Ma 119

ACTA
POLYTECHNICA
SCANDINAVICA

MATHEMATICS AND COMPUTING SERIES No. 119

Adaptive methods for on-line recognition of isolated hand-written characters

VUOKKO VUORI

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory of Computer and Information Science

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Computer Science and Engineering for public examination and debate in Auditorium T2 at Helsinki University of Technology (Espoo, Finland) on the 14th of December, 2002, at 12 o'clock noon.

Espoo 2002

Vuori, V., **Adaptive methods for on-line recognition of isolated handwritten characters**. Acta Polytechnica Scandinavica, Mathematics and Computing Series No. 119, Espoo 2002, 93 pp. Published by the Finnish Academies of Technology. ISBN 951-666-619-1. ISSN 1456-9418. UDC 004.93:651.921:519.7:003

Keywords: handwriting recognition, isolated Latin characters, on-line, adaptation, clustering, allographs, Dynamical Time Warping (DTW), Learning Vector Quantization (LVQ), Self-Organizing Map (SOM), the k nearest neighbors (k-NN) rule.

ABSTRACT

The main goal of the work presented in this thesis has been the development of an on-line handwriting recognition system which is able to recognize handwritten characters of several different writing styles and is able to improve its performance by adapting itself to new writing styles. The recognition method should be applicable to hand-held devices of limited memory and computational resources. The adaptation process should take place during normal use of the device, not in some specific training mode. For the usability aspect of the recognition system, the recognition and adaptation processes should be easily understandable to the users.

The first part of this thesis gives an introduction to the handwriting recognition. The topics considered include: the variations present in personal handwriting styles; automatic grouping of similar handwriting styles; the differences between writer-independent and writer-dependent as well as on-line and off-line handwriting recognition problems; the different approaches to on-line handwriting recognition; the previous adaptive recognition systems and the experiments performed with them; the recognition performance requirements and other usability issues related to on-line handwriting recognition; the current trends in on-line handwriting recognition research; the recognition results obtained with the most recent recognition systems; and the commercial applications.

The second part of the thesis describes an adaptive on-line character recognition system and the experiments performed with it. The recognition system is based on prototype matching. The comparisons between the character samples and prototypes are based on the Dynamical Time Warping (DTW) algorithm and the input characters are classified according to the k Nearest Neighbors (k-NN) rule. The initial prototype set is formed by clustering character samples collected from a large number of subjects. Thus, the recognition system can handle various writing styles. This thesis work introduces four DTW-based clustering algorithms which can be used for the prototype selection. The recognition system adapts to new writing styles by modifying its prototype set. This work introduces several adaptation strategies which add new writer-dependent prototypes into the initial writer-independent prototype set, reshape the existing prototypes with a Learning Vector Quantization (LVQ)-based algorithm, and inactivate poorly performing prototypes. The adaptations are carried out on-line in a supervised or self-supervised fashion. In the former case, the user explicitly labels the input characters which are used as training samples in the adaptation process. In the latter case, the system deduces the labels from the recognition results and the user's actions. The latter approach is prone to erroneously labeled learning samples.

The different adaptation strategies were experimented with and compared with each other by performing off-line simulations and genuine on-line user experiments. In the simulations, special attention has been paid to the various erroneous learning situations likely to be encountered in real world handwriting recognition tasks. The recognition system is able to improve its recognition accuracy significantly on the basis of only a few additional character samples per class. Recognition accuracies acceptable in real world applications can be attained for most of the test subjects.

This work also introduces a Self-Organizing Map (SOM)-based method for analyzing personal writing styles. Personal writing styles are represented by high-dimensional vectors, the components of which indicate the subjects' tendencies to use certain prototypical writing styles for isolated characters. These writing style vectors are then visualized by a SOM which enables the detection and analysis of clusters of similar writing styles.

PREFACE

The work presented in this thesis has been carried out at the Laboratory of Computer and Information Science (CIS) in the Department of Computer Science and Engineering of Helsinki University of Technology during the years 1998–2002. This work has been funded by the National Technology Agency of Finland, Foundation for Promotion of Technology, and Nokia Foundation.

I wish to thank my supervisor Professor Erkki Oja and the other members of our research group at the CIS-laboratory, Dr. Jorma Laaksonen and M.Sc. Matti Aksela. I also would like to thank Dr. Jari Kangas and Mr. Jukka Yrjänäinen from Nokia Research Center. They all have provided valuable insights and been most supportive. Jorma has spent many long hours proof reading and perfecting my writings and helped to solve various technical problems with an amazing patience. Matti has done an enormous work in implementing the character recognition system into PDA devices. I also owe my thanks to my colleagues, friends, relatives, and the other volunteers who took part in the collection of the local character databases and in the user experiments. I like to thank all my co-workers for all the fun, the interesting and inspiring discussions, the unhurried lunch and coffee hours, and the ever prevailing Friday afternoon atmosphere.

Finally, I wish to thank my family and Mikko for all the support and for bearing my ups and downs. I bet, they now know about handwritten characters and pattern recognition more than they ever cared for.

Helsinki, August 2002

Vuokko Vuori

Vuokko Vuori

Contents

Abstract **2**

Preface **4**

List of abbreviations **7**

1 Introduction to the thesis **8**

1.1 Goals and scope of the thesis . . . 8
1.2 Contributions of the thesis . . . 9
1.3 Outline of the thesis . . . 9
1.4 Included publications . . . 10

2 Introduction to handwriting recognition **13**

2.1 Handwriting style variations . . . 14
2.1.1 Variations of characters . . . 14
2.1.2 Alignment of characters . . . 16
2.1.3 Personal background factors . . . 16
2.1.4 Situational factors . . . 17
2.1.5 Material factors . . . 17
2.1.6 Constraints on writing . . . 17
2.2 On-line vs. off-line handwriting recognition . . . 18
2.3 Writer-independent and writer-dependent handwriting recognition . . . 19
2.4 Automatic grouping of similar handwriting styles . . . 20
2.4.1 Automatic writing style clustering with on-line data . . . 21
2.4.2 Automatic writing style clustering with off-line data . . . 22
2.5 On-line handwriting data acquisition and processing . . . 22
2.5.1 Raw data . . . 23
2.5.2 Segmentation . . . 23
2.5.3 Preprocessing and normalization methods . . . 25
2.5.4 Features and representations . . . 26
2.6 On-line handwriting recognition methods . . . 27
2.6.1 Statistical methods . . . 28
2.6.2 Feature space classifiers . . . 29
2.6.3 Prototype-based methods . . . 30
2.6.4 Neural methods . . . 31
2.6.5 Structural and syntactical methods . . . 33
2.6.6 Generative models of handwriting . . . 35
2.7 Adaptive recognition systems for on-line handwriting . . . 36
2.7.1 Adaptation in prototype-based systems . . . 37
2.7.2 Adaptation in statistical systems . . . 40
2.7.3 Adaptation in neural systems . . . 42
2.7.4 Conclusions and remarks . . . 44
2.8 Language modeling . . . 45
2.9 Usability issues . . . 46
2.10 Summary: on-line handwriting recognition today . . . 47

3 Description of the developed adaptive on-line character recognition system 50
3.1 Data collection and databases 51
3.2 Preprocessing and normalization methods 53
3.3 Measuring dissimilarity with Dynamic Time Warping 56
3.3.1 DTW-based dissimilarity measures 57
3.3.2 Experimental results for the dissimilarity measures 58
3.3.3 Transforming dissimilarities into similarities 59
3.4 Creation of a prototype set 59
3.4.1 Clustering algorithms for prototype selection 61
3.4.2 Clustering indices for determining the number of prototypes 61
3.4.3 Experimental results for the clustering algorithms and indices 62
3.5 Classification 63
3.5.1 The k Nearest Neighbors rule 63
3.5.2 Speed-up methods for the k-NN classifier 65
3.6 Adaptation strategies 66
3.6.1 Add prototypes 67
3.6.2 Reshape prototypes 67
3.6.3 Inactivate prototypes 67
3.6.4 Controlled adaptation 68
3.6.5 Self-supervised labeling of the learning samples 69
3.6.6 Experimental results for the adaptation strategies 72
3.7 Writing style clustering 77
3.7.1 Personal writing style vectors 78
3.7.2 Visualization of the writing style vectors with the Self-Organizing Map 78
3.7.3 Experimental results 79

4 Conclusions 81

References 84

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
BMU	Best-matching Map Unit
BP	Back Propagation
CH	Calinski and Harabasz (clustering index)
CNN	Convolutional Neural Networks
DB	Davies and Bouldin (clustering index)
DTW	Dynamic Time Warping
EM	Expectation-Maximization
HMM	Hidden Markov Model
k-NN	k Nearest Neighbors
LVQ	Learning Vector Quantization
MAP	Maximum A Posteriori
MLLR	Maximum Likelihood Linear Regression
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
MS-TDNN	Multi-State Time Delayed Neural Network
NN	Neural Network
OCR	Optical Character Recognition
PCA	Principal Component Analysis
PDA	Personal Digital Assistant
SAT	Stroke-based Affine Transformation
SOC	Self-Organizing Classifier
SOM	Self-Organizing Map
ST	Spatio-Temporal
ST-MLP	Spatio-Temporal Multi-Layer Perceptron
SVM	Support Vector Machine
TDNN	Time Delayed Neural Network
RBF	Radial Basis Function
WD	Writer-Dependent
WI	Writer-Independent

1 INTRODUCTION TO THE THESIS

1.1 Goals and scope of the thesis

The scope of the thesis is the adaptive on-line recognition methods for isolated characters, the emphasis being on prototype-based recognition methods and Latin characters, i.e. lower and upper case letter and digits. Isolated characters can be obtained from a handwritten text which has been produced in such a way that there is no need for automatic segmentation but it is clear which parts of the pen trace belong to which characters prior to their classification. Alternatively, isolated characters can be segmented manually from cursively written words.

The main goal of the work presented in this thesis has been the development of an on-line handwriting recognition system which is able to recognize handwritten characters of several different writing styles and is able to improve its performance by adapting itself to new writing styles. The recognition method should be applicable to hand-held devices of limited memory and computational resources. The adaptation process should take place during normal use of the device, not in some specific training mode. For the usability aspect of the recognition system, the recognition and adaptation processes should be easily understandable to the users. In such a case, the recognition errors seem less random, and therefore, are less annoying to the users. In addition, the users have some insight how to adapt their writing styles to be more easily recognized by the system. If the users understand the exact way how the system adapts to new writing styles, they can control the adaptation process and explicitly teach the recognition system.

In order to create a recognition system which works well with all kinds of writers already in the very beginning of its use, it is essential to establish as much knowledge as possible on the different writing styles. One of the goals of the thesis was to gain such knowledge by analyzing large databases of character samples collected from several subjects and to enumerate the different writing styles. The performed analyses included simple but laborious manual examinations of the whole databases and not so human labour-intensive final inspections of the results of various automatic clustering algorithms applied to the databases. As a result, a prototypical character was selected for each distinctly different style of writing a character of each class.

The prototypical characters are used as a core of the recognition system. The prototype-based recognition system can easily be adapted to new writing styles by modifying its prototype set as the users provide new character samples. The adaptation can be carried out in a supervised or self-supervised fashion. In the former case, the user explicitly labels the input characters which are used as training samples in the adaptation process. In the latter case, the system deduces the labels from the recognition results and the user's actions. The latter approach is prone to erroneously labeled learning samples. In this thesis, different on-line adaptation strategies, both supervised and self-supervised ones, were experimented with and compared with each other by performing off-line simulations and genuine on-line user experiments.

Another approach to improve the ability of the recognition system to handle different writing styles is to devise a method for dividing the personal writing styles into meaningful groups and then to dedicate a specialized recognizer for each group. One of the goals of the thesis was to study if such a grouping of personal writing styles can be observed by measuring the subjects' tendencies to use different prototypical styles for writing isolated characters.

1.2 Contributions of the thesis

The main contributions of this thesis are:

- A comprehensive literature survey of the various topics related to automatic recognition of handwriting, including the generative models of handwriting, the sources of variations within and between personal handwriting styles, the automatic grouping of similar personal handwriting styles, the handwriting data acquisition and processing methods, the performance requirements and the other usability issues concerning handwriting recognition, the recognition methods themselves, and the results obtained with the most recent recognition systems. The emphasis of the survey is on on-line handwriting data and adaptive recognition methods for isolated Latin characters.

- The development of an adaptive prototype-based recognition system for isolated Latin characters. The initial prototype set has been formed by clustering character samples provided by a large number of subjects. Thus, the recognition system can handle various writing styles already at the beginning. The recognition system can adapt to new writing styles in a self-supervised fashion by modifying its prototype set during its normal use. The recognition system is able to improve its recognition accuracy significantly on the basis of only a few additional character samples per class written by the current user.

- The performance of the developed recognition system and its adaptation strategies have been evaluated in off-line simulations, and more importantly, in genuine user experiments. In the simulations, special attention has been paid to the various erroneous learning situations likely to be encountered in real-world handwriting recognition tasks.

- The Self-Organizing Map (SOM)-based method for analyzing personal writing styles. Personal writing styles are represented by high-dimensional vectors, the components of which indicate the subjects' tendencies to use certain prototypical writing styles for isolated characters. These writing style vectors are then visualized by a SOM which enables the detection and analysis of clusters of similar writing styles.

1.3 Outline of the thesis

This thesis consists of:

- An introductory chapter in which the motivations, scope, goals, and main contributions of the thesis work are described.

- A literature survey chapter which gives the essential background knowledge for the on-line handwriting recognition problem, reviews the state-of-the-art methods used in the various stages of the recognition process and recognition results reported in the literature, introduces alternative adaptation approaches, and discusses usability issues concerning handwriting recognition.

- A chapter which describes the various components of the adaptive prototype-based on-line character recognition system developed during the thesis project and the experiments performed with them, and reports the main results and the conclusion drawn from the individual experiments.

- A chapter which gives the main conclusions of this thesis work.
- A set of original publications giving the details of the methods and results.

1.4 Included publications

The published conference and journal articles listed below are included in this thesis. This section briefly describes the motivations and contents of the articles and declares the contributions of the author of this thesis. The articles are introduced here in nearly the same order as the works described in them have been carried out – article 7 is the only exception.

1. Vuori, V., J. Laaksonen, E. Oja, and J. Kangas (2001a). Experiments with adaptation strategies for a prototype-based recognition system for isolated handwritten characters. *International Journal on Document Analysis and Recognition 3*(3), 150–159.
2. Vuori, V., J. Laaksonen, and J. Kangas (2002). Influence of erroneous learning samples on adaptation in on-line handwriting recognition. *Pattern Recognition 35*(4), 915–925.
3. Vuori, V., J. Laaksonen, E. Oja, and J. Kangas (2000b). Controlling on-line adaptation of a prototype-based classifier for handwritten characters. In *Proceedings of the 15th International Conference on Pattern Recognition*, Volume 2, pp. 331–334.
4. Vuori, V., M. Aksela, J. Laaksonen, E. Oja, and J. Kangas (2000a). Adaptive character recognizer for a hand-held device: implementation and evaluation setup. In *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, pp. 13–22.
5. Vuori, V., J. Laaksonen, E. Oja, and J. Kangas (2001b). Speeding up on-line recognition of handwritten characters by pruning the prototype set. In *Proceedings of 6th International Conference on Document Analysis and Recognition*, pp. 501–505.
6. Vuori, V. and J. Laaksonen (2002). A comparison of techniques for automatic clustering of handwritten characters. In *Proceedings of the 16th International Conference on Pattern Recognition*, Volume 3, pp. 168–171.
7. Vuori, V. and E. Oja (2000). Analysis of different writing styles with the self-organizing map. In *Proceedings of the 7th International Conference on Neural Information Processing*, Volume 2, pp. 1243–1247.
8. Vuori, V. (2002). Clustering writing styles with a self-organizing map. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, pp. 345–350.

Journal article 1 describes several important parts of the prototype-based recognition system developed during the thesis project: 1) the preprocessing and normalization methods for isolated on-line characters, 2) a clustering algorithm for selecting prototypical characters, 3) various alternative dissimilarity measures which can be used both in the prototype selection and in the prototype-based recognition of unknown characters, and 4) alternative supervised on-line adaptation strategies which modify the prototype set by adding new prototypes, reshaping and inactivating existing prototypes as users provide new character samples. After selecting the best-performing combination of the preprocessing and normalization methods and the dissimilarity measure, the performances of the adaptation strategies were evaluated and compared with each other by performing off-line simulations of the first data collection. In addition, the performance of the most promising, i.e. fastest learning strategy was evaluated in genuine on-line user experiments. The author of the thesis had a major role in the design and implementation of all the methods but the clustering algorithm. She organized the second data collection, and designed and performed all the experiments. The first coauthor helped in all these tasks, planned and implemented the clustering algorithm, and provided useful comments in the reporting phase as did the two other coauthors of the article.

The other journal article 2 carries on the topics of article 1. The goal of the work reported in this article was to examine how well the on-line adaptation strategies would perform in more realistic situations in which the correctness of all the learning samples cannot be guaranteed. In this work, the erroneously labeled learning samples were assumed to occur with a certain probability and to be caused by either uncorrected recognition errors or wrongly interpreted writing mistake corrections. The influences of the erroneous learning samples on the recognition performance and the size of the prototype set of the adaptive recognition system were studied via simulations. The author of this thesis both planned and performed all the experiments. The coauthors helped in the reporting phase.

Conference paper 3 also tackles the problem of erroneous learning samples in the adaptation process. The purpose of the work described in the article was to study and experiment with controlling methods which can increase the classifier's tolerance to malformed or mislabeled learning samples and limit the growth of the prototype set. The introduced controlling methods either set an upper limit for the number of prototypes per class or switch the adaptation of a particular character class on or off depending on the earlier performance of the classifier. Again, the coauthors participated just in the reporting of the work and the author of this thesis designed, implemented and experimented with the controlling methods.

Conference paper 4 describes a character recognition system implemented in a hand-held PDA (Personal Digital Assistant) device for performing genuine user experiments with a self-supervised adaptation method. In the experiments, the prototype set of the recognizer was adapted on-line to each user's personal writing style by adding new prototypes into the prototype set and reshaping the existing ones. Adaptation was performed after a user had finished his or her answer to a question posed by a special questionnaire program. The adaptation process was supervised by the user's reactions to the recognition results and other indirect information obtained from the user interface of text input. The article also discusses the practical problems encountered in the implementation of a computationally heavy recognition method into a device with limited memory resources and computational power. The author of this thesis devised the adaptation method, performed the experiments with the constrained character matching algorithm, and made a tool for analyzing the results of the user experiments. The second author of the article implemented all the required methodology in the hand-held device and programmed the questionnaire program. All the coauthors helped in the reporting.

Conference paper 5 introduces a new preprocessing method for on-line characters and a two-phased recognition scheme aimed to speed up the prototype-based recognition. The new preprocessing method heavily down-samples the data points of on-line characters so that only the corners and some other geometrically meaningful points of the pen trace are kept. As the computational complexity of the character and prototype matching method is proportional to the square of the average number of data point per character, down-sampled characters can be handled considerably faster than characters not preprocessed in such a way. The two-phased recognition scheme is based on that fact. In the first phase, the prototype set is pruned and ordered on the basis of preclassification performed with heavily down-sampled characters and prototypes. In the second phase, the final classification is performed without any down-sampling and on the basis of a reduced set of prototypes. Two down-sampling methods, a basic linear method and the introduced nonlinear method, have been compared with each other. The author of this thesis devised the down-sampling methods and the two-phased classification scheme and performed the experiments. The coauthors took part in the reporting phase of the work.

The main goal of the work reported in conference paper 6 was to develop automatic methods for finding a set of prototypical characters which would represent well the different writing styles present in a large character database. One of the major obstacles in achieving this goal was the uneven representation of different writing styles in the database: there were hundreds of examples of some common styles but only a few samples of the rare styles. The work describes four hierarchical clustering algorithms and two clustering indices and the experiment performed with them in order to find a good prototype set which covers all the different writing styles for isolated characters present in the database but does not contain many redundant prototypes. The work was mostly done by the author of this thesis, the second author of the article provided useful insights in the planning of one of the clustering algorithms and provided assistance in preparing the article.

Conference papers 7 and 8 describe a method for examining clustering properties of personal handwriting styles. Handwriting styles were represented by vectors, the components of which reflect the writer's tendencies to use certain prototypical writing styles for isolated characters. The resulting very high-dimensional writing style vectors were analyzed by projecting them nonlinearly onto a two-dimensional plane by using Self-Organizing Map (SOM) algorithm. The main difference between the two works is the size of the analyzed database. The former work was a pilot study with 45 subjects in which the viability of the representation method for handwriting styles was examined. As the results were promising and much larger database with character samples from over 700 subjects became available to us, the same ideas were reused in the latter work in order to test scalability of the approach. Both works were nearly solely carried out by the author of this thesis while the second author of article 7 contributed in the reporting phase.

2 INTRODUCTION TO HANDWRITING RECOGNITION

Traditionally, interactions between humans and computers have been based on a display, printer, keyboard, and pointing device. However, a keyboard can be very inconvenient when the device is only slightly bigger or the same size as the human palm. In addition, a keyboard is difficult to integrate in small devices and it usually determines the size of the whole apparatus. This is especially true when the number of the characters is very high as in Chinese and Japanese languages. A pointing device, for example a track ball or a pen, is insufficient or very slow when used alone in applications in which textual input is also desired.

Because of these problems, new methods for input have been developed, for example systems that recognize speech and handwriting. Because both are very natural ways to communicate, people can easily learn to use them. Unfortunately, these recognition tasks are not that easy for computers whose artificial intelligence is different from that of humans. Neither one of the recognition problems has been completely solved yet. Very high accuracy is needed for such systems before they can be commonly accepted. Handwriting recognition is the more attractive input method, especially in noisy environment and when privacy is needed.

Automatic on-line recognition of handwritten text has been an on-going research problem for over four decades. Naturally, the first approaches were rather simple because of immature technology for recording pen point movements, and the limited memory and computational resources of the existing computers. In those early years, there were no mass-market applications and therefore no serious demand for on-line handwriting recognition systems. However, due to the emergence and increasing popularity of hand-held computers, such as Personal Digital Assistants (PDAs) and advanced cellular phones, on-line handwriting recognition has been gaining more and more interest since the 1990's. (Tappert et al. 1990)

At first sight, handwriting recognition does not appear to be a difficult problem. A recognition system should just choose the correct answer, usually the one that most resembles the written one, from a limited set of characters or words. Unfortunately, this approach faces a number of difficulties. The most prominent problem in handwriting recognition is the vast variation in personal writing styles. There are also a lot of variation within a writing style of one person. The These variations depend for example on the context of the writing, writing equipment, writing situation, and the mood of the writer. The writing style may also evolve with time or practice. The performance of the automatic recognition system thus depends heavily on how well the different personal writing styles and their variations are modeled.

A recognition system should be insensitive to meaningless variations and still be able to distinguish different but sometimes very similar looking characters. Recognition systems should, at least in the beginning, be able to recognize many writing styles. Such multi-user systems usually have problems with recognition accuracy. One way to increase performance is adaptation, which means that the system learns its user's personal writing style. Alternatively, multiple recognition systems each specializing in handling a group of writing styles sharing some group-specific properties can be designed. The way of obtaining the final recognition decision from the recognition results of the specialized system would then depend on which group of writing styles the current user's style of writing belongs to.

This chapter gives an introduction to the handwriting recognition problem and reviews the state-of-the-art methods used in the various stages. In addition, the applications and some usability issues of handwriting recognition are discussed. The emphasis is on on-line recognition of isolated Latin characters written in natural handwriting style and on adaptive methods. Section 2.1 discusses the variations present in personal handwriting styles. Section 2.2 introduces the two main categories of handwriting recognition problems: on-line and off-line recognition. Section 2.3 is about the differences between writer-independent and writer-dependent handwriting recognition problems. Section 2.4 considers the automatic grouping of similar handwriting styles. The collection, segmentation, preprocessing, normalization, features, and representations of on-line handwriting data are considered in section 2.5. Section 2.6 discusses the different approaches to on-line handwriting recognition and describes recognition systems based on various methodologies. Section 2.7 considers issues related to adaptation and reviews the previously developed adaptive recognition systems and the experiments performed with them. Section 2.8 briefly discusses the language models used in handwriting recognition. Section 2.9 considers the recognition performance requirements and other usability related issues of handwriting recognition systems. Section 2.10 discusses the current trends in on-line handwriting recognition research, reviews what kind of recognition accuracies can be obtained with the recent recognition methods, and briefly describes some commercial recognition softwares.

2.1 Handwriting style variations

A writing style is based on the alignment and variable forms of characters. Both on-line and off-line handwritten characters have enormous variety in shape compared to machine-printed characters. Handwriting style variations occur mostly between different writers but also within the handwriting of any individual writer. Handwritten characters may be regarded as distorted versions of idealized character models, which are called *allographs*, and the distortions can be interpreted to be caused by several factors. According to Kuklinski (1984), sources of variations can be classified into personal background factors, situational factors, and material factors. These factors affect both the generation and recognition of the characters. Therefore, they should be considered in the collection of handwriting data which is used in designing of a handwriting recognition system. The handwriting data should be produced in situations similar to those in which the recognition system will be used.

2.1.1 Variations of characters

Handwritten characters can vary in both their static and dynamic properties. Static properties are the underlying, ideal models of the characters, the allographs, and the geometrical properties such as relative positions and sizes of the strokes, corners, retraces, ornamentals, sizes and aspect ratios of the characters, and the general slant of the writing. Dynamic properties are more involved with the generative aspects of the characters. Characters can look similar although their number of strokes, and the drawing order and direction of the strokes may vary considerably. (Tappert et al. 1990)

Allographs are alternative ways of writing a character; Figure 1 shows some examples for digit '4'. The reasons why a writer uses some particular allographs can be positional, contextual, dialectical, and stylistic (Herrick 1979). In some languages, the position of a character in a word matters. Different allographs are systematically used for characters which are written in isolation or which are in the middle, beginning, or end of a word (Herrick 1979). In the scripts of the Araboid genus, for example the Naskhi script, there are four different position-dependent allographs for almost every letter. In Bengali script, each

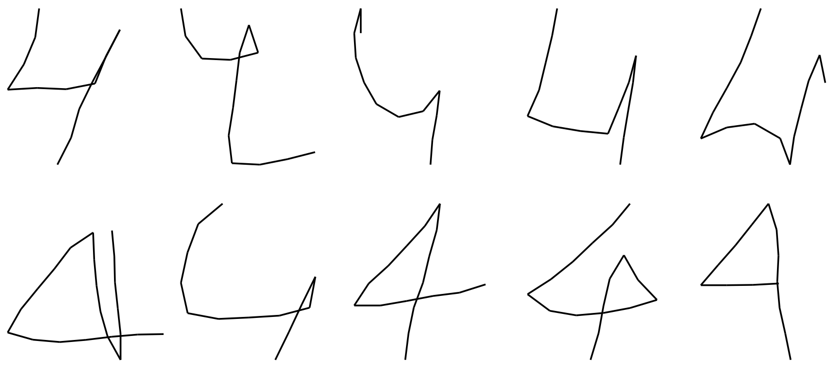

Figure 1: Some examples of alternative ways of writing digit '4'.

vowel letter has two allographs: one for vowels which form an entire syllable by themselves, and one for those which form a syllable together with one or more consonants. Writer's aspirations to optimize the ease and the speed of writing or the legibility of the script effect on which allographs are used. Thus, there are some preference differences between left-handed and right-handed writers for using allographs (Ansell 1979). Also, different allographs can be used for a letter when it is a part of a very frequently occurring letter combination or when it is used in some other context. Sometimes, the letters in frequent combinations are so slurred together that perhaps the combinations of letters should be considered as single characters instead (Wing 1979). Some languages have dialectical variations in their written form (Herrick 1979). For example, there are two dialects of Devanagari script used in India. The different dialects use different allographs for several letters. In practice, many writers use mixtures of the two dialects. Naturally, writers can choose to use different allographs also for purely stylistic reasons.

The number of strokes in a character varies as sequential strokes have a tendency to be connected within a character. This is caused by pen lifting failures. Stroke connecting is more likely to occur between sequential vertical or horizontal strokes whose ends are close to each other than between two strokes of which one is horizontal and the other is vertical (Kuklinski 1984). Stroke connecting is very common in Asian languages, for example with the Kanji characters, as these characters typically consist of several short and straight strokes (Kobayashi et al. 2001). *Retraces* of strokes are also caused by the errors in detecting pen liftings. Very short retraces at the beginning and end of the strokes are called *hooks* and *continuations*, respectively. Retraces, hooks, and continuations are illustrated in Figures 2 and 3. On the other hand, the number of strokes in a character increases when a stroke is broken by an unintentional pen lift.

The directions of the strokes vary if a writer tries to minimize the time the pen is lifted up or has some other personal preferences for using certain drawing directions. Usually, vertical strokes are drawn before horizontal strokes and strokes on the left before strokes on the right (Ward and Kuklinski 1988). Vertical or tilted linear strokes are usually drawn from top to bottom as they are taught to be written in that way in primary schools. Northeast- and southeast-ward handwriting movements are preferred to southwest- and northwest-ward movements due to the motoric properties of right wrist and hand (Lorette 1999).

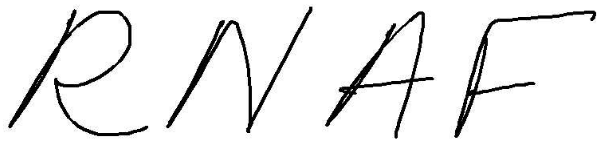

Figure 2: Characters with retraces.

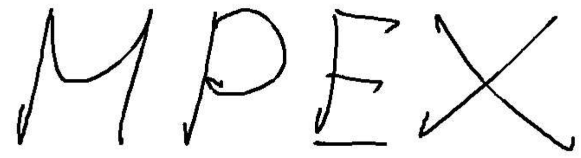

Figure 3: Characters with hooks and continuations.

In addition to the stroke number variations, the shape of the strokes varies – nominally straight strokes can be curved as bows or s-curves. The shapes of loops and cusps change as loops have a tendency to collapse into cusps and, on the contrary, cusps can change into loops. Distinguishing cusps and loops is a difficult problem because they often produce similar features (Ward and Kuklinski 1988). These and other more or less random variations in the shapes of strokes may be attributed to the inherent accuracy and speed limitations of the fine motorics of the hand and wrist (Wing 1979).

2.1.2 Alignment of characters

Handwriting styles can be classified into four basic classes: *boxed discrete*, *spaced discrete*, *run-on discrete*, and *connected* or *pure cursive* handwriting (Tappert 1984). The discrete handwriting styles are sometimes called *printing.* Boxed discrete style means that all characters are written in guideline boxes. Spaced discrete style means that the writing is less constricted and the characters are separated spatially by a significant space or temporally by a predefined time difference between the characters. In the run-on discrete style, the characters can touch each other but a stroke cannot be included in more than one character. This means that pen must be lifted at least between characters. In the cursive style, all the letters in one word are connected. Usually, natural handwriting style is a mixture of discrete and cursive styles, called *mixed* handwriting style (Bellegarda et al. 1994; Fujisaki et al. 1991). Some examples of text samples with differently aligned characters are shown in Figure 4. All the samples have been written by the same writer.

2.1.3 Personal background factors

One of the most important personal background factors of writing style variations is handedness as left-handed and right-handed writers use muscle and tendon groups involved in writing differently. For example, left-handers prefer to draw horizontal lines from right to left and right-handers prefer to do it in the opposite direction. This can be explained by the fact that writers usually prefer dragging the pen behind the hand to pushing it ahead.

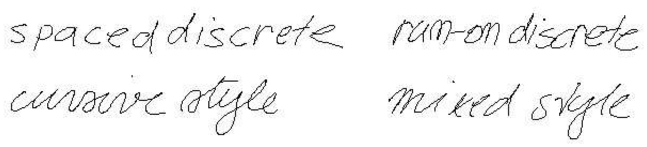

Figure 4: Different alignments of characters.

Also age and health affect motor control of writing and thereby the resulting writing style. The education and origin of the writer have important roles in the writing style because different type characters are taught in different schools. In addition, some professions affect the person's handwriting style, especially when a neat writing style is needed. (Kuklinski 1984; Ward and Kuklinski 1988)

2.1.4 Situational factors

Some examples of situational factors are stress, haste, motivation, distractions from the writing task, and the method of presentation (Kuklinski 1984; Wing 1979). Haste can be caused by several reasons and it has major effects on style as many people have different styles for different writing speeds. In addition, fatigue and mood affect the writing speed. The motivation of the writer also determines how carefully characters are written, for example, the address of an important letter is very likely to be written carefully while an ordinary shopping list can be almost unreadable even to the writer himself. Text can be written differently depending on whether it is self generated or copied. For example, Kassel (1995) mentioned in his thesis an interesting copying effect which he observed while designing a data collection setup: the subjects were copying the font of the visual writing prompt. Naturally, this had a limiting effect on the variability of character shapes and sizes. An example of a situational factor which has an effect on how the writers use different character allographs is the feedback provided by a handwriting recognition system. If the system does not recognize its user's natural handwriting style, the user is tempted to try out alternative styles in order to increase the system's performance. This phenomenon is called *user adaptation.*

2.1.5 Material factors

The writing instrument, surface, and form constitute the material factors of the writing style. The writing instrument has an effect on the writing style depending on its size and overall comfort. Writing surface's friction and position also affect the style. The form factors, such as the size of the blank writing area, the length of the writing line, or the size of the writing boxes for characters, can have a dramatic effect on the handwriting style (Kuklinski 1984).

2.1.6 Constraints on writing

As the variations are the key problem of automated handwriting recognition, writing style is usually more or less constrained in such systems. Usually, characters are written in boxes to ease their separation, or the system provides guiding lines to help the users write more

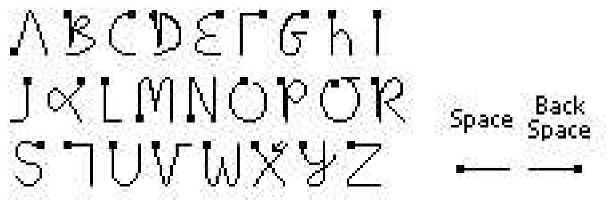

Figure 5: Graffiti characters.

consistently. These constraints are designed to improve the performance of the recognizer and they can be seen as mediate *a priori* decision rules. If the constraints are too strict, they may cause the system to be rejected by the users. According to experiments carried out by Suen (1979), the constraints do affect the style and speed of writing and the recognition results.

In the most constrained systems, users are expected to use specific character allographs. These systems perform very well assuming that users do write in the required style. The drawback of such systems is that users tend to write in their own styles and, as personal writing styles vary a lot, these systems are awkward for most of the users (Ward and Blesser 1985). Figure 5 shows an example of a constrained writing style, namely Graffiti character set used in PDA devices running Palm Operating System. The Graffiti characters are so called *unistroke characters* meaning that they are drawn with a single stroke. The pen should not be lifted within characters. The advantages of using unistroke characters is that there is no need for any other character segmentation than the pen lifts. Therefore, characters can be even drawn on each other and thus only a small input area is necessary. In addition, unistroke character are faster to write and less prone to recognition errors than ordinary characters (Goldberg and Richardson 1993). The drawback of unistroke characters is that they do not always represent natural ways of writing and thus they need to be learned and remembered. In addition, a requirement for simple shapes limits the number of the character classes which can be input using unistroke characters.

2.2 On-line vs. off-line handwriting recognition

There are two categories of handwriting recognition problems: *on-line* and *off-line* problems (Tappert et al. 1990). These categories are based on the nature of the handwriting data. On-line recognition means that handwriting is collected and recognized in real time, i.e. at the same time it is produced. The writing medium is usually a tablet or a flat display which can capture information on the location and motion of a pen-like pointing device moving on its surface. The locations and movements of the pen point, and possibly its pressure on the writing surface, are frequently sampled and sent to the recognition system. The applications of on-line handwriting recognition include various kinds of interactive user-interfaces in which a method for textual input is needed but a keyboard would not be a practical solution, for example as in the case hand-held computers. For good surveys concerning on-line handwriting recognition, see articles (Nouboud and Plamondon 1990; Tappert et al. 1990; Plamondon and Srihari 2000).

In the case of off-line recognition, handwriting has been produced using an ordinary pen and paper well before its recognition. Thus, the off-line recognition methods use scanned images of the handwriting. The features used in the recognition are first enhanced and then extracted from bitmap images by means of digital image processing. Off-line handwriting recognition is often called Optical Character Recognition (OCR) (Tappert et al. 1990). OCR includes also the recognition of machine printed characters. Off-line handwriting recognition methods are used for automatic conversion of paper documents to electronic ones which then may be interpreted or postprocessed by computers. Typical applications of off-line handwriting recognition are used for handling of huge amounts of information in paper form, for example automatic sorting of mail (D'Amato et al. 2000) and handling of financial documents such as cheques (Gorski et al. 1999). For more information on off-line handwriting recognition, see review articles (Arica and Yarman-Vural 2001; Govindan 1990; Trier et al. 1996; Plamondon and Srihari 2000; Vinciarelli 2002).

The main advantage of on-line handwriting data over off-line is the dynamic information on writing process. Off-line data is just static images of handwriting. The image pixels do not contain any information on the writing direction or the writing order of the strokes. The values of pixels tell only if the pen point has ever visited the locations the pixels correspond to. It is not always clear which pixels belong to which strokes, or even what is the number of strokes. The quality of off-line data depends heavily on how well the pen trace image can be segmented from the background. In addition, too thick or smudged pen trace cannot capture small details of characters. In the case of off-line data, the pen trace is captured together with an image of the paper it has been written on. Thus, special image processing algorithms are needed for removing the background information and enhancing the actual handwriting information. None of these problems occurs with on-line data. The on-line data requires less memory resources than off-line data as only the coordinates of the sampled pen point positions and possibly some other features are stored.

However, on-line data does not contain explicit higher-order spatial information which is readily available in off-line data, for example, whether two data points far from each other in the time domain are close to each other in the spatial domain. Therefore, broken and delayed strokes, such as a cross in letter 't' or a dot in letter 'i', and other completions and insertions which are added to the main bodies of characters with some delay, are more problematic in on-line than in off-line data. In addition, the differences in writing order and direction of strokes of similar looking characters are not always useful information but just natural and meaningless variation.

On-line data can be easily converted to off-line data of high quality, for an example see (Laaksonen et al. 1999). Thus, recognition methods developed for off-line handwriting can be used for on-line data too, and probably with better results due to the lack of typical visual noise. The best results can be obtained if features extracted from both on-line and off-line representations of handwriting data are used together (Jaeger et al. 2001; Prevost and Milgram 1997; Yaeger et al. 1998).

2.3 Writer-independent and writer-dependent handwriting recognition

Handwriting recognition systems can be divided into two categories on the basis of the data they have been trained with. In this thesis, the categories are *writer-independent* and *writer-dependent* recognition systems. In the case of writer-independent system, training data is collected from different subjects than those who will be the end-users of the recognition system. Therefore, a writer-independent system should be trained with data collected from as many subjects as possible with various kinds of writing styles in order to guarantee that the recognition system will perform well with all the potential users. On the contrary, writer-dependent systems are specialized in recognizing only certain writing styles. A writer-dependent system is trained with data collected from the same writers whose handwriting the system will be recognizing in the future use. The training database of a writer-dependent system should be large enough to contain all the important variations and peculiarities of the writing styles in its repertoire. As the writer-dependent recognition systems do not try to model all the possible variations and features present in natural handwriting, higher recognition accuracies can be obtained for individual subjects than with writer-independent systems based on the same recognition methodology (see Subrahmonia, 2000 for results obtained with Hidden Markov Models). A writer-independent system can serve as a starting point in the development of a writer-dependent system if it can be adapted to new writing styles. That kind of adaptation will be dealt with in more detail in section 2.7.

2.4 Automatic grouping of similar handwriting styles

One way to try to improve the performance of a handwriting recognition system is to cluster writers with similar writing styles and to design a specialized system for each of the writing style clusters. Such a multi-expert system should also include a way of automatically identifying in which of handwriting style clusters the current user belongs to and which of the experts should be used for recognizing his or her handwriting. Of course, such a *style-dependent* recognition system can be further adapted into a writer-dependent one. The advantage of the grouping of similar writing styles and style-dependent recognition is that less handwriting data is required from the current user. This is important as it might not be practical, and it definitely does not improve the system's walk-up acceptance, to collect a large amount of data before the system starts to perform well with a new user. For the same reason, the automatic writing style identification should not depend on what are the particular words and character which have been written and it should not require a large amount of data to work reliably.

Subrahmonia (2000) argues that the writing style clustering should not be done independently of the handwriting models used for recognition. Otherwise, there is a possibility that writing styles similar in respect to the writing style clustering algorithm will not correspond to similar handwriting models. In such a case, it would be unrealistic to assume higher recognition accuracies with a style-dependent system than with a writer-independent one. Most likely, a writer-independent system trained with all the available data would perform better than a style-specific trained only with some subset of the data. The experiment performed by Kassel (1995) is a good example of such a phenomenon. Kassel carried out an experiment in order to study whether a grouping of the writers according to their gender or handedness would improve recognition rates. He used a database of isolated characters collected from 159 subjects. The character samples were represented by a fixed number of evenly distributed coordinate points of the pen trace. The statistical recognition system modeled the character classes with mixtures of Gaussian distribution with diagonal covariance matrices. The results of the experiments showed that the assumption of some handwriting properties specific to gender or handedness is not completely unjustified: recognition systems trained with data collected solely from female, right- or left-handed subjects performed best with the test subjects belonging to the same categories. However, the best recognition results for all the writer categories were obtained when the recognition system was trained with all the available data.

In the next two sections, some earlier works on automatic grouping of personal writing styles are reviewed. Writing style clustering has also been studied in this thesis, namely in (Vuori and Oja 2000, included publication 7) and (Vuori 2002, included publication 8). These works are described and their result are discussed in more detail in section 3.7.

2.4.1 Automatic writing style clustering with on-line data

In her experiments, Subrahmonia (2000) used Hidden Markov Models (HMMs) (Rabiner 1989) for representing isolated handwritten characters. She compared the performances of the recognition system based on writer-independent, writer-dependent and style-dependent character models. The similarity measure between the personal writing styles was based on the principal angles of the writer-dependent feature spaces, on the likelihood ratios of writer-dependent character models, or on the linear combination of the two. According to the results, the highest recognition accuracies were achieved with writer-dependent character models and the style-dependent models performed significantly better than the writer-independent models. On average, the error rates of the writer-dependent and style-dependent systems were 41% and 25–29%, respectively, of the error rate of the writer-independent system. The experiments were performed with a database containing handwriting samples from 113 subjects. It should be noted, that Subrahmonia used only a single HMM per character class in her experiments. The comparison between the writer-independent and writer-dependent systems would have have been fairer, if she had trained a HMM for each character allograph instead. A single writer hardly uses all the character allographs present in the multi-writer database and thus the modeling task of a single HMM is much easier in the case of a writer-dependent recognition systems than in the case of writer-independent system. Nevertheless, her results are encouraging for using style-dependent recognition systems. Subrahmonia did not comment how to select the best performing style-dependent system for a new user.

Schomaker et al. (1994) have developed a method suitable for automatic writing style classification or writer identification. They modeled handwriting on stroke-level with Self-Organizing Map (SOM) algorithm (Kohonen 1997). The pen trajectories were segmented into strokes according to the velocity minima and a set of prototypical strokes was found with the SOM algorithm. Isolated characters were modeled by using sequences of the prototypical strokes and probabilistic stroke transition networks. The personal writing styles were represented by the subjects' histograms of prototypical stroke usage. Schomaker et al. claimed that the histograms of different writers really were different while the histograms estimated by using different writing samples of the same writer were comparable with each other. They applied an agglomerative clustering algorithm to 60 histograms formed from a database containing character samples written by 30 subjects. Each subject had written two sets of handwriting samples and two histograms were formed per subject. At the lowest level of the clustering, the histograms corresponding to the same writers formed their own isolated clusters. Groupings of handwriting styles which could be explained by the gender and nationality of the writers were observed at the higher levels of clustering. A random subset of only 40 prototypical strokes was sufficient for error-free writer identification.

Vuurpijl and Schomaker (1997b) have devised a method for detecting generic writing styles, i.e. mixed, cursive, and discrete hand print. Their method is based on three features: a cursivity index and the average distances of the observed strokes to cursive and hand print prototypical strokes. The cursivity index reflects the writer's tendency to produce isolated hand print characters or fully-connected cursive script. It measures what is the average relative difference between the number of letters and the number of pen lifts in a word. The value of cursivity index is one for writers using pure cursive style and zero for those using pure hand printing style. For writers with mixed style, the values of the cursivity index are somewhere between the two extremes. The prototypical cursive and hand print strokes were obtained by training separately two SOMs, one for subjects who were known to use cursive style and the other for subjects using hand print style. Personal writing styles were classified into the three generative style categories by combining the three features nonlinearly into one feature and thresholding its values. Vuurpijl and Schomaker performed experiments with their method by using a database containing handwriting samples from 187 subjects. According to the results, the introduced method was able to distinguish hand print and cursive styles from each other perfectly. Only some mixed and cursive styles, or mixed and hand printing styles, were confused with each other. These errors are quite understandable and unavoidable as the borders between the pure and mixed styles are rather vague also for human experts.

2.4.2 Automatic writing style clustering with off-line data

Crettez (1995) has studied handwriting style recognition for off-line data. He used a set of nearly one thousand scanned images of literal amounts on cheques, in total nearly four thousand words, as a handwriting database. Crettez used following features to characterize handwritten words: thickness of the pen trace, height of the main body of the word, spatial density of characters, four main directions of the pen trace directions (reflecting the general slant, inter- and intra-letter ligatures, retroactive and reverse bindings, and pseudo-horizontal parts of tracing), and the relative intensities of the main directions. Crettez applied a fuzzy clustering algorithm to the database and found out that the local optima for the number of handwriting style families were 10, 23, 34, and 44. According to his results, if 23 style families were defined, approximately 30% of the samples were written in homogeneous style, i.e. the words in them had similar degrees of membership to the style families. Therefore, his method would not be suitable for writer identification but for selecting a proper recognizer expert for each word.

Bouletreau et al. (1997) have also studied writing style classification methods for off-line handwriting. They characterized personal writing styles with four parameters based on the fractal properties of the handwriting data. The parameters were derived from two fractal dimensions defined for different scalings. The fractal dimensions measure the irregularity and fragmentation of the pen trace. By performing a statistical test, they found out that the parameters were highly writer-specific and stable over time. They did not comment at all how many writing style families could be found on the basis of the four parameters. Their method would be well suited for writer identification.

2.5 On-line handwriting data acquisition and processing

The following sections will discuss the acquisition of on-line handwriting data and its different processing stages performed prior to the actual recognition stage. Section 2.5.1 concentrates on data collection, section 2.5.2 introduces different approaches to the segmentation of the handwriting data into characters, section 2.5.3 is about preprocessing and normalization methods used for isolated character samples, and section 2.5.4 discusses different types of features and representations used for on-line handwriting data. It should be emphasized that these processing stages are not independent from each other and should be planned together. Naturally, the choices made and performances of the methods used in the earlier stages affect what can be done and achieved in the later stages.

2.5.1 Raw data

A typical format of on-line handwriting data is a sequence of coordinate points of the moving pen point. In addition to location, the pressure between the pen point and writing surface can be measured. Sometimes, not only the events when the pen point is actually touching the writing surface are detected and recorded but handwriting data is collected also when the pen point is hovering just above the writing surface. Connected parts of the pen trace in which the pen point is touching the writing surface, or the pressure between them exceeds some threshold value, are called *strokes*. (Note that sometimes trace segments defined not only by pen lifts but also by other critical points of the pen trace such as curvature maxima, velocity minima, and local horizontal minima are called strokes.) If the touching detection is too sensitive and is activated with too low pressures, hooks and retraces are likely to appear in the beginnings and the ends of strokes and some strokes might be unintentionally connected. On the contrary, if the pressure limit is set too high for touching detection, strokes tend to break accidentally.

The pen trace is usually sampled with a constant rate and thus data points are evenly distributed in time but not in space. When the speed of writing is slow, the sample points are located densely on the true pen trace, whereas quick writing produces more sparsely located points. Typically, writing speed slows down in sharp corners, in other points of extremal curvature, at the beginning and the end of the stroke, but also by hesitation and pausing of the writer. Sampling rate and resolution should be so high that the sampled data points represent the true pen trace faithfully. Naturally, the selection of suitable level of sampling rate and resolution depends on the writing speed and the scale of the meaningful pen trace features. If sampling rate is too low, odd corners will be introduced on the sampled pen trace and some of the real corners and miniscule trace features can be missed. In practice, sampling resolution has to be finite and that causes some errors in the recorded pen point locations. If sampling resolution is too low, the sampled pen traces are jagged. Data collection hardware can sometimes introduce erroneous points clearly out of the real pen trace. Such points are called *wild points* (Guerfali and Plamondon 1993). Wild points can be introduced for example when the writer rests his or her hand on the pressure sensitive writing surface.

Figure 6 shows some examples of characters collected with different sampling rates and resolutions. Characters in the upper row are collected with sufficiently high resolutions as there is no jagging in the captured pen traces. Characters in the lower row are collected by using much poorer resolution and the captured pen traces are clearly jagged. The sampling rates seem to be high enough for all but perhaps the two last characters in the upper row as the captured pen traces in those cases have some corners which most probably were not in the real pen traces. Effects of varying writing speed on data point density can be seen most clearly in the second character on the left in the upper row.

2.5.2 Segmentation

Before anything else, handwriting data is segmented at least into words. That is not difficult as words are spatially separated from each other. The major difficulty lies in the segmentation of words into characters, especially in the case of cursive writing style. The segmentation approaches can be either *external* or *internal* (Tappert et al. 1990).

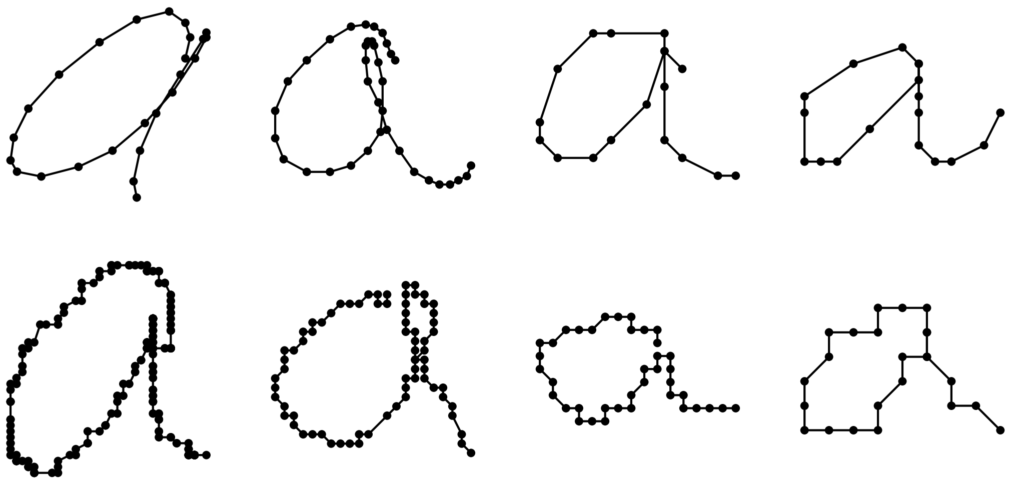

Figure 6: Examples of character collected with different temporal sampling rates and spatial resolutions.

In internal approaches, the segmentation and recognition of handwriting are performed simultaneously. Internal segmentation approach is used in the recognition of whole words written in cursive or mixed style. Typically, the recognition methods are based on Hidden Markov Models (HMMs) (Rabiner 1989), Time Delayed Neural Networks (TDNN) (Haykin 1999, pp 640–663), Multi-State Time Delayed Neural Networks (MS-TDNN) (Jaeger et al. 2001), or Convolutional Neural Networks (CNN) (Bengio et al. 1993). First, these models are used for evaluating character probabilities, or some other matching scores, for highly overlapping segments of the handwriting data. The best segmentation and the corresponding character interpretations are then found by using dynamic programming techniques, for example Viterbi algorithm (Theodoridis and Koutroumbas 1999, pp 309–312), and language models. For some examples of internal segmentation approaches, see (Bengio et al. 1995; Guyon et al. 1992; Hu et al. 2000; Jaeger et al. 2001; LeCun and Bengio 1994; Schenkel et al. 1995).

In external approaches, the segmentation of words into characters is performed prior to the recognition. External segmentation of cursive or mixed style writing is usually based on critical points of the pen trace such as pen lifts, curvature maxima, velocity minima, and local horizontal minima. External segmentation provides either a single solution or several tentative solutions which are then evaluated and ranked by the recognition system. The latter approach has been used, for example, by Morasso et al. (1992). By posing some constraints on the writing style, external segmentation can be made much easier. The recognition system might require that characters are separated by pen lifts, time pauses, or empty spaces. Usually, some writing guidelines are provided to make the segmentation easier for the writer. Typically, characters are written into boxes or onto lines marked with ticks. Without any guidelines, users tend to connect successive characters (Goldberg and Goodisman 1991).

2.5.3 Preprocessing and normalization methods

The purpose of preprocessing and normalization methods is to simplify the recognition task by reducing the amount of information, eliminating imperfections, and removing uninformative variations in handwriting data. In this section, only methods useful in recognition of isolated characters are concerned. Methods used for whole words or longer portions of handwritten text, for example baseline drift and slant normalization methods, are omitted here. Such methods have been reviewed by Guerfali and Plamondon (1993) along the other principal preprocessing and normalizing methods used for on-line handwriting data.

On-line handwriting data collected with a high sampling rate contains redundant information as the pen trace could be represented well enough with fewer data points. Therefore, the data points are usually either *down-sampled* or *resampled*. Down-sampling means that only some of the original data points are kept while the rest are abandoned. In linear down-sampling, the data points to be kept are selected in a linear manner from the data point sequence. For example, only every nth point can be kept. Nonlinear down-sampling methods select the data points in a nonlinear fashion instead. For example, the criterion for keeping a data point can be a minimum distance along the pen trace to the previous preserved point. Resampling means that new data points are calculated on the basis of the original ones. For example, new spatially equidistant data points can be obtained by interpolation from temporally equidistant original data points. For both resampling and down-sampling methods, it is important not to lose data points of geometrical or perceptual importance, for example corner points, which are useful for the recognition. Several methods for detecting such points have been developed, for some examples, see (Brault and Plamondon 1993; Freeman and Davis 1977; Kobayashi et al. 2001; Li and Hall 1993; Pavlidis et al. 1997; Zhang et al. 2000). Resampling and down-sampling may destroy or severely distort implicit dynamic information in the original data points. Therefore, all dynamic features should be evaluated prior to such preprocessings.

Various preprocessing methods have been designed for removing imperfections in the on-line handwriting data caused by hardware problems or erratic hand motions. Accidentally broken strokes can be detected by the angular continuity, high velocity and temporal and spatial proximity of their endpoints. Broken strokes can be simply concatenated or a straight line segment can be interpolated between their endpoints. The jagging phenomenon caused by too low a sampling resolution can be reduced by various smoothing methods. These methods are typically based on low-pass filtering, i.e. weighted averaging of neighboring data points. Also more sophisticated methods based on curve fitting, for example on local cubic spline approximation (Hu et al. 1996), have been developed. It should be noted that smoothing operations reduce also the real angular discontinuities of the pen trace. Wild points can be detected by high velocities out of the limits of normal hand motions. Hooks can be detected by their location, size, and large angular variations. For some examples of hook removal methods, see (Kharma and Ward 2001; Loy and Landay 1982; Narayanaswamy 1996; Veltman and Prasad 1994).

Handwritten character samples must be normalized in respect to the variations in their size and location. Otherwise, they cannot be reasonably compared with each other and would not be compatible with the recognition system. Size normalization of isolated characters is typically just a linear scaling to a standard height preserving the aspect ratios of the characters. The size normalization improves the recognition rates in general but makes the recognition of similarly-shaped upper and lower case letters very difficult without any contextual information. Translation variations are normalized by moving the centers of the characters into the origin of the coordinate system. A character center can be defined to be the center of mass of the data points or the center point of the smallest box drawn around the character. The general slant of handwriting can be estimated from a histogram of the directions of the pen point movements. However, as a single character does not contain much directional information as opposed to a whole word, usually no deslanting is performed for isolated characters.

If characters are compared to each other or recognized in a stroke-wise manner, variations in the drawing order and direction of the strokes must be considered. These variations can be normalized by reordering and reversing the strokes in some systematic way. For an example, see Narayanaswany's Ph. D. thesis in which strokes are reordered from left-to-right (Narayanaswamy 1996). Such a normalization approach is better suited for characters consisting of simple straight strokes, for example the Kanji characters, than for characters made of more complicated curvilinear strokes. Alternatively, different stroke order and direction variations can be tried out in an exhaustive way. This brute force solution can be computationally too demanding for practical applications. A more sophisticated approach is to set some limits on how much the stroke order can vary from the standard stroke order, as was done by Lin et al. (1993) in case of isolated Chinese characters. The most common approach is to handle the different stroke order and direction variations as character subclasses, or allographs, and to train the recognition system with a database large enough to be likely to include all the principal variations.

2.5.4 Features and representations

The recognition of handwritten patterns is based on the features extracted from the preprocessed and normalized raw data. The features are selected so that they represent the handwriting well and emphasize the inter-class differences and intra-class similarities. In the case of on-line handwriting data, features developed for off-line data are also applicable as on-line data can easily be converted into off-line data. This section concentrates on features which are specific to on-line data. The off-line features are reviewed in (Arica and Yarman-Vural 2001; Trier et al. 1996).

Typically on-line handwriting data is represented by time series, i.e. sequences of feature vectors corresponding to the data points of the sampled pen trace. In the simplest case, the only features are the x- and y-coordinates of the sampled pen point positions. Point-oriented feature vectors are often augmented with some higher level features. Mere positional features of pen trace are not sufficient if some of the character classes differ from each other only by small-sized subpatterns as in the case of the Kanji characters (Kobayashi et al. 2001). The additional data point features can be of either local or global nature. Local features are estimated by using only a few neighboring data points. Typical examples of local features include the tangential direction and curvature of the pen trace. Global features can capture phenomena of larger scale than local features. Global features are based on data points temporally farther away from each other, and in some cases, on data points close to each other only spatially. For example, Hu et al. (2000) have introduced global features which indicate whether a data point belongs to a loop, or measure how close the data point is to a crossing or a cusp. These global features are used along with some local features in a HMM-based recognition system. Jaeger et al. (2001) have also used local and global features together in their MS-TDNN-based recognition system. They used a global feature which indicates whether a data point is below a delayed stroke, and a feature which measures whether there is an ascender or descender above or below the data point. They used also low-resolution images centered on the data points to capture their spatial contexts.

In some cases, higher-level feature extraction and recognition are integrated into one process and they are learned together. Typically such recognition systems are based on TDNN in which the first neuron layers act as high-level feature detectors and the last layers act as a nonlinear classifier. Feature detection is made position and time-shift invariant by weight sharing (Bengio et al. 1995; Guyon et al. 1992; Jaeger et al. 2001; LeCun and Bengio 1994; Matíc et al. 1993).

On-line handwriting data patterns can also be represented by simple feature vectors, symbol strings, or graph structures. The components of feature vectors can either indicate the existence or absence of certain features or give measured values for them. These features can be as simple as the number of strokes and the rough orientations and the lengths of the strokes, or more complicated, such as the existence of loops, cusps, and different kinds of junctions of strokes. The major drawback of feature vectors is that they are not well suited for representing either the temporal or the two-dimensional spatial structure of handwriting. They are also rather inflexible for representing handwriting samples of varying complexity and varying number of interesting features, for example Asian characters.

The temporal structure of handwriting can be captured with symbol strings. Such strings are obtained by dividing handwriting data into successive segments which are then coded by symbols. The symbols can correspond to different types of lines, curves, loops, cusps, and corners. Such symbol strings can be seen as a special case of time series representations of the pen trace and are often called *chain codes* (Freeman 1961). Graphs can well represent the spatial structure of handwriting. Their nodes can represent different subpatterns, for example strokes, and their arcs different relationships between the subpatterns.

2.6 On-line handwriting recognition methods

On-line handwriting recognition has been an actively researched problem for decades during which several different approaches have been attempted in order to achieve systems able to perform well with various kinds of natural writing styles. Some of the applied methods have been developed especially for on-line handwriting recognition but most of them are borrowed from other fields of pattern recognition, image analysis, and signal processing. Especially many ideas have been taken from automatic speech recognition which shares many similarities with on-line handwriting recognition: the temporal nature of the signal to be recognized and the underlying language of the message being communicated (Starner et al. 1994).

The following sections describe various approaches to handwriting recognition. Even though the recognition methods are here divided into distinct categories, the different categories share many similarities. Ultimately, all the different recognition methods try to define decision boundaries in the pattern space which minimize the number of misrecognitions or the expected cost involved in the decisions. In addition, in practice many of the developed handwriting recognition systems are hybrids based on several different techniques. For example, computationally simpler methods are often used for preclassification and final decision is made with more sophisticated and computation-intensive methods in order to decrease computational burden and speed-up the recognition process. Sometimes, the recognition problem can be divided into subtasks which can be dealt more or less separately with recognition methods belonging to different categories. Each of the recognition method categories have their special properties, advantages and limitations. These will be discussed next. In addition, some application examples will be briefly reviewed.

2.6.1 Statistical methods

In statistical methods variations present in natural handwriting are assumed to be of stochastic nature. Statistical recognition methods are based on the prior probabilities of the classes, class-specific probabilities of the observed handwriting samples, and Bayesian decision theory. In a statistical recognition system the classification criterion is so designed that it minimizes the probability of erroneous classifications or the expected cost involved in the decisions, both in the correct and incorrect ones. The classification criterion of a statistical classifier defines decision boundaries in the pattern space. These decision boundaries can be approximated with various computational techniques. Thus, the decision boundaries defined by other classification methods, say the feature space classifiers, prototype-based methods, or neural methods discussed in the following sections, can be seen as approximations of the optimal decision boundaries of a statistical classifier.

Statistical methods can be divided into two categories: parametric and nonparametric methods. Parametric methods assume that the handwriting samples are statistical variables from distributions which can be characterized with sets of parameters and each class has its own distribution parameters. The parameters are estimated from training data. For example, Odaka et al. (1982), Arakawa (1983), and Loy and Landay (1982) have modeled character classes with Gaussian distributions. Odaka et al. (1982) and Arakawa (1983) represented characters with a fixed number of spatially equidistant x- and y-coordinate points or Fourier coefficient per each stroke, respectively. Loy and Landay (1982) represented the characters by chain codes. In all these three studies, characters were preclassified by the dimensionality of their representation.

Nonparametric methods do not assume any specific type or form for the class distributions. Instead, the required distributions are estimated directly from the training data, for example with histograms or kernel functions such as Radial Basis Functions (RBFs) or Parzen windows. Also the k Nearest Neighbors (k-NN) rule (Fix and Hodges 1951; Cover and Hart 1967) can be seen as a nonparametric statistical classification method, see the discussion in section 3.5.1. Parametric distributions are computationally more easy to handle than nonparametric ones as only the distribution parameters need to be stored and the class probabilities for a handwriting sample can then be calculated by using well-defined functions. However, they run the risk that the underlying assumptions for the types of distributions are improper. For example, a naive model might neglect important correlations between the features and a more complex model might have too many parameters to be estimated reliably from a limited amount of training data.

Hidden Markov Models (HMMs) are currently the most widely used parametric statistical method applied to on-line handwriting recognition. HMMs were initially introduced and studied in the late 1960's and early 1970's. HMMs have also been called Markov sources and probabilistic functions of Markov chains. HMMs are the state-of-art method in speech recognition and they have been gaining more and more popularity also in on-line handwriting recognition since early 1990's. HMMs are especially well suited for the recognition of cursive handwriting (Rabiner 1989; Plamondon and Srihari 2000). HMMs have been applied to on-line handwriting recognition by the following researchers: (Beigi et al. 1994; Bellegarda et al. 1994; Biem 2001; Brakensiek et al. 2001; Connell and Jain 2002; Hu et al. 1996; Hu et al. 2000; Narayanaswamy 1996; Pitrelli et al. 2001; Rigoll et al. 1996; Senior and Nathan 1997; Starner et al. 1994; Subrahmonia et al. 1996; Veltman and Prasad 1994).

HMMs are used for describing both the statistical properties and temporal structures of time series. HMMs are based on Markov processes whose states are not directly observable. The observations, i.e. variables which form the time series, are considered as probabilistic functions of these hidden states. A HMM can represent several things in a handwriting recognition system. Early HMM-based on-line recognition systems had separate models for each word. As the number of word models increases linearly with the size of the lexicon, systems based on character models have become more popular (Nathan et al. 1993; Veltman and Prasad 1994). As the characters usually contain patterns that are common to more than one character, stroke and other subcharacter models can also be used and concatenated to form letter models (Hu et al. 1996). Training of the recognition system is easier when the number of handwriting models is small. In the case of a large vocabulary, it can be difficult to obtain an adequate training set as several samples for each word model are needed for estimating the statistical parameters accurately and reliably.

A basic HMM is characterized by the following parameters: 1) the number of states in the model, 2) the number of distinct observations per state, 3) the state transition probabilities, 4) the probability distribution of the observation for each state, and 5) initial state distribution (Rabiner 1989). HMMs which are used in handwriting recognition are typically so-called left-to-right models (Hu et al. 1996). These models have the property that the state index increases or remains unchanged as the time elapses. A direct path, i.e. the state index increases by one in every state transition, through such a model corresponds to the average realization of the modeled process. Self loops, i.e. the state index remain unchanged, correspond to lengthened versions of the process and state skips correspond to shortened versions (Nathan et al. 1993). In speech recognition systems, the observation probabilities for each state are usually mixtures of continuous probability densities. However, discrete probability densities seem to be better suited for on-line handwriting (Bellegarda et al. 1995; Rigoll et al. 1996).

2.6.2 Feature space classifiers

When handwriting samples are represented by feature vectors they can be seen as points in a multidimensional feature space. If good features can be found, the handwriting samples will form separate clusters in the feature space so that the samples in the same cluster belong to the same class. So, each class will occupy certain regions in the feature space and the classification process can be based on the division of the feature space. The division can be defined by decision boundaries in a linear or nonlinear fashion. In the former case, the class-specific regions are defined by linear hyperplanes, and in the latter case, by more complex nonlinear manifolds. In both the cases, the division is usually defined so that the number of erroneously classified training samples is minimized. A nonlinear division of feature space can also be achieved by first mapping the feature vectors nonlinearly to higher dimensional space, then defining an optimal linear classifier in the new feature space, and finally defining the corresponding nonlinear classifier in the original feature space. This is the underlying idea of many recognition methods, for example in Support Vector Machines (SVMs) (Haykin 1999, pp 318–351), and in neural methods such as RBF (Haykin 1999, pp 256–317) and Multi-Layer Perceptron (MLP) (Haykin 1999, pp 156–254) networks. (Theodoridis and Koutroumbas 1999)

The success of the statistical and feature space classifiers depends heavily on the feature selection. In fact, if the features are good, any recognition method based on this feature-space-division idea will work well. The decision boundaries, or the free parameters of the the recognition system, are usually found with iterative optimization techniques which are computation-intensive and might get stuck in suboptimal points in the parameter space. These recognition methods are not ideal for systems which should be able to quickly adapt to a new writing style or to learn incrementally.

2.6.3 Prototype-based methods

In prototype-based methods, the classification of an unknown handwriting sample is performed by matching it to a set of known samples, the prototypes, and selecting the class according to the most similar prototypes. The following issues have to be carefully planned in order to achieve good recognition performance: 1) a representation for handwriting samples which emphasizes the inter-class differences and intra-class similarities, 2) a similarity or dissimilarity measure sensitive to meaningful variations and ignoring meaningless variations between handwriting samples of different classes, and 3) a representative prototype set covering all the principal style variations such as character allographs. Naturally, the devisings of these three basic components of any prototype-based classifier system are very dependent on each other. On of the most commonly applied decision rule prototype based system is the k Nearest Neighbors (k-NN) rule (Fix and Hodges 1951; Cover and Hart 1967): the classification of an unknown character is made according to the majority of its k most similar prototypes in the prototype set. Prototype-based recognition methods are well suited for adaptive systems. When some new training samples are obtained, the prototypes can be adjusted to represent them better. There is no need to retrain or adapt the whole recognition system but just the prototypes of the same classes as the new samples. Incremental learning can be achieved just by including the new samples into the prototype set as soon as their class labels have been determined. The prototypes, the similarity measure, and the decision rule of a prototype-based classifier together divide the pattern space into a tessellation so that each region corresponds to a certain pattern class. The decision boundaries thus defined by a prototype-based classifiers can be seen as approximations of the optimal decision boundaries of a statistical classifier.

The simplest prototype-based approach is to represent the handwriting samples with feature vectors, compare them by using Euclidean distance, and use the whole labeled training database as a prototype set. This basic approach has several obvious drawbacks. A fixed-sized feature vector might be a too simple model to represent all the temporal and spatial aspects of the handwriting samples, especially if the samples are of varying complexity. Euclidean distance treats all the features equally and implicitly assumes that the features are not dependent on each other and they are normally distributed with same variance around the values of the prototypes. Though the whole training database is guaranteed to include as many as possible writing style variations, it can make the recognition process computationally too heavy for practical applications. The recognition time depends linearly on the size of the prototype set as an unknown sample is compared to each of the prototypes. Even though several shortcuts and speedups can be used in the matching process, the most sensible approach is to use only the minimum number of prototypes. The prototype set can be created either algorithmically or by a human expert. The prototype set creation topic will be discussed in more detail in section 3.4.

Typically, on-line handwriting samples are represented by time series consisting of vectors of various trajectory features and the similarity measures used in the prototype-based recognition systems are able to compare representations of different lengths. Very often, the similarity measures are based on a dynamic programming algorithm which finds the nonlinear point-to-point correspondence between the data points of two time series which is optimal in sense of the sum of the point-wise matching costs. Dynamic Time Warping

(DTW) (Sankoff and Kruskal 1983), also known as *elastic matching*, is one of the most commonly used time series matching algorithms for on-line handwriting. The DTW algorithm was introduced to on-line handwriting recognition in the beginning of 1980's and since then, many variations to the basic algorithm have been developed. The DTW algorithm has also been used in this thesis work, for the details, see section 3.3. For more examples of recognition systems in which the DTW algorithm has been applied, see articles (Karnaugh et al. 1981; Lu and Brodersen 1984; Tappert 1984; Kurtzberg 1987; Tappert 1991; Andrianasy and Milgram 1995; Scattolin 1995; Li and Yeung 1997; Prevost and Milgram 1997; Connell and Jain 2001; Kobayashi et al. 2001).

Of course, there are many other matching methods for on-line handwriting samples than the popular DTW algorithm. For example, Pavlidis et al. (1997) and Webster and Nakagawa (1998) have designed prototype-based recognition systems in which the matching methods of cursive words or characters are physics-inspired. Handwritten shapes are considered to be made of virtual wires or springs and their matching takes place through the stretching and bending of the components of the shape models. The transformation energy required for making two shape models similar to each other is considered to be the dissimilarity measure between the handwriting samples. In the recognition system developed by Wilfong et al. (1996) handwritten characters are represented by fixed-size feature vectors consisting of the x- and y-coordinates of spatially equidistant points on the sampled pen traces. Character samples derivable from each other by translation, dilation and rotation are considered to be equal. Each character sample spans a two-dimensional subspace and the distance between two character samples is the angle between the corresponding subspaces. Wakahara and Odaka (1997) have introduced a similarity measure based on affine transformations of characters represented by a fixed number of x- and y-coordinate points. The variations between an unknown character and a prototype character are explained by Stroke-based Affine Transformation (SAT) components and residual components. The SAT components are caused by affine transformations of the characters, such as translation, rotation, scaling, and shearing. The SAT components are considered to explain the intraclass variations, whereas the residual components represent the interclass variations. First, the unknown characters are prerecognized and the prototype set is pruned by using a dissimilarity measure derived from both the SAT and residual components. The final classification is performed with a dissimilarity based only on the residual components.

2.6.4 Neural methods

Artificial Neural Networks (ANNs or NNs) are widely used for handwriting recognition. A good but quite dated review of various neural methods applied to handwriting recognition is given by Morasso et al. (1992). Some of the more recent neural approach will be cited and briefly described in this section and in the coming section 2.7.3. The whole recognition process, from data normalization to higher-level feature extraction and to pattern classification, can be based on neural methods. Alternatively, only some specific subtasks can be performed with neural methods. Neural recognition systems which are responsible for the whole recognition process are in some sense relying on the same basic idea as the systems based on prototype matching. Instead of explicitly storing prototypes, neural networks are associative memories in which the handwriting information contained in the training samples is stored implicitly in the weights of a neural network. Again, the neural classifiers, which divide their input space into regions corresponding to certain pattern classes, can be seen as approximations of statistical classifiers. In fact, the outputs of a neural network are unbiased estimates of the *a posteriori* probabilities of the patters classes if the neural network has one output neuron per each class, the desired response of an output neuron is one when the input pattern belongs to the class the neuron is assigned to and zero otherwise, and the neural network is trained so that MSE (Mean Square Error) between the actual and desired responses is minimized (Theodoridis and Koutroumbas 1999, pp. 73–75). Neural and statistical methods share the risk of satiation where the system yields an average but incomplete representation of all possible handwriting styles. This problem can be alleviated by paying special attention that the common writing styles are not overly represented in the training set or by having distinct models or subclasses for alternative writing styles.

Yaeger et al. (1998) have developed a recognition system for handwritten characters which is based on a MLP network. The input of the MLP is a combination of static and dynamic features. The static features are the gray-scale image and aspect ratio of an unknown character. The dynamic features are the number of strokes and some directional features for a fixed number of pen trajectory points. The four layers of the network are fully connected, except the input layer and the first hidden layer. The first hidden layer consists of neurons which are specialized in extracting higher-level features from the different input feature groups. For example in the case of image features, formation of translation invariant feature detectors is enforced by weight sharing. The MLP is trained by using a modified version of the back-propagation (BP) algorithm (Haykin 1999, 161–175). The authors present a number of training schemes which improve the recognition system's ability to handle rare writing styles, under-represented classes, and malformed input characters.

Articles (Mozayyani et al. 1998; Vaucher et al. 2000) describe a neural recognition system for isolated characters. The time-dependent inputs of the neural network are the four basic direction components (up, down, left, right) of the pen point movements at each sampled data point on the pen trace. The first hidden layer of the neural network consists of so-called Spatio-Temporal (ST) neurons which act as accumulators. The output of a ST neuron is zero until the cumulative sum of the inputs reaches a predefined threshold value. After that, the accumulated value is sent out as an output of the neuron and the accumulator is reset to zero. ST neurons have only limited temporal memories and the arrived input impulses are attenuated exponentially over time. The ST neurons are thus able to detect line segments of certain direction and length. A MLP network operates on the outputs of ST neurons once all the data points of the character have been handled by the ST neurons. The output layer of the MLP has two neurons for each character class so that the recognition system could better handle multiple writing styles.

Schomaker (1993) describes two variations of a recognition system for on-line cursive handwriting. Both the variations are based on the Self-Organizing Map (SOM) algorithm. The first variation of the system models handwriting on stroke-level and the second variation on character-level. In the stroke-based system, cursive writing is first segmented into strokes by local velocity minima of the moving pen point. Strokes are then represented by 14 features and modeled with a stroke-SOM. In the character-based system, cursive writing is tentatively segmented into characters by the local velocity minima. Characters consist of 1–6 strokes. They are represented by sequences of 30 x- and y-coordinate points and modeled with an allograph-SOM. The stroke-SOM and allograph-SOM provide multiple scored interpretations for a single input stroke or for a group of consecutive strokes, respectively. The character-based system is computationally more demanding than the stroke-based system due to both the higher dimensionality of the modeled data and the multiple matchings per stroke. The best recognition performance can be achieved by finding the character segmentation hypothesis with a stroke-based system and performing the final classification with a character-based recognizer.

Morasso et al. (1995) have developed a recognition system for cursive handwriting which is based on a variation of the SOM algorithm entitled Self-Organizing Classifier (SOC) (Morasso et al. 1992). The main difference between SOM and SOC is that the latter is able to learn incrementally. Also in that system, handwriting is segmented into strokes by velocity minima. Strokes are represented by eight-dimensional feature vectors. Characters consisting of different numbers of strokes are modeled with separate SOCs. The SOCs provides multiple interpretations with confidence values for the strokes. The final classification is performed by using a special language model.

In many systems, a neural method is used for higher-level feature extraction and the final classification of handwritten patterns is performed with some other method. For example, Guyon et al. (1992), Matíc et al. (1993), and Platt and Matíc (1997) have first trained a TDNN to recognize isolated characters and then replaced or augmented the last neuron layer of the network with a k-NN classifier, Support Vector Machine (SVM) classifier, or RBF Network classifier, respectively. By this way, they have combined the advantages of neural and prototype-based methods: the neural part of the recognition system is well suited for modeling higher-level spatial and temporal features of handwriting and the decision making part of the system is able to learn incrementally.

2.6.5 Structural and syntactical methods

Structural recognition methods are well suited for recognition tasks in which the structures of the patterns are of paramount importance. The measured feature values or knowledge about their existence or absence do not always provide enough information for the classification of a pattern and some additional information on the relations between the features, or on the structure of the pattern, is needed. Structural methods are especially well suited for Asian languages in which characters are more complex than Latin alpha-numerical characters but consist of rather simple subpatterns (Jung and Kim 2000; Kim and Kim 2001; Lin et al. 1993; Yang 1998). The structural, class-specific rules are often formed by human experts as they are rather difficult to learn automatically. This is especially true for syntactical recognition system in which pattern structures are described by grammars of formal languages and the recognition is based on parsing. Such methods are not well suited for recognition systems which should be able to automatically learn completely new writing styles. However, just the lowest level components of the structural and syntactical handwriting models can be adapted to better represent new handwriting sampling.

Usually in the structural recognition methods developed for on-line handwriting, handwriting samples just have structural representations or are described by a set of structural features and their classification is performed with prototype-based or feature space classifiers. Different kinds of symbol strings, or chain codes, are widely used structural representations of on-line handwriting (Bontempi and Marcelli 1994; Duneau and Dorizzi 1994; Kerrick and Bovik 1988; Qian and Truemper 1998; Li and Yeung 1997; Nouboud and Plamondon 1991; Wang and Gupta 1991; Yuen 1996). Chain coding means that the captured pen trace is divided into temporally adjacent segments which are then coded. The segmentation can be based on pen lifts, various local extreme points, or just evenly spaced points on the pen trace. The codes can represent straight line segments of equal lengths with discrete directions, lengths of line segments and continuous angles between them, different types of curved strokes, loops, and corners etc. If the codes have only discrete values, the handwriting patterns can be represented by symbol strings. Chain codes are one-dimensional structural models which can represent the temporal structure of on-line handwriting very well. Their ability to capture the two-dimensional spatial structures of handwriting is rather limited and it heavily depends on how complicated features the codes correspond to. Classification of chain-coded handwriting patterns can be performed in several ways, for example with statistical methods(Loy and Landay 1982), string comparison (Nouboud and Plamondon 1991), or dynamic programming methods (Yuen 1996).

Kerrick and Bovik (1988) have designed a hierarchical on-line character recognition system which is based on structural features of three levels and on prototype matching. The low-level features are the existences and locations of the endpoints and T-crossings of the strokes. The location features can have only four values which correspond to the quadrants of the bounding boxes drawn around the characters. These low-level features are easy to extract from the character samples. The intermediate-level features, the corners and their locations, are found by a chain code-based algorithm. In this case, a corner is an area of sustained high stroke curvature. The intermediate-level features are computationally more demanding than the low-level features. In the first stage of the recognition process, the prototype set is pruned by testing the absence or presence of particular low- and intermediate-level features in given bounding box quadrants according to a binary decision tree. In the next stage, an unknown character sample is matched stroke-wise against the remaining prototypes by using high-level features. Each stroke is associated with a shape which can be either straight or curved. A straight stroke can have four orientations. Curved strokes are divided into four segments of equal lengths. The average chain code value is computed for each segment and the shape of the stroke is coded by four digits. Two strokes are considered equal if they both are straight and have the same orientation, or, if the digits of their shape codes are the same or differ only by one unit. If several prototypes of different classes match to the unknown character, also the aspect ratios of the characters are compared and some additional class-dependent feature tests are performed.

The structural recognition method for on-line characters developed by Chan and Yeung (1999) is based on a flexible prototype matching scheme. Characters are composed of strokes which are represented by sequences of the following primitives: straight line segments, counter-clockwise or clockwise curved segments, loops, and dots. The first three primitives are associated with an eight-valued direction feature. Each character class is represented by several prototypes. The prototype matching scheme allows structural deformations of four levels. A matching procedure starts from the strictest deformation level and proceeds to the next and more relaxed level if no matching prototypes are found for the unknown character sample. At the first level of matching, no deformations are allowed and the structures of the prototype and the unknown character samples have to match exactly. At the second level of matching, some primitive type deformations are allowed: a straight line segment can be matched with both types of curved segments. At the third matching level directional deformations are allowed: directional features of the primitives can differ by one unit. The fourth matching level allows both primitive type and directional deformations. This matching scheme does not always produce a single solution as some of the alphanumeric character classes have exactly similar structures. In those cases, the final classification is based on additional class-specific features, such as the relative positions or height differences of the strokes.

Jung and Kim (2000) have combined neural networks and graph representations in their recognition system for on-line Korean Hangul characters. The Hangul characters are composed of from two to four graphemes. The graphemes correspond to vowels and consonants and are considered to be either simple or complex. In total, there are 50 different graphemes. A Hangul character has a clear two-dimensional structure and there are six basic structures how the graphemes can be arranged into a character. There are 11 172 possible characters of which 3 000 are in daily use and 500 cover 99% of the Korean texts. Jung's and Kim's recognition system first produces several tentative grapheme segmentations each of which are then recognized by a TDNN. After that, the unknown character is represented by a graph whose nodes correspond to scored and labeled grapheme segments and arcs to the transition probabilities between the graphemes. A character sample is recognized and its segmentations into graphemes are determined simultaneously by finding the most probable path through its graph representation by using the Viterbi algorithm (Theodoridis and Koutroumbas 1999, pp 309–312).

2.6.6 Generative models of handwriting

Some of the handwriting recognition methods reported in the literature are based on generative models of handwriting. For example, some recognition systems are based on the idea that the handwriting process can be described by a model of hand movements and different classes correspond to different model parameters and model structures. Plamondon and Srihari (2000) divide the generative models of handwriting into two groups: *oscillatory* and *discrete* models. In oscillatory models, handwriting is considered to be produced by oscillating hand movements. In order to obtain differently shaped pen traces, the amplitudes, phases, and frequencies of the oscillations are modulated. With these models, discontinuous movements, such as isolated strokes, are explained by interruptions of the oscillation process and abrupt changes of the oscillation parameters. Identified oscillation parameters can serve as features in on-line handwriting recognition. In discrete models, hand movements are seen as superimpositions of simple basic strokes. Continuous movements are explained by the time-overlapping of the discrete strokes. With discrete handwriting models, an identified model structure provides the basis for the segmentation of the handwriting data into meaningful subpatterns. Next, some examples of oscillatory and discrete models developed for on-line handwriting are described briefly.

Singer and Tishby (1994) have considered handwriting as modulations of a simple cycloidal pen point motion, described by two coupled oscillations with a constant linear drift along the line of writing. With their model, a pen trace can be encoded by the slow modulations of the amplitudes and phases of the oscillations. Pen traces are first segmented by points of zero vertical velocity. Next, amplitude and phase modulations are estimated for each segment. Finally, these estimates are quantized to a small number of levels. By this procedure, continuous pen point movements can be represented by time series of discrete motor parameters. A quite similar oscillatory modeling approach has been described by Beigi (1997).

Lorette (1999) describes a discrete generative model for cursive handwriting. According to his model, the fundamental units of cursive handwriting are basic downstrokes, beginning and ending strokes, within-letter and between-letter ligatures, letters, and words. Basic downstrokes are vertical strokes drawn in downward direction. They constitute the kernels of the letter structures. Beginning and ending strokes are the first and last strokes of the letter structures. Within-letter ligatures are cusps, small loops, and arcs which connect the strokes belonging to the same letter structure. Word structures are formed by concatenating several letter structures so that strokes belonging to different letter structures are connected by arcs called between-letter ligatures. Between-letter and within-letter ligatures can be invisible if they correspond to hand movements in which the pen is lifted up. These fundamental units are segmented by pen lifts, by points of curvature inflection or discontinuation, and by local curvature and vertical position extrema.

Plamondon and Guerfali (1998) have developed a generative model for handwriting which is based on kinematic theory of rapid human movements. According to this discrete handwriting model, basic strokes are produced by controlling the pen point velocity by synchronously activating an agonist and antagonist neuromuscular system. The neuromuscular systems are considered to have impulse responses which can be described with lognormal functions. Basic strokes are characterized by velocity vectors defined by nine parameters. Fluent cursive writing consists of superimposed basic strokes. On-line handwriting can be encoded by extracting the underlying basic strokes and estimating their model parameters at each data point on the sampled pen trace. This generative model has been applied in Scriptôt handwriting teaching system aimed for primary school children (Djeziri et al. 2002).

2.7 Adaptive recognition systems for on-line handwriting

The need for adaptive recognition methods is obvious, no matter should the recognition system be a writer-dependent or writer-independent one. The major problem of the writer-independent systems is that there will always be some users whose writing styles were under-represented or not at all contained in the training database. Therefore, even if a writer-independent system performed well on average, its recognition accuracy would be intolerably low for some of the potential users. This problem can be alleviated by adapting the writer-independent recognition system to the writing styles of its end-users.

In the case of writer-dependent systems, the end-users have to write all the training samples. Depending on the applied recognition methodology and the level of abstraction of the handwriting models, that means something from one sample up to tens of samples per each pattern class or handwriting model. In addition to a tedious data collection, the users have to wait for the system to be trained from scratch. The users would need to provide fewer training samples if the recognition system were first trained with handwriting samples collected from other writers and then just slightly modified to better handle the particular writing style of the current user. In fact, better recognition accuracies can be achieved with writer-dependent recognition systems if they are initially trained with data collected from several other subjects (Nathan et al. 1995; Yaeger et al. 1998). In addition, with most of the recognition methods, the adaptation of the recognition system requires considerably less computation and memory resources than the whole training process and can therefore be performed on-line.

Style-specific recognition systems are kind of compromises between writer-independent and writer-dependent systems. They can handle various personal writing styles better than writer-independent systems but are less specialized than writer-dependent systems. However, the selection process of the most suitable style-specific recognition system for the current user requires some handwriting samples and it can be seen as one form of system adaptation. Naturally, also the recognition performance of a style-specific system can be improved by adapting it to a writer-dependent system.

Adaptive recognition systems can be divided into different categories depending on whether the adaptation is performed in a supervised, unsupervised, or self-supervised manner. In the supervised methods, the classes of all the training samples are assumed to be known and the recognition system is adapted so that it can better recognize these labeled handwriting samples. On the contrary, in the unsupervised methods, the classes of the training samples are not known and the recognition system is tuned on the basis of some other criteria than the recognition accuracy. For example, if handwriting is modeled on subcharacter level, the handwriting models can be tuned so that they better represent the new handwriting samples regardless of their classes. In the self-supervised methods, the true class labels are not available but the handwriting samples are labeled by the recognition system itself on the basis of some indirect information. For example, the recognition system might deduce the class labels from the user's actions and responses to the recognition results as will be described in section 3.6.5.

Another categorizing feature of the adaptive systems is the mode of the adaptation: whether the system is adapted on-line or off-line. On-line adaptation is performed continuously as the user provides new handwriting samples and the recognition performance improves gradually. On the contrary, off-line adaptation is carried out in batch runs after the user has provided a certain number of training samples, possibly in a special training mode of the system. Therefore, there will be some delays before the new training samples have any effect, and the performance and the behavior of the recognition system change more suddenly and in more drastic ways. That might be confusing for the users, especially, if they consciously try to alter their writing styles to be better recognized by the system. Another fact that speaks for on-line adaptation during the normal use of the device is that the subjects tend to write differently in training sessions than when performing real tasks (Goldberg and Goodisman 1991).

The adaptation process of a handwriting recognition system should be fast, robust, incremental, and understandable for the user. An adaptive recognition system should notice if there are any problems in the recognition of some of the classes and adapt itself to new writing styles on the basis of as few as possible training samples. The adaptation process should not be too sensitive to erroneous learning samples, both mislabeled and malformed ones, as they cannot be completely avoided in real life. *Incremental learning* means that the system is able to learn something completely new, for example can establish new allograph models for handwritten characters, instead of just perfecting its ability to handle the handwriting styles it already knows to some extent. The ability to learn incrementally is of paramount importance if the recognition system has not been trained with handwriting samples collected from its end-users. If the users understand how the adaptation process works, they can take the most advantage of it and even intentionally train the system. Otherwise, the users might be tempted to write in some unnatural or exaggerated way which would help neither the recognition nor the adaptation process. In the worst case, adaptation might turn out to be more harmful than useful.

Several adaptive handwriting recognition systems have been reported in the literature. The following sections briefly review how adaptation has been realized and experimented with in prototype-based systems, statistical systems, and in systems based on neural networks. Some of these recognition systems have already been discussed in the earlier sections 2.6.3, 2.6.1, and 2.6.4. The recognition accuracies reported for the different recognition systems are not comparable with each other and they are given here only for judging the beneficial effects of the adaptation.

2.7.1 Adaptation in prototype-based systems

A prototype-based recognition system can be adapted to new writing styles by modifying the prototype set in the following ways: 1) new handwriting samples provided by the current user can be added to the prototype set, 2) existing prototypes can be adjusted to better represent the current user's writing style, and 3) unused or poorly performing prototypes can be inactivated. The first form of adaptation enables the recognition system to learn incrementally. It is likely to improve the recognition rates quickly but also to increase the recognition time as the computational complexity of a prototype-based system depends linearly on the number of prototypes. The second form of adaptation provides more gradual improvements of the recognition accuracy and it might not be sufficient if the end-users' writing styles differ drastically from the writing styles the system was initially trained to handle. The third form of adaptation is necessary if some of the training samples are erroneous, either malformed or just incorrectly labeled. Otherwise, the adaptation can turn out to be more harmful than useful. (Vuori et al. 2002, included publication 2)

Kurtzberg and Tappert (1981) have developed a prototype-based recognition system for isolated characters. In their system, characters are represented by time series of the pen point's normalized heights from the baseline of the writing and tangential angles of the pen trace. The unknown characters are compared to the prototypes by using a DTW-based method called elastic matching. The classification of the unknown characters is based on the 1-NN rule. The initial prototype set is formed in a special training mode in which the system prompts the user to write specific characters. After that, the recognition system is adapted to the new writing style by adding all the misrecognized character samples and the correctly classified character samples for which the ratio of distances to the best-matching correct and to the second best-matching incorrect prototype is within a given tolerance into the prototype set. Tappert (1984) developed the recognition system further by extending it to handle also cursive words and by adding two new features to the time series representation of the characters: the normalized horizontal and vertical offsets of the pen point from the mass center of the character. Tappert performed experiments with isolated characters written by 6 subjects not familiar with the recognition system. The recognition system was trained independently for each subject. The writer-dependent error rates for the 26 upper and 26 lower case Latin letters and the 10 digits were between 0.3% and 4.3% if there were separate prototype sets for the three character groups, and 1.2% and 8.9% if there was only one prototype set. Tappert performed experiments also with cursive words written carefully by 3 subjects familiar with the recognition system. The recognition system was able to correctly segment all the word samples into characters and the letter-level writer-dependent recognition error rates were between 2.1% and 3.6%. These early results were really promising and they showed that the DTW-based distance measure is suitable for the recognition of both isolated characters and cursive words.

Fujisaki et al. (1991) have developed a recognition system for run-on discrete handwriting. Their system first classifies strokes and then generates character hypotheses which are verified with a DTW-based method. The system's recognition performance was evaluated for 82 character classes (upper and lower case Latin letters, digits, and special characters) and 12 subjects. Each subject wrote approximately 1 300 and 1 500 character samples which were used as a writer-dependent prototype set and a test set, respectively. The subjects were advised to write similar characters of different classes differently and not to use cursive style, i.e. connect adjacent characters. The average writer-dependent character error rates were 9.3%, 3.7%, or 2.8% for the recognition problem of 82 classes, upper case letters, or digits, respectively. Fujisaki et al. experimented also with on-line adaptation with the same 12 subjects: the average recognition accuracy leveled off after the subjects had written four character samples per character class.

Bontempi and Marcelli (1994) have designed a prototype-based character recognition system in which on-line character samples are represented by symbol strings and a writer-independent prototype set is first formed and then adapted into a writer-dependent one with a genetic learning algorithm. In their system, character samples are segmented into four components, each of which is encoded by eight three-valued symbols. The first two symbols represent the curvature of the component, the next three symbols represent the writing direction, the following symbol represents the relative length of the component in respect to the total length of the pen trace, and the last two symbols represent the spatial relationships between the current and successive component. The coding scheme is so designed that small variations in the character's shape correspond to small variations in the character's symbol string representation. The recognition system stores the representations and class labels of all the misrecognized character samples and adapts its prototype set if the error rate and the number of stored character samples are high enough for some character class. The correct class labels are provided by the user. The genetic learning algorithm can add new writer-specific prototypes into the prototype set. However, it does not alter or remove the original writer-independent prototypes. This way, the recognition system is able to learn incrementally and will not lose its ability to recognize the handwriting styles not used by its current user. In the experiments performed by Bontempi and Marcelli, an initial writer-independent prototype set was learned from a database written by 50 subjects and containing lower and upper case letters and digits. The number of the initial writer-independent prototypes was 20 per each character class. In the adaptation experiments, the threshold error rate and the number of misrecognized character samples required for the activation of the adaptation process were 80% and 20, respectively. Without any adaptation, an average error rate of 1.9% and rejection rate of 14.7% were achieved for over 10 new subjects. With adaptation, these figures were significantly better, namely 0.8% and 6.5%. However, these results were obtained by using the same data sets for both the adaptation and testing of the recognition system.

In the on-line cursive word recognition system developed by Duneau and Dorizzi (1994), on-line handwriting is represented by concatenated letter prototypes. On-line handwriting data is segmented into strokes which are represented by five-dimensional feature vectors. A stroke can belong only to one letter. The structure of the stroke sequence, or a letter hypothesis, is obtained by encoding the strokes by five-valued symbols. These structures capture the rough shapes of the letters. The initial writer-independent prototype set was formed with an incremental clustering algorithm applied separately to the sets of character samples of certain class and structure. Each prototype has a confidence value which reflects its representativity and discrimination ability. The prototype set and the confidence values of the prototypes are adapted regularly after the user has provided a certain number of word samples. The adaptation process has three phases: 1) the word samples are segmented into letters which are used for updating the letter prototype set with the incremental clustering algorithm, 2) the confidence values of the prototypes are updated, and 3) prototypes whose confidence values are smaller than a given threshold value are removed. The first phase of the adaptation can alter the existing prototypes, generate new prototypes, or even new structures, or has no effect at all. In the third phase of adaptation, only those letter prototypes which have existed for long enough can be removed. Duneau and Dorizzi performed experiments in which the initial writer-independent prototype set was formed by clustering the letters of 10 000 word samples provided by ten subjects and was then used for the recognition of 1 000 word samples written by one new subject. Without any adaptation, the word error rate was approximately 19%. The word error rate improved significantly due to the adaptation. The letter prototype set was adapted after each set of 100 word. The word error rate was 17% for the first and 6% for the second set of 100 words. After the sixth set of 100 words, the word error rate stayed below 2%. The lexicon used in the experiment contained 10 000 English words.

On-line cursive handwriting is represented by sequences of character prototypes also in the recognition system developed by Qian and Truemper (1998). This recognition system has three modules: a preprocessing module, an interpretation module, and a learning module. The preprocessing module segments the captured pen trace into strokes according to pen lifts, local maxima and minima of the y-coordinate, and cusp points. Strokes are then classified into five shape categories and the connections between successive strokes are divided into three categories. In addition, the average slope is calculated for all strokes and the average curvature for strokes belonging to two particular shape categories. Thus, pen traces are represented by chain codes. The interpretation module identifies several, possibly overlapping, letter candidates from the chain code representation and scores them by using a prototype matching algorithm. A word score is obtained by finding the optimal sum the letter scores minus the lengths of the unidentified parts of the chain code. The word with the highest score is the recognition result. The learning module adapts the character prototypes according to the recognition results or word labels supplied by the user. The learning module segments the word samples into letters in the same way as the interpretation module and localizes the chain code segments which were erroneously recognized. Each erroneously recognized segment can give rise to one of the following forms of adaptation: 1) the segment can be added into the prototype set, 2) the least frequently used prototype can be replaced by the segment, and 3) the parameters of an existing prototype can be adjusted so that the matching score between the prototype and chain code segment is increased. Qian and Truemper formed an initial prototype set by collecting four samples per each of 26 lower case letter class from a single writer. After that, they tested the recognition performance of the system with sets of 100 English words written by four new subjects. The adaptation of the letter prototypes improved the system's recognition accuracy significantly as the average word error rate was 34.5% without and 18.7% with adaptation. The lexicon used in the experiments comprised 10 000 words.

2.7.2 Adaptation in statistical systems

Statistical handwriting recognition systems can be adapted into new writing styles by updating the probability distributions on the basis of new handwriting samples. Incremental learning is not as easy and fast as with prototype-based methods because new statistical models cannot be learned reliably with only one or a few samples.

Loy and Landay (1982) have performed an early experiment with an adaptive statistical character recognition system. In their recognition system, alphanumeric characters were represented by chain codes and 69 allograph classes were modeled by Gaussian distributions of diagonal covariance matrices. The dimensionalities of the chain code representations of the allograph classes were between 1 and 6. The database used in a simulated on-line adaptation experiment was written by a single subject and contained several character samples per each of the allograph classes. The initial variances of the Gaussian distributions were estimated from character samples written by some other subjects. The means were apparently initialized by the chain code representations of the first samples of each allograph class written by the single test subject. Each time a character sample was correctly recognized, the mean and the variance of the corresponding Gaussian distribution were updated. According to the experiments, the sum of the error rate and rejection rate was approximately 4.5% after the recognition system had seen 2 samples per each allograph class. After 4 samples per each allograph class, the sum stayed between 2.5% and 1%. These results were really promising, especially as the system's recognition performance improved so quickly. However, there is no guarantee that the results would generalize to other test subjects.

Subrahmonia et al. (1996) have carried out experiments in which a HMM-based writer-independent recognition system for cursive on-line handwriting is adapted into writer-dependent one. Their recognition system had one HMM per each character class. Word HMMs were formed by concatenating character HMMs. The size of the lexicon was approximately 22 000 words. The HMMs' state-dependent probability distributions for observations were mixtures of Gaussian distributions with diagonal covariance matrices. The total number of model parameters was reduced by parameter tying, i.e. the Gaussian distributions were shared across all the states of all the character HMMs. First, all the observation vectors in the training database were clustered with the C-means algorithm (Duda and Hart 1973) in order to find initial parameter values for the shared Gaussian distributions. Next, the character HMMs were trained with the expectation-maximization (EM) algorithm (Haykin 1999, pp 382). The writer-independent character models were trained with a handwriting database written by approximately 100 subjects. Next, the writer-independent models were adapted into writer-dependent models by training them with 4 000 additional labeled word samples collected from each of the 5 new writers. The means, covariance matrices and mixture coefficient of the character HMMs were updated. The recognition performance of the system was measured with test sets of approximately 400 words. These experiments showed dramatic improvements in the personal word error rates during the first 500 words. In the end of the adaptation, the word error rates had decreased about 60% on average. Nathan et al. (1996) have performed supplementary adaptation experiments with the same recognition system and training scheme, and 7 new subjects. The result were similar to those reported by Subrahmonia et al.

Also Senior and Nathan (1997) have run adaptation experiment with a very similar statistical recognition system. The main difference to the experiments carried out by Subrahmonia et al. and Nathan et al. was that the Maximum Likelihood Linear Regression (MLLR) algorithm was used for the training of the HMMs instead of the EM algorithm. With the MLLR algorithm, the writer-independent estimates for the means and variances of the Gaussian distributions of the HMMs had to be adapted into writer-dependent estimates separately from each other. The mixture coefficients were not updated. Senior and Nathan performed adaptation experiments with 12 subjects each of whom had provided 200 word samples for the adaptation and about 500 word samples for the recognition performance evaluation. They experimented with both a supervised and a self-supervised adaptation scheme. In the latter scheme, the word samples used in the adaptation were labeled according to their recognition results. These experiments showed that the both adaptation schemes were able to decrease the word error rate significantly. After 200 word samples, the word error rate had decreased, on average, by approximately 12% and 8% with the supervised and self-supervised scheme, respectively.

Connell and Jain (2002) have designed a recognition system for on-line handwriting which is based on allograph HMMs, i.e. the system has several HMMs per each character class. Also in this system, word-level HMMs are formed by chaining character-level HMMs. The allograph HMMs are left-to-right models with a single Gaussian distribution per stage. The covariance matrices of the Gaussian distributions are diagonal. This is made justifiable by decorrelating the features with a Principal Component Analysis (PCA)-based (Haykin 1999, 396–404) method. The allograph HMMs were trained with the Baum-Welch algorithm (Rabiner 1989) and discretely written character samples and cursive word samples which have been manually segmented into characters. The writer-independent character allographs were found with an iterative clustering algorithm. The adaptation process was partly self-supervised: all the training samples had character class labels and the allograph labels were determined by the most probable writer-independent allograph HMMs of the same class. Connell and Jain performed adaptation experiments with both isolated characters and cursively written words. Only those allograph HMMs for which there were enough training samples were adapted. In the experiments with lower case letters, 84 allograph HMMs were first trained with a database which contained approximately 80 500 character samples written by over 100 subjects. These writer-independent allograph HMMs were further adapted into the writing styles of 11 new subjects each of which had provided over 2 000 character samples. The data from each writer was randomly split into training and test sets of approximately equal sizes. The experimental results showed that due to the adaptation the personal recognition error rates decreased by 54% on average. In the experiments with cursive words, the writer-independent allograph HMMs were adapted with word samples written by 8 subjects. The size of the lexicon was 483 words and the number of character classes was 93. Each subject provided approximately 600 words so that there were at least 5 samples per each character class. Also in these experiments, significant improvements in the error rate were gained due to the adaptation: the word error rate decreased by 9%. In the both sets of experiments, much better recognition results were obtained if multiple HMMs were used instead of a single HMM per each character class. (See (Connell 2000) for more details.)

Brakensiek et al. (2001) have developed a HMM-based recognition system for on-line cursive handwriting and compared different adaptation techniques. In their recognition system, one writer-independent left-to-right HMM was trained for each character class by using the Baum-Welch algorithm and approximately 25 000 word samples written by 145 subjects. The state-dependent probability distributions for the observations were modeled with mixtures of Gaussian distributions. The writer-independent HMMs were adapted to the writing styles of 21 new subjects each of which had provided two sets of approximately 100 word samples used for the adaptation and testing of the system. The adaptation was performed with several algorithms, the maximum a posteriori (MAP) (Theodoridis and Koutroumbas 1999, pp 31–33), MLLR, or EM. According to their experiments with supervised adaptation, the MAP and EM algorithms clearly outperform the MLLR algorithm and 100 word samples are not enough to adapt all the parameters of the HMMs: the word error rate actually rose when the means and variances of the Gaussian distributions were updated simultaneously. The word error rate improved by 12%, 33%, and 39% when only the means of the Gaussian distributions were adapted with the MLLR, MAP, and EM algorithm, respectively. Brakensiek et al. performed also some experiments with self-supervised adaptation by using the EM algorithm and word samples labeled according to the recognition results of the writer-independent recognition system to update the means of the Gaussian distributions. In this way, the word error rate was decreased by 30%. The recognition results cited here were obtained by using a lexicon of approximately 2 200 German words.

2.7.3 Adaptation in neural systems

Recognition systems based on neural networks can be adapted to new writing styles in two different ways. The free parameters of the network, i.e. the synaptic weights associated with the connections between the neurons and the bias terms of the activation functions of the neurons, can be adjusted so that the network performs better with the handwriting samples of the current user. Alternatively, the structure of the network can be altered by adding new neurons, removing existing neurons, or changing the connections between the neurons. Only the latter form of adaptation enables incremental learning.

Guyon et al. (1992), Matíc et al. (1993), and Platt and Matíc (1997) have developed adaptive character recognition systems based on neural methods. They all have first trained a TDNN to recognize isolated characters and then replaced or augmented the last neuron layer of the resulting network with a k-NN, SVM, or RBF classifier. The recognition system developed by Guyon et al. (1992) was initially trained with character samples written by 350 subjects. The output layer of the TDNN was then replaced by a 3-NN classifier so that the weight vectors of the output neurons were used as initial writer-independent prototypes for the corresponding character classes. Correspondingly, character samples were represented by the outputs of the neurons in the second last layer. Prototypes and unknown character samples were compared with each other by using the dot product as a similarity measure. The 3-NN classifier was adapted to new writing styles by adding the misrecognized character samples to the prototype set. The correct class labels of the learning sample were provided by user. The average character error rate of the writer-independent system was 2.8% for upper case letters and digits written by some new test subjects. The letters and digits were recognized separately. The writer-independent recognition system was adapted into writing styles of 7 new test subjects each of which wrote in total 450 upper case letters and special characters. The writer-dependent recognition system was adapted to handle 7 new character classes not known by the initial writer-independent system. At the end of the adaptation, the character error rate was in the best cases below 0.1% and on average between 1% and 2%. According to Guyon et al., their recognition system was able to learn new character classes and new writing styles on the basis of only a few new samples.

Matíc et al. (1993) performed adaptation experiments with the same writer-independent character recognition system and the same 7 subjects as Guyon et al. (1992). Instead of a k-NN classifier, they replaced the last neuron layer of the TDNN with a SVM. The SVM consisted of 36 hyperplanes each of which separated one of the 26 upper case letter and 10 digit classes from the rest of the character classes. The hyperplanes of a SVM are defined by weighted sums of training patters called supporting patterns. The supporting patterns are the training samples closest to the class boundaries. The training of a SVM involves the solution of a quadratic optimization problem with linear constraints. The SVM of the writer-independent recognition system was adapted into new writing styles by retraining its hyperplanes by using only the supporting patterns and the misclassified new training samples. At the end of the adaptation, the character error rate was on average 2.5%. Thus, the k-NN classifier is not only computationally more simple to adapt to new writing styles but it achieves on average higher recognition accuracy than the SVM classifier. However, the recognition process itself is computationally more complex with the k-NN classifier.

Also Platt and Matíc (1997) performed adaptation experiments with the writer-independent TDNN-based character recognition system developed by Guyon et al. (1992). They augmented the output neuron layer of the TDNN with a RBF network. The output vectors of the TDNN were used as the input vectors of the RBF network. The components of the output vector of the whole recognition system were calculated by adding differently weighted sums of the outputs of the RBF neurons to the components of the output vector of the TDNN. The recognition system was adapted to new writing styles by adding new RBF neurons or modifying the weights associated with the outputs of the RBF neurons. A new RBF neuron was added if the Euclidean distance between the output of the TDNN and its nearest RBF center exceeded a certain threshold. Otherwise, the weights were adjusted so that the output of the whole system was closer to the desired output. The initial writer-independent recognition system was trained to handle in total 72 classes of upper and lower case letters, digits, and special characters. Platt and Matíc performed experiment with supervised adaptation and 5 subjects each of which wrote from 39 to 93 word samples. The word error rate decreased by 45% due to the adaptation. At the end of the adaptation, the size of the RBF network was between 3 and 25 neurons.

Schomaker et al. (1993) have developed a recognition system which models on-line cursive handwriting with SOMs. First, a writer-independent recognition system is obtained by training a SOM with strokes extracted from manually labeled word samples written by 17 subjects. Several alternative stroke interpretations are attached to each neuron of the resulting stroke-SOM. The recognition of cursive words is based on the stroke-SOM and a stroke transition network which defines the possible stroke sequences and their allograph interpretations. For a more detailed description of their system, see also article (Schomaker 1993) and the previous section 2.6.4. The writer-independent recognition system is then adapted to new writing styles by updating the stroke transition network. The reference vectors of the neurons of the stroke-SOM are not updated. The adaptation process has three stages: 1) initial allograph probability adaptation, 2) allograph labeling, and 3) final allograph probability adaptation. In the first adaptation stage, the stroke-SOM and stroke transition network are used for the recognition of a new set of word samples. The recognition system provides a list of 20 candidate output words for each new word sample. If the correct recognition result is among these candidates, the probabilities of the matching allographs in the stroke transition network are gradually increased until the correct word is at the top of the candidate list or a maximum number of iterations is reached. In the second stage, the allographs of the unrecognized words are manually labeled by the user. However, completely idiosyncratic shapes are left without labels. The third stage of adaptation is the same as the first one but the starting point is the initially adapted stroke transition network and all of the new word samples are used for its final adaptation. Schomaker et al. performed adaptation experiments with 16 subjects, each of which provided from 49 to 180 word samples for both the training and testing of the recognition system. The recognition system was adapted separately for each subject. Before any adaptation, the word error rate was 65% on average. After adaptation, it was 37%. Approximately 45% of the training samples used for the adaptation had to be manually labeled. The size of the lexicon used in the experiments varied between 5 000 and 7 000 words.

Yaeger et al. (1998) have performed adaptation experiments with their MLP-based recognition system for isolated characters (see also section 2.6.4). The recognition system was first trained with character samples written by 45 subjects. The character samples were of 95 classes of upper and lower case letters, digits, and special characters. The resulting writer-independent recognition system was then separately adapted to the writing styles of three new subjects. The character error rate of the writer-independent system was on the average 21.3%. After adaptation, the average character rate had decreased down to 7.2%. If the recognition system was trained from scratch for each new subject, the average character error rate was slightly higher, 7.4%.

2.7.4 Conclusions and remarks

The error rates given in the three earlier sections for various handwriting recognition systems are not comparable with each other for several reasons. First of all, the systems were trained and tested with different data. In all the cases, there were significant variations in the error rates obtained for different writers both before and after the adaptation of the recognition system. That can be explained by the fact that the writing styles of some of the subjects are easy to recognize and learn both by humans and machines whereas some of the writing styles are nearly illegible scribble. In the case of easy writing styles, characters of different classes have distinct and consistent shapes which are commonly used by many writers. The recognition systems posed different constraints on the writing styles. The handwriting databases used in the experiments were collected with various writing equipment and therefore the handwriting samples were of varying quality. In addition, the number of classes varied between the recognition systems. In the case of cursive word recognition, also the size of the lexicon, the used language models and other postprocessing methods have a significant effect on the error rates. In despite of all these facts, the results obtained with the different recognition systems show clearly that personal error rates can be significantly decreased by adapting a writer-independent recognition system to a writer-dependent one. However, these previous experiments have been just simulations of on-line adaptation with perfectly labeled training samples and therefore the effects which erroneous learning samples or user adaptation would have on the adaptation process and development of the recognition accuracy have not been considered at all. For a summary of the recognition performances obtained with the most recent and promising systems, see Tables 1 and 2 in section 2.10.

2.8 Language modeling

Handwritten text can be divided into basic units such as characters, syllables, words, phrases, sentences, lines, and paragraphs. The syntactic and semantic meaning of a textual unit is defined by its context, i.e. the surrounding units. The use of contextual knowledge facilitates the recognition of the otherwise very confusing parts of the text. Contextual knowledge can be represented by the following language models: a list of acceptable character combinations (character N-grams), probabilities of character N-grams, a list of acceptable words (lexicon), rules for acceptable words in certain global or local context, a list of acceptable word combinations (word N-grams), probabilities of word N-grams, and a grammar describing the syntax of the language (Rose and Evett 1995; Srihari 1985). For examples of recognition system in which language models have a central role, see (Beigi 1992a; Beigi 1992b; Clergeau and Plamondon 1995; Fujisaki et al. 1991; Hu et al. 1996; Yaeger et al. 1998).

Language modeling is of paramount importance in the recognition of cursive handwriting, especially in the character segmentation phase and in the postprocessing of recognition results. The recognition of cursive words can be performed more quickly and accurately if the tentative segmentations which lead to very unlikely character combinations or to words which are not included in the lexicon are pruned away as early in the recognition process as possible. Language models can also be used in the recognition of isolated characters and they usually improve the error rates (Tappert et al. 1990). However, in some applications the existing contextual information is very limited. The used language can be unknown and the input text totally unconstrained so that it may not have any apparent meaning. A situation like this is very likely to occur in a PDA application in which the users typically write down names and nicknames, addresses, telephone numbers, codes and other short personal notes in multiple languages. However, some simple language-independent models, for example case rules, can be very useful in the postprocessing of the recognition results also in such applications.

2.9 Usability issues

The recognition accuracy is one of the key factors determining the acceptability of a handwriting recognition system and the whole application in which it is implemented. Other important usability factors are the reliability, ease of the use, and understandability of the recognition system. The required recognition rate for a handwriting recognizer is very high – even higher than humans can perform. Experiments with good typists and a special keyboard that introduced random errors with a predefined rate have shown that writers tolerate errors up to 1% while 0.5% is unnoticeable and 2% is intolerable (Guyon and Warwick 1996). These error rates are clearly lower than the average error rate of human readers, which is approximately 4% according to the experiments reported by Neisser and Weene (1960) and Parizeau and Plamondon (1994). However, the relationship between the user acceptance of a pen-based interface and its recognition accuracy is highly task-dependent: a recognition accuracy not adequate for writing tasks involving large vocabularies might be totally acceptable for more limited tasks such as form-filling (Frankish et al. 1995). The reliability means that the recognition system is not likely to make mistakes which might go unnoticed by the user but rather rejects suspicious input samples than falsely recognizes them. High reliability is important, for example, when a user is inputting passwords, financial amounts, telephone numbers, or e-mail addresses.

The ease of use means that characters can be input in a natural writing style and speed, it is easy to learn and remember how to use the recognition system and how to write in the required way, and the recognition errors and writing mistakes can be corrected in a natural and effortless ways which do not disturb the writing process too much. As human handwriting speed is 12–33 words per minute (MacKenzie et al. 1994), the recognition time should be less than about 300 ms per character. Otherwise, quick writers need to wait for the recognition results. The ease of correction was found to be at least as important usability issue as the recognition accuracy in the usability study of various text entry methods for hand-held devices conducted by Guyon et al. (1995). Some of the potential users might be intimidated if the recognition system requires that the characters are written and segmented in some specific way. However, the user experiments performed by MacKenzie and Zhang (1997) with a recognition system that required special Graffiti characters and with 25 subjects showed that the subjects willing to learn are able to master the Graffiti characters quickly and do not easily forget what they have learned. The average walk-up recognition accuracy of Graffiti recognizer was 87%. After five minutes practice and a week later without any additional practice, average accuracies as high as 97% were reached.

The understandability of the recognition process is important because the users are able to alter their handwriting styles to be better recognized only if they have some insight into the possible causes of the recognition errors. It helps if the trace of the pen is shown to the users – otherwise the users do not know whether the recognition errors were due to hardware problems in capturing the pen trace, careless writing, or the limited capabilities of the recognition system to handle certain character allographs. The recognition errors which do not seem random are less annoying to the users (Webster and Nakagawa 1998). If the users understand how the recognition system adapts itself to new writing styles, they can take the control of the adaptation process and so explicitly train the system. Naturally, if the recognition system makes no mistakes, there is no need it to be understandable for its users. However, this is hardly ever the case.

2.10 Summary: on-line handwriting recognition today

These days, on-line handwriting recognition is very commonly used as a textual input method in palm-sized PDA devices. There exist several on-line handwriting recognition softwares for that purpose, for example PenReader by Paragon Software (PenReader 2002), CalliGrapher, the latest version of which is called Transcriber, by ParaGraph (CalliGrapher 2002), Jot by Communication Intelligence Corporation (Jot 2002), smARTwriter by Advanced Recognition Technologies (smARTwriter 2002), MyScript by Vision Objects (MyScript 2002), TealScript by TealPoint Software (TealScript 2002), Decuma Latin by Decuma (Decuma Latin 2002), and Graffiti by Palm (Graffiti 2002). Of these mentioned recognition systems, Graffiti requires the most constrained style of writing as it has only one unistroke shape for each character class. The other systems have several alternative prototypes of more natural writing styles for each character class. CalliGrapher, smARTwriter, and the latest version of MyScript can recognize also mixed and cursive writing styles. MyScript, TealScript, Decuma Latin, CalliGrapher, smARTwriter, and JOT have special modes in which the recognizer can be customized. In CalliGrapher, users can inactivate prototypes or state whether they are used often or rarely. In JOT, some of the character classes have prototypes of exactly the same shape and the users can specify how these shapes should be classified. MyScript, TealScript, and Decuma Latin softwares have special dialogs in which the users can examine the recognition system's prototype set and delete or replace prototypes by their preferred writing styles. The users of smARTwriter can train the system to better recognize their personal writing styles by writing a certain text prompted by the system. Due to the commercial nature of these systems, there are not scientific articles publicly available in which the recognition performances of these systems would be evaluated and compared with each other by using the same data, or, in which the applied recognition methodologies would be described in detail.

The current trend in the research of on-line handwriting recognition seems to be the development of writer-independent large vocabulary systems for natural cursive or mixed style handwriting. The emphasis of the research is on statistical and neural methods, especially on HMMs and TDNNs. These recognition methods are used together with linguistic modeling techniques as the mere pen trace information in cursive handwriting is often too ambiguous for the successful recognition of all the characters. Lately, adaptive recognition methods have been gaining more and more interest. Some recent recognition results reported in the literature for isolated Latin characters and words of cursive or mixed handwriting style are given in Tables 1 and 2. In case of isolated characters, the reported recognition performances of some of the systems are very high and are comparable to the error rates reported for humans. Recognition systems described in (Bengio et al. 1995; Guyon et al. 1992; Matíc et al. 1993; Prevost and Milgram 1997) seem especially promising. The error rates reported for cursive or mixed-style words are in most cases too high for any practical application. However, the systems reported in (Bengio et al. 1995; Jaeger et al. 2001) seem to perform very well with multiple writers and large lexicons.

Table 1: Recognition results reported in the literature for isolated Latin characters. Results are averaged over different subjects. Abbreviations WD and WI come from writer-dependent and writer-independent, respectively. Notation WI→WD means that an initial writer-independent recognition system was adapted into a writer-dependent one.

Article	*Methods*	*Classes*	*Comments*	*Results*
(Bengio et al. 1995)	CNN, HMM	0-9, a-Z, 33 symbols	WI	error rates 1.4% (0-9), 4.2% (a-z), 2.9% (A-Z), 4.3% (symbols)
(Biem 2001)	HMM	0-9, a-Z, 30 symbols	WI	error rate 8.3%, 17 subjects
(Bontempi and Marcelli 1994)	chain codes	0-9, a-Z	WI→WD	error rate 1.9%→0.8%, rejection rate 14.7%→6.5%, 10 subjects
(Chan and Yeung 1999)	chain codes, decision trees	0-9, a-Z	WD?	error rates 1.4% (0-9), 1.5% (A-Z), 2.6% (a-z), 1.9% (62 classes)
(Connell and Jain 2001)	HMM	a-z	WI→WD	error rate 23.8%→11%, 11 subjects
(Fujisaki et al. 1991)	DTW	0-9, a-Z, 20 symbols	WD	error rates 2.8% (0-9), 3.7% (A-Z), 9.3% (82 classes), 12 subjects
(Guyon et al. 1992)	TDNN, *k*-NN	0-9, A-Z, 7 symbols	WI→WD	error rate 2.8% (no symbols) → 1-2%, 7 subjects
(Hu et al. 2000)	HMM	0-9, a-Z	WI	error rates 3.2% (0-9), 6.4% (A-Z), 14.1% (a-z)
(Li and Yeung 1997)	DTW, chain codes	0-9, a-Z	WI, constr. writing style	error rate 7.9%, rejection rate 1.1%, 15 subjects
(Matíc et al. 1993)	TDNN, SVM	0-9, A-Z, 7 symbols	WI→WD	error rate 2.8% (no symbols) → 2.5%, 7 subjects
(Mozayyani et al. 1998)	ST-MLP	a-z	WD, constr. writing style	error rate ∼1%, 15 subjects
(Nouboud and Plamondon 1991)	chain codes	0-9, A-Z, 23 symbols	WD	error rate ∼4%, 15 subjects
(Prevost and Milgram 1997)	DTW	0-9, A-Z	WD?, on-line & off-line features	error rates 1.4% (0-9), 2.9% (A-Z)
(Scattolin 1995)	weighted DTW	0-9	WD	error rate 0.5%, rejection rate 10.9%, 33 subjects
(Schenkel et al. 1995)	TDNN, HMM	A-Z	WI	error rate 1.8%/11.4% with/without lexicon, 25 subjects
(Vuori et al. 2002)	DTW	a-z,å,ä,ö, 0-9	WI→WD	error rate ∼15%→1.3%, 24 subjects
(Wilfong et al. 1996)	curve matching	A-Z	WD	error rate 2%, 8 subjects
(Yaeger et al. 1998)	MLP	0-9, a-Z, 33 symbols	WI→WD	error rate 21.3%→ 7.2%, 3 subjects
(Yuen 1996)	chain codes	a-z	WD	error rate 10%, 1 subject

Table 2: Recognition results reported in the literature for cursive or mixed handwriting styles. Results are averaged over different subjects. Abbreviations WD and WI come from writer-dependent and writer-independent, respectively.

Article	*Methods*	*Lexicon*	*Comments*	*Results*
(Bengio et al. 1995)	CNN, HMM	25 461 words	WI	error rate 3.2%
(Brakensiek et al. 2001)	HMM	2 200 words	WI→WD	error rate 13.7% → 8.4% (supervised) or 9.6% (self-supervised), 21 subjects
(Connell and Jain 2001)	HMM	483 words	WI→WD	error rate 29.2%→26.5%, 8 subjects
(Duneau and Dorizzi 1994)	chain codes	10 000 words	WI→WD	error rate 19% → <2%, 1 subject
(Hu et al. 2000)	HMM	500, 1 000, 2 000, 5 000, 10 000, or 20 000 words	WI	error rate 8.2%, 9.5%, 12.8%, 16.8%, 20.2%, 23.7% (sorted by the size of the lexicon), 100 subjects
(Jaeger et al. 2001)	MS-TDNN	5 000, 20 000, or 50 000 words	WI	error rate 4.0% 6.6%, 8.8% (sorted by the size of the lexicon)
(Morasso et al. 1995)	SOC	409, 1 893, or 7 480 words	WD	error rate 26%, 32%, 41% (sorted by the size of the lexicon), 2 subjects
(Nathan et al. 1996)	HMM	22 000 words	WI→WD	error rate 32.8%→12.3%, 7 subjects
(Pavlidis et al. 1997)	curve matching	100 words, 7 gestures	WD	error rate 8.5%, 12 subjects
(Qian and Truemper 1998)	chain code	10 000 words	WI→WD	error rate 34.5%→18.7%, 4 subjects
(Schomaker et al. 1993)	SOM	5 000-7 000 words	WI→WD	error rate 65%→37%, 16 subjects
(Senior and Nathan 1997)	HMM	22 000	WI→WD	error rate 28.7% → 26.4% (supervised) or 25.4% (self-supervised), 12 subjects
(Wilfong et al. 1996)	curve matching	32 words	WD	error rate 6.5%, 19 subjects

3 DESCRIPTION OF THE DEVELOPED ADAPTIVE ON-LINE CHARACTER RECOGNITION SYSTEM

The main goal of this thesis work was to develop a practical on-line recognition system for handwritten characters. The system ought to be able to handle multiple writing styles in the beginning of its use and to quickly adapt to new writing styles and so improve its recognition performance during normal use. Prototype-based recognition systems are well suited for adaptation and, most importantly, they can learn incrementally new writing styles on the basis of only a single character sample. If the prototype set is cleverly formed, i.e. there are no redundant prototypes but every prototype represents a distinct writing style, the computational complexity of a prototype-based recognition system is of the same degree than the computational complexity of any recognition system that is based on allograph models. These were the main reasons for choosing a prototype-based approach in this thesis work. For usability reasons, the comparison of the input characters and prototypes and the decision making process should be understandable to the users. This is true for the DTW matching algorithm and the k-NN decision rule. In addition, the recognition results reported for them in the previous literature have been most promising. These are the reasons why they have been applied also in this thesis work. The various components of the developed recognition system and the experiments performed with them have already been reported in the eight publications included in this thesis, namely in articles (Vuori and Oja 2000; Vuori et al. 2000a; Vuori et al. 2000b; Vuori et al. 2001a; Vuori et al. 2001b; Vuori et al. 2002; Vuori and Laaksonen 2002; Vuori 2002). Thus, this chapter gives just a general overview of the recognition system, describes the applied methods briefly, and reports only the main results of the experiments.

The components of the adaptive character recognition system are shown and their interactions are illustrated in Figure 7. The information flow through the system begins at the data collection where subjects write isolated characters with a special input equipment to be described later. The input characters are written either on a special tablet or on the pressure sensitive display of a PDA device. Next, the input characters are preprocessed and normalized. After that, they are either stored in the database for later use or they are directed for the recognition to the DTW-matcher. The DTW-matcher compares the input characters with all the labeled character prototypes and directs the matching results to the classification and adaptation units. The k-NN classifier sends the recognition results to the user interface. The adaptation unit modifies the prototype set of the k-NN classifier on the basis of the user's reactions to the recognition results, the input characters, and the usages and performances of the prototypes. The prototypes can be selected by a human expert or by a DTW-based clustering algorithm from a database containing character samples from several subjects. The recognition system also includes a DTW- and prototype-based method for representing personal writing styles by vectors and a SOM-based method for their visualization and clustering analysis.

The following sections are organized as follows: section 3.1 describes the collection and contents of the databases used in the experiments; section 3.2 introduces the preprocessing and normalization methods developed during the thesis project; section 3.3 outlines the various DTW-based dissimilarity measures defined for on-line characters and describes how the dissimilarity measures of infinite range can be transformed into similarity measures of finite range; section 3.4 introduces the clustering algorithms and indices developed for automatic prototype selection; section 3.5 reintroduces the k-NN rule and its mathematical foundation and describes some speed-up methods for the classification process; section 3.6

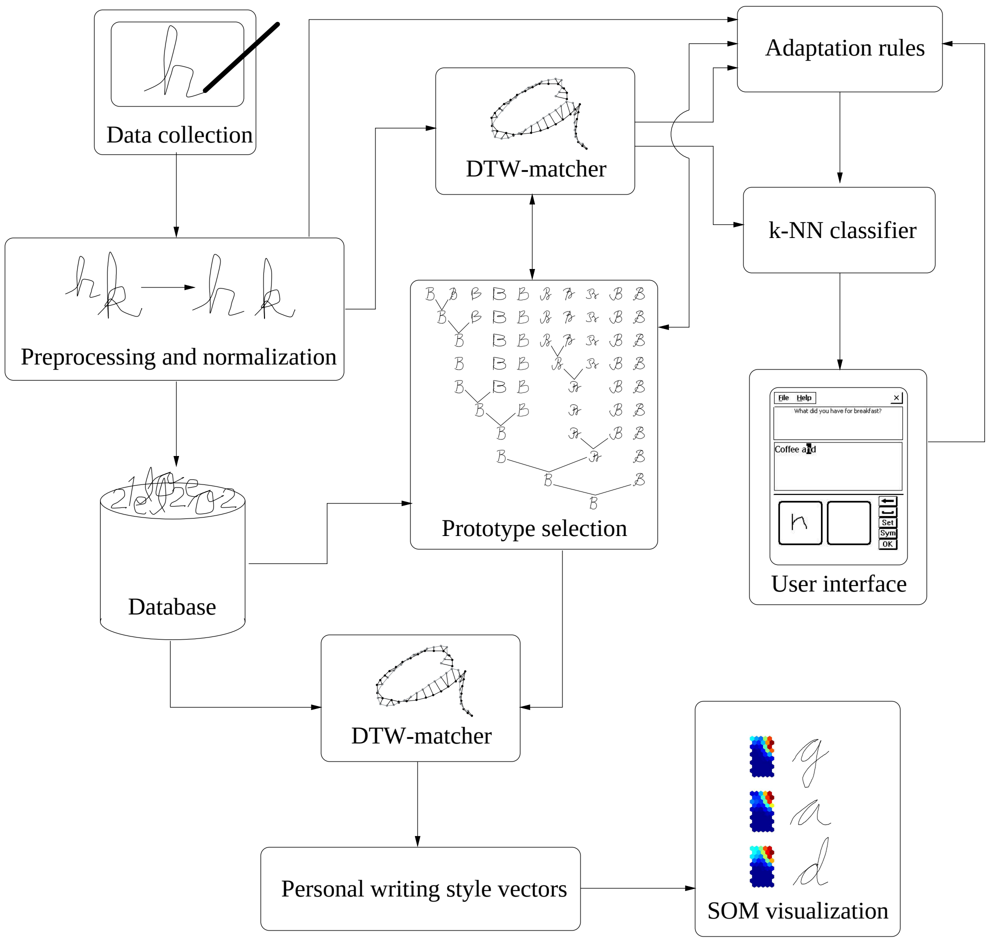

Figure 7: General overview of the developed recognition system. The arrows represent interactions between the various components of the system.

describes the various adaptation strategies of the k-NN classifier; section 3.7 introduces the formation and SOM-based analysis of personal writing style vectors.

3.1 Data collection and databases

The first experiments were performed with databases collected in the Laboratory of Computer and Information Science at Helsinki University of Technology, Finland. These databases are thus called local Databases 1–4. The latest experiments have been carried out by using two large public databases.

The character samples of the local databases were collected with pressure sensitive Wacom ArtPad II tablet attached to a Silicon Graphics (SGI) Indy workstation (Database 1) or SGI Octane workstation (Databases 2–4). The resolution of the tablet is 100 lines per millimeter and the sampling rate is at maximum 205 data points per second. The loci of the pen point movements consists of the x- and y-coordinates, the pen's pressure against the writing surface, and recording time. All data were saved in UNIPEN format (Guyon

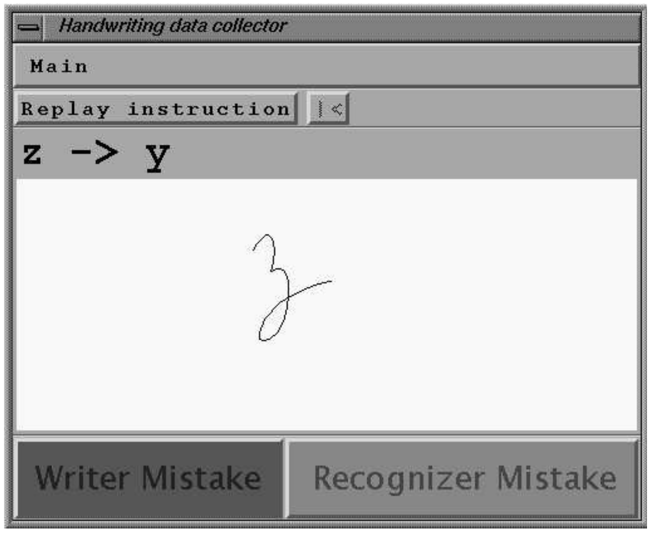

Figure 8: The user interface of the data collection program used in the collection of local Databases 2–4.

et al. 1994). The essential details of the local databases are summarized in Table 3.

The first local database (Database 1) consists of characters which were written without any visual feedback: the trace of the pen was shown neither on the tablet nor on the screen. Therefore, the pressure level thresholding the pen movements into either pen-up or pen-down movements was individually set for each writer. Some of the characters were written in alphabetical order, but most of them according to a dictation of a short story. The characters of Database 1 are upper and lower case Latin characters, upper and lower case versions of the three diacritical characters used in the Finnish language, digits, and some special characters. The special characters have not been used in the thesis project. Database 1 consists of approximately 10 400 character samples written by 22 subjects. The distribution of the character samples in respect to the characters classes is uneven and roughly the same as in the Finnish language.

The rest of the local databases (Databases 2–4) were collected with a program which showed the pen trace on the screen and recognized the characters on-line. The minimum writing pressure for showing the trace of the pen on the screen and detecting pen down movements was the same for all writers. All the samples were checked manually and those which were reported as writer's mistakes or clearly incorrect were abandoned. The user interface of the data collection program is illustrated in Figure 8. Database 2 was collected with a very poor recognizer. The database consists of approximately 13 200 character samples with nearly even distribution. The first five writers wrote 2 040 characters (30 samples per a class) but the amount of work was found to be intolerable and so the total number of character samples was reduced down to 1 020 for the rest of the writers. Database 3 consists of Database 2 and character samples written by eight additional writers. For them, a bit better adaptive recognizer was used. The total number of character samples in the Database 3 is approximately 21 200. The adaptive recognizer was further improved for the collection of Database 4 which consists of approximately 8 100 characters written by eight new writers. The total number of subjects who participated in the collection of

Table 3: Summary of the local databases.

Database	*Subjects*	*Left-handed*	*Females*	*Items*	*Classes*
Database 1	22	1	1	∼ 10 400	a–z, A–Z, å, ä, ö, Å, Ä, Ö, 0–9, and special characters
Database 2	8	2	5	∼ 13 200	a–z, A–Z, å, ä, ö, Å, Ä, Ö, 0–9
Database 3	16	2	5	∼ 21 200	a–z, A–Z, å, ä, ö, Å, Ä, Ö, 0–9
Database 4	8	0	5	∼ 8 100	a–z, A–Z, å, ä, ö, Å, Ä, Ö, 0–9

the local databases is 45.

The two public databases are called IRONOFF (Viard-Gaudin et al. 1999) and UNIPEN train_r01_v07 (Guyon et al. 1994). Only the isolated digits and upper and lower case letters were used in the experiments. The two databases were combined into one, all the character samples were checked manually and obviously erroneous ones were removed. The removed samples were segmented or labeled incorrectly, contained wild points or small extra strokes, or were somehow malformed or written in a very unnatural way. Some typical examples of bad handwriting samples which were cleaned from the public databases are illustrated in Figure 9. In total, 3 174 erroneous samples were found. The total number of samples in the cleaned database is 130 831. These samples have been written by 728 subjects. The subjects were of various ages and from several countries and both handedness groups were represented. The character samples of the public databases have been collected with pressure-sensitive displays or tablets which are able to record the x- and y-coordinates of a moving pen point. As the public databases had several contributors, the character samples are of significantly varying quality.

3.2 Preprocessing and normalization methods

During the thesis project, the following preprocessing methods have been developed and experimented with: *DuplicatePoints*, *SpuriousPoints(f)*, *Decimate(n)*, *Interpolate(n)*, *EvenlySpacedPoints(d)*, *NPointsPerStroke(n)*, and *ExtremePoints(d)*. *DuplicatePoints* is a nonlinear down-sampling method which removes data points whose coordinate values are exactly the same as those of the preceding data point. *SpuriousPoints* is a nonlinear down-sampling method which removes spurious data points caused by hardware problems of the data collection equipment, for its detailed description, see (Vuori et al. 2000a, included publication 4). *Decimate* is a linear down-sampling method which keeps only every nth data point. *Interpolate* method works in the opposite way: it interpolates n new spatially equidistant points between the successive original data points. *EvenlySpacedPoints* is a resampling method which interpolates new spatially equidistant data points so that the Euclidean distances between successive data points are d. *NPointsPerStroke* is a linear resampling method which interpolates n equidistant data points per each stroke. Therefore, the new data points are more sparsely distributed in long strokes than in short ones. *ExtremePoints* is a nonlinear down-sampling method which keeps only those data points

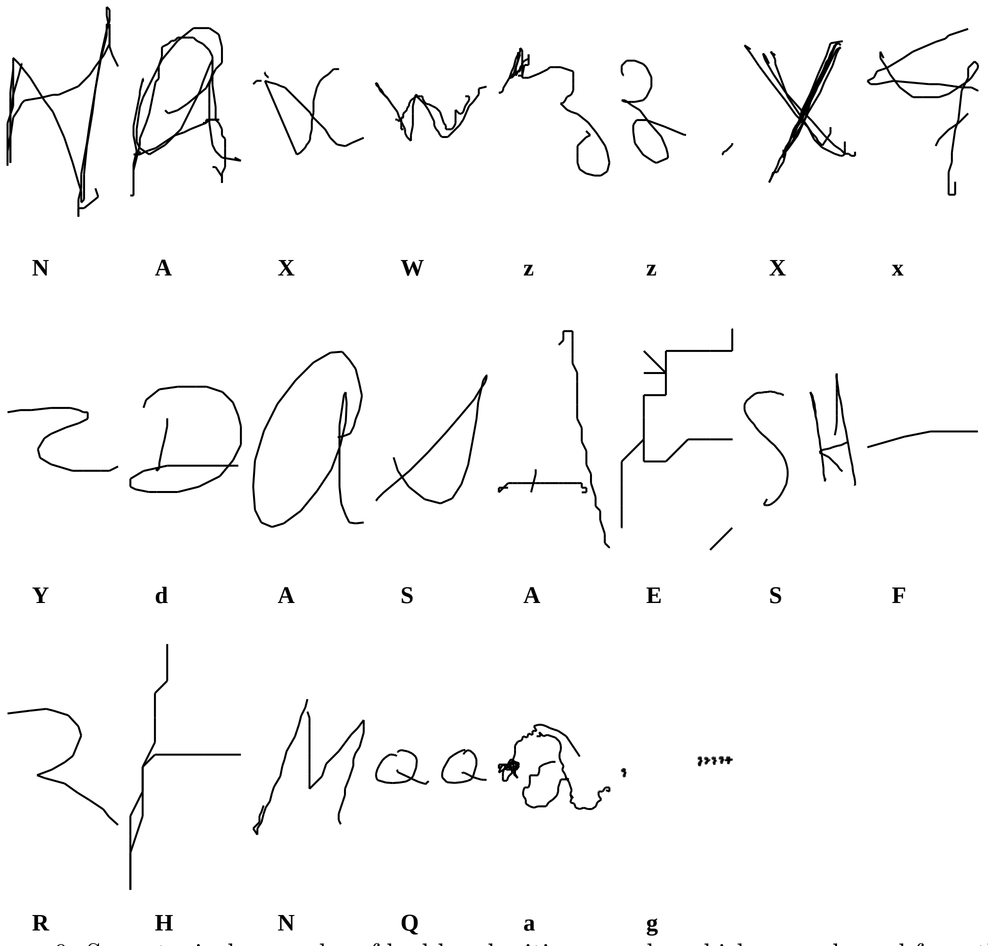

Figure 9: Some typical examples of bad handwriting samples which were cleaned from the public databases.

in which the horizontal or vertical component of pen point velocity changes sign, becomes zero, or ceases to be zero. It also requires that there is a minimum spatial distance between successive retained data points.

Most of the experiments with the different preprocessing methods were performed only with the local databases containing character samples of high quality in respect to sampling frequency, resolution and spurious data points. In all the experiments, character samples were first preprocessed with the *DuplicatePoints* method. This preprocessing had either only a slightly beneficial or no effect at all on the performance of the recognition system. The *SpuriousPoints* method was only applied in the palm-top implementation (Vuori et al. 2000a, included publication 4), because for some unknown reason the data acquisition in that environment produces clearly erroneous data points. However, they are rare and easily removable by using this nonlinear filtering.

Preprocessing methods *Decimate*, *Interpolate*, and *EvenlySpacedPoints* were compared to each other in the series of experiments reported in (Vuori et al. 2001a, included publication 1). In that work, various combinations of preprocessing and normalization methods and dissimilarity measures were experimented with and the best average recognition rates were obtained either by using the *DuplicatePoints* and *Decimate* methods together or the former method alone. This is an interesting result, as it indicates that the implicit dynamic information which is contained in temporally equidistant data points is beneficial. It also shows that the sampling frequency used in the collection of the local databases was sufficiently high.

Preprocessing methods *NPointsPerStroke* and *EvenlySpacedPoints* were compared to each other in an unpublished experiment performed while designing an assignment for the Pattern Recognition course that the author lectured at the Helsinki University of Technology. A writer-independent prototype set was formed from the digit samples of Database 1 and the digit samples of Databases 3 and 4 were used as a test set. According to this small-scale experiment, the *EvenlySpacedPoints* method yielded slightly better recognition results, probably because the *NPointsPerStroke* method emphasizes too much small strokes and dots, positions and shapes of which may vary considerably. Therefore, the only advantage of the *NPointsPerStroke* methods seems to be the fact that it produces fixed-length time series representations for the character samples.

In the work reported in (Vuori et al. 2001b, included publication 5), the *Decimate* and *ExtremePoints* preprocessing methods were compared with each other. In the performed experiments, the error rates obtained by using the *ExtremePoints* method were less dependent on the value of the down-sampling parameter and there was less variation between different writers. The both preprocessing methods were efficient in reducing the average recognition time. In the case of the *Decimate* method, the recognition accuracy clearly deteriorated and its variation between the writers increased if the value of the decimation parameter n was increased. When the *ExtremePoints* method was used as a preprocessing method, the recognition accuracy was more or less the same with the different values of the distance parameter d. If the parameter values were set so that the recognition time was approximately the same with the two preprocessing methods, *ExtremePoints*(d) was better than *Decimate*(n) in respect to the total error rate, the average error rate calculated for the personal error rates of different writers and their standard deviation. Figure 10 illustrates the effects of the *ExtremePoints*(d) and *Decimate*(n) methods. This shows that the latter preprocessing method keeps the corners in their original locations better than the former method when the number of preserved data points is approximately the same in the preprocessed character samples.

During this thesis project, the following normalization algorithms have been developed: *MassCenter*, *BoundingBoxCenter*, and *MinMaxScaling*. *MassCenter* moves the mass center of the data points of the character sample into the origin of the coordinate system. *BoundingBoxCenter* does the same to the mass center of the bounding box. *MinMaxScaling* is a linear scaling which preserves the aspect ratios of the character samples and sets the longer sides of the bounding boxes to a constant value. This size normalization treats upper and lower case letters equally. Thus, problems will arise if the lower and upper case versions of the letters differ only by their size. If preprocessing and normalization methods are used together, the size normalization has to be performed first, then the preprocessing operations, and finally the translation normalization.

According to the experiments with a DTW-based character recognition system reported in (Vuori et al. 2001a, included publication 1), *MassCenter* produces better recognition

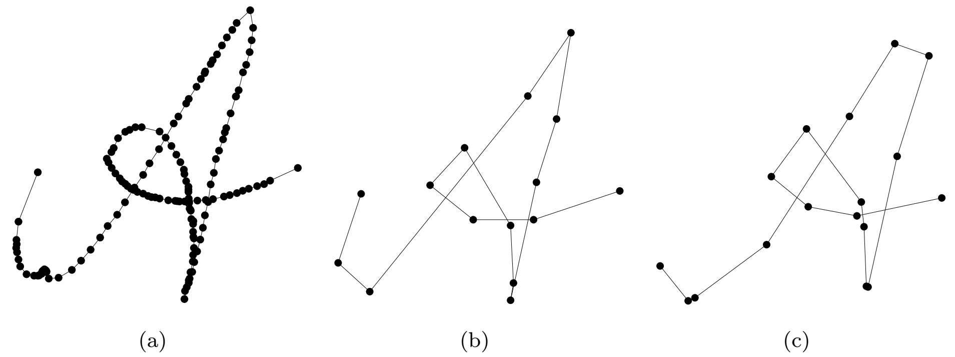

Figure 10: (a) Character sample prior to preprocessing. The same character sample preprocessed with (b) *Decimate(*8*)* or (c) *ExtremePoints(*20*)* method.

results than *BoundingBoxCenter*. Evidently, *MassCenter* is more robust against elongated strokes and dot positioning and therefore finds the centers of the main bodies of the character samples more robustly than *BoundingBoxCenter*. In all the experiments, significantly better result were obtained with *MinMaxScaling* method than with no size normalization at all. Even though the most of the recognition errors, 34–44%, were confusions between the upper and lower case versions of the same letter when the letters and digits were recognized together.

3.3 Measuring dissimilarity with Dynamic Time Warping

The DTW algorithm is a nonlinear matching method for time series. It was developed originally for speech recognition in the beginning of 1970's (Sankoff and Kruskal 1983) and was introduced for the recognition of handwriting in the early 1980's (Kurtzberg and Tappert 1981). DTW-based matching methods have been used successfully also in on-line signature verification (Martens and Claesen 1997; Sato and Kogure 1982; Wirtz 1998). The DTW algorithm can be used for the comparison of all kinds of time series. It finds the optimal time warping, i.e. the point-to-point correspondence between two time series, which minimizes the sum of point-wise matching costs. The DTW-based matching methods are insensitive to the distortions caused by the varying writing speed, for example, to the variations in the total number and distribution of data points in a character sample. However, these methods are not sensitive only to the variations in the shapes of the strokes but also to the variations in the drawing order and direction of the strokes.

The DTW-based matching methods are well suited for on-line handwriting recognition due to their transparency to the users. Let us assume that on-line handwriting is represented by a time series of the coordinate points of the moving pen point and the DTW matchings of the input characters and the prototypes are illustrated as in Figure 11. It is very easy to see how the shapes of the input characters should be modified to better match the correct prototypes.

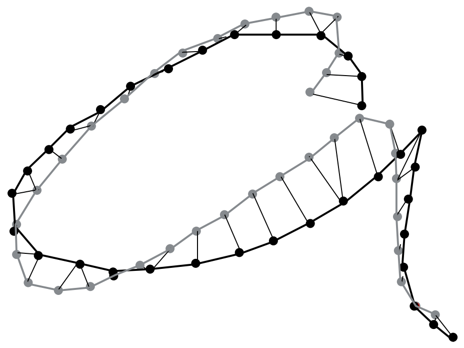

Figure 11: The point-to-point correspondence of two characters established with the DTW algorithm.

3.3.1 DTW-based dissimilarity measures

Six dissimilarity measures based on the DTW algorithm and called *Point-to-point*, *Normalized point-to-point*, *Point-to-line*, *Normalized point-to-line*, *Kind-of-area* and *Simple-area* distances were developed during this thesis project. All the six dissimilarity measures are defined on stroke basis. If the number of strokes in two characters are different, the dissimilarity measure between the characters is defined to be infinite. Otherwise, it is the sum of the dissimilarity measures between the corresponding strokes. The main difference between the dissimilarity measures is in the costs of matching two data points. All the matching costs and continuity and boundary constrains of the DTW algorithm are symmetric. Therefore, the dissimilarity measures are symmetric. The continuity constraints require that the data points are matched in the same order as they have been produced. In addition, all data points are matched at least once and several data points can be matched against one. According to the boundary constraints, the first and last data points of the strokes are matched against each other or to lines interpolated between the first, or last, two data points.

Point-to-point distance uses the squared Euclidean distance between the data points as the matching cost. There are five variants of it. In the basic version, data points are represented only by their x- and y-coordinates. In the first two variants, also the magnitude or x- and y-components of the pen point velocity are used as data point features. In the first speed-up variant, constraints for the nonlinearity of the optimal time warping path are stricter than in the basic version. The other speed-up variant compares the lengths of the strokes first and sets the dissimilarity between the characters samples infinite if they differ too much. The two speed-up variants of *Point-to-point* distance are described in more detail in section 3.5.2.

In the case of *Point-to-line* distance, the data points are matched to lines interpolated between the data points. The matching cost is the minimum squared Euclidean distance between the line and the point. All the data points except the first and last ones of one of the strokes are matched. *Normalized point-to-point* and *Normalized point-to-line* distances are otherwise similar to *Point-to-point* and *Point-to-line* distances, respectively, but the sums of the matching costs are divided stroke-wise by the number of matchings, i.e. the length of the warping path. In some applications, these distances are further normalized by dividing them by the number of strokes.

Kind-of-area(n,m) distance also matches data points against data points. However, the matching cost is a product of two terms. The first term is the nth power of the Euclidean distance between the data points. The second term is the mth power of the sum of the Euclidean distances from the matched data points to their neighboring data points in the time series. *Kind-of-area*(2,0) and *Point-to-point* distances are equivalent. As illustrated in Figure 11, each matching of data points, except the first one, defines a new triangle or quadrangle. *Simple-area* distance uses the areas of these polygons as the matching costs.

The main difference between the dissimilarity measures is their sensitivity to the dynamic variations of handwriting, such as speed and acceleration, and the timing of the sampling process. These factors affect *Point-to-point* distance most. *Point-to-line* distance is not so sensitive to the phase of sampling due to the interpolations. In case of *Normalized point-to-point* and *point-to-line* distances, the effects of dynamic variations are reduced by the normalization. The sensitivity of *Kind-of-area*(n,m) distance to the dynamic variations can be controlled with the parameter m. Provided that the sampling frequency is high, the area between two strokes does not depend much on the dynamics of the writing. Therefore, *Simple-area* distance depends the least on the dynamic properties of the strokes.

3.3.2 Experimental results for the dissimilarity measures

The basic version of *Point-to-point*, *Normalized point-to-point*, *Point-to-line*, *Normalized point-to-line*, *Kind-of-area* and *Simple-area* were compared to each other using different preprocessing and normalization in the experiments reported in (Vuori et al. 2001a, included publication 1). The basic version of *Point-to-point* performed best on average. *Normalized point-to-point* performed nearly as well, its averaged error rate was only 1% higher. *Point-to-line* and *Normalized point-to-line* both produced the averaged error rate 7% higher than the averaged error rate of the basic *Point-to-point.* With *Kind-of-area* and *Simple-area* the averaged error rate was 15% and 87% higher, respectively.

The basic version and the first two variants of the *Point-to-point* dissimilarity measure were compared to each other in an unpublished recognition experiment in which a writer-independent prototype set was formed from the samples of Database 1 and the lower case letters and digits of Databases 3 and 4 were used as a test set. The data points were represented by the x- and y-coordinates of the pen point position and the magnitude or the x- and y-components of the pen point velocity. The lowest averaged error rate was obtained when the positional features were weighted by a factor of 1 and the velocity components by a factor of 0.25 in the squared Euclidean distances between the matched data points. This optimal averaged error rate was only 1% lower than the averaged error rate obtained with the basic version of *Point-to-point.*

The basic version and the two speed-up variations of the *Point-to-point* dissimilarity measure were compared to each other in the experiments reported in (Vuori et al. 2000a, included publication 4). According to the experimental results, the first speed-up variation was able to decrease the recognition time nearly by 30% while the error rate remained practically the same and less than 1% of the character samples got rejected. The second speed-up variation was able to decrease the recognition time by 11% while the error rate rose by less than 1% and the rejection rate was less than 0.01%.

3.3.3 Transforming dissimilarities into similarities

The DTW-based dissimilarity measures discussed in the previous section have ranges from zero to infinity and their values depend on the number of strokes. In addition, the unnormalized versions of *Point-to-point* and *Point-to-line* distances depend on the number of data points in the strokes. However, in the most cases, it is not interesting which are the most dissimilar character samples but which are the most similar ones. Therefore, a better resolution should be used for the small dissimilarity measures than for the large ones. The dissimilarity measures calculated for all kinds of character pairs can be made comparable with each other by simply normalizing them by the number point-wise matching costs and by the number of strokes. For example, in the studies reported in (Vuori and Oja 2000, included publication 7) and (Vuori 2002, included publication 8), personal writing styles were characterized by vectors, the components of which indicated how much the subjects' character samples resembled certain prototypical writing styles for isolated characters. In these works, the similarities between the characters were obviously much more important than the dissimilarities. In addition, it was important that all the components of the writing style vectors had the same range. Otherwise, the components with larger ranges would have had a dominating effect on the results.

For the aforementioned reasons, two transformations which turn *Normalized point-to-point* distances of infinite range into similarity measures of finite range have been developed. The similarity measures obtained with the two transformations have the following properties. In both transformations, the *Normalized point-to-point* distance between the character samples is normalized by the number of strokes and thus the similarity measures are independent of the number of strokes and data points. Both the transformations result similarity measures with ranges from zero to one. The parameters of the first transformation were selected so that for the most of the character samples the similarity measure for the best-matching correct prototype was close to one whereas the similarity values evaluated for the rest of the prototypes were close to zero. The parameter of the second transformation was set so that that the distribution of the similarity measures between character samples and their best-matching correct prototypes was approximately even. The former transformation produces more or less binary-valued similarity measures which just tell whether two characters are considered similar or not. The similarity measures obtained with the latter transformation have better resolution and really describe how similar the two characters are.

3.4 Creation of a prototype set

In the case of prototype-based recognition systems, the better the different writing styles are covered and represented by the prototypes, the higher accuracies are achieved. However, the prototype set should not contain too many redundant prototypes as the recognition time depends linearly on its size. The prototype set can be formed automatically by applying a clustering algorithm to a large training database containing character samples written be several subjects. In such an approach, the clustering algorithm divides the training database into groups of similar character samples, or clusters, and selects a prototype for each cluster which represents all the character samples in that cluster. Clustering of the training samples is beneficial with other recognition methods too. For example, better recognition results can be expected if each writing style is modeled with its own HMM instead of using one model per class and for several, sometimes significantly different, writing styles (Connell 2000).

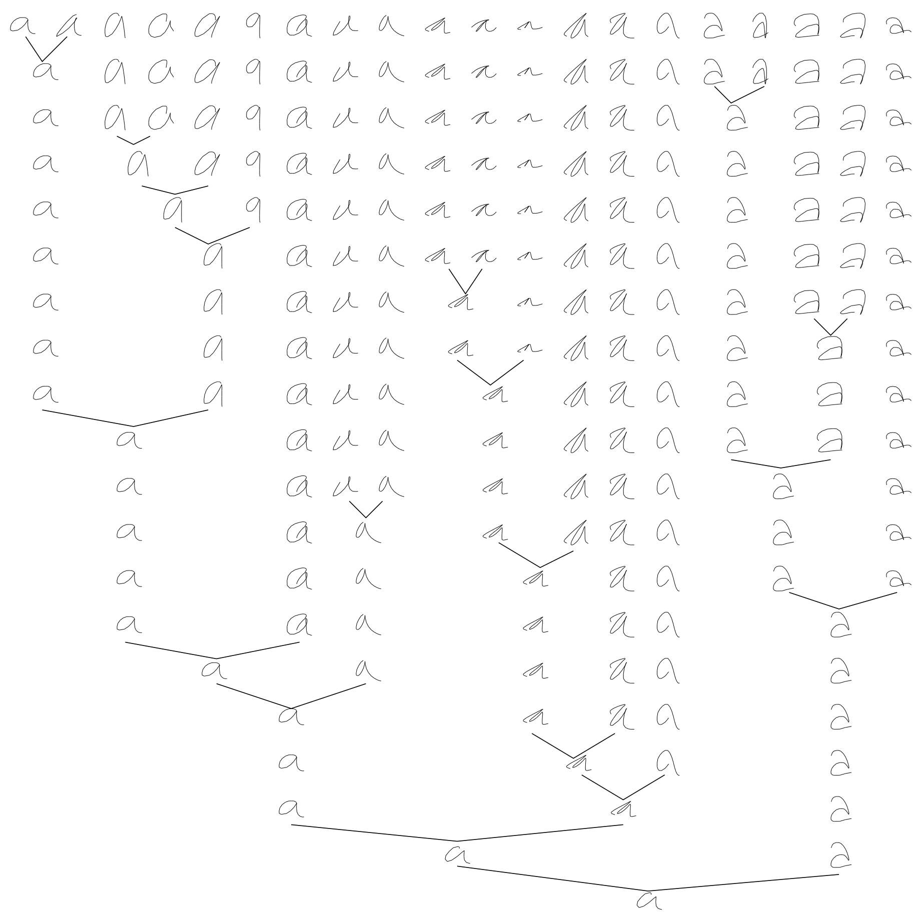

Figure 12: A visualization of a hierarchical clustering found by the *TreeClust* algorithm. Each row corresponds to one clustering solution and the lines show which clusters are merged to get the next solution. The shown character samples are cluster prototypes.

Often, one of the major obstacles in finding a good clustering or a prototype set is the uneven representation of different writing styles in the database: there can be hundreds of examples of some common styles but only a few samples of the rare styles. Another difficult problem is the determination of the number of clusters. The number of clusters can be selected automatically on the basis of some clustering index which measures the similarities and dissimilarities of the character samples within and between the clusters. Alternatively, the number of clusters can be decided by a human expert. Figure 12 shows how the progress of an agglomerative clustering can be visualized with a tree-like structure. From such a visualization, it is easy to see whether the number of clusters is too high or low. In the former case, the prototypes of different clusters are too similar to each other. In the latter case, there are no prototypes for the writing styles which could be observed in the earlier stages of the clustering algorithm.

During the thesis project, four hierarchical clustering algorithms for prototype selection and two clustering indices which can be used with the clustering algorithms for automatic determination of the number of prototypes have been developed. They all use the DTW algorithm to compare character samples. The clustering algorithms and indices, as well as the experimental results obtained with them are discussed briefly in the next three sections.

The topic of automatic prototype selection has been considered previously in (Anonymous 1989; Bontempi and Marcelli 1994; Connell and Jain 1998; Duneau and Dorizzi 1994; Li and Yeung 1997; Morasso et al. 1993; Nathan et al. 1993; Prevost and Milgram 2000; Schomaker 1993; Vuurpijl and Schomaker 1997a).

3.4.1 Clustering algorithms for prototype selection

The developed clustering algorithms are called *TailCutting*, *TreeClust*, *CMeans*, and *MinSwap*. *TailCutting* is a semiautomatic clustering algorithm, i.e. the number of clusters, or prototypes, must be predefined by the user. This algorithm starts from a situation in which all the samples belong to the same cluster. The number of clusters is increased iteratively until the predefined count is reached or the samples run out. The iterative algorithm starts by finding the centers of the clusters, i.e. the items which yield the minimum sum of dissimilarity measures between themselves and all other items belonging to the same cluster. Next, all the items are ordered into an increasing series according to their distance to the center item of their cluster. A new cluster is formed of the items furthest from their cluster centers. The number of items in the new cluster is determined by minimizing a splitting criterion function. Before the next cluster splitting, the cluster centers are iteratively recalculated and each item is assigned to the cluster center nearest to it. This is repeated until a stationary partition is reached. A detailed description of the algorithm can be found from (Laaksonen et al. 1999) and (Vuori et al. 2001a, included publication 1).

Clustering algorithms *TreeClust*, *CMeans* (Duda and Hart 1973), and *MinSwap* are agglomerative and hierarchical. There are two version of CMeans algorithm called *CMeans 1* and *CMeans 2*. With all these algorithms, clusters are represented by prototypes which are the samples having the minimum sum of distances to the other samples in the same cluster. *TreeClust*, *MinSwap*, and *CMeans 2* start form a situation in which all the samples are prototypes, i.e. form their own clusters, while in the beginning of the *CMeans 1* algorithm, only a random subset of the samples is selected to be the initial prototype set.

As the clustering algorithms proceed, the number of clusters is reduced by merging of clusters. In *TreeClust*-, *CMeans 1*-, and *CMeans 2* algorithms those two clusters whose prototypes are most similar to each other are merged into one. *MinSwap* algorithm tries several alternative mergings, first the clusters with the most similar prototype pair, then the clusters with the next similar pair, etc. A new prototype is selected among the samples which belong to the new cluster. After that, *MinSwap*, *CMeans 1*, and CMeans 2 reassign the samples into the clusters according to the closest prototypes and then reselect the prototypes. This is continued until a stable division is found. *MinSwap* performs the same procedure but also calculates how many of the samples are swapped out from the new cluster into the other clusters, or vice versa, and selects that merging which gives rise to the minimum number of these swappings.

3.4.2 Clustering indices for determining the number of prototypes

The clustering indices are called *DB* (Davies and Bouldin 1979) and *CH* (Calinski and Harabasz 1974). The *DB* and *CH* clustering indices have been modified from their original version so that the squared Euclidean distances between two samples or a sample and cluster prototype are replaced by DTW-based distances. In addition, the sample which has the minimum sum of distances to all the other samples is used as a mean. The *DB* index was selected to be used in the experiments of this thesis work because it is one of the most commonly used clustering indices.

The DTW-based *DB* index can be calculated as follows. First, evaluate DB_{ij}

$$DB_{ij} = \frac{S_i + S_j}{M_{ij}} \tag{1}$$

where M_{ij} is the distance between the prototypes of the ith and jth cluster and S_i is the average of the distances between the prototype P_i of the ith cluster and the n_i samples $\{C_i^j\}_{j=1}^{n_i}$ belonging to that cluster, or

$$S_i = \frac{1}{n_i} \sum_{j=1}^{n_i} d(P_i, C_i^j) \tag{2}$$

where $d(\cdot, \cdot)$ is the DTW-based distance. The *DB* index is then given by the following equation:

$$DB(k) = \frac{1}{k} \sum_{i=1}^{k} \max_{1 \le j \le k, j \ne i} DB_{ij} \tag{3}$$

where k is the number of clusters. A sensible number of clusters can be determined by minimizing the *DB* index.

The *CH* index was selected as it outperformed several other clustering measures, including the *DB* index, in a thorough comparison performed by Milligan and Cooper (1985). The numerator part of the *CH* index measures how much the cluster prototypes differ from the mean sample and the denominator part tells how much the samples differ from their cluster prototypes. Therefore, a reasonable number of clusters can be found by maximizing the *CH* index. The DTW-based *CH* index can be evaluated as follows:

$$CH(k) = \frac{\sum_{i=1}^{k} n_i d(P_i, M)/(k-1)}{\sum_{i=1}^{k} \sum_{j=1}^{n_i} d(P_i, C_i^j)/(n-k)}, \tag{4}$$

where $d(\cdot, \cdot)$ is the DTW-based distance, k is the number of clusters, n is the total number of samples, n_i is the number of samples in the ith cluster, P_i is the prototype of the ith cluster and M is the mean sample and C_i^j is the jth sample in the ith cluster.

3.4.3 Experimental results for the clustering algorithms and indices

Clustering algorithms *TailCutting* and *TreeClust* were compared with each other in two sets of experiments. In these experiments, the prototypes were selected among the character samples of local Database 1 by using either the *TailCutting* or *TreeClust* algorithm. The prototype sets were then evaluated and compared with each other by using them for the recognition of the character samples included in local Databases 3 and 4. In the first set of experiments, the two clustering algorithms were used for selecting seven prototypes per each lower and upper case letter and digit class. The error rates obtained by using prototypes selected by the *TailCutting* or *TreeClust* algorithm were 23.1% or 24.8%, respectively. In the next set of experiments, the number of prototypes per character class varied between 1 and 7. The *TailCutting* algorithm performed better also in this case as the error rates obtained by using prototypes selected by the *TailCutting* or *TreeClust* algorithm were

24.8% or 25%, respectively. However, these small differences between the error rates are quite insignificant in practice.

Clustering algorithms *TreeClust*, *MinSwap*, *CMeans 1*, and *CMeans 2* were compared with each other in the experiments reported in (Vuori and Laaksonen 2002, included publication 6). The main objective of those experiments was to find clearly distinct prototypes for each character class which would represent well all the writing styles present in the large public databases. Therefore, no attention was paid to how well the prototype set could capture differences between the classes, i.e. would perform in a classification task. Missing rare styles was considered to be a much worse problem than over-represented common styles. Instead of recognition rates, different clustering results were judged and compared with each other on the basis of the knowledge on the writing styles established during the manual examination and cleaning of the character database.

In these experiments the *DB* and *CH* clustering indices rarely agreed and a manual examination of the clustering results revealed that neither one of them did make much sense in practice. Therefore, the number of prototypes could not be decided automatically. The prototypes found with the different clustering algorithms for common writing styles were very similar with each other. However, the different clustering algorithms did miss different rare writing styles. *CMeans 1* could not find the rare styles at all unless they were selected as initial prototypes, which was not very probable as the common styles were heavily over-represented in the training database. *CMeans 2* performed little better as initially all the samples were prototypes. However, due to the reassignments of the samples into the clusters, the small clusters of high internal variance were often absorbed into the big clusters. There were no clear difference between *CMeans 2* and *MinSwap* – perhaps there should be some kind of normalization by the size of the new cluster for the number of swappings. *MinSwap* algorithm evidently favors merging small clusters. Of all the four algorithms *TreeClust* could best preserve the small clusters of rare styles in the merging process. However, assignment of the samples into the clusters is always final and the prototypes of common styles were not always the best representatives of all the samples of that style. Instead of selecting one general prototype, *TreeClust* algorithm tended to select a couple of more extreme samples of the same style. This problem can be alleviated by fine-tuning the final prototypes for example with the Learning Vector Quantization (LVQ) algorithm as in (Vuori and Oja 2000, included publication 7).

3.5 Classification

The adaptive character recognition system developed during this thesis project is based on the k-NN rule. This simple decision rule is based on distances between the unknown sample and the prototypes but it can also be interpreted as a statistical classification rule based on the estimates of the *a posteriori* probabilities of the classes. Section 3.5.1 gives the mathematical justification of that interpretation. Section 3.5.2 introduces the speed-up methods developed for the k-NN classification process.

3.5.1 The k Nearest Neighbors rule

The k Nearest Neighbors rule was first introduced by Fix and Hodges (1951) and further analyzed by Cover and Hart (1967). The k-NN rule is a data-driven approach and a completely parameter-free method as it does not rely on any assumptions on the underlying distributions of the classes. The classification is based only on the known classifications of the training samples which form the prototype set: an unknown sample is classified according to majority of the classes of its k nearest prototypes. The volumes of the regions spanned by the k nearest neighbors depend on how densely the prototypes are distributed around the unknown character sample. This is the main difference compared to other nonparametric statistical classification methods, for example the Parzen window method (Duda and Hart 1973), which estimate the class probabilities on the basis of the training samples observed in regions of fixed volume. The mathematical background of the k-NN rule is described next (Schalkoff 1992).

The k-NN rule is based on the following assumption: as the number of the training samples n increases to infinity, the number of training samples k_n which fall into a given region V_n also increases to infinity, or

$$\lim_{n\to\infty} k_n \to \infty, \tag{5}$$

so that the ratio k_n/n approaches the probability that a sample falls in that region. On the basis of that assumption, the probability distribution $p(x)$ of the samples can be estimated as follows:

$$\frac{P(\text{a sample falls in the neighborhood of } x \text{ with volume } V_n)}{V_n} = \frac{k_n/n}{V_n} \approx p(x). \tag{6}$$

According to the Bayes rule,

$$P(\omega_i|x) = \frac{P(\omega_i)P(x|\omega_i)}{P(x)} = \frac{P(x,\omega_i)}{P(x)} = \frac{P(x,\omega_i)}{\sum_{j=1}^{C} P(x,\omega_j)} \tag{7}$$

where C is the number of classes. The required probabilities $P(x,\omega_i)$ can be estimated by using (6) in the following way:

$$P(x,\omega_i) \approx \frac{k_i/n_v}{V} \tag{8}$$

where n_v is the total number of the training samples fallen in region V and k_i is the number of samples belonging to the class ω_i. By substituting (8) into (7), the following approximation of $P(\omega_i|x)$ will be obtained:

$$P(\omega_i|x) \approx \frac{(k_i/n)/V}{\sum_{j=1}^{C}(k_j/n)/V} = \frac{k_i}{k}. \tag{9}$$

As has been shown above, the k-NN rule corresponds to a classification carried out by choosing the most probable class according to the estimates of the *a posteriori* probabilities of the classes. Cover and Hart (1967) have shown that any other k-NN rule, $k \neq 1$, does not have an error probability lower than the 1-NN rule if every interclass distance is greater than any intraclass distance. If the classes are overlapping in the feature space and the training set is large, a k-NN rule with $k \neq 1$ performs better than the 1-NN rule as it gives more accurate estimates of the *a posteriori* probabilities of the classes. Cover and Hart (1967) have also proven that the error probability of the k-NN rule has both upper and lower limits. If R^* is the optimal error probability obtained with the Bayes rule and the true probability distributions of the classes and the size of the training set approaches infinity, the bounds of the error probability R of the k-NN rule are given by the following inequality:

$$R^* \leq R \leq R^*(2 - \frac{M}{M-1}R^*) \tag{10}$$

where M is the number of classes.

The drawbacks of the k-NN rule are its memory requirements and computational complexity which depend linearly on the size of the prototype set. Therefore, the prototype sets used in this work have been formed by using clustering algorithms, although, better recognition accuracies would have been obtained with the whole training database. The applied clustering algorithms select several prototypes for each character class but each prototype represent a different writing style. Thus there can be only one correct prototype among the nearest prototypes of a character sample. For this reason, the classifications have always been performed according to the 1-NN rule. It should be noted that if the prototype set contains only one prototype per each different writing style, the computational complexity and the memory requirements of the recognition system are of the same degree as they would be with any recognition system based on allograph models.

3.5.2 Speed-up methods for the k-NN classifier

There are other ways to reduce the computational complexity of the k-NN classification procedure besides limiting the size of the prototype set. During the thesis project, two approaches have been taken in order to speed-up the recognition process. In the first approach, the computational complexity of the individual matchings is reduced by posing stricter constraints for the DTW matching. In the second approach, the number of required complete DTW matchings is reduced. This is achieved by ordering the prototypes so that the DTW distances to the well-matching prototypes will be evaluated in the early stages of the search and a DTW matching is interrupted as soon as it becomes clear that the current prototype is not among the k best-matching ones.

During the thesis project, two additional constraints for the *Point-to-point* DTW matching have been introduced. The first constraint requires that the matched strokes have to be of somewhat similar length. This constraint was formulated as follows: the strokes cannot be matched if

$$(N_2 \geq \alpha N_1 + \beta) \quad \text{or} \quad (N_1 \geq \alpha N_2 + \beta), \tag{11}$$

where N_1 and N_2 are the lengths of the strokes. Pruning parameter $\alpha \geq 1$ determines how much N_1 and N_2 are allowed to differ relatively whereas pruning parameter $\beta \geq 0$ sets an upper limit for the absolute difference between N_1 and N_2. The other constraint regulates how much the DTW matching is allowed to differ from linear matching. The strictness of this constraint can be regulated with a single parameter $1 \geq c \geq 0$. This constraint was formulated as follows: the ith data point of stroke 1 and the jth data point of stroke 2 can be matched if

$$\frac{N_2}{N_1}i - cN_2 \leq j \leq \frac{N_2}{N_1}i + cN_2. \tag{12}$$

where N_1 and N_2 are again the lengths of strokes. If $c = 0$, linear matching is the only feasible solution. Figure 13 illustrates how this constraint reduces the number of feasible point-to-point matching between the strokes. According to the experimental result reported in (Vuori et al. 2000a, included publication 4), the additional constraints decreased the average recognition time of a single character significantly, see also section 3.3.2.

The prototypes can be ordered prior to full-scale DTW matching on the basis of the rough shape of their first stroke or on the basis of a fast preclassification. In the former ap-

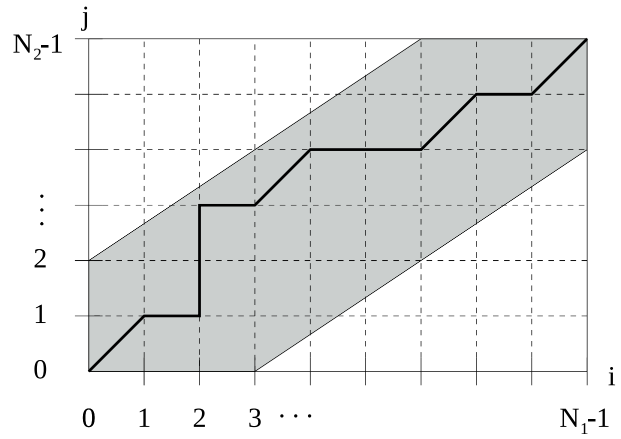

Figure 13: A DTW matching the nonlinearity of which has been constrained by equation (12) and parameter $c = 1/3$. All the feasible point-to-point matchings of the strokes are inside the shaded area. An example of a feasible solution is plotted with a bold line.

proach, the prototypes and unknown character samples are divided into sixteen categories determined by the quadrants of the coordinate plane in which the starting and ending points of their first stroke are located (Vuori et al. 2002, included publication 2). Each category is represented by four binary-valued bits so that if the starting or the ending point of the first stroke is moved to the neighboring quadrant, only one bit changes its value. The distance between two categories is the count of bit differences in their representations. The prototypes which have the smallest category distance to the unknown sample are matched first. According to an unpublished, small-scale experiment, roughly 70% of the best-matching prototypes belonged to the same category as the unknown character. In the latter approach, the preclassification is performed with heavily down-sampled characters and thus the DTW matchings can be performed very fast. The final classification is then performed with the N best-matching prototypes found in the preclassification. According to the experimental results reported in (Vuori et al. 2001b, included publication 5), the two-phased classification scheme speeds-up the recognition process significantly, by about 76%, without increasing the recognition error rate at all.

3.6 Adaptation strategies

A prototype-based recognition system can be adapted to new writing styles basically in three ways: by adding new prototypes, inactivating confusing prototypes, and reshaping existing ones. Several adaptation strategies which combine these three basic operations in different ways have been experimented in this thesis work. These experiments have been reported in included publications 1–4 (Vuori et al. 2000a; Vuori et al. 2000b; Vuori et al. 2001a; Vuori et al. 2002). Included publication 1 (Vuori et al. 2001a) introduces four adaptation strategies which are called *Add*, *Lvq*, *Hybrid*, and *Inactivate*. Included publication 2 (Vuori et al. 2002) describes three new variations of *Inactivate*-strategy which are called *Inactivating Prototypes 2-4*. Included publication 3 (Vuori et al. 2000b) introduces five controlled adaptation strategies called *PLAdd*, *PLAddAndLvq*, *PLAddOrLvq*, *OFAdd*, and *OFAddAndLvq*.

3.6.1 Add prototypes

Adaptation strategy *Add*(k) examines the classes of the k prototypes nearest to the input character. The input character is added to the prototype set if any one of the k nearest prototypes belongs to a wrong class. Problems arise if the existing prototypes represent writing styles which the current writer uses but not for the class the prototypes belong to, or, if the writer uses very similar writing styles for different classes. For example, the writing styles used for upper case letter 'O' and digit '0', or lower case letter 'l' and digit '1', can be exactly the same, or, '0' can be distinguished from 'O' by a diagonal stroke and '1' from 'l' by a small hook. In such problematic cases, the adaptation strategy adds new prototypes in vain: the neighborhood of the k nearest prototypes can consists of prototypes of very similar writing styles but of different classes no matter how many new prototypes have been added.

3.6.2 Reshape prototypes

When a character input by the current user is basically similar to a prototype of the correct class, for example, it has the same number and order of strokes, but is of slightly different shape, the existing prototype can be reshaped instead of adding the input character to the prototype set. This can be performed with an adaptation strategy called *Lvq*(α) based on a modified version of the Learning Vector Quantization (LVQ) algorithm (Kohonen 1997; Laaksonen et al. 1998). The *Lvq*(α) strategy works as follows. First, a point-to-point correspondence between the strokes of the input character and the best-matching prototype is established with the DTW algorithm. Next, the prototype is reshaped by moving its data points. If the prototype belongs to the same class as the input character, the data points of the prototype are moved toward the corresponding points of the input character. Otherwise, the data points are moved in the opposite direction. The positive learning coefficient α controls the magnitude of these modifications.

Adaptation strategy *Hybrid*(k, α) combines *Add* and *Lvq* strategies. The k nearest prototypes of the input character are examined. If any one of them belongs to the same class as the input character, the nearest prototype is modified with the *Lvq*(α) strategy. Otherwise, the input character is added to the prototype set. With this adaptation strategy, the recognition system will be able to learn completely new writing styles but the increase in the size of the prototype set should be moderate compared to the pure *Add*(k) strategy.

3.6.3 Inactivate prototypes

Adaptation strategy *Inactivate*(N, G) inactivates those prototypes which are more harmful than useful. Some character classes tend to be confused, for example 'g' and '9', as some writers write them in exactly the same way. Prototypes which are similar to characters input by the current user but which belong to wrong classes, i.e. are not used in correct classifications, should be inactivated or forgotten. After each recognition, *Inactivate*(N,G)-strategy checks if the prototype nearest to the input character should be inactivated: if its goodness value g is below a given limit G and it has been the nearest prototype for at least N times, it is removed from the set of active prototypes. The goodness value is defined as follows:

$$g = \frac{N_{corr} - N_{err}}{N_{corr} + N_{err}}, \tag{13}$$

Table 4: Summary of the properties of the prototype inactivation methods. A bullet in the second or third column indicates whether the method keeps an account of the number of times a prototype has been the best matching one or among the k best matching ones. A bullet in the fourth or fifth column marks if the usage limit is the same for all prototypes or whether there are separate limits for the original prototypes and for those which have been included into the prototype set due to adaptation. The last column tells whether the goodness limit G is adjustable or fixed to zero.

Inactivation method	*1-NN*	*k-NN*	*1 usage limit*	*2 usage limits*	*G*
Inactivate	●		●		*adjustable*
Inactivating Pr. 1	●		●		*fixed*
Inactivating Pr. 2		●	●		*fixed*
Inactivating Pr. 3	●			●	*fixed*
Inactivating Pr. 4		●		●	*fixed*

where N_{corr} and N_{err} are the numbers of times when the prototype has been the nearest one and its class has been correct or incorrect, respectively. Parameters G and N control the strictness and reaction rate of the inactivation rule, respectively.

Also the *Inactivating Prototypes 1–4* strategies help the recognition system to recover from confusing or erroneous prototypes. Each of these methods keeps an account for each prototype of the times it has been used, i.e. been the nearest prototype (*Inactivating Prototypes 1* and *3*), or among the k nearest ones (*Inactivating Prototypes 2* and *4*), and of the times its class in these cases has been incorrect or correct. After the user has input a character which then has been classified and the correctness of the classification deduced from the user's ensuing activities, potential inactivation of the prototypes involved in the classification decision can take place. A prototype is inactivated if the following three conditions are fulfilled simultaneously: 1) it has been used sufficiently many times, i.e. it has reached its usage limit N, 2) its class has been incorrect more often than correct, and 3) it is not the last prototype of its class. *Inactivating Prototypes 1*- and *2*-methods apply the same usage limit for all prototypes whereas *Inactivating Prototypes 3*- and *4*-methods have separate limits for the original prototypes and for those which have been included into the prototype set later on due to adaptation. Note, *Inactivating Prototypes 1* with usage limit N and *Inactivate*$(0,N)$ are equal. The properties of the prototype inactivation methods are summarized in Table 4. All the prototype inactivation strategies can be used together with the other adaptation strategies. Erroneous learning samples cause errors in the prototype accounts. Therefore, some prototypes will be inactivated sooner, or later, than they would if all the samples were correctly labeled. In the worst case, prototypes are inactivated even though they are performing well with correct data.

3.6.4 Controlled adaptation

The adaptation strategies can be controlled in two ways: an upper limit can be set for the number of prototypes per class, or the adaptation of a particular class can be switched on or off depending on the classifier's performance. Adaptation strategy *PLAdd* (*PL* stands for prototype limit) works as *Add* if the number of prototypes is less than the upper limit N_{U}. Otherwise, the adaptation is turned off. The *PLAddAndLvq* and *PLAddOrLvq* strategies are similar to the *Hybrid* and *Add* strategies, respectively, but if the upper limit for prototypes has been reached, they perform only prototype reshaping. Adaptation strategies *OFAdd* and *OFAddAndLvq* (*OF* stands for on/off) are based on another controlling principle. At the beginning of the adaptation, they are similar to the *Add* and *Hybrid* strategies. However, adaptation is switched off for an individual character class if the input samples belonging to that class have been classified successfully N_{off} times in a row. On the other hand, adaptation is switched back on after N_{on} consecutive unsuccessful recognitions.

3.6.5 Self-supervised labeling of the learning samples

All the adaptation strategies described above require that the input characters are labeled, i.e. their classes are known. Yet, the class labels of the input characters are generally not available unless the recognition system is being adapted in some special training mode. However, the labeling of the input samples can be performed in a self-supervised fashion according to the user's behavior and reactions to the recognition results. In the simulation reported in included publications 2 and 3 (Vuori et al. 2002; Vuori et al. 2000b) it was assumed that the device in which a character recognition system has been implemented has an input subprogram with the following properties:

1. Input characters are written into the desired positions on the display. Or alternatively, the user first selects the input position by pointing it with a pen and then writes the characters into a special writing area. In either case, it is always clear whether the pen is used as a pointing device or for textual input.

2. Handwritten characters are recognized and replaced by the machine-printed recognition results right after they have been input.

3. Recognition errors and writing mistakes are corrected by inputting a new character on top of the machine-printed character.

4. The input text can be edited by drawing special symbols or gestures. For example, the user might like to remove, move, copy, or paste some parts of the text.

5. All input characters and the their input times and positions are stored.

6. The recognition result of the latest input character is assumed to be the correct class for all the characters input into the same cursor position.

7. Adaptation is carried out character by character after the recognition results for a text entity, for example a line, has been accepted by the user.

Such an arrangement is illustrated in Figure 14. The machine-printed characters are recognition results, and the handwritten characters beneath them are the corresponding inputs. The topmost handwritten character is the latest input attempt. For example, the first character, ‘H’, was recognized correctly after the third attempt. All the three characters input into the same position are labeled according to the recognition result of the latest character and will be used as learning samples when the user has finished the text sequence. As recognition errors and spelling mistakes are corrected by the user in the same way, some of the input characters can be assigned to wrong classes. This could be avoided by using different methods for the two operations but it would be quite inconvenient for the user.

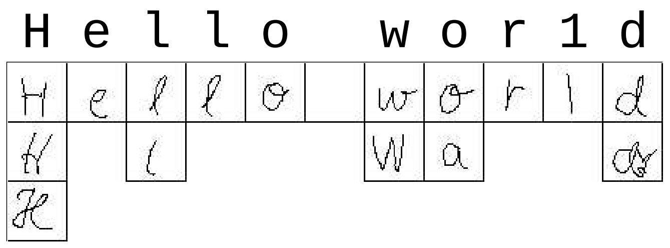

Figure 14: An example of data input. The topmost handwritten characters are the latest inputs into the input locations. The machine printed characters are the corresponding recognition results.

Error sources. Erroneous learning situations of three types can take place in this kind of data input. First, the user may not notice all the recognition errors or does not care to correct them. This situation is called the error type *A*. Such a situation occurs in the example shown in Figure 14: the second character from right was intended to be letter ‘l’ but it was misrecognized and accepted as number ‘1’. According to user experiments carried out with the questionnaire program described in (Vuori et al. 2000a, included publication 4), the recognition errors in which the case of a letter is confused tend to be ignored by the users. Second, if the user makes writing mistakes and corrects them, some of the learning samples will be incorrectly labeled. This is the error type *B*. In the example, the user has first misspelled the word ‘world’ as ‘warld’ and then replaced the correctly-recognized letter ‘a’ with ‘o’. The adaptation system does not now know whether the first character, recognized as ‘a’ was meant to be ‘o’ and misrecognized, or not. Third, carelessly written and thus atypical or malformed characters, such as the first input attempt of letter ‘d’ in the example, give rise to bad learning samples. Also hardware problems can cause erroneous samples. Erroneous samples are introduced for example when there are problems in detecting whether a pen is touching the writing surface or not. In such cases, there can be erroneously segmented characters and characters whose strokes are accidentally broken, connected or completely missing. In addition, the writing equipment can introduce so-called wild points which are spurious data points clearly away from the true trace of the pen. Malformed character samples can be introduced also when there are some problems in interpreting whether the user is using the pen as a pointing device or for writing. The malformed samples are considered to be of the error type *C*.

Hand-held implementation. For evaluation purposes, the character recognition system was implemented in a hand-held device and used as a text input method in a questionnaire program (Aksela 2000; Vuori et al. 2000a, included publication 4). The user interface of the program is illustrated in Figure 15. The program asks the user questions which cannot be answered with a single word. The user inputs one character at a time in either of the two writing areas and the recognition results are shown in the text area above. Successive characters input in the same writing area are separated by a time threshold. The system is able to recognize lower and upper case letters and digits. In addition, a

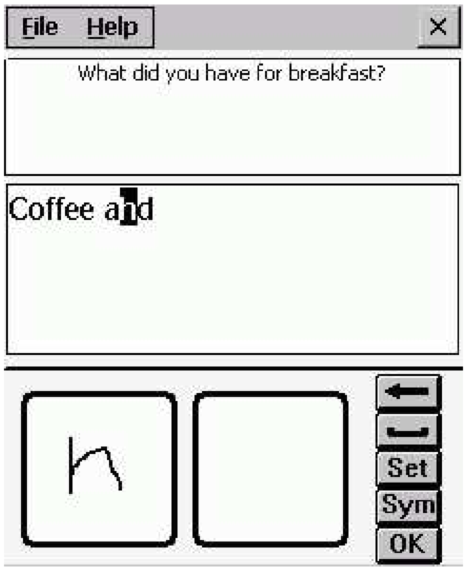

Figure 15: User interface of the questionnaire program. Questions to be answered are shown in the uppermost part of the window. Recognition results are shown in the middle part. Location of the text cursor is indicated with a blinking vertical line and selected text partitions are highlighted with black. Input characters are written into the two rectangular areas in the lower part of the window.

single horizontal line drawn from left to right or vice versa is recognized as a space or backspace, respectively. These two symbols can also be inserted with special buttons. If the system is not able to classify the input character (no prototype can be matched due to prototype pruning or continuity constraints), the user is asked to choose the correct class from a table of all the available characters.

The text cursor can be relocated by pointing the text with the pen. Text partitions can be selected by drawing a horizontal line over the text. These simple functions enable the editing of the text. For example, the user can correct the recognition results by selecting a character and rewriting it. So that the system would not make the same mistake twice in a row, the rewritten characters will not be recognized into the same class as its predecessor. Alternatively, user can make the correction by picking out the correct class from the table prompted by the *Set*-button. Special symbols, such as '?', '% ', '(', ',' etc., can be inserted in a similar manner with the *Sym*-button. More than one character can be deleted at the same time by first selecting the characters and then replacing them with a single character, space, or backspace. When the answer is ready, it is submitted with the *OK*-button.

The recognition system is adapted between questions. Learning samples are labeled according to the recognition results of the latest characters written in the same positions or the class set by the user. All the letters and digits of the submitted answers are used as learning samples. Deleted characters and characters labeled according to them are abandoned from the learning set. However, if a single character is deleted and replaced with a new character immediately, both characters are kept in the learning set. With this policy, the user can change his mind on what he is writing without confusing the adaptation process.

Table 5: The selected parameter values and error rates for the adaptation strategies *Add*, *Lvq*, Hybrid, and *Inactivate*, and for the design and verification set.

Add	*Lvq*	*Hybrid*	*Inact.*	$E^{tot}_{des.}$	$E^{tot}_{verif.}$	$E^{final}_{verif.}$
				14.33	14.13	14.06
4				2.93	3.12	1.81
	0.3			6.81	9.89	8.63
4			3,0	2.95	3.04	1.56
		3,0.3		3.06	4.18	2.50
		3,0.3	16,0	3.33	4.26	2.75

3.6.6 Experimental results for the adaptation strategies

Perfectly labeled learning samples and Add, Lvq, Hybrid, and Inactivate. The first experiments with adaptation strategies *Add*, *Lvq*, *Hybrid*, and *Inactivate* have been reported in (Vuori et al. 2001a, included publication 1). These experiments were carried out by using lower case letters and digits. An initial writer-independent prototype set was formed from the character samples of Database 1. First, character samples written by 16 subjects of Database 3 were recognized one by one and in the same order as they had been collected. The prototype-based recognition system was adapted after each character sample. Parameter values which yielded the lowest total error rates with the design database were selected for the adaptation strategies and the recognition results were then verified with character samples written by 8 additional subjects of Database 4. The results of these experiments are summarized in Table 5. The figures in the first four columns are the parameter values used for the adaptation strategies. The other figures are various error percentages. $E^{tot}_{des.}$ and $E^{tot}_{verif.}$ are the total error rates for all the characters of the design and verification set. $E^{final}_{verif.}$ is the error rate for the last 200 characters of each writer.

From Table 5 it can be seen that all the adaptation strategies could improve the recognition accuracy significantly. Adaptation strategy *Add* yielded the lowest total error but it increased the size of the prototype considerably. The *Hybrid* strategy improved the recognition rates nearly as much as the *Add* strategy and added only a few new prototypes. Adaptation strategy *Lvq* was not sufficient when used alone as it cannot learn writing styles which are fundamentally different from those represented by the initial prototypes. The *Inactivate* strategy was rarely used and it did neither really improve the recognition rates nor limit the size of the prototype set. This was not an unexpected result as erroneous samples had been removed from the databases.

Erroneous learning samples and Add, Hybrid, and Inactivating Prototypes 1–4. The experiments performed with adaptation strategies *Inactivating Prototypes 1–4* are reported in article (Vuori et al. 2002, included publication 2). In these experiments, the *Inactivating Prototypes 1–4* strategies were used together with *Add* and *Hybrid* strategies in on-line adaptation simulations in which some of the learning samples were erroneously labeled. The lower case letters and digits of Databases 3 and 4 were used as a test set. The initial writer-independent prototype set was formed from the characters samples of Database 1. Two types of erroneous labeling situations were concerned. In simulations where the erroneously labeled were of type A, some of the character samples were erroneously labeled according to their recognition results. In simulations where the erroneously labeled were of type B, some of the character samples were labeled to arbitrary incorrect classes.

In the first set of simulations, the risk of erroneous learning samples was minimized by performing adaptation after the successful recognitions, and with a given error probability, also after some unsuccessful recognition. Thus, the erroneous learning samples were of type A. No prototype inactivation strategy was applied in these experiments. According to the results of the experiments, the prototype-based recognition system is able to improve its performance significantly on the basis of the successfully recognized characters. Without any adaptation the recognition error rate was on average approximately 15%. If the *Add* or *Hybrid* strategy was applied and all the recognition errors were noticed, the final error rate was 6.5% or 6.6%, respectively. The number of erroneous learning samples had a nearly linear effect on the final recognition error rate. However, even if all the recognition errors were unnoticed, the final error rates achieved with the adaptive systems did not exceed the final error rate obtained with a nonadaptive system.

In the second set of simulations, all the character samples were used as learning samples. Otherwise, the experiments were similar to the experiments of the first set and the erroneous samples were of type A. These experiments showed that the misrecognized input characters are very valuable. When all learning samples were correctly labeled, both the *Add* and *Hybrid* strategy were able to reduce the recognition error rate considerably more than in the first set of experiments as the final error rates for *Add*- and *Hybrid*-strategy were 1.3% and 1.6%, respectively. If the probability for labeling misrecognized learning samples incorrectly was 0.5, the final error rate was approximately 3.2% with the *Add* strategy and 2.6% with the *Hybrid* strategy. These result are still significantly better than those attained in the first set of experiments. If all misrecognized learning samples were erroneously labeled, the results of the first and second set of experiments were similar to each other.

In the third set of simulations, erroneous learning samples were of type B. The relation between the recognition error rate and the probability of erroneous learning samples was almost linear. If ten percent of the learning samples were labeled incorrectly, the final error rate was 14.6% or 18.4% for the *Add* and *Hybrid* strategy, respectively. The results obtained with the *Add* strategy were so much better probably because it did not ruin prototypes by reshaping them wrongly as the *Hybrid* strategy did. With both the strategies the recognition error rate always decreased in the beginning of the adaptation but started to increase soon if the probability of erroneous learning samples was too high, more than approximately 3-4 percent, and was at the end even worse than without adaptation. These experiments showed that erroneous learning samples with arbitrary labels are much more harmful than erroneous learning samples labeled according to their recognition results.

In the fourth set of simulations, prototype inactivation methods *Inactivating Prototypes 1-4* were applied together with the *Add* or *Hybrid* strategy. A prototype inactivation method was applied right after each recognition and application of either the *Add* or *Hybrid* strategy. These experiments showed that the prototype inactivation methods can help the prototype-based recognition system to recover from the erroneous learning samples. There was no significant change in the system's recognition performance whether a separate lower usage limit was used for the prototypes added due to adaptation (*Inactivating Prototypes 3* and *4*) or whether the same limit was used for all prototypes (*Inactivating Prototypes 1* and *2*). This result showed that it is equally important to inactivate confusing initial prototypes and to recover from the erroneous learning samples. The best results were obtained if the prototype inactivation strategies were applied aggressively: prototypes should be in-

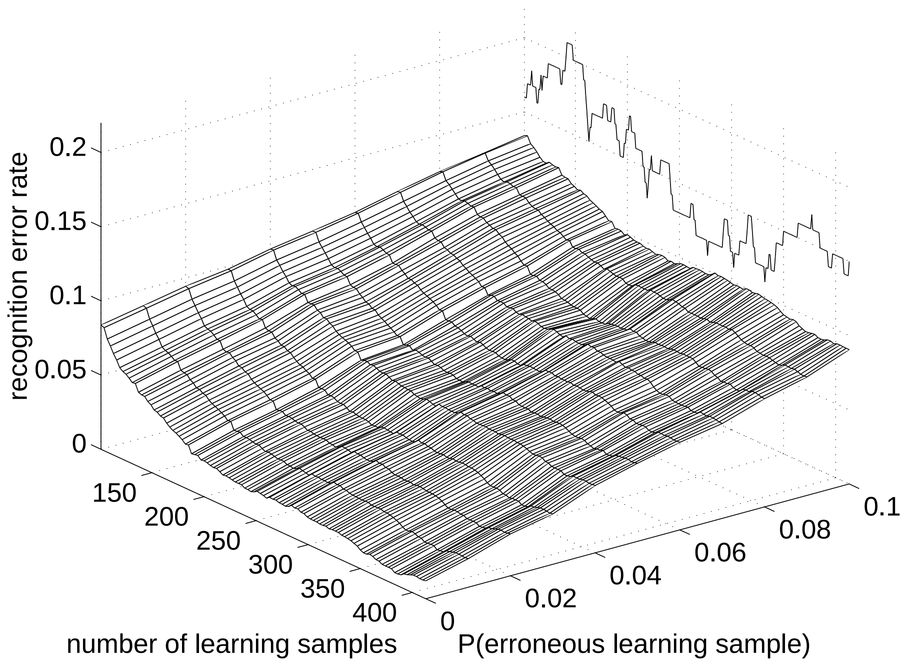

Figure 16: Evolution of the recognition error rates as a function of the number of learning samples and the probability of erroneous learning samples of type B. The surface corresponds to error rates obtained by using the *Hybrid* and *Inactivating prototypes 1* strategies. The background curve corresponds to the error rate of a nonadaptive system.

activated right after they have contributed to only one or two unsuccessful recognitions. It was interesting to see that the recognition rates obtained with the *Hybrid* strategy were only slightly worse than those obtained with the *Add* strategy even though all its modifications on the prototype set could not be completely reversed by the prototype inactivation methods. If ten percent of the learning samples were incorrectly labeled, the final error rate obtained with the *Add* strategy and *Inactivating Prototypes 1* or *2* was 8.3% or 7.5%, respectively. With the *Hybrid* and *Inactivating Prototypes 1* strategy these figures were 9.5% and 7.8%. The evolution of the recognition error rates as a function of the number of learning samples and the probability of erroneous learning samples of type B is illustrated in Figure 16 for the *Hybrid* and *Inactivating Prototypes 1* strategy.

Erroneous learning samples and PLAdd, PLAddAndLvq, PLAddOrLvq, OFAdd, and OFAddAndLvq. The experiments with the controlled adaptation strategies *PLAdd*, *PLAddAndLvq*, *PLAddOrLvq*, *OFAdd*, and *OFAddAndLvq* have been reported in (Vuori et al. 2000b, included publication 3). In these experiments, the initial writer-independent prototype set was formed from the character samples of Database 1 and the lower case letters of Databases 3 and 4 were used as a test set.

The experiments with adaptation strategies which set an upper limit for the number of prototypes per class (*PLAdd*, *PLAddAndLvq*, and *PLAddOrLvq*) showed that there is no need to add many new prototypes per class. Best results were obtained if only two new prototypes were allowed per class and adaptation was continued as pure LVQ-learning after the upper limit for prototype count was reached. *PLAdd*-strategy was outperformed by *PLAddOrLvq*-strategy in respect to initial, final, and total error rates, and growth percentage of the prototype set. *PLAddAndLvq*-strategy increased the size of the prototype set clearly less than *PLAddOrLvq*-strategy. Error rates obtained with the latter strategy were in general better than those obtained with the former strategy. *PLAddOrLvq*-strategy was able to decrease the system's sensitivity to the erroneous learning samples of both type A and B and the system's ability to learn in the error-free situation was preserved.

The idea of controlling the adaptation by switching it on and off did not turn out to be as good an idea as limiting the number of new prototypes. *OFAdd*-strategy did reduce the harmful effects of erroneous learning samples and control the growth of the prototype set. However, the error rate for the simulation with perfectly labeled learning samples increased. Best results were obtained when $N_{\text{on}} = 2$ and $N_{\text{off}} = 3$. *OFAddAndLvq*-strategy did not outperform its uncontrolled version, *AddAndLvq*-strategy, in respect to the recognition rates with any parameter values.

The results of experiments in which the most promising controlled adaptation strategies and their uncontrolled versions (*Add*, *AddAndLvq*, *PLAddAndLvq*, *PLAddOrLvq*, *OFAdd*) were applied together with the inactivation strategy *Inactivate(1,0)* (or *Inactivate prototypes 1*) were congruent with the results of the previous experiments. *Inactivate*-strategy was successful in controlling the growth of the prototype set. It increased the initial error rate by approximately one percentage unit. Its effects on the final error rate were insignificant when some of the learning samples were erroneously labeled according to their recognition results, but it was truly beneficial in eliminating prototypes introduced due to arbitrarily-labeled learning samples.

Genuine on-line user experiments. The *Add*(4) strategy was applied in the collection of Database 4. This can be considered as a genuine on-line test of that adaptation strategy. To evaluate the effects of the adaptation, the characters were recognized afterwards off-line with a nonadaptive system. In these experiments, the initial writer-independent prototype set was formed from the character samples of Database 1 and the test set consisted of the upper and lower case letter and digits written by the 8 subjects of Database 4. Following observations were made on the basis of these experiments: 1) The initial accuracy of the recognizer is determined by the writer-independent prototype set and it is high for those writers whose writing style was covered by the initial prototype set. 2) The benefits of the adaptation, and thus the final accuracy, depend on the consistency and nature of the writing style. 3) The initial rate of improvement was high if the initial accuracy was low. 4) The increase in the size of the prototype set was more prominent if the final error rate was high. 5) The writers did not change their writing styles so that the characters would have resembled more the initial writer-independent prototypes and thus been better recognized. On the contrary, the final error rate was on average slightly higher than the total error rate. It seems that instead of trying to improve the recognition results by writing more carefully and consistently, some of the users tended to demand more and more from the system as it learned and their writing styles became less careful, even sloppy. This would probably not have occurred with a nonadaptive recognition system and if the users had had some other motivations than to finish the tiresome experiment as soon

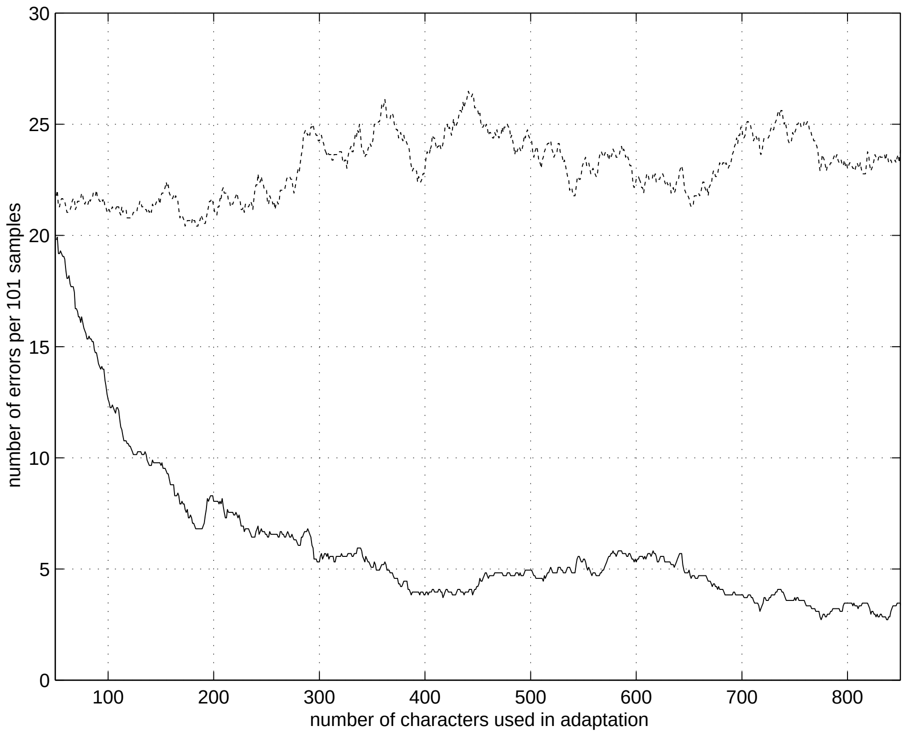

Figure 17: Evolution of the error rate during data collection in which the characters were recognized on-line. In the true, adaptive data collection, which corresponds to the lower solid curve, input characters were added to the prototype set if one of the four nearest prototypes belonged to wrong class. The higher, dashed curve corresponds to the nonadaptive simulation of the data collection.

as possible. Figure 17 illustrates how the error rate evolved during the data collection and its off-line simulation with a nonadaptive recognition system. It is clearly visible how the recognition error rate decreases very rapidly from its initial level of approximately 20% to the level of approximately 3% during the time when first 400 characters are input to the system. Meanwhile, the error rate of the nonadaptive recognition system fluctuates or even increases.

Also the *Hybrid* strategy has been applied in real on-line user experiments. All but one of the nine users who participated in the experiments with the adaptive questionnaire program, which is described in (Vuori et al. 2000a, included publication 4) and in previous section 3.6.5, noticed that the recognition system was able to improve its recognition performance. The recognition rates of the system could not be calculated directly as the true classes of the input characters were not known. However, clear improvements in the recognizer's performance could be observed in the following performance indices: submitted characters vs. all input characters, characters submitted with one attempt vs. all input characters, or characters submitted with one attempt vs. all submitted characters. The evolution of the ratio of characters submitted with a single attempt to all submitted characters during the run of the questionnaire program is illustrated in Figure 18. At least four of the subjects told that they had made some conscious efforts in order to improve

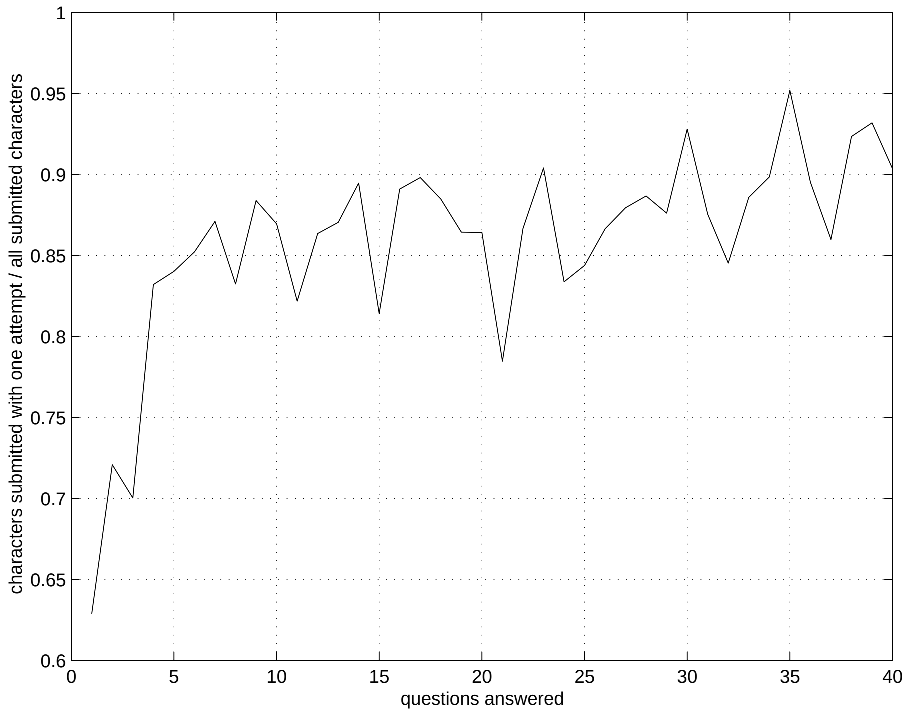

Figure 18: Evolution of the ratio of characters submitted with a single attempt to all submitted characters during the run of the questionnaire program. These results, averaged for 9 subjects, show that the self-supervised adaptation scheme was able to improve the recognition accuracy of the system significantly.

the recognition rates. Only one of them had clearly misunderstood how the recognition system worked and wrote just bigger letters instead of differently shaped letters in order to get them recognized correctly.

3.7 Writing style clustering

Included publications 7 and 8 (Vuori and Oja 2000; Vuori 2002) describe a method which has been developed for examining clustering properties of personal handwriting styles. In this method, personal handwriting styles are represented by vectors, the components of which reflect the writers' tendencies to use certain prototypical writing styles for isolated characters. The very high dimensional writing style vectors are analyzed by projecting them nonlinearly onto a two-dimensional plane by using the SOM algorithm (Kohonen 1997). The main difference between the two works is the size of the analyzed database. The former work is a pilot study with the 45 subjects of the local Databases 1–4 in which the viability of the representation method for handwriting styles is examined. As the results were promising and the much larger public databases UNIPEN and IRONOFF with character samples from over 700 subjects became available to us, the same ideas were reused in the latter work in order to test scalability of the approach.

3.7.1 Personal writing style vectors

Writers' tendencies to use the prototypical styles for isolated characters were measured by the average similarity values between their character samples and prototypes. For each writer, the average similarity value for each prototype was calculated by: 1) evaluating the similarity measures between the prototype and all the writer's character samples of the same class and having the same number of strokes, 2) summing up the similarity values, and 3) finally dividing the sum by the number of its terms. The average similarity values were concatenated into a writing style vector. The dimensionality of a writing style vector is the same as the size of the prototype set.

If a subject had no samples at all for some class, all the average similarity values corresponding to that class were considered to be missing from the writing style vector and did not have any effect in the training of the SOM. If a writer had only one character sample for some class, his or her tendencies to use the prototypical styles of that class were estimated by a single similarity value instead of the average. In such cases, the writer's tendencies to use the prototypical styles consisting of a different number of strokes than the collected sample were zero. In addition, a single sample led to an assumption that the writer uses only the writing style corresponding to the best-matching prototype as the similarity values between the sample and the other prototypes are in most cases very close to zero. For the same reason, the sum of the average similarity values of prototypes of the same class and having the same number of strokes is rarely over one.

3.7.2 Visualization of the writing style vectors with the Self-Organizing Map

SOM (Kohonen 1997) is a neural network in which the neurons are connected to each other so that they form a regular lattice. Each neuron acts both as an input and output neuron and is associated with a reference vector. The reference vectors are compared with the network's input. The outputs of the neurons depend on how similar the input and reference vectors are. The neuron, the reference vector of which is most similar to the input vector, is called the best-matching map unit (BMU). During the training of the network, the reference vectors of the BMUs and their neighboring neurons are updated so that they better represent the input vectors, in this work the writing style vectors. Due to such training, different neurons will specialize in representing different areas of the input space. In addition, neurons near to each other in the neuron lattice tend to correspond to areas close to each other in the input space. Therefore, a SOM can be seen as a nonlinear mapping from the input space to the lower-dimensional lattice space. The SOM's ability to represent the training data faithfully depends on the true dimensionality of the data set and on the size and dimensionality of the neuron lattice.

The U-matrix (Ultsch and Siemon 1989) of a SOM is helpful in detecting clusters on the map. Its coloring is based on the distances between neighboring map units. Areas in which the neighboring map units are similar to each other are colored with dark gray, whereas light shades indicate that the differences between the neighboring units are more significant. Therefore, clusters of personal writing styles can be seen on the U-matrix as dark areas surrounded by lighter areas. The SOM can also be visualized with images colored according to the values of the components of the reference vectors. These images are called component planes. Component planes show how the tendencies to use the corresponding prototypical character styles vary over the map.

3.7.3 Experimental results

In the first study, 22% of the 327 prototypical styles found from the local databases were used only by one of 45 writers. In the second study, the analysis of the 728 writing style vectors showed that approximately 32% of the 2 591 prototypical styles were used by a single subject. These results show that the personal writing styles are likely to consist of character shapes which cannot be learned from a character database collected from other writers, even if the database is rather large as in the second study. Therefore, in order to achieve satisfactory recognition result for all kinds of writers, a recognition system based on character prototypes has to be able to learn new writing styles.

The correlations between writing styles of single characters found by looking at the component planes of the SOM were intuitively pleasing and were congruent with the prior human knowledge which was not used in the construction of the map, see Figure 19. In the first study with the smaller database, a clear distinction between subjects who use cursive style and those whose style is a mixture of print and block styles could be made. In the second study, the analysis of the U-matrix of the SOM showed that several clusters of writers can be found. However, the interpretation of these clusters was not straightforward: the component planes were indicating high tendencies of the writers to use the corresponding prototypes in the locations of several clusters and the areas where alternative styles were likely to be used were often overlapping. The differences between the alternative styles had to be drastic enough, for example different number or drawing order of the strokes, in order to see clear negative correlation between the corresponding component planes.

On the basis of these two studies, following conclusions were drawn. The results of the two studies justify the use of the knowledge on the writing styles of other writers in the adaptation of a recognition system into a new writing style only to some extent. On the basis of the analysis of ranges of the component planes established in the second study, the number of character prototypes which might explain the writing style clusters is rather small. Therefore, on the basis of only a few arbitrary character samples the knowledge in which cluster a new writer belongs cannot be established and the prototype set cannot be pruned effectively without compromising the recognition accuracy.

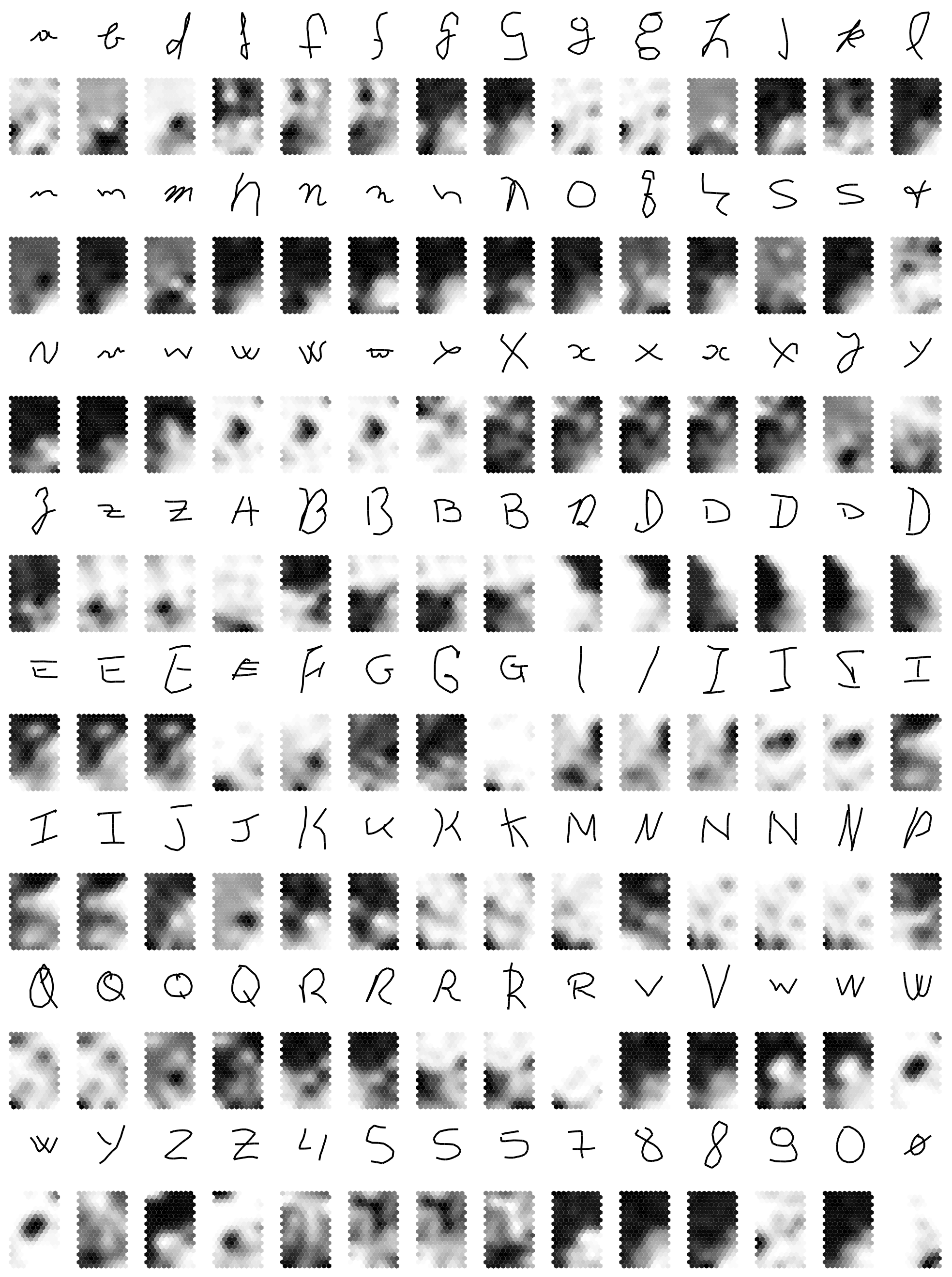

Figure 19: Some interesting component planes of the SOM representing the personal writing styles of 728 subjects. The dark areas of the component planes indicate that writers mapped in that location on the SOM have a high tendency to use the prototypical writing style shown above the components plane.

4 CONCLUSIONS

The main goals of the thesis work have been met well. The developed on-line recognition system works with various kinds of writers as its initial prototype set represents several different writing styles. This thesis work proposes several clustering algorithms which can be used for automatic prototype selection or as a useful visualization tool for a human expert. This thesis also introduces several on-line adaptation strategies and reports various experiments performed with them. Special attention has been paid on the applicability and robustness of the adaptation methods. Typically, by using a few hundreds of input characters written by a new user, the error rate for lower case letters and digits can be decreased from approximately 15% down to less than 2% which is an acceptable level.

Most importantly, the recognition system has been implemented in a hand-held device and the functionality of the introduced self-supervised adaptation scheme has thus been proven not only in off-line simulations but also in genuine on-line user experiments. The clustering analyses performed with the SOM algorithm showed that groups of similar personal writing styles can be found. However, it seems that the best initial performance and fastest improvements cannot be achieved by first selecting a style-specific recognition system and then adapting it to the end-user's writing style but by adapting directly a general writer-independent recognition system into a writer-dependent one.

In the literature survey part of this thesis, the various recognition systems developed for the on-line handwritten character recognition problem have been divided into the following categories: feature space classifiers, prototype-based classifiers, statistical classifiers, neural networks-based classifiers, structural methods-based classifiers, and generative models-based classifiers. The pros and cons of the methods belonging to these categories have been discussed. The most popular, recent, and unique methods suitable for on-line recognition and adaptation have been described more closely. Special attention has been paid on how easily the recognition systems can be adapted into new writing styles. This thesis briefly describes the on-line adaptation experiments with Latin characters and their main results which have been reported in the literature previously. The prototype-based classifiers seem to be best suited for textual input as they can learn incrementally and adapt themselves to new writing styles on the basis of only a few new samples per character class. In addition, such systems are easily understandable to the users, which makes the recognition errors less annoying and more easily avoidable.

The comparison between the various recognition systems reported in the literature is difficult: they have been tested with different data, character sets, and subjects; they pose different constraints on the writing style; and some of the results are writer-dependent while the others are writer-independent. All these factors make the comparison between the recognition results presented in this thesis work and in other research works rather unreasonable. However, if all these differences are ignored, the results of the nonadaptive experiments carried out in this thesis work seem to be somewhat worse than those reported in the literature for basically similar applications of the DTW algorithm. On the other hand, due to adaptation, recognition accuracies high enough to be acceptable for real-world applications have been attained for most of the subjects. The local handwriting databases used in the experiments reported in this thesis have been used also in some other handwriting recognition experiments performed in our laboratory. The DTW-based classifier developed during this thesis project clearly outperforms the chain code classifier described in (Aksela 2000), the Euclidean 1-NN classifier and the adaptive Local Subspace Classier described in (Laaksonen et al. 1999), and the SVM and HMM-SVM classifiers described in (Girdziušas 2001). The recognition error rates reported for these other approaches are two or three times higher than the error rates reported for the DTW-based recognition system. The most distinct feature of the recognition system presented in this thesis work compared to other recognition systems reported in the literature is its ability to learn in a self-supervised manner. The recognition accuracies could be further improved by using contextual information and language modeling. For example, most of the recognition errors could be avoided if it were known in advance whether the input characters are lower or upper case letters or digits.

The DTW algorithm is very well suited for on-line handwriting recognition as it produces high recognition rates and it can easily be visualized to the users. However, there are certain differences in ways how humans and the DTW-based dissimilarity measures compare handwritten characters. To humans, some small-sized differences, say hooks or ornaments at the ends or beginnings of strokes, or minuscule differences in the relative positions of the strokes and corner points, can be very important whereas in the sense of the DTW-based dissimilarity measures they are insignificant. In addition, the DTW-based dissimilarity measures are sensitive to dynamic variations such as writing speed which are more or less meaningless to humans. The differences between the human and DTW-based machine perception can be seen also in the ways how they select prototypes representing the different writing styles of isolated characters. Humans tend to select as prototypes character samples which are more or less ideal, i.e. the strokes are neatly oriented and positioned, there are no unnecessary hooks, the curves are symmetric and smooth etc. The developed automatic clustering algorithms and indices are designed so that they select prototypes which are different from each other and are kind of average samples of groups of similar characters in the sense of the DTW dissimilarity measure. For these reasons, the clustering algorithms and indices were not able to find the same prototypes as a human expert. However, the clustering algorithms did reduce the amount of manual labour significantly as a good prototype set could be formed from their combined results. A human expert can easily select the prototypes for the different writing styles contained in a large character database by examining the tree-like visualizations of the results of the agglomerative and hierarchical clustering algorithms.

The SOM-based analysis of personal writing styles showed that clusters of similar writing styles can be found when handwriting is modeled on character-level. However, as the number of character prototypes which explain the writing style clusters is rather small, the cluster in which a new writer belongs cannot be determined on the basis of only a few character samples of arbitrary classes. Most likely, faster improvements in the recognition accuracies can be achieved by using one general writer-independent prototype set and adapting it to new writing styles right away instead of first selecting the style-specific prototype set which best represents the writing style of the current user among several alternative prototype sets. Perhaps, automatic writing style recognition would be more beneficial in an approach in which handwriting is modeled on subcharacter level. The number of necessary subcharacter models would be lower than the number of allograph models, and personal writing styles could be classified on the basis of fewer character samples which would not need to be of some specific classes. In such a case, the differences between the writing style classes might be something like: vertical strokes are drawn upright or tend to be slanted to left or right, horizontal strokes are drawn from left to right or in the opposite direction, etc.

The experiments performed with the on-line adaptation strategies and the SOM-based analysis of personal writing styles showed that the gradual LVQ-based prototype reshaping is not sufficient alone when a writer-independent prototype set is adapted into a writer-dependent one. New prototypes are needed as the personal writing styles of the end-users of the recognition system are very likely to contain character shapes not included in the initial writer-independent prototype set, even if it has been formed of character samples written by several hundreds of subjects. However, only one or two new prototypes per each character class should be enough. If the adaptation process is not supervised, there is always a risk of erroneous learning samples and thus erroneous prototypes. The risk of erroneous learning situation can be minimized by using only those input characters which seem to be correctly recognized as learning samples. This approach has been shown to improve the recognition rates significantly. However, much better recognition performances can be achieved if the input characters whose recognition results have been corrected by the user are used as learning samples too, even if they are the character samples most likely to be mislabeled or somehow malformed. In such a case, methods for detecting and inactivating erroneous or poorly performing prototypes are necessary. This thesis work proposes several such methods and they make the recognition system more robust against erroneous learning samples.

In some of the current commercial character recognition systems, the prototype sets can be customized in special learning modes which give no rise to erroneous learning samples. This thesis work presents a self-supervised scheme for labeling the input characters so that the adaptation can be performed during the normal writing task. There is no reason why the developed self-supervised adaptive character recognition system should not be equipped also with a special learning tool or mode which enables the inspection of the prototype set and its user-supervised adaptation. On the contrary, that would help the users to gain insight into the recognition process and, of course, give them more control over the adaptation process.

References

Aksela, M. (2000). Handwritten Character Recognition: A Palm-top Implementation and Adaptive Committee Experiments. Master's thesis, Helsinki University of Technology.

Andrianasy, F. and M. Milgram (1995). A new learning scheme for the recognition of dynamical handwritten characters. In *Proceedings of the 5th IEEE Workshop on Neural Networks for Signal Processing*, Cambridge, MA, USA, pp. 371–379.

Anonymous (1989). Global prototype establishment procedure for handwritten character recognition. *IBM Technical Disclosure Bulletin 31*(9), 114–116.

Ansell, M. (1979). Handwriting classification in forensic science. *Visible Language 13*(3), 239–251.

Arakawa, H. (1983). On-line recognition of handwritten characters – alphanumerics, Hiragana, Katakana, Kanji. *Pattern Recognition 16*(1), 9–16.

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

Beigi, H. S. M. (1992a). Character prediction for on-line handwriting recognition. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, Volume 2, Toronto, Canada, pp. TM10.3.1–4.

Beigi, H. S. M. (1992b). A flexible template language model and its application to handwriting recognition. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, Volume 1, Toronto, Canada, pp. WA1.28.1–4.

Beigi, H. S. M. (1997). Pre-processing the dynamics of on-line handwriting data, feature extraction and recognition. In A. C. Downton and S. Impedovo (Eds.), *Progress in Handwriting Recognition*, pp. 191–198. World Scientific Publishers.

Beigi, H. S. M., K. Nathan, G. J. Clary, and J. Subrahmonia (1994). Challenges of handwriting recognition in Farsi, Arabic and other languages with similar writing styles an on-line digit recognizer. In *Proceedings of the 2nd Annual Conference on Technological Advancements in Developing Countries*, New York.

Bellegarda, E. J., J. R. Bellegarda, D. Nahamoo, and K. S. Nathan (1994). A fast statistical mixture algorithm for on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*(12), 1227–1233.

Bellegarda, E. J., J. R. Bellegarda, D. Nahamoo, and K. S. Nathan (1995). A discrete parameter HMM approach to on-line handwriting recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pp. 2631–2622. IEEE.

Bengio, Y., Y. LeCun, and D. Henderson (1993). Globally trained handwritten word recognizer using spatial representation, convolutional neural networks and hidden Markov models. In J. Cowan and G. Tesauro (Eds.), *Advances in Neural Information Processing Systems*, Volume 6. Morgan Kaufmann.

Bengio, Y., Y. LeCun, G. Nohl, and C. Burges (1995). LeRec: A NN/HMM hybrid for on-line handwriting recognition. *Neural Computation 7*(5), 1289–1303.

Biem, A. E. (2001). Minimum classification error training of hidden Markov models for handwriting recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, Volume 3, Salt Lake City, Utah, USA, pp. 1529–1532.

Bontempi, B. and A. Marcelli (1994). A genetic learning system for on-line character recognition. In *Proceedings of International Conference on Pattern Recognition*, Volume 2, pp. 83–87.

Bouletreau, V., N. Vincent, R. Sabourin, and H. Emptoz (1997). Synthetic parameters for handwriting classification. In *Proceedings of 5th International Conference on Document Analysis and Recognition*, Volume 1, pp. 102–106. IEEE.

Brakensiek, A., A. Kosmala, and G. Rigoll (2001). Comparing adaptation techniques for on-line handwriting recognition. In *Proceedings of 6th International Conference on Document Analysis and Recognition*, Seattle, Washington, USA, pp. 486–490.

Brault, J.-J. and R. Plamondon (1993). Segmenting handwritten signatures at their perceptually important points. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*(9), 953–957.

Calinski, T. and J. Harabasz (1974). A dendrite method for cluster analysis. *Communications in Statistics 3*, 1–27.

CalliGrapher (2002, June). CalliGrapher, a handwriting recognition software by ParaGraph. http://www.paragraph.com/calligrapher.html.

Chan, K.-F. and D.-Y. Yeung (1999). Recognizing on-line handwritten alphanumeric characters through flexible structural matching. *Pattern Recognition 32*, 1099–1114.

Clergeau, S. and R. Plamondon (1995). Integration of lexical and syntactical knowledge in a handwriting-recognition system. *Machine Vision and Applications 8*, 249–259.

Connell, S. D. (2000). *Online handwriting recognition using multiple pattern class models.* Ph. D. thesis, Michigan State University.

Connell, S. D. and A. K. Jain (1998). Learning prototypes for on-line handwritten digits. In *Proceedings of the 14th International Conference on Pattern Recognition*, pp. 182–184.

Connell, S. D. and A. K. Jain (2001). Template-based online character recognition. *Pattern Recognition 34*, 1–14.

Connell, S. D. and A. K. Jain (2002, March). Writer adaptation for online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*(3), 329–346.

Cover, T. M. and P. E. Hart (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory IT-13*(1), 21–27.

Crettez, J.-P. (1995). A set of handwriting families: style recognition. In *Proceedings of 3th International Conference on Document Analysis and Recognition*, Montreal, Canada, pp. 489–494.

D'Amato, D., E. Kuerbert, and A. Lawson (2000). Results from a performance evaluation of handwritten address recognition system for the United States Postal Service. In L. R. B. Schomaker and L. G. Vuurpijl (Eds.), *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, Amsterdam, pp. 189–198. Nijmegen: International Unipen Foundation. ISBN 90-76942-01-3.

Davies, D. L. and D. W. Bouldin (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence 1*(2), 224–227.

Decuma Latin (2002, June). Decuma Latin, a handwriting recognition software by Decuma. http://www.decuma.com/pages/products/latin_recognition.html.

Djeziri, S., W. Guerfali, R. Plamondon, and J. M. Robert (2002). Learning handwriting with pen-based systems: computational issues. *Pattern Recognition 35*(5), 1049–1057.

Duda, R. O. and P. E. Hart (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons.

Duneau, L. and B. Dorizzi (1994). On-line cursive script recognition: a system that adapts to an unknown user. In *Proceedings of International Conference on Pattern Recognition*, Volume 2, pp. 24–28.

Fix, E. and J. L. Hodges (1951). Discriminatory analysis—nonparametric discrimination: Consistency properties. Technical Report Number 4, Project Number 21-49-004, USAF School of Aviation Medicine, Randolph Field, Texas.

Frankish, C., R. Hull, and P. Morgan (1995). Recognition accuracy and user acceptance of pen interfaces. In *Proceedings ACM CHI'95 Conference on Human Factors in Computing Systems*.

Freeman, H. (1961). On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers 10*(2), 260–268.

Freeman, H. and L. S. Davis (1977). A corner-finding algorithm for chain-coded curves. *IEEE Transactions on Computers*, 287–303.

Fujisaki, T., T. E. Chefalas, J. Kim, C. C. Tappert, and C. G. Wolf (1991). On-line run-on character recognizer: Design and performance. *International Journal of Pattern Recognition and Artificial Intelligence 5*(1&2), 123–137.

Girdziušas, R. (2001). Discriminative On-line Recognition of Isolated Handwritten Characters. Master's thesis, Helsinki University of Technology.

Goldberg, D. and A. Goodisman (1991). Stylus user interfaces for manipulating text. In *Proceedings of the ďth Annual ACM Symposium on User Interface Software and Technology*, pp. 127–135.

Goldberg, D. and C. Richardson (1993). Touch-typing with a stylus. In *Conference proceedings on Human factors in computing systems, INTERCHI'93*, pp. 80–89.

Gorski, N., V. Anisimov, E. Augustin, O. Baret, and D. Price (1999). A2iA check reader: a family of bank check recognition systems. In *Proceedings of 5th International Conference on Document Analysis and Recognition*, pp. 523–526.

Govindan, V. K. (1990). Character recognition – a review. *Pattern Recognition 23*(7), 671–683.

Graffiti (2002, June). Graffiti, a handwriting recognition software by Palm. http://www.palm.com/products/input/.

Guerfali, W. and R. Plamondon (1993). Normalizing and restoring on-line handwriting. *Pattern Recognition 26*(3), 419–430.

Guyon, I., D. Henderson, P. Albrecht, Y. LeCun, and J. Denker (1992). Writer independent and writer adaptive neural network for on-line character recognition. In S. Impedovo and J. C. Simon (Eds.), *From Pixels to Features III: Frontiers in Handwriting Recognition*, pp. 493–506. Elsevier Science Publishers.

Guyon, I., Y. LeCun, and C. Warwick (1995). Usability study of text entry interfaces for wireless personal organizers and communicators. In *Conference proceedings of Interfaces to Real and Virtual Worlds*, Montpellier, France.

Guyon, I., L. Schomaker, R. Plamondon, M. Liberman, and S. Janet (1994). Unipen project of on-line data exchange and recognizer benchmark. In *Proceedings of International Conference on Pattern Recognition*, pp. 29–33.

Guyon, I. and C. Warwick (1996). Joint EC-US survey of the state-of-the-art in human language technology. http://cslu.cse.ogi.edu/HLTsurvey/.

Haykin, S. (1999). *Neural Networks – A comprehensive foundation* (2 ed.). Prentice-Hall, Inc.

Herrick, E. M. (1979). Letters with alternative basic shapes. *Visible Language 13*(2), 133–142.

Hu, J., M. K. Brown, and W. Turin (1996). HMM based on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*(10), 1039–1045.

Hu, J., S. G. Lim, and M. K. Brown (2000). Writer independent on-line handwriting recognition using an HMM approach. *Pattern Recognition 33*(1), 133–147.

Jaeger, S., S. Manke, J. Reichert, and A. Waibel (2001, March). Online handwriting recognition: the NPen++ recognizer. *International Journal on Document Analysis and Recognition 3*(3), 169–180.

Jot (2002, June). Jot, a handwriting recognition software by Communication Intelligence Corporation. http://www.cic.com/products/jot/.

Jung, K. and H. J. Kim (2000). On-line recognition of cursive Korean characters using graph representation. *Pattern Recognition 33*, 399–412.

Karnaugh, M., J. M. Kurtzberg, and C. C. Tappert (1981). Improved parameter set for adaptive symbol recognition. *IBM Technical Disclosure Bulletin 24*(1B), 769–771.

Kassel, R. H. (1995). *A comparison of approaches to on-line handwritten character recognition*. Ph. D. thesis, Massachusetts Institute of Technology.

Kerrick, D. D. and A. C. Bovik (1988). Microprocessor-based recognition of handprinted characters from a tablet input. *Pattern Recognition 21*(5), 525–537.

Kharma, N. N. and R. K. Ward (2001). A novel invariant mapping applied to handwritten Arabic character recognition. *Pattern Recognition 32*, 2115–2120.

Kim, H.-Y. and J. H. Kim (2001). Hierarchical random graph representation of handwritten character and its application to Hangul recognition. *Pattern Recognition 34*(2), 187–201.

Kobayashi, M., S. Masaki, O. Miyamoto, Y. Nakagawa, Y. Komiya, and T. Matsumoto (2001). RAV (reparametrized angle variations) algorithm for online handwriting recognition. *International Journal on Document Analysis and Recognition 3*(1), 181–191.

Kohonen, T. (1997). *Self-Organizing Maps*, Volume 30 of *Springer Series in Information Sciences*. Springer-Verlag. Second Extended Edition.

Kuklinski, T. T. (1984). Components of handprint style variability. In *Proceedings of the 7th International Conference on Pattern Recognition*, pp. 924–926.

Kurtzberg, J. M. (1987). Feature analysis for symbol recognition by elastic matching. *IBM Journal of Research and Development 31*(1), 91–95.

Kurtzberg, J. M. and C. C. Tappert (1981). Symbol recognition system by elastic matching. *IBM Technical Disclosure Bulletin 24*(6), 2897–2902.

Laaksonen, J., M. Aksela, E. Oja, and J. Kangas (1999). Adaptive Local Subspace Classifier in on-line recognition of handwritten characters. In *Proceedings of International Joint Conference on Neural Networks 1999*.

Laaksonen, J., J. Hurri, E. Oja, and J. Kangas (1998). Comparison of adaptive strategies for on-line character recognition. In *Proceedings of International Conference on Artificial Neural Networks*, pp. 245–250.

Laaksonen, J., V. Vuori, E. Oja, and J. Kangas (1999). Adaptation of prototype sets in on-line recognition of isolated handwritten Latin characters. In S.-W. Lee (Ed.), *Advances in Handwriting Recognition*, pp. 489–497. World Scientific Publishing.

LeCun, Y. and Y. Bengio (1994). Word-level training of handwritten word recognizer based on convolutional neural networks. In *Proceedings of International Conference on Pattern Recognition*, Volume 2, pp. 88–92.

Li, X. and N. Hall (1993). Corner detection and shape classification of on-line handprinted kanji strokes. *Pattern Recognition 26*(9), 1315–1333.

Li, X. and D.-Y. Yeung (1997). On-line handwritten alphanumeric character recognition using dominant points in strokes. *Pattern Recognition 30*(1), 31–44.

Lin, C.-K., K.-C. Fan, and F. T.-P. Lee (1993). On-line recognition by deviation-expansion model and dynamic programming matching. *Pattern Recognition 26*(2), 259–268.

Lorette, G. (1999). Handwriting recognition or reading? what is the situation at the dawn of the 3rd millenium? *International Journal on Document Analysis and Recognition 2*(1), 2–12.

Loy, W. and L. Landay (1982). An on-line procedure for recognition of handprinted alphanumeric characters. *Pattern Recognition 4*(4), 422–427.

Lu, P.-Y. and R. W. Brodersen (1984). Real-time on-line symbol recognition using a DTW processor. In *Proceedings of 7th International Conference on Pattern Recognition*, Volume 2, pp. 1281–1283.

MacKenzie, I. S., B. Nonnecke, S. Riddersma, C. McQueen, and M. Meltz (1994). Alphanumeric entry on pen-based computers. *International Journal of Human-Computer Studies 41*, 775–792.

MacKenzie, I. S. and S. Zhang (1997). The immediate usability of graffiti. In *Proceedings of Graphics Interface'97*, Toronto, Canada, pp. 129–137. Canadian Information Processing Society.

Martens, R. and L. Claesen (1997). Dynamic programming optimization for on-line signature verification. In *Proceeding of 4th International Conference on Document Analysis and Recognition*, pp. 653–656. IEEE.

Matíc, N., I. Guyon, and V. Vapnik (1993). Writer-adaptation for on-line handwritten character recognition. In *Proceedings of International Conference on Pattern Recognition and Document Analysis*, Tsukuka, Japan, pp. 187–191. IEEE Computer Society Press.

Milligan, G. W. and M. C. Cooper (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika 50*(2), 159–179.

Morasso, P., L. Barberis, S. Pagliano, and D. Vergano (1993). Recognition experiments of cursive dynamic handwriting with self-organizing networks. *Pattern Recognition 26*(3), 451–460.

Morasso, P., A. Pareto, and S. Pagliano (1992). Neural models for handwriting recognition. In S. Impedovo and J. C. Simon (Eds.), *From Pixels to Features III: Frontiers in Handwriting Recognition*, pp. 423–440. Elsevier Science Publishers.

Morasso, P., A. Pareto, and V. Sanguineti (1992). SOC: A self-organizing classifier. In I. Aleksander and J. Taylor (Eds.), *Artificial Neural Networks 2*, Volume 2, pp. 1223–1226. North-Holland.

Morasso, P. G., M. Limoncelli, and M. Morchio (1995). Incremental learning experiments with SCRIPTOR: an engine for on-line recognition of cursive handwriting. *Machine Vision and Applications 8*, 206–314.

Mozayyani, N., A. Baig, and G. Vaucher (1998). A fully-neural solution for on-line handwritten character recognition. In *Proceedings of International Joint Conference on Neural Networks*, Volume 2, pp. 160–164.

MyScript (2002, June). MyScript, a handwriting recognition software by Vision Objects. http://www.visionobjects.com/gb/business/products/myscript.asp.

Narayanaswamy, S. (1996). *Pen and Speech Recognition in the User Interface for Mobile Multimedia Terminals*. Ph. D. thesis, University of California, Berkley.

Nathan, K., J. R. Bellegarda, D. Nahamoo, and E. J. Bellegarda (1993). On-line handwriting recognition using continuous parameter hidden Markov models. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, Volume 5, pp. 121–124.

Nathan, K. S., H. S. M. Beigi, G. J. C. J. Subrahmonia, and H. Maruyama (1995). Real-time on-line unconstrained handwriting recognition using statistical methods. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pp. 2619–2622. IEEE.

Nathan, K. S., J. Subrahmonia, and M. P. Peronne (1996). Parameter tying in writer-dependent recognition of on-line handwriting. In *Proceedings of International Conference on Pattern Recognition*, pp. 28–32. IEEE.

Neisser, U. and P. Weene (1960). A note on human recognition of hand-printed characters. *Information and Control* (3), 191–196.

Nouboud, F. and R. Plamondon (1990). On-line recognition of handprinted characters: survey and beta tests. *Pattern Recognition 23*(9), 1031–1044.

Nouboud, F. and R. Plamondon (1991). A structural approach to on-line character recognition: System design and applications. *International Journal of Pattern Recognition and Artificial Intelligence 5*(1&2), 311–335.

Odaka, K., H. Arakawa, and I. Masuda (1982). On-line recognition of handwritten characters by approximating each stroke with several points. *IEEE Transactions on Systems, Man and Cybernetics 12*(6), 898–903.

Parizeau, M. and R. Plamondon (1994). Machine vs humans in a cursive script reading experiment without linguistic knowledge. In *Proceedings of International Conference on Pattern Recognition*, Volume 2, pp. 93–98.

Pavlidis, I., R. Singh, and N. P. Papanikolopoulos (1997). An on-line handwritten note recognition method using shape metamorphosis. In *Proceedings of 5th International Conference on Document Analysis and Recognition*, Volume 2, pp. 914–918. IEEE.

PenReader (2002, June). PenReader, a handwriting recognition software by Paragon Software. http://www.penreader.com/technologies/Handwriting.html.

Pitrelli, J. F., J. Subrahmonia, and B. Maison (2001). Toward island-of-reliability-driven very-large-vocabulary on-line handwriting recognition using character confidence scoring. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pp. 1525–1528.

Plamondon, R. and W. Guerfali (1998). The generation of handwriting with delta-lognormal synergies. *Biological Cybernetics 78*(2), 119–132.

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

Platt, J. C. and N. P. Matíc (1997). *Advances in Neural Information Processing Systems*, Chapter A Constructive RBF Network for Writer Adaptation. Number 9. MIT Press.

Prevost, L. and M. Milgram (1997). Static and dynamic classifier fusion for character recognition. In *Proceedings of 5th International Conference on Document Analysis and Recognition*, Volume 2, pp. 499–506.

Prevost, L. and M. Milgram (2000). Modelizing character allographs in omni-scriptor frame: a new non-supervised clustering algorithm. *Pattern Recognition 21*, 295–302.

Qian, G. and K. Truemper (1998). A rapidly learning interpretation system for on-line cursive handwriting. Working paper, University of Texas at Dallas.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pp. 267–295.

Rigoll, G., A. Kosmala, J. Rottland, and C. Neukirchen (1996). A comparison between continuous and discrete density hidden Markov models for cursive handwriting recognition. In *Proceedings of International Conference on Pattern Recognition*, Volume 2, pp. 205–209. IEEE.

Rose, T. G. and L. J. Evett (1995). The use of context in cursive script recognition. *Machine Vision and Applications 8*, 241–248.

Sankoff, D. and J. B. Kruskal (1983). *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison.* Addison-Wesley.

Sato, Y. and K. Kogure (1982). Online signature verification based on shape, motion, and writing pressure. In *Proceedings of International Conference on Pattern Recognition*, Volume 2, pp. 823–826. IEEE.

Scattolin, P. (1995). Recognition of handwritten numerals using elastic matching. Master's thesis, Concordia University, Canada.

Schalkoff, R. (1992). *Pattern recognition: statistical, structural and neural approaches.* John Wiley & Sons, Inc.

Schenkel, M., I. Guyon, and D. Henderson (1995). On-line cursive script recognition using time-delay neural networks and hidden Markov models. *Machine Vision and Applications 8*(4), 215–223.

Schomaker, L. (1993). Using stroke- or character-based self-organizing maps in the recognition of on-line, connected cursive script. *Pattern Recognition 26*(3), 443–450.

Schomaker, L., G. Abbink, and S. Selen (1994). Writer and writing-style classification in the recognition of online handwriting. In *IEE Colloquium (Digest) Proceedings of the European Workshop on Handwriting Analysis and Recognition: European Perspective.*

Schomaker, L., E. Helsper, H. Teulings, and G. Abbink (1993). Adaptive recognition of online, cursive handwriting. In *International Conference on Handwriting and Drawing.*

Senior, A. and K. Nathan (1997). Writer adaptation of a HMM handwriting recognition system. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, Volume 2, pp. 1447–1450.

Singer, Y. and N. Tishby (1994). Dynamical encoding of cursive hanwriting. *Biological Cybernetics 71*(3), 227–237.

smARTwriter (2002, June). smARTwriter, a handwriting recognition software by Advanced Recognition Technologies.
http://www.artrecognition.com/products/ds_smartwriter.htm.

Srihari, S. N. (1985). *Computer text recognition and error correction*, Chapter Introduction, pp. 1–4. IEEE Computer Society Press.

Starner, T., R. S. J. Makhoul, and G. Chou (1994). On-line cursive handwriting recognition using speech recognition methods. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, Volume 5, pp. 125–128. IEEE.

Subrahmonia, J. (2000). Similarity measures for writer clustering. In L. R. B. Schomaker and L. G. Vuurpijl (Eds.), *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, Amsterdam, pp. 541–546. Nijmegen: International Unipen Foundation. ISBN 90-76942-01-3.

Subrahmonia, J., K. Nathan, and M. P. Peronne (1996). Writer dependent recognition of on-line unconstrained handwriting. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, pp. 3478–3481. IEEE.

Suen, C. Y. (1979). A study on man-machine interaction problems in character recognition. *IEEE Transactions on Systems, Man, and Cybernetics 9*(11), 732–737.

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

Tappert, C. C. (1984). Adaptive on-line handwriting recognition. In *Proceedings of 7th International Conference on Pattern Recognition*, pp. 1004–1007. IEEE.

Tappert, C. C. (1991). Speed, accuracy, and flexibility trade-offs in on-line character recognition. *International Journal of Pattern Recognition and Artificial Intelligence 5*(1&2), 79–95.

TealScript (2002, June). TealScript, a handwriting recognition software by TealPoint Software. http://www.tealpoint.com/softscrp.htm.

Theodoridis, S. and K. Koutroumbas (1999). *Pattern Recognition*. Academic Press.

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

Ultsch, A. and H. P. Siemon (1989). Exploratory data analysis: Using Kohonen networks on transputers. Technical Report 329, Univ. of Dortmund, Dortmund, Germany.

Vaucher, G., A. R. Baig, and R. Séguier (2000). A set of neural tools for human-computer interactions: Application to the handwriting character recognition, and visual speech recognition problems. *Neural Computing & Applications 9*, 297–305.

Veltman, S. R. and R. Prasad (1994). Hidden Markov models applied to on-line handwritten isolated character recognition. *IEEE Transactions on Image Processing 3*(3), 314–318.

Viard-Gaudin, C., P. M. Lallican, S. Knerr, and P. Binter (1999). The IRESTE on/off (IRONOFF) dual handwriting database. In *Proceedings of 5th International Conference on Document Analysis and Recognition*, Bangalore, India, pp. 455–458.

Vinciarelli, A. (2002). A survey on off-line cursive word recognition. *Pattern Recognition 35*(7), 1344–1446.

Vuori, V. (2002). Clustering writing styles with a self-organizing map. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, pp. 345–350.

Vuori, V., M. Aksela, J. Laaksonen, E. Oja, and J. Kangas (2000a). Adaptive character recognizer for a hand-held device: implementation and evaluation setup. In L. R. B. Schomaker and L. G. Vuurpijl (Eds.), *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, Amsterdam, pp. 13–22. Nijmegen: International Unipen Foundation. ISBN 90-76942-01-3.

Vuori, V. and J. Laaksonen (2002). A comparison of techniques for automatic clustering of handwritten characters. In *Proceedings of the 16th International Conference on Pattern Recognition*, Volume 3, pp. 168–171.

Vuori, V., J. Laaksonen, and J. Kangas (2002). Influence of erroneous learning samples on adaptation in on-line handwriting recognition. *Pattern Recognition 35*(4), 915–925.

Vuori, V., J. Laaksonen, E. Oja, and J. Kangas (2000b). Controlling on-line adaptation of a prototype-based classifier for handwritten characters. In *Proceedings of the 15th International Conference on Pattern Recognition*, Volume 2, pp. 331–334.

Vuori, V., J. Laaksonen, E. Oja, and J. Kangas (2001a). Experiments with adaptation strategies for a prototype-based recognition system for isolated handwritten characters. *International Journal on Document Analysis and Recognition 3*(3), 150–159.

Vuori, V., J. Laaksonen, E. Oja, and J. Kangas (2001b). Speeding up on-line recognition of handwritten characters by pruning the prototype set. In *Proceedings of 6th International Conference on Document Analysis and Recognition*, Seattle, Washington, USA, pp. 501–505.

Vuori, V. and E. Oja (2000). Analysis of different writing styles with the self-organizing map. In *Proceedings of the 7th International Conference on Neural Information Processing*, Volume 2, pp. 1243–1247.

Vuurpijl, L. and L. Schomaker (1997a). Finding structure in diversity: A hierarchial clustering method for the categorization of allographs in handwriting. In *Proceedings of 5th International Conference on Document Analysis and Recognition*, pp. 387–393. IEEE.

Vuurpijl, L. and L. Schomaker (1997b). *Progress in Handwriting Recognition*, Chapter Coarse writing-style clustering based on simple stroke-related features, pp. 37–44. World Scientific Publishers.

Wakahara, T. and K. Odaka (1997). On-line cursive kanji character recognition using stroke-based affine transformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*(12), 1381–1385.

Wang, P. S.-P. and A. Gupta (1991). An improved structural approach for automated recognition of handprinted character. *International Journal of Pattern Recognition and Artificial Intelligence 5*(1&2), 97–121.

Ward, J. R. and B. Blesser (1985). Interactive recognition of handprinted characters for computer input. *IEEE Computer Graphics and Applications 9*(5), 24–37.

Ward, J. R. and T. Kuklinski (1988). A model for variability effects in handprinting with implications for design of handwriting character recognition systems. *IEEE Transactions on Systems, Man, and Cybernetics 18*(3), 438–451.

Webster, R. G. and M. Nakagawa (1998). An interface-oriented approach to character recognition based on a dynamical model. *Pattern Recognition 31*(2), 193–203.

Wilfong, G., F. Sinden, and L. Ruedisueli (1996). On-line recognition of handwritten symbols. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*(9), 935–940.

Wing, A. M. (1979). Variability in handwritten characters. *Visible Language 13*(3), 283–298.

Wirtz, B. (1998). Method for the dynamic verification of an autograph character string on the basis of a reference autograph character string. United States Patent, patent number 5,730,468.

Yaeger, L. S., B. J. Webb, and R. F. Lyon (1998). Combining neural networks and context-driven search for on-line, printed handwriting recognition in the newton. *AAAI's AI Magazine 19*(1), 73–89.

Yang, Y.-Y. (1998). Adaptive recognition of Chinese characters: Imitation of psychological process in machine recognition. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans 28*(3), 253–265.

Yuen, H. (1996). A chain coding approach for real-time recognition of on-line handwritten characters. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, Volume 6, pp. 3426–3429. IEEE.

Zhang, K., I. Pratikakis, J. Cornelis, and E. Nyssen (2000). Using landmarks to establish a point-to-point correspondance between signatures. *Pattern Analysis & Applications 3*(1), 69–75.