z)	369 (83)	114 (37)
Total L	31,554	10,028
Total U	6,485	2,666

NIST SD19 Database

The SD19 is composed of 3,669 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.

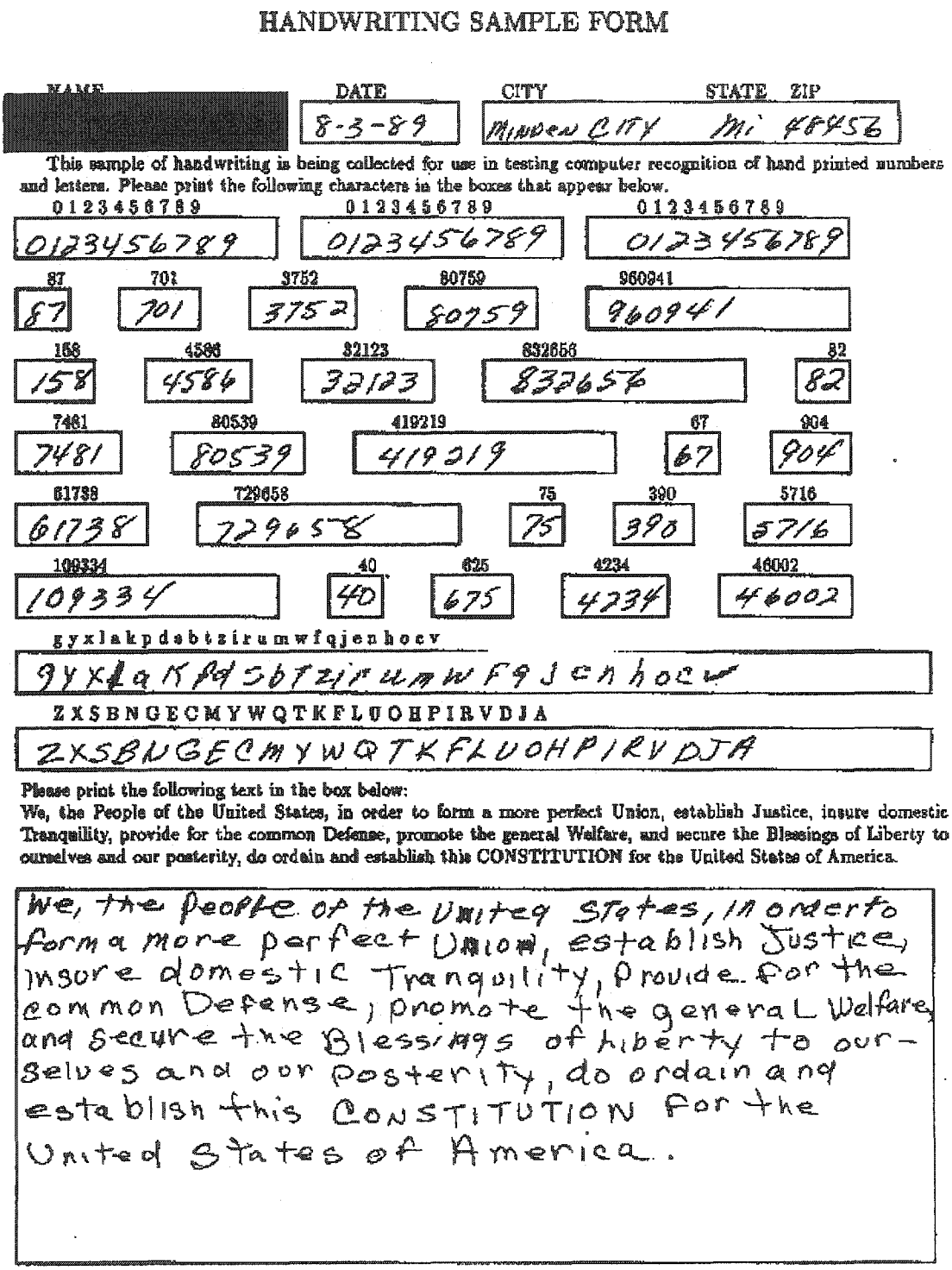
Table XXXII

HSF series distribution

Series	No. 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	3,669

An example of a full-page NIST form or HSF page is shown in Figure 50. 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 XXXIII.

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 [redacted] DATE 8-3-89 CITY Minden City STATE MI ZIP 48456

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

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 e b t z i r u m w f q j e n h o c v

gyxlakpdebtzirumwfqjenhocv

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 ourselves and our Posterity, do ordain and establish this CONSTITUTION for the United States of America.

Figure 50 Handwriting Sample Form (HSF full-page form)

Table XXXIII

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.

Numerical Fields

As part of his doctoral research, Britto Jr. [45] developed a system to extract strings of digits from NIST forms. He built a test set that contains 12,802 distributed into six classes: 2-digit (2,370), 3-digit (2,385), 4-digit (2,345), 5-digit (2,316), 6-digit (2,169), and 10-digit (1,217) string, respectively. These data were extracted from hsf_7 series.

For the experiments conducted on digit string recognition, the test set we have used is a subset of 2-digit strings of hsf_7 series. In this case, the test set contains 986 images of 2-digit strings related to the lexicon of 2-digit day. For training and validation, we have used 195,000 and 28,124 images of isolated digits extracted from hsf_{0,1,2,3}.

APPENDIX 2

Hidden Markov Models

The Hidden Markov Model Concept

A Hidden Markov Model (HMM) is a doubly stochastic process, with an underlying stochastic process that is not observable (hence the word hidden), but can be observed through another stochastic process that produces the sequence of observations [96]. The hidden process consists of a set of states connected to each other by transitions with probabilities, while the observed process consists of a set of outputs or observations, each of which may be emitted by each state according to some output probability density function (pdf). Depending on the nature of this pdf, several kinds of HMMs can be distinguished. If the observations are naturally discrete or quantized using vector quantization [41], and drawn from an alphabet or a codebook, the HMM is called discrete. If these observations are continuous we are dealing with a continuous HMM, with a continuous pdf usually approximated by a mixture of normal distributions. In the context of this thesis we consider discrete HMMs.

In some applications, it is more convenient to produce observations by transitions rather than by states. Furthermore, it is sometimes useful to allow transitions with no output in order to model, for instance, the absence of an event in a given stochastic process. If we add the possibility of using more than one feature set to describe the observations, we must modify the classic formal definition of HMMs [96]. These modifications can be found in [122] and they are also described in this appendix. In this case, the following parameters are required:

- T: length of the observation sequence $O = \{o_0, o_1, \ldots, o_{T-1}\}$, where $o_t = (o_t^0, o_t^1, \ldots, o_t^{P-1})$, the observation o_t^p at time t being drawn from the p^{th} finite feature set, and $p = 0, 1, \ldots, P-1$.
- N: number of states in the model.
- M_p: number of possible observation symbols for the p^{th} feature set.

- $S = \{s_0, s_1, \ldots, s_{N-1}\}$: set of possible states of the model.
- $Q = \{q_t\}$: q_t denotes the state at time t.
- $V_p = \{v_0^p, v_1^p, \ldots, v_{M-1}^p\}$: codebook or discrete set of possible observation symbols corresponding to the p^{th} feature set.
- $A = \{a_{ij}\}$, $a_{ij} = P[q_{t+1} = s_j | q_t = s_i]$: probability of going from state s_i at time t to state s_j at time $(t+1)$, and at the same time producing a real observation o_t at time t.
- $A' = \{a'_{ij}\}$, $a'_{ij} = P[q_t = s_j | q_t = s_i]$: probability of null transition from state s_i at time t to state s_j at time t, producing null observation symbol Φ. Note here that there is no increase over time since no real observation is produced.
- $B_p = \{b_{ij}^p(k)\}$, $b_{ij}^p(k) = P[o_t^p = v_k^p | q_t = s_i, q_{t+1} = s_j]$: output pdf of observing the k^{th} symbol in the p^{th} feature set when a transition from state s_i at time t to state s_j at time $(t+1)$ is taken. If we assume the P output pdfs are independent (multiple codebooks), we can compute the output probability $b_{ij}(k)$ as the product of P output probabilities:

$$b_{ij}(k) = \prod_{p=0}^{P-1} b_{ij}^p(k) \tag{2.1}$$

- $\pi = \{\pi_i\}$, $\pi_i = P[q_0 = s_i]$: initial state distribution. In general, it is more convenient to have predefined initial and final states s_0 and s_{N-1} that do not change over time. In this case, $\pi_0 = 1$ and $\pi_i = 1, 2, \ldots, N-1$.

A, A', B_p, and π obey the stochastic constraints:

$$\sum_{j=0}^{N-1} [a_{ij} + a'_{ij}] = 1 \qquad \sum_{k=0}^{M_p-1} b_{ij}^p(k) = 1 \qquad \sum_{i=0}^{N-1} \pi_i = 1 \tag{2.2}$$

$$p = 0, 1, \ldots, P-1$$

It can be seen that a complete specification of an HMM requires specification of two model parameters, N and M, specification of observation symbols, and the specification of the four sets of probability measures A, A', B_p and π where $p = 0, 1, \ldots, P-1$. For convenience, we use the compact notation $\lambda = (A, A', B_p, \pi)$ to indicate the complete parameter set of the model. This parameter set, of course, defines a probability measure for O, i.e., $P(O|\lambda)$, which we discuss along this Appendix.

Types of HMMs

There are two important types of HMMs: ergodic and left-right or Bakis model [96]. The ergodic model is a specific case of a fully-connected model when all a_{ij} are positive. In this type of model, the states are interconnected in such a way that any state can be reached from any other state. Figure 51(a) shows a 4-state ergodic HMM model. The left-right model presents an important kind of state interconnection for text recognition modeling, which has the property:

$$a_{ij} = 0, \quad j < i \tag{2.3}$$

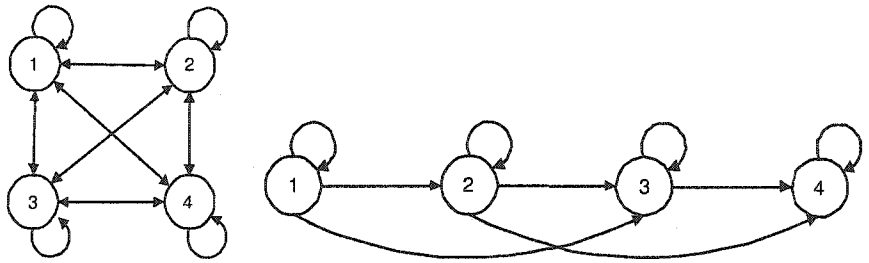
This property means that no transitions are allowed to states whose indices are lower than that of the current state. Since the state sequence must begin in state 0 (and end in state $N-1$), the initial state probabilities have the following property:

$$\pi_i = \begin{cases} 0, & i \neq 0 \\ 1, & i = 0 \end{cases} \tag{2.4}$$

Often, with left-right models, additional constraints are used in order to avoid great changes in state indices, such as:

$$a_{ij} = 0, \quad j > i + \Delta \tag{2.5}$$

The value of Δ is used as a limit for jumps. For instance, in 51(b), Δ is two, that is, no jumps of more than two states are allowed.



(a) (b)

Figure 51 Types of HMMs: (a) Ergodic model, and (b) Left-right model

The Three Basic Problems for HMMs

Given a model, three basic problems of interest must be solved for the model to be useful in real-word applications. These problems are the following:

- The evaluation problem: given an observation sequence $O = (o_0, o_1, \ldots, o_{T-1})$, and a model $\lambda = (A, A', B, \pi)$, how do we compute $P(O|\lambda)$, the probability of O given λ ?
- The decoding problem: given the observation sequence $O = (o_0, o_1, \ldots, o_{T-1})$, and the model λ, how do we find the optimal state sequence in λ that has generated O ?

- The training problem: given a set of observation sequences and an initial model λ, how can we re-estimate the model parameters so as to increase the likelihood of generating this set of sequences ?

The Evaluation Problem

To compute $P(O|\lambda)$, we modify the well-known Forward-Backward procedure [96] to take into account the assumption that symbols are emitted along transitions, the possibility of null transitions, and the use of multiple codebooks. Hence, we define the forward probability $\alpha_t(i)$ as:

$$\alpha_t(i) = P(o_0, o_1, \ldots, o_{t-1}, q_t = s_i|\lambda) \tag{2.6}$$

where $\alpha_t(i)$ is the probability of the partial observation sequence $(o_0, o_1 \ldots, o_{t-1})$ (until time $t-1$) and the state s_i reached at time t given the model λ. $\alpha_t(i)$ can be inductively computed as follows.

1. *Initialization*

$$\alpha_0(0) = 1.0 \tag{2.7}$$

$$\alpha_0(j) = \sum_{i=0}^{N-1} a'_{ij}\alpha_0(i) \quad j = 0, 1, \ldots, N-1$$

given that s_0 is the only possible initial state.

2. *Induction*

$$\alpha_t(j) = \sum_{i=0}^{N-1} \left[a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_{t-1}) \right) \alpha_{t-1}(i) + a'_{ij}\alpha_t(i) \right] \tag{2.8}$$

$$j = 0, 1, \ldots, N-1 \quad and \quad t = 1, 2, \ldots, T$$

3. *Termination*

$$P(O|\lambda) = \alpha_T(N-1) \tag{2.9}$$

given that s_{N-1} is the only possible terminal state. Similarly, we define the backward probability $\beta_t(i)$ by:

$$\beta_t(i) = P(o_t, o_{t+1}, \ldots, o_{T-1}|q_t = s_i, \lambda) \tag{2.10}$$

where $\beta_t(i)$ is the probability of the partial observation sequence from the time t to the end, given state s_i reached at time t and the model λ. $\beta_t(i)$ can also be inductively computed as follows.

1. *Initialization*

$$\beta_T(N-1) = 1.0 \tag{2.11}$$

$$\beta_T(i) = \sum_{j=0}^{N-1} a'_{ij}\beta_T(j) \quad i = 0, 1, \ldots, N-1$$

given that s_{N-1} is the only possible terminal state.

2. *Induction*

$$\beta_t(i) = \sum_{j=0}^{N-1} \left[a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t) \right) \beta_{t+1}(j) + a'_{ij}\beta_t(j) \right] \tag{2.12}$$

$$t = T-1, T-2, \ldots, 0 \quad \textit{and} \quad i = 0, 1, \ldots, N-1$$

3. *Termination*

$$P(O|\lambda) = \beta_0(0) \tag{2.13}$$

given that q_0 is the only possible initial state.

The Decoding Problem

The decoding problem is solved using a near-optimal procedure, the Viterbi algorithm, by looking for the best state sequence $Q = (q_0, q_1, \ldots, q_T)$ for the given observation sequence $O = (o_0, o_1, \ldots, o_{T-1})$. Again, we modify the classic algorithm [96] in the following way.

$$\delta_t(i) = \max_{q_0, q_1, \ldots, q_{t-1}} P[q_0, q_1, \ldots, q_t, q_t = s_i, o_0, o_1, \ldots, o_{t-1}|\lambda] \tag{2.14}$$

where $\delta_t(i)$ is the probability of the best path that accounts for the first t observations and ends at state s_i at time t. The function $\Psi_t(i)$ is defined to recover the best state sequence by a procedure called Backtracking. $\Psi_t(i)$ e $\delta_t(i)$ can be recursively computed as follows.

1. *Initialization*

$$\delta_0(0) = 1.0 \tag{2.15}$$

$$\Psi_0(0) = 0$$

$$\delta_0(j) = \max_{0 \le i \le N-1} [\delta_0(i) a'_{ij}] \quad j = 0, 1, \ldots, N-1$$

$$\Psi_0(j) = \arg \max_{0 \le i \le N-1} [\delta_0(i) a'_{ij}] \quad j = 0, 1, \ldots, N-1$$

given that s_0 is the only possible initial state.

2. *Recursion*

$$\delta_t(j) = \max_{0 \le i \le N-1} \left[a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_{t-1}) \right) \delta_{t-1}(i); a'_{ij} \delta_t(i) \right] \tag{2.16}$$

$$t = 1, 2, \ldots, T \quad \textit{and} \quad j = 0, 1, \ldots, N-1$$

$$\Psi_t(j) = \arg \max_{0 \le i \le N-1} \left[a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_{t-1}) \right) \delta_{t-1}(i); a'_{ij} \delta_t(i) \right] \tag{2.17}$$

$$t = 1, 2, \ldots, T \quad \textit{and} \quad j = 0, 1, \ldots, N-1$$

3. *Termination*

$$P^* = \delta_T(N-1) \tag{2.18}$$

$$q_T^* = (N-1) \tag{2.19}$$

given that s_{N-1} is the only possible terminal state.

4. *Backtracking procedure*

$$q_t^* = \Psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 0 \tag{2.20}$$

As shown above, except for the Backtracking procedure, Viterbi and Forward procedures are similar. The only difference is that the summation is replaced by maximization.

The Training Problem

The main strength of HMMs is the existence of a procedure called the Baum-Welch algorithm [43, 96] that iteratively and automatically adjusts HMM parameters given a training set of observation sequences. This algorithm, which is an implementation of the Expectation-Maximization algorithm [80], guarantees the model to converge to a local maximum of the probability of observation of the training set according to the maximum likelihood estimation criterion. This maximum depends strongly on the initial parameters.

To re-estimate HMM parameters, we first define $\xi_t^1(i,j)$, the probability of being in state s_i at time t and in state s_j at time $(t+1)$, producing a real observation O_t, given the model and the observation O, and $\xi_t^2(i,j)$, the probability of being in state i at time t and in state j at time t, producing the null observation Φ, given the model and the observation O.

$$\xi_t^1(i,j) = P(q_t = s_i, q_{t+1} = s_j | O, \lambda) \tag{2.21}$$

$$\xi_t^2(i,j) = P(q_t = s_i, q_t = s_j | O, \lambda) \tag{2.22}$$

The development of these quantities leads to:

$$\xi_t^1(i,j) = \frac{\alpha_t(i) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t) \right) \beta_{t+1}(j)}{P(O|\lambda)} \tag{2.23}$$

$$\xi_t^2(i,j) = \frac{\alpha_t(i) a'_{ij} \beta_t(j)}{P(O|\lambda)} \tag{2.24}$$

We also define $\gamma_t(i)$ as the probability of being in state s_i at time t, given the model and the observation sequence.

$$\gamma_t = P(q_t = s_i | O, \lambda) \tag{2.25}$$

$\gamma_t(i)$ is related to $\xi_t^1(i,j)$ and $\xi_t^2(i,j)$ by the following equation:

$$\gamma_t = \sum_{j=0}^{N-1} [\xi_t^1(i,j) + \xi_t^2(i,j)] = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \tag{2.26}$$

The re-estimations of HMM parameters $\{a_{ij}\}$, $\{a'_{ij}\}$ e $\{b_{ij}^p(k)\}$ are:

$$\overline{a_{ij}} = \frac{\text{expected number of transitions from } s_i \text{ at time } t \text{ to } s_j \text{ at time } (t+1)}{\text{expected number of being in } s_i} \tag{2.27}$$

$$\overline{a'_{ij}} = \frac{\text{expected number of transitions from } s_i \text{ at time } t \text{ to } s_j \text{ at time } t \text{ and observing } \Phi}{\text{expected number of being in } s_i} \tag{2.28}$$

$$\overline{b_{ij}^p(k)} = \frac{\text{expected number of symbols } v_k^p \text{ in transition from } s_i \text{ at time } t \text{ to } s_j \text{ at time } (t+1)}{\text{expected number of transitions in } s_i \text{ at time } t \text{ to } s_j \text{ at time } (t+1)} \tag{2.29}$$

Given the definitions of $\xi_t^1(i,j)$, $\xi_t^2(i,j)$, and $\gamma_t(i)$, it is easy to see, when we are using one observation sequence O:

$$\overline{a_{ij}} = \frac{\sum_{t=0}^{T-1} \xi_t^1(i,j)}{\sum_{t=0}^{T} \gamma_t(i)} = \frac{\sum_{t=0}^{T-1} \alpha_t(i) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t)\right) \beta_{t+1}(j)}{\sum_{t=0}^{T} \alpha_t(i)\beta_t(i)} \tag{2.30}$$

$$\overline{a'_{ij}} = \frac{\sum_{t=0}^{T-1} \xi_t^2(i,j)}{\sum_{t=0}^{T} \gamma_t(i)} = \frac{\sum_{t=0}^{T-1} \alpha_t(i) a'_{ij} \beta_t(j)}{\sum_{t=0}^{T} \alpha_t(i)\beta_t(i)} \tag{2.31}$$

$$\overline{b_{ij}^p(k)} = \frac{\sum_{t=0}^{T-1} \delta(o_t^p, v_k^p) \xi_t^1(i,j)}{\sum_{t=0}^{T-1} \xi_t^1(i,j)} = \frac{\sum_{t=0}^{T-1} \delta(o_t^p, v_k^p) \alpha_t(i) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t)\right) \beta_{t+1}(j)}{\sum_{t=0}^{T-1} \alpha_t(i) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t)\right) \beta_{t+1}(j)} \tag{2.32}$$

where $\delta(x,y) = \begin{pmatrix} 1 & if x = y \\ 0 & if x \neq y \end{pmatrix}$

For a set of training sequences $O(0), O(1), \ldots, O(U-1)$ (size U), as is usually the case in real world applications, the above formulas become:

$$\overline{a_{ij}} = \frac{\sum_{u=0}^{U-1} \sum_{t=0}^{T-1} \xi_t^1(i,j,u)}{\sum_{u=0}^{U-1} \sum_{t=0}^{T} \gamma_t(i,u)} = \frac{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T-1} \alpha_t(i,u) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t^p(u))\right) \beta_{t+1}(j,u)}{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T} \alpha_t(i,u)\beta_t(i,u)} \tag{2.33}$$

$$\overline{a'_{ij}} = \frac{\sum_{u=0}^{U-1} \sum_{t=0}^{T-1} \xi_t^2(i,j,u)}{\sum_{u=0}^{U-1} \sum_{t=0}^{T} \gamma_t(i,u)} = \frac{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T-1} \alpha_t(i,u) a'_{ij} \beta_t(j,u)}{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T} \alpha_t(i,u)\beta_t(i,u)} \tag{2.34}$$

$$\overline{b_{ij}^p(k)} = \frac{\sum_{u=0}^{U-1} \sum_{t=0}^{T-1} \delta(o_t^p(u), v_k^p) \xi_t^1(i,j,u)}{\sum_{u=0}^{U-1} \sum_{t=0}^{T-1} \xi_t^1(i,j,u)} \tag{2.35}$$

$$\overline{b_{ij}^p(k)} = \frac{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T-1} \delta(o_t^p(u), v_k^p) \alpha_t(i,u) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t^p(u)) \right) \beta_{t+1}(j,u)}{\sum_{u=0}^{U-1} \frac{1}{P(u)} \sum_{t=0}^{T-1} \alpha_t(i,u) a_{ij} \left(\prod_{p=0}^{P-1} b_{ij}^p(o_t^p(u)) \right) \beta_{t+1}(j,u)}$$

In the above equations, the index u is introduced into α, β, ξ^1, ξ^2, and γ to indicate the observation sequence $O(u)$ currently used. Note that a new quantity $P(u) = P(O(u)|\lambda)$ appears, since this term is now included in the summation and cannot be eliminated as before.

If we define the current model as $\lambda = (A, A', B_p, \pi)$ and the re-estimated model as $\overline{\lambda} = (\overline{A}, \overline{A'}, \overline{B_p}, \overline{\pi})$, and we iteratively use $\overline{\lambda}$ in place of λ and repeat the re-estimation calculation, we can then improve the probability of O being observed from the model until some limiting point is reached. The final result of this re-estimation procedure is a maximum likelihood estimate of the HMM. It should be pointed out that the Forward-Backward algorithm leads to a local maxima only, and that in most problems of interest, the likelihood function is very complex and has many local maxima.

APPENDIX 3

Weighted Least Square Approach

The approach we have used to eliminate the undesirable maximum (minimum) points is based on the weighted least square technique [81] and it works as follows. After the maxima (minima) have been detected, they are adjusted to a straight line. A common way would be apply the least square method using the mathematical model $y = ax + b$, where a is the skew of the straight line, b is the intercept, and x and y are the coordinates of the maxima (minima).

The criterion of minimizing the sum of the squares of the remainder, v_k, must be applied when there are more than two maxima (minima) in the handwriting, i.e., $n > 2$. In mathematical notation, the least square method is defined by:

$$minimize(\sum_{k=1}^{n} v_k^2) = minimize_{(a,b\in\Re)}(\sum_{k=1}^{n}[ax_k + b - y_k]^2) \tag{3.1}$$

When different weights are applied to the remainder, the weighted least square method is defined by minimizing:

$$\sum_{k=1}^{n} v_k^2 p_k \tag{3.2}$$

where p_k is the weight of the corresponding remainder v_k. The parameters are obtained, both for the weighted case and for the case in which the weights are considered units, by deriving the above equation in relation to the parameters and equaling them to zero.

The method we have used to detect the undesirable maxima (minima) is known in geodesy as changing weights [65]. It consists in decreasing, at each iteration, the weight of the points of the remainder which surpasses a given pre-fixed iteration value. One of the weighting function method [65] is:

$$p_{k+1} = p_k \begin{cases} p_k e^{-(\frac{|v_k|}{3\sigma})} & if |v_k| \geq 3\sigma \\ p_k & if |v_k| < 3\sigma \end{cases} \quad with \quad \sigma = \sqrt{\frac{\sum_{k=1}^{n} p_k v_k^2}{n-2}} \tag{3.3}$$

where σ is the standard deviation.

As a rule, the first iteration is performed without weights. In our case, a different approach was adopted to increase the efficiency of the method. Such approach consists initially in defining and employing weights since the first iteration. The criterion adopted to define the weights p_k, with $k = 1, \ldots, n$, is:

$$p_k = \frac{1}{d_k^2} \tag{3.4}$$

where $d_k = \frac{\delta_k}{min(\delta_k)}$, with $k = 1, \ldots, n$ and $\delta_k = (y_k - y_{k-1})^2 + (y_k - y_{k+1})^2$.

Then, the weight of the remainders greater than the standard deviation value is decreased for the first two iterations. For the next iterations, the criterion adopted is to decrease the weight of the remainders surpassing 3σ. Furthermore, better results were achieved using 2σ instead of 3σ in the weighting function exponential denominator.

APPENDIX 4

Vector Quantization Process

To use the HMMs described in Appendix 2, we need to represent an image as a sequence of discrete symbols. To do so, each feature vector extracted from the image that contains real values (low-level-feature) needs to be quantized to one of the discrete symbols (high-level feature) available in a previously computed codebook. To create this codebook, it is necessary to apply the concept of vector quantization [41].

Let us assume that $X = [X_1, X_2, \ldots, X_d]$ is a d-dimensional vector whose components $\{X_k, 1 \leq k \leq d\}$ are real-valued, continuous-amplitude random variables. In vector quantization, the vector X is mapped onto another real-valued, discrete-amplitude d-dimensional vector Z. It is used to say that X is quantized as Z. This can be denoted by:

$$Z = q(X) \tag{4.1}$$

where $q()$ is the quantization operator. Z takes one of a finite set of values $W = \{Z_i, 1 \leq i \leq L\}$ where $Z_i = [Z_{i1}, Z_{i2}, \ldots, Z_{id}]$. The set W is referred to as the codebook, L is the size of codebook (or number of levels in the codebook), and $\{Z_i\}$ is the set of codewords. To design a codebook, we divide the d-dimensional space of the original random vector X into L regions or cells $\{C_i, 1 \leq i \leq L\}$ and associate with each cell C_i a vector Z_i. The quantizer then assigns the codeword Z_i if X is in C_i.

$$q(X) = Z_i, \quad if \quad X \in C_i \tag{4.2}$$

The mapping of X onto Z results in a quantization error, and a distortion measure $d(X, Z)$ can be defined between X and Z to measure the quality of quantization.

In this case, we have used the Euclidean distance which is the most commonly used distortion measure.

Quantization is optimal when the overall average distortion defined in Equation 4.5 is minimized over all L-levels of the quantizer. There are two necessary conditions for optimality. The first condition is that the optimal quantizer is realized by using a nearest neighbor selection rule.

$$q(X) = Z_i, \quad iff \quad d(X, Z_i) \leq d(X, Z_j), \quad j \neq i, \quad 1 \leq j \leq L \tag{4.3}$$

This means that the quantizer selects the codeword vector that results in the minimum distortion with respect to X. The second condition for optimality is that each codeword Z_i is chosen to minimize the average distortion in cell C_i. Let us consider $\{X(n), 1 \leq n \leq M\}$ as a set of training vectors and K_i as a subset of vectors located in cell C_i. The average distortion D_i in cell C_i and the overall average distortion $D_{overall}$ are then given by:

$$D_i = \frac{1}{K_i} \sum_{X \in C_i} d(X, Z_i) \tag{4.4}$$

$$D_{overall} = \sum_{i=1}^{L} D_i \tag{4.5}$$

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

$$Z_i = cent(C_i) \tag{4.6}$$

Z_i is then obtained from:

$$Z_i = \frac{1}{K_i} \sum_{X \in C_i} X \tag{4.7}$$

One well-known method for codebook design is an iterative clustering algorithm known in the pattern recognition literature as the K-Means algorithm [41]. The basic idea of such an algorithm is to divide the set of training vectors into L clusters $C_i\{1 \leq i \leq L\}$ in such a way that the two necessary conditions for optimality are satisfied. The algorithm can be described as follows:

- Initialization step: choose randomly a set of initial centroids $(Z_i, 1 \leq i \leq L)$.
- Classification step: classify each element of training vectors $X(n)$ into one of the clusters C_i by choosing the nearest codeword $Z_i(X \in C_i, iff \quad d(X, Z_i) \leq D(X, Z_j) \quad for \quad all \quad j \neq i)$.
- Codebook updating step: update the codeword of each cluster by computing the centroid of the training vectors in each cluster $(Z_i = cent(C_i), 1 \leq i \leq L)$.
- Termination step: if the decrease in the overall distortion $D_{overall}$ at the current iteration relative to the overall distortion at the previous iteration is below a certain threshold, stop; otherwise go to the classification step.

APPENDIX 5

Directional Distance Distribution

To each of the pixels in the binary input pattern map (Figure 52), two sets of 8 bytes which we call the W (White) set and B (Black) set are allocated as shown in Figure 53. 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)).

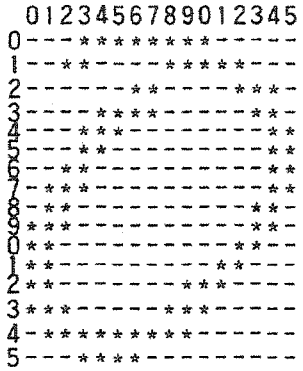


Figure 52 A sample pattern

For the sample pattern in Figure 52, the pixel at coordinates (8,2) will have the WB encoding at the top of Figure 53. 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.

Figure 53 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

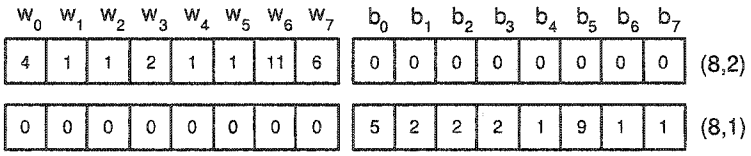


Figure 53 An example of WB encoding

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. For more details see [89].

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] A. Arika and F. T. Yarman-Vural. Optical character recognition for cursive handwriting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):801–813, 2002.

[3] 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, August 2002.

[4] S. Bandyopadhyay and U. Maulik. Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition*, 35:1197–1208, 2002.

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

[6] H. Bunke, M. Roth, and E. G. Schukat-Talamazzini. Off-line cursive handwriting recognition using hidden markov models. *Pattern Recognition*, 28(9):1399–1413, 1995.

[7] 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, August 2002.

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

[9] R. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7), 1996.

[10] Y. K. Chen and J. F. Wang. Segmentation of single- or multiple-touching handwritten numeral string using background and foregound analysis. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 22(11):1304–1317, 2000.

[11] W. Cho, S. W. Lee, and J. H. Kim. Modeling and recognition of cursive words with hidden markov models. *Pattern Recognition*, 28(12):1941–1953, 1995.

[12] D. L. Davies and W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:224–227, 1979.

[13] J. J. de Oliveira Jr., J. M. de Carvalho, C. Freitas, and R. Sabourin. Evaluating nn and hmm classifiers for handwritten word recognition. In *Proc. of 15^{th} Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, Fortaleza-Brazil, October 2002. IEEE Computer Society.

[14] J. J. de Oliveira Jr., J. M. de Carvalho, C. Freitas, and R. Sabourin. Feature sets evaluation for handwritten word recognition. In *Proc. 8^{th} International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, pages 446–451, Niagara on the Lake-CA, August 2002.

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

[16] M. Devaney and A. Ram. Efficient feature selection in conceptual clustering. In *Proc. of 14^{th} International Conference on Machine Learning*, pages 92–97, Morgan Kaufmann, 1997.

[17] 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.

[18] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* John Wiley and Sons, second edition edition, 2001.

[19] J. G. Dy and C. E. Brodley. Feature subset selection and order identification for unsupervised learning. In *Proc. 17^{th} International Conference on Machine Learning*, Stanford University-CA, July 2000.

[20] 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.

[21] R. Fan, L. Lam, and C. Y. Suen. Processing of date information on cheques. In *Proc. 5th International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, pages 207–212, Colchester-U.K, September 1996.

[22] J. T. Favata, S. N. Srihari, and V. Govindaraju. Off-line handwritten sentence recognition. In *Proc. 6th International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, pages 171–176, Taegon-Korea, August 1998.

[23] C. Freitas. *O Uso de Modelos Escondidos de Markov para Reconhecimento de Palavras Manuscritas.* PhD thesis, Pontifícia Universidade Católica do Parana, Curitiba-Brazil, 2001.

[24] C. Freitas, F. Bortolozzi, and R. Sabourin. Handwritten isolated word recognition: An approach based on mutual information for feature set validation. In *Proc. 6th International Conference on Document Analysis and Recognition (ICDAR)*, pages 665–669, Seattle-USA, September 2001.

[25] C. Freitas, M. Morita, L. S. Oliveira, E. Justino, A. Yacoubi, E. Lethelier, F. Bortolozzi, and R. Sabourin. Base de dados de cheques bancarios brasileiros. In *Proc. of XXVI Conferencia Latinoamericana de Informatica*, Atizapan de Zaragoza-Mexico, 2000.

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

[27] 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.

[28] 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 Transactions on Fuzzy Systems*, 3(1):83–95, 1995.

[29] J. Ghosh. Multiclassifier systems: Back to the future. In *Proc. 3rd International Workshop on Multiple Classifier Systems*, pages 1–15, Cagliari-Italy, June 2002.

[30] A. M. Gillies. Cursive word recognition using hidden markov models. In *Proc. Fifth U.S. Postal Service Advanced Technology Conference*, pages 557–562, 1992.

[31] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, Mass., Addison-Wesley, 1989.

[32] N. Gorski, V. Anisimov, E. Augustin, O. Baret, and S. Maximov. Industrial

bank chech processing: the A2iA checkreader. *International Journal on Document Analysis and Recognition*, 3:196–206, 2001.

[33] F. Grandidier. *Un Nouvel Algorithme de Sélection de Caractéristiques- Application à la Lecture Automatique de l'ecriture Manuscrite.* PhD thesis, École de Technologie Supérieure, Montreal-Canada, Janvier 2003.

[34] D. Guillevic and C. Y. Suen. Cursive script recognition applied to the processing of bank cheques. In *Proc. of 3th International Conference on Document Analysis and Recognition*, pages 11–14, Montreal-Canada, August 1995.

[35] D. Guillevic and C. Y. Suen. HMM-KNN word recognition engine for bank cheque processing. In *Proc. of 14th International Conference on Pattern Recognition (ICPR)*, pages 1526–1529, Brisbane-Australia, August 1998.

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

[37] 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.

[38] K. Han and I. K. Sethi. An off-line cursive handwritten word recognition system and its aplication to legal amount interpretation. In S.Impedovo et al, editor, *International Journal of Pattern Recognition and Artificial Intelligence*, pages 757–770. World Scientific, 1997.

[39] L. Heutte, J. Moreau, B. Plessis, J. Plagmaud, and Y. Lecourtier. Handwritten numeral recognition based on multiple feature extractors. In *Proc. 2nd International Conference on Document Analysis and Recognition (ICDAR)*, pages 167–170, Tsukuba-Japan, October 1993.

[40] T. K. Ho. Multiple classifier combination: Lessons and next steps. In A. Kandel and H. Bunke, editors, *Hybrid Methods in Pattern Recognition*, pages 171–198. World Scientific, 2002.

[41] X. D. Huang, Y. Ariki, and M. A. Jack. *Hidden Markov Models for Speech Recognition.* Edinburgh Univ. Press, 1990.

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

[43] F. Jelinek. *Statistical Methods for Speech Recognition.* MIT Press, 1997.

[44] 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.

[45] A. Britto Jr. *A Two-stage HMM-Based Method for Recognizing Handwritten Numeral Strings.* PhD thesis, Pontifícia Universidade Católica do Parana, Curitiba-Brazil, 2001.

[46] A. Britto Jr., R. Sabourin, F. Bortolozzi, and C. Y. Suen. A two-stage hmm-based system for recognizing handwritten numeral strings. In *Proc. 6th International Conference on Document Analysis and Recognition (ICDAR)*, pages 396–400, Seattle-USA, September 2001.

[47] 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, August 2002.

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

[49] 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.

[50] G. Kim and V. Govindaraju. Handwritten phrase recognition as applied to street name images. *Pattern Recognition*, 31(1):41–51, 1998.

[51] H. Y. Kim and J. h. Kim. Handwritten korean character recognition based on hierarchical randon graph modeling. In *Proc. 6th International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, pages 557–586, Taegon-Korea, August 1998.

[52] J. H. Kim, K. K. Kim, and C. Y. Suen. An HMM-MLP hybrid model for cursive script recognition. *Pattern Analysis and Applications*, 3:314–324, 2000.

[53] Y. S. Kim, W. N. Street, and F. Menczer. Feature selection in unsupervised learning via evolutionary search. In *Proc. 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 365–369, Boston-USA, August 2000.

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

[55] S. Knerr and E. Augustin. A neural network-hidden markov model hybrid for cursive word recognition. In *Proc. of 14th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 1518–1520, Brisbane-Australia, August 1998.

[56] A. L. Koerich. *Large Vocabulary Off-Line Handwritten Word Recognition.* PhD thesis, École de Technologie Supérieure, Montreal-Canada, August 2002.

[57] M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, 33:25–41, 2000.

[58] S. Kumar, J. Ghosh, and M. M. Crawford. Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis and Applications*, 5(2):210–220, 2002.

[59] A. Kundu, Y. He, and M. Chen. Alternatives to variable duration hmm in handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1275–1280, 2002.

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

[61] 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.

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

[63] 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.

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

[65] A. Leick. *GPS Satellite Surveying.* John Wiley and Sons, second edition edition, 1995.

[66] M. Leroux. *Reconnaissance de Textes Manuscrits à Vocabulaire Limité avec Application à la Lecture Automatique des Chèques.* PhD thesis, Université de Rouen, France, Décembre 1991.

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

[68] 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.

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

[70] U. Mahadevan and R. C. Nagabushnam. Gaps matrices for word separation in handwritten lines. In *Proc. 3th International Conference on Document Analysis and Recognition (ICDAR)*, pages 124–127, Montreal-Canada, August 1995.

[71] U. Marti and H. Bunke. Text line segmentation and word recognition in a system for general writer independent handwriting recognition. In *Proc. 6th International Conference on Document Analysis and Recognition (ICDAR)*, pages 159–163, Seattle-USA, September 2001.

[72] U. Marti and H. Bunke. Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system. *International Journal of Pattern Recognition and Artifical Intelligence*, 15(1):65–90, 2001.

[73] 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.

[74] 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.

[75] 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.

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

[77] 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.

[78] P. Mitra, C. A. Murthy, and S. K. Pal. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):301–312, 2002.

[79] M. A. Mohamed and P. Gader. Handwritten word recognition using segmentation-free hidden markov modeling and segmentation-based dynamic programming techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):548–554, 1996.

[80] T. K. Moom. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, pages 42–52, 1996.

[81] M. Morita, J. Facon, F. Bortolozzi, S. J. A. Garnes, and R. Sabourin. Mathematical morphology and weighted least squares to correct handwriting baseline skew. In *Proc. 5^{th} International Conference on Document Analysis and Recognition (ICDAR)*, pages 430–433, Bangalore-India, September 1999.

[82] M. Morita, E. Lethelier, A. El Yacoubi, F. Bortolozzi, and R. Sabourin. An hmm-based approach for date recognition. In *Proc. Fourth IAPR International Workshop on Document Analysis Systems (DAS)*, pages 233–244, Rio de Janeiro-Brazil, December 2000.

[83] M. Morita, E. Lethelier, A. El Yacoubi, F. Bortolozzi, and R. Sabourin. Recognition of handwritten dates on bankchecks using an hmm approach. In *Proc. 13^{th} Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, pages 113–120, Gramado-Brazil, October 2000.

[84] M. Morita, L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C.Y. Suen. An HMM-MLP hybrid system to recognize handwritten dates. In *Proc. International Joint Conference on Neural Networks (IJCNN)*, pages 867–872, Honolulu-USA, May 2002.

[85] M. Morita, R. Sabourin, F. Bortolozzi, and C.Y. Suen. Segmentation and recognition of handwritten dates. In *Proc. 8th International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, pages 105–110, Niagara on the Lake-CA, August 2002.

[86] M. Morita, R. Sabourin, F. Bortolozzi, and C.Y. Suen. A recognition and verification strategy for handwritten word recognition. To appear in Proc. 7th International Conference on Document Analysis and Recognition (ICDAR), 2003.

[87] M. Morita, R. Sabourin, F. Bortolozzi, and C.Y. Suen. Unsupervised feature selection using multi-objective genetic algorithms for handwritten word recognition. To appear in Proc. 7th International Conference on Document Analysis and Recognition (ICDAR), 2003.

[88] M. Morita, A. El Yacoubi, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Handwritten month word recognition on Brazilian bank cheques. In *Proc. 6th International Conference on Document Analysis and Recognition (ICDAR)*, pages 972–976, Seattle-USA, September 2001.

[89] 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.

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

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

[92] 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 (ICPR)*, volume 1, pages 568–571, Quebec City-Canada, August 2002.

[93] N. Otsu. A threshold selection method from gray-level histogram. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[94] J. Park and V. Govindaraju. Use of adaptive segmentation in handwritten

phrase recognition. *Pattern Recognition*, 35:245–252, 2002.

[95] J. Park, V. Govindaraju, and S. N. Srihari. Efficient word segmentation driven by unconstrained handwritten phrase recognition. In *Proc. 5th International Conference on Document Analysis and Recognition (ICDAR)*, pages 605–608, Bangalore-India, September 1999.

[96] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of IEEE*, 77(2):257–286, 1989.

[97] S. Rao. *Optimization Theory and Application.* New Delhi:Wiley Eastern Limited, 1991.

[98] C. Scagliola. Search algorithms for the recognition of cursive phrases without word segmentation. In *Proc. 6th International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, pages 123–132, Taegon-Korea, August 1998.

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

[100] G. Seni and E. Cohen. External word segmentation of off-line handwritten text lines. *Pattern Recognition*, 27(1):41–52, 1994.

[101] A. W. Senior and A. J. Robinson. An off-line cursive handwriting recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):309–321, 2002.

[102] 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 (ICPR)*, volume 2, pages 455–458, 1997.

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

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

[105] S. N. Srihari, V. Govindaraju, and A. Shekhawat. Interpretation of handwritten address in us mailstream. In *Proc. 2nd International Conference on Document Analysis and Recognition (ICDAR)*, pages 291–294, Tsukuba-Japan, October 1993.

[106] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sort-

ing in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1995.

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

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

[109] C. Y. Suen, Q. Xu, and L. Lam. Automatic recognition of handwritten data on cheques - fact or fiction ? *Pattern Recognition Letters*, 20(13):1287–1295, 1999.

[110] H. Takahashi and T. D. Griffin. Recognition enhancement by linear tournament verification. In *Proc. 2nd International Conference on Document Analysis and Recognition (ICDAR)*, pages 585–588, Tsukuba-Japan, October 1993.

[111] L. Talavera. Feature selection as a preprocessing step for hierarchical clustering. In *Proc. 16th International Conference on Machine Learning*, pages 389–397, Morgan Kaufmann, 1999.

[112] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York-USA., 1995.

[113] J. Vesanto and E. Alhoniemi. Clustering of the self-organization map. *IEEE Transactions on Neural Networks*, 11(3):586–600, 2000.

[114] X. Wang, V. Govindaraju, and S. Srihri. Holistic recognition of handwritten character pairs. *Pattern Recognition*, 33:1967–1973, 2000.

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

[116] S. L. Xie and M. Suk. On machine recognition of hand-printed chinese character by feature relaxation. *Pattern Recognition*, 21(1):1–7, 1988.

[117] Q. Xu. *Automatic Segmentation and Recognition System for Handwritten Dates on Cheques.* PhD thesis, Concordia University, Montreal-Canada, December 2002.

[118] Q. Xu, J. H. Kim, L. Lam, and C. Y. Suen. Recognition of handwritten month words on bank cheques. In *Proc. 8th International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, pages 111–116, Niagara on the Lake-CA,

August 2002.

[119] Q. Xu, L. Lam, and C. Y. Suen. A knowledge-based segmentation system for handwritten dates on bank cheques. In *Proc. 6th International Conference on Document Analysis and Recognition (ICDAR)*, pages 384–388, Seattle-USA, September 2001.

[120] A. El Yacoubi. *Modélisation Markovienne de L'écriture Manuscrite Application à la Reconnaissance des Adresses Postales.* PhD thesis, Université de Rennes 1, France, Septembre 1996.

[121] A. El Yacoubi, M. Gilloux, R. Sabourin, and C. Y. Suen. An hmm-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):752–760, 1999.

[122] A. El Yacoubi, R. Sabourin, M. Gilloux, and C. Y. Suen. Off-line handwritten word recognition using hidden markov models. In L.C. Jain and B. Lazzerini, editors, *Knowledge Techniques in Character Recognition.* CRC Press LLC, April 1999.

[123] M. El Yacoubi, M. Gilloux, and J. M. Bertille. A statistical approach for phrase location and recognition within a text line: An application to street name recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):172–188, 2002.

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

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

[126] L. Q. Zhang and C. Y. Suen. Recognition of courtesy amounts on bank cheques based on a segmentation approach. In *Proc. 8th International Workshop on Frontiers of Handwriting Recognition (IWFHR)*, pages 298–302, Niagara on the Lake-CA, August 2002.

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

[128] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary

algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.