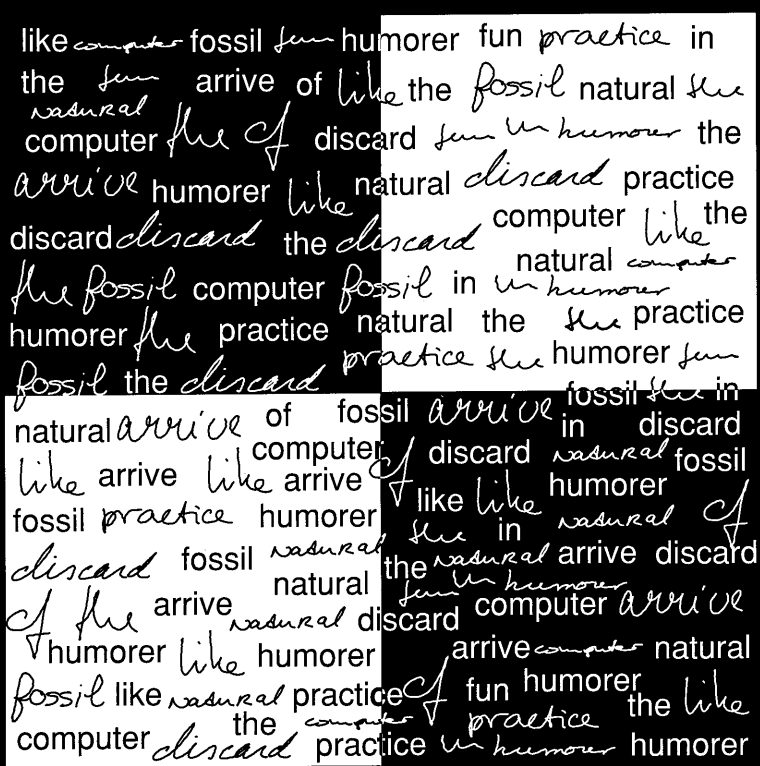
Handwriting recognition and verification : a hidden Markov approach

Handwriting Recognition and Verification

A Hidden Markov Approach

J.G.A. Dolfing



Handwriting Recognition and Verification

A Hidden Markov Approach

The cover picture symbolizes the essence of the current thesis: the processing of handwriting and the investigation of how to translate handwritten text to ASCII text.

Handwriting Recognition and Verification

A Hidden Markov Approach

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de Rector Magnificus, prof.dr. M. Rem, voor een commissie aangewezen door het College voor Promoties in het openbaar te verdedigen op vrijdag 20 november 1998 om 16.00 uur

door

Jannes Gijsbertus Arnoldus Dolfing

geboren te Amersfoort

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr. E.H.L. Aarts
en
prof.dr. J. Wessels

Copromotor: dr. R. Häb-Umbach

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Dolfing, Jannes Gijsbertus Arnoldus

Handwriting Recognition and Verification: A Hidden Markov Approach
Jannes Gijsbertus Arnoldus Dolfing. -
Eindhoven: Eindhoven University of Technology
Thesis Eindhoven. - With index, ref. - With summary in Dutch
ISBN 90-74445-38-1
Subject headings: pattern recognition, handwriting recognition, signature verification, on-line handwriting, hidden Markov model, feature extraction, biometrics.

The work described in this thesis has been carried out at the Philips Research Laboratories in Eindhoven, the Netherlands, as part of the Philips Research programme.

Preface

When I received my computer science degree, the question arose whether to pursue a Ph.D. degree at the university or to join a company. This was resolved quickly when the opportunity came to join the Philips Nat.Lab. However, the idea of getting a Ph.D. was revived and I had the opportunity to combine the best-of-both-worlds.

Over the years, I have worked with numerous people without whom this work would not have been possible. Therefore, I would like to express my gratitude towards them.

First of all, I would like to thank Emile Aarts for his support and encouragement in the process of doing research and writing this thesis. Our stimulating discussions resulted in new ideas and approaches and generally learned me a lot about the nature of research.

Next, I would like to thank Arne Duwaer. His constant support and the PAID tablets were two important factors in the success of this work.

Further, I would like to thank Reinhold Häb-Umbach who learned me a lot about speech recognition techniques.

I am also grateful to the people that are or have been involved in the PAID and ROSE project, in particular to Walter Slegers, who shared an office with me for the past five years, and Anita, Gert-Jan, Kofi, Paul, Rudi, Pieter, Koos and Peter. I also thank Marco van Oosterhout, for his contributions to the signature verification work, the document examiners of the National Forensic Laboratory in Rijswijk, the 'Algoritmen Club' and all persons who contributed handwriting and signature samples.

In addition to the involved people, I would like to thank the management of Philips Research, in particular the department NB&SP, for giving me the opportunity to carry out the research described in this thesis.

Finally, I would like to thank my family and friends for their continuing support and interest in my work.

Eindhoven, November 1998 Hans J.G.A. Dolfing

Contents

1 Introduction **1**
- 1.1 On-line handwriting recognition and verification 2
- 1.2 Informal description of the problem 6
- 1.3 Design philosophy 7
- 1.4 Related work 9
- 1.5 Thesis outline 14

2 A Conceptual Model of Handwriting **15**
- 2.1 Properties 16
- 2.2 Generation 19
- 2.3 Reading 22
- 2.4 Acquisition 23
- 2.5 Representation 28

3 A Theory of Hidden Markov Models **31**
- 3.1 Markov Models 33
- 3.2 Hidden Markov Models 34
- 3.3 Model Structure 35
- 3.4 Evaluation 38
- 3.5 Training 44
- 3.6 Advanced topics 53
- 3.7 Problem Definition 61

4 Handwriting recognition **63**
- 4.1 Representation 64
- 4.2 Models 78
- 4.3 Algorithmic aspects 82
- 4.4 Experimental framework 88

5 Recognition Experiments **91**
- 5.1 Data 91
- 5.2 Parameter selection 93

5.3 Character recognition 95
5.4 Word recognition 111
5.5 Sentence Recognition 124

6 Signature Verification **129**
6.1 Introduction 130
6.2 Representation 137
6.3 Model 144

7 Verification Experiments **149**
7.1 Data 150
7.2 Parameters 152
7.3 Experiments 153
7.4 Discussion 171

8 Conclusion **173**

Bibliography **177**

Author Index **190**

Subject Index **195**

Samenvatting **198**

Curriculum Vitae **199**

1

Introduction

This thesis is concerned with the problem of handwriting recognition and verification based on hidden Markov models. Within the scope of improving man-machine interfaces, we investigate the potentials of novel hardware and software features. The general idea underlying our studies is based on the belief that man-machine interaction can be improved by departing from a situation in which the machine adapts to the user rather than the user to the machine.

Handwriting is only one form of communication among people. Others are speech, sign language and non-verbal communication like facial expressions. In comparison to handwriting, speech has a more temporary character. In this thesis, we focus on handwriting processing in contrast to processing typed or printed text, which is called optical character recognition (OCR). Historically, handwriting has been used for more than 2000 years to permanently record messages. Many cultures have developed their own symbols and spoken words which has lead to a situation today where handwriting is expressed in culturally dependent symbols and writing direction. Example symbol types are shown in Figure 1.1 and include Chinese, Kanji (Japanese), Hangul (Korean), Arabic, Cyrillic, Latin. Examples of different writing directions are Kanji which is written top-to-bottom, Arabic which is written right-to-left and Latin which is written left-to-right. The Egyptian hieroglyphs demonstrate that a combination of left-to-right and right-to-left is also possible.

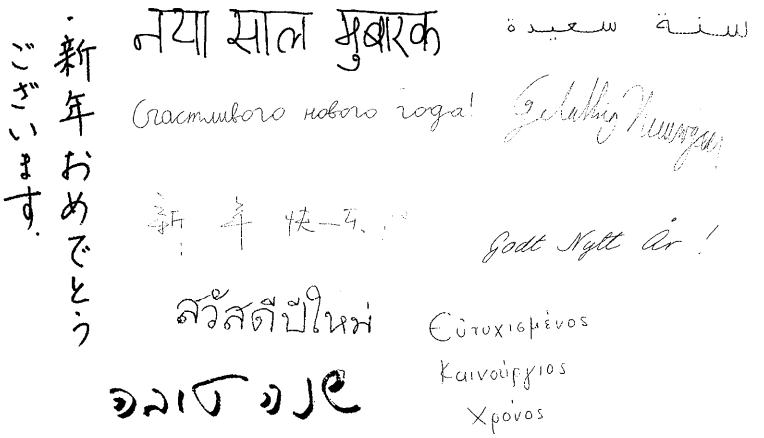


Figure 1.1. Collected handwriting samples from several cultures. The text is 'Happy New Year' in several alphabets and languages. From left-to-right and top-to-bottom we have Japanese, Hindi, Arabic (Maroc), Cyrillic (cursive style), Latin (Dutch), Chinese (Xing Shu, cursive), Latin (Norwegian, cursive), Thai, Greek, and Hebrew.

Handwritten signatures have been used for centuries for identifying the writer of a letter. Most signatures are legible, but some have been transformed into an illegible personal symbol. Nowadays, some countries, such as the US or Japan, have strict requirements regarding signature legibility. In the case of European signatures, however, the written name is often distorted into a symbol-like signature.

Handwriting and speech have largely complementary properties. While speech is used for fast communication, handwriting is used for personal note-taking. This is reflected in the types of application in which speech recognition is used in dialogue systems and dictation while handwriting recognition is used in electronic agendas for instance. Signatures are used in banking to authorize checks. It is possible to enhance automatic teller machines (ATMs) to use signatures instead of or in combination with PIN codes for authorization. Signatures are also used in combination with credit card transactions. The algorithms and hardware discussed in this thesis offer a possibility to check the written signatures more accurately. An improved signature verification procedure can be used to attack the annual $1.3 billion credit card fraud [Port, 1996].

1.1 On-line handwriting recognition and verification

On-line handwriting processing with the use of a transparent digitizer is a central topic in this thesis. We divide the processing of on-line handwriting into handwrit-

ing recognition and verification. Handwriting recognition is loosely defined as the task which converts handwritten text to a computer readable form like ASCII text whereas handwriting verification denotes the task of using a piece of handwriting to verify the identity of a writer.

We discriminate between *on-line* and *off-line* handwriting processing on the basis of the available information about the writing process which depends on the recording technique. In off-line handwriting processing, the handwriting is recorded such that only the *static* image of the handwriting is available. In on-line handwriting processing, not only the static image is recorded but also the *temporal* and *dynamic* information like pressure and pen-inclination. Examples are writing on paper which provides the input for off-line handwriting processing and writing on a special device, a digitizer, which forms the input for on-line handwriting processing.

We discriminate between *opaque* and *transparent* digitizers. Opaque digitizers sample the handwriting and show the image on a separate monitor. In the case of transparent digitizers, the writing is done on the display itself. Such a setup is sometimes referred to as *electronic paper* because the electronic ink appears under the pen tip almost as with real pen and paper.

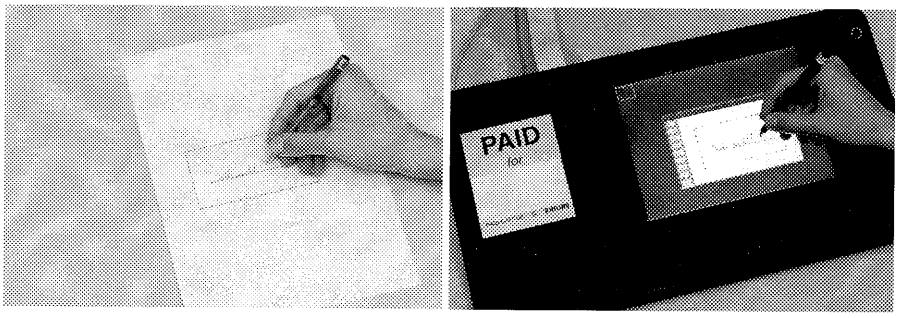


Figure 1.2. Left: writing on paper. Right: writing on transparent digitizer.

The on-line recorded handwriting contains temporal information due to regular sampling with up to 200 points per second (pps) depending on the digitizer and contains information about the temporal order of pen strokes. In addition, the sampling often includes pressure and tilt information for each sampled point where tilt or pen-inclination is the angle of the pen relative to the digitizer surface.

The digitizer will normally convert the handwriting in a regular stream of (x,y) coordinates, which is processed by a computer. At Philips Research, a special IC is used to digitize the handwriting in an equidistant stream of $(x,y,pressure,tilt)$ information [Duwaer, 1993], which enables us to explore the use of the additional dynamic information about the writing process.

The handwriting recognition is executed by analyzing the handwriting, extract-

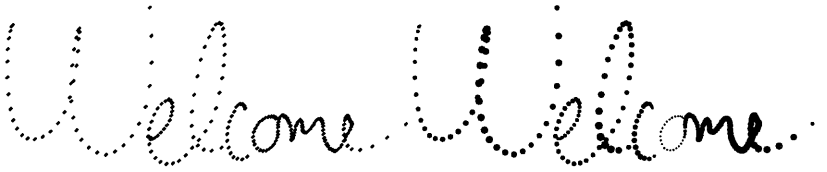


Figure 1.3. The word 'welcome' sampled equidistant in time. Larger dots indicate higher pressure. Left: sampled as (x, y) coordinates. Right: sampled as $(x, y, pressure)$ coordinates.

ing the information and presenting it in computer readable form. This process highly depends on the used language and symbols. Because we know that reading our own handwriting is often easier than reading other handwritings, we discriminate between *writer-dependent* and *writer-independent* recognition.

Since the objective is to impose few limitations on the writer's handwriting, we have to deal with different *writing styles*. In the order of increasingly difficult recognition, we have, prior of all, *boxed* input style, where boxes are provided in which the user can place the characters. This is common in the completion of forms. In the absence of boxes, we speak of *printed* style, which is a sequence of discrete characters. When the discrete characters start overlapping, we call the style *run-on*. The next stage is when all the characters of a word are connected which is known as *cursive* or *connected* handwriting. The most common type of handwriting is a mix of printed and cursive, which is called *mixed* or *unconstrained* style.

1.1.1 Handwriting properties

Although the purpose of handwriting is to convey written messages, this does not mean that the handwriting interpretation is unique. Often, parts of a handwritten text are difficult to read. An example is the infamous doctor's prescription. The reason for the difficult interpretation is not only an unclear handwriting but also because handwriting is inherently variable and ambiguous.

Variability is one of the most challenging problems. No two persons will ever write the same character exactly the same way. For any Latin character, several different shapes are used, e.g., 'a' and '*a*'. These different appearances are called *allographs*. In addition, personal variations change the character appearance.

Ambiguity is the problem which prevents a 100% accurate recognition of handwriting. Characters of the same shape may have different meanings dependent on the context. For example, a small circle can be interpreted as a lowercase 'o', an uppercase 'O', a degree symbol '^{o}', a dot on an 'i' or even a full stop.

Ligatures are connections between two characters. They are another source of confusion because it is possible to connect two characters in such a way that the combination resembles another character. An example is the word 'in', which resembles an 'm' written in cursive style when the dot on the 'i' is omitted.

Overspecification is the problem that parts of handwriting, especially of cursive words, may be interpreted as characters which are not a part of the word. An example is the word 'and', where the vertical bar of the 'd' may be interpreted as an 'l'.

Temporal information contained in the sampled handwriting is often helpful to interpret the handwriting but causes extra problems in other cases. Temporal information allows us to discriminate clockwise and anti-clockwise circles in on-line handwriting. As we will see, this is useful in signature verification, but less appropriate in handwriting recognition, where an 'o' remains an 'o' regardless of the writing direction.

Delayed strokes are another issue. Because we know the timing, we can see whether the dots on the 'i's in the word 'minimum' were put on the 'i's immediately after writing the body of the 'i's or after the word was completed. In the latter case, the dots are called 'delayed strokes'. Other examples of potential delayed strokes are the dot on a 'j' or the horizontal bar of a 't'.

In the case of writing on paper (off-line), the delayed strokes are not important because the image of a word does not show us the order in which the composing strokes were put on paper. In on-line handwriting it is a problem because we expect to find the character data in one piece and not split into body parts and dots. Therefore, delayed strokes in handwriting are a kind of *discontinuity*. In spoken language, this problem does not occur since speech is continuous in time.

1.1.2 Signatures

Handwriting can be used to identify individuals and verify claimed identities. In this study, we concentrate on signature verification because signatures have been an accepted mean of identification for hundreds of years, as demonstrated by the examples in Figure 1.4 from a book by Martinet [1790].

Identification means that one of several possible identities is chosen as the correct one. *Verification* is the process in which we examine a given identity as result of which the identity is either accepted or rejected. Abraham, Dolan, Double & Stevens [1991] explained that verification is possible based on knowledge, possession of a key or a characteristic. There is a whole set of personal characteristics, often defined as *biometrics*, which can be used for the purpose of identification or verification. This includes fingerprints, eye-retina, DNA but also handwriting and specifically signatures [Miller, 1994]. Therefore, it is no coincidence that these characteristics are used in *forensic science* in attempts to solve criminal cases.

Table 1.1. Qualities of the different biometrics.

Biometric	Reliability	Acceptability	Cost
Fingerprint	4	3	2
Hand geometry	3	4	3
Eye retina	5	2	1
Iris	5	2	2
Face	3	4	3
Voice	3	5	3
Handwriting	2	4	4
Keystroke	1	4	5
Signature	3	5	4

Table 1.1 presents an attempt to relate a number of biometric properties on a five point scale. To obtain these figures, we used the discussions by Miller [1994] and Fairhurst, Cowley & Sweeney [1994] in combination with our opinion about the biometric properties under consideration. The *reliability* scores the robustness against fraud. *Acceptability* is a (psychological) score to indicate the willingness of general public to accept the use of such a biometric property. The best score for reliability and acceptability is five. *Cost* indicates the manufacturing cost based on the components where five indicates low cost. Although storage of biometric templates is another design criterion, it has not been included in Table 1.1 because the used techniques and algorithms heavily effect the storage requirements.

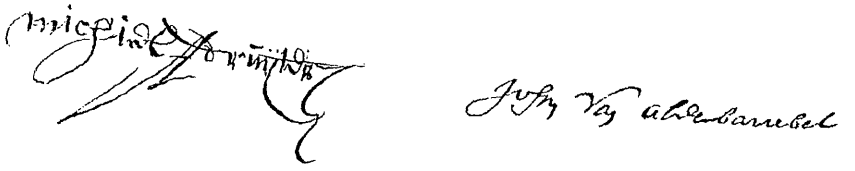


Figure 1.4. Ancient signatures. Left: M. de Ruijter (1613). Right: J.van Oldebarneveld (1599).

1.2 Informal description of the problem

We concentrate on on-line handwriting recognition and verification which are two related aspects of on-line handwriting processing. Both on-line recognition and verification use the same input, i.e., digitized handwriting. Although the input is the same, on-line recognition and verification are two different types of pattern recognition problems which is why we derive two problem descriptions.

With respect to on-line handwriting recognition, we concentrate on the recog-

nition of Latin character sequences resulting in ASCII text. An example is the Dutch version of 'Happy new year' in Figure 1.1. We do not impose constraints on the writing style which is why the use of context is essential in this approach.

First, we aim at the minimization of the writer-independent error rate for unconstrained handwriting recognition. We emphasize writer-independent recognition because it has more practical importance than writer-dependent recognition. Further, we concentrate on recognition performance rather than speed since a poor recognition performance cannot be compensated by speed.

Next, we require the on-line handwriting recognition system to be *generic* and *scalable*. Generic in the sense that we want to recognize characters, words and sentences with a single system. Scalable in the sense that the size of the written text should have no effect on its recognition.

These objectives lead to the following problem description: given a set of Latin character models, context and a sampled handwriting as a sequence of x and y coordinates, how do we compute the sequence of characters, words, sentences which is the most likely interpretation independent of writer, size and style?

With respect to on-line handwriting verification, we concentrate on signature verification because signatures are a widely accepted biometric property useful for identification and verification.

First, we aim at making the verification system secure against impostors. We assume that there is a given identity which has to be checked against the written signature. Second, signature verification is used for different purposes with different security measurements. Therefore, we have to balance the security requirement against the rejection of genuine signatures because users don't like to be refused access without obvious reason.

These objectives lead to the following description of the signature verification problem: given a sampled handwritten signature as a stream of $(x, y, pressure, tilt)$ coordinates, how do we exploit the extra dynamic data to minimize the vulnerability with respect to accidental and deliberate, professional forgers, i.e., to minimize the risk of false acceptance and false rejection?

1.3 Design philosophy

To solve the problems described in the previous section, we employ stochastic classification techniques. Our approach uses the Bayesian decision rule for minimum error rate, discussed in Chapter 3, and is implemented with the hidden Markov model (HMM) as a model for both handwriting recognition and verification.

There are a number of reasons to use hidden Markov models. First, these models are very suitable to recognize input signals with varying time-flow. Second, hidden Markov models are the most common and successful techniques in (con-

tinuous) speech recognition. Because handwriting and speech signals are comparable in the sense that they are both mostly continuous in time, we can most likely employ the hidden Markov models with similar success as in speech recognition. Based on the same argument and the fact that hidden Markov models have also proven their feasibility for speech verification, we expect that we can employ hidden Markov models also successfully in signature verification. Third, the use of hidden Markov models to classify handwriting and speech input opens an opportunity to combine both handwriting and speech processing in one classifier system. Finally, the use of the Bayesian decision rule implemented with the hidden Markov model allows us to simultaneously combine all knowledge sources, especially the handwriting models and context knowledge, into the classification process.

The above reasons are important because we have learned in the past that successful handwriting recognition emphasizes the use of context in handling ambiguities, avoids preliminary segmentations and controls the inherent ambiguity of handwriting by delaying the decision on the interpretation for as long as possible. Similar lessons have been learned in speech recognition. Therefore, we employ speech recognition techniques also in handwriting recognition including the use of a robust, statistical model with data-driven training and implicit segmentations.

We propose a system architecture as shown in Figure 1.5 which contains several independent parts that are handwriting representation, model, classification and context.

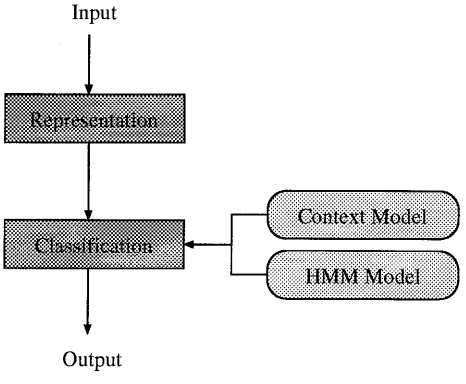


Figure 1.5. System architecture.

Representation. The interface between the sampled handwriting and the hidden Markov model is the representation, which has to catch the handwriting's characteristics in order to achieve successful processing. It is obvious that different characteristics are important for recognition and verification. Since the recognition should be writer-independent, the representation is designed to discard irrelevant

personal differences. In contrast, the verification process exploits personal variations which have to be modeled in the representation. Nevertheless, the model and the search algorithm should be able to handle both representation types. The handwriting characteristics are discussed in detail in Chapter 2.

Model. The hidden Markov model should be able to describe parts of handwriting such as characters and signatures. Although a signature can probably be treated in one piece, a handwritten text is a sequence of characters of unknown length. Therefore, we have to choose the unit to be modeled, e.g., character part, character or word.

Classification. The classification involves the construction of a problem space which is searched for the best solution. An effective search algorithm is important because the number of handwriting interpretations may be huge. This is because handwriting is highly ambiguous as already explained above. A sequence of ambiguous characters forbids any preliminary interpretation. The decision regarding the identity of the handwriting is delayed until we have enough information to be reasonably sure. The consequence of this approach of *delayed decision* is to consider many interpretations (hypotheses) at the same time.

Real-time recognition is also useful. This means that the recognition results should be available as soon as the writing stops. Since it is not possible to say in advance when the writing will stop, the classification will have to maintain hypotheses during each step of the writing process.

Context. Context is the information which provides clues for the correct interpretation of handwriting in addition to the handwriting itself. Typical contexts are a dictionary and grammar. A dictionary contains all character sequences which make up a legal word while a grammar tells us which word sequences are allowed or likely. Such information is essential because it limits the possible number of interpretations and guides the search process to a correct solution.

1.4 Related work

We start this section with an outline of the handwriting recognition field by presenting a number of overview articles of handwriting recognition, together with related techniques like digitizers and language modeling. Then we concentrate on the literature of handwriting recognition in general in Subsection 1.4.1 followed by handwriting recognition based on hidden Markov models in Subsection 1.4.2. We conclude with Subsection 1.4.3 which contains literature related to signature verification based on hidden Markov models.

First of all, Tappert, Suen & Wakahara [1990] give an excellent overview of

the state-of-the-art in handwriting recognition technology. Plamondon & Lorette [1989] and Leclerc & Plamondon [1994] give similar reviews of the state-of-the-art in signature verification technology. Suen, Berthod & Mori [1980] and Mori, Yamamoto & Yasuda [1984] concentrate on the state-of-the-art in character recognition. Finally, Mori, Suen & Yamamoto [1992] give a survey of the related field of optical character recognition.

The hardware used in on-line handwriting recognition to digitize the handwriting is very important. Ward & Schultz [1993] and Quinnell [1995] discuss the properties and techniques used in a digitizer. Tappert, Fox, Kim, Levy & Zimmerman [1986] discuss general hardware requirements for handwriting recognition on transparent tablets. A recent development is that a number of handwriting recognition approaches have learned from speech recognition techniques especially those based on hidden Markov models. Rabiner & Juang [1993] and Lee [1989] discuss in depth speech recognition technology based on hidden Markov models.

Language models are used to exploit the language structure as recognition context in order to minimize ambiguity. The original study of Shannon [1951] studied the information content of the English language. One of the results of this study is that the English language contains lots of redundant information. Estimated is 50% redundant, i.e., that a sequence of characters reduces the information context of the next character by 50%. The study by Witten & Bell [1990] is an excellent example of a study on the information content of character and word-based language models.

In the past, baseline studies on human reading performance have been conducted to find out what kind of performance is realistic for handwriting recognition systems. Edelman, Flash & Ullman [1990] studied human reading performance without context as described in the following quote

> "In comparison, people recognize correctly 96.8% of handprinted characters [Neisser and Weene 1960], 95.6% of discretized handwriting [Suen 1983] and about 72% of cursive strings (see [Edelman 1988] appendix 1)".

1.4.1 Handwriting Recognition

During the last decades, numerous techniques and models have been developed to recognize handwritten text. These techniques can be compared on basis of the following issues: image type (on-line or off-line), data type (characters, words or sentences), style (boxed input, printed, cursive or mixed), culture (Latin, Kanji, Chinese, etc) and context (character context, language models).

Although we concentrate on on-line recognition, it is useful to compare this with off-line recognition techniques as discussed in Senior [1994]. Specific ex-

amples are presented by Simon [1992], Bozinovic & Srihari [1989], Guberman & Rozentsveig [1976], and Kovalevsky [1980]. Simon [1992] discusses off-line word recognition based on regular and singular features where the shortest path from left to right through the handwritten word, i.e., the axis, is the regular part while the singular parts constitute of the complementary parts. Bozinovic & Srihari [1989] discuss an off-line handwriting recognition system based on hierarchical techniques and heuristics including the detection of segmentation points to split a word in its composing characters. Guberman & Rozentsveig [1976] discuss heuristics to retrieve the order of pen-strokes from the image of handwritten text. Kovalevsky [1980] discusses dynamic programming [Bellman, 1957] and correlation techniques to recognize isolated typed characters, isolated handprinted characters and sequences of typed characters.

Handwriting is used for producing text, but also other symbols. While Rubine [1991] explores the recognition of handwriting gestures, Leedham [1990] discusses the recognition of shorthand symbols. Wolman [1992] and Sicard [1992] give examples of the recognition of handwritten and printed music recognition, respectively. Winkler & Lang [1996] explore an approach to the recognition of handwritten, mathematical symbol expressions based on hidden Markov models.

Both statistical and structural approaches are explored for character recognition. Guyon, Albrecht, LeCun, Denker & Hubbard [1991], Le Cun [1990], and Le Cun [1993] discuss character recognition based on *statistical* pattern recognition like neural networks. Guyon, Vapnik, Boser, Bottou & Solla [1992] explore the use of structural risk minimization for character recognition. Examples of *structural* pattern recognition approaches to handwriting recognition are given by Ali & Pavlidis [1977], who demonstrate a digit recognizer, and Stallings [1977] who explores the use of a formal model of the picture of a Chinese character.

Word-based recognition approaches for discrete type handwriting is discussed by Fujisaki, Chefalas, Kim, Tappert & Wolf [1991] and Fujisaki, Beigi, Tappert, Ukelson & Wolf [1992], who use elastic matching for basic character recognition. Weissman, Schenkel, Guyon, Nohl & Henderson [1994] discuss the recognition of handprinted words and use a time delay neural network (TDNN) for character matching. Seni, Srihari & Nasrabadi [1994] demonstrate another TDNN-based approach to on-line, cursive word recognition. Schomaker [1993], Boes, Fogaroli, Maslin, Keil & Whitrow [1990], and Wright [1990] present examples of the recognition of cursive, mixed or unconstrained handwriting. Other approaches based on hidden Markov models and neural networks are discussed later.

An example of statistical language modeling for handwriting recognition is given by Guyon & Pereira [1995], who explore a character-based language model with variable context size. A different approach to context modeling is demonstrated by Evett, Wells, Keenan, Rose & Whitrow [1992].

1.4.2 Recognition based on hidden Markov models

Starting with the work of Jelinek [1976] and Bahl, Jelinek & Mercer [1983], hidden Markov models are well established in speech recognition and have recently gained attention in handwriting recognition. An early example of work in this field is presented by Nag, Wong & Fallside [1986]. A unified system for speech and handwriting recognition is described by Kaltenmeier, Class, Regel-Brietzmann, Caesar, Gloger & Mandler [1993], where hidden Markov models are used to recognize speech and off-line, handwritten words on postcards. Bose & Kuo [1992] describes the use of hidden Markov models in an OCR system. Winkler [1996] discusses the recognition of handwritten mathematical symbols. On the hardware side, Glinski [1987] discusses a dedicated processor for pattern recognition based on hidden Markov models.

Bellegarda, Nathan, Nahamoo & Bellegarda [1993] and Bellegarda, Bellegarda, Nahamoo & Nathan [1994] discuss on-line character recognition based on hidden Markov models. They use continuous hidden Markov models to model characters. One or more states per character are used to tradeoff performance and speed. Kim & Park [1996] show an example of off-line, Korean character recognition.

Unconstrained, on-line word recognition based on hidden Markov models are reported by Nathan, Beigi, Subrahmonia, Clary & Maruyama [1995] and Dolfing & Haeb-Umbach [1997]. While Nathan et al. [1995] use a two-stage search where the first stage reduces dictionary size, Dolfing and Haeb-Umbach [1997] employ a one-stage search. Starner, Makhoul, Schwartz & Chou [1994] discuss on-line, handwritten sentence recognition with a speech recognition system.

Similar recognition systems for word recognition based on speech recognition techniques and time delay neural networks are discussed by Manke & Bodenhausen [1994], Manke, Finke & Waibel [1995] and Schenkel, Guyon & Henderson [1994]. Seiler, Schenkel & Eggimann [1996] explore different representations on the same data to compare off-line and on-line handwriting recognition.

Off-line recognition systems based on hidden Markov models are discussed by Cho, Lee & Kim [1995], Bunke, Roth & Schukat-Talamazinni [1995], and Oh, Ha & Kim [1995]. Cho et al. [1995] emphasize the use of ligature models in the word model and achieves up to 71% correct words with a dictionary containing 10,000 words. Bunke et al. [1995] explore off-line recognition with cooperative writers and a small vocabulary of 150 words and achieved 98% correct answers. Other off-line word recognition work has been done by Kundu, He & Bahl [1989], Chen, Kundu & Zhou [1992], He, Chen & Kundu [1992], and Kundu & Bahl [1988].

Various recent publications discuss details of on-line handwriting recognition with hidden Markov models. Ratzlaff et al. [1996] and Manke et al. [1996] dis-

cuss improvements on efficient search techniques. Beigi et al. [1994] and Dolfing & Haeb-Umbach [1997] discuss size normalization in a hidden Markov model context. Bellegarda et al. [1995] explore modeling issues like the difference between discrete and continuous hidden Markov models while Senior, Nathan & Subrahmonia [1996] explore duration modeling. Instead of a writer-independent system as discussed by Nathan et al. [1995], Subrahmonia et al. [1996] discuss a writer-dependent recognition system. Finally, Dolfing [1998] compares the benefits of ligature models and contextual character models in a writer-independent hidden Markov model context while Kosmala, Rottland & Rigoll [1997] investigate contextual character models in a writer-dependent experiment.

1.4.3 Signature Verification

There are only few studies of signature verification based on hidden Markov models. Among these we mention the studies by Yang [1995], Yang, Widjaja & Prasad [1995], and Paulik & Mohankrishnan [1993]. More general statistical models for signature verification are explored by Hastie, Kishon, Clark & Fan [1991] and also Clark, Hastie & Kishon [1990].

In contrast to the few studies of hidden Markov model based signature verification, there is a fair amount of literature on speaker verification with hidden Markov models. Naik [1990] and also Naik [1994] give overviews of speaker verification. Jacobs & Setlur [1994] and Naik, Netsch & Doddington [1989] demonstrate specific studies on speaker verification with hidden Markov models. These studies include comparisons of dynamic time warping (DTW) and approaches based on hidden Markov models.

1.4.4 Consequences

The previous sections have demonstrated that a number of handwriting recognition and verification systems based on hidden Markov models have been investigated. However, the character and word error rates of on-line handwriting recognition for mixed-style and unconstrained handwriting input using a very large vocabulary is still not satisfactory. Therefore, this thesis concentrates on minimizing the character and word error rates in the context of the recognition of handwritten characters, words and sentences. The approach is to integrate additional context and handwriting-specific knowledge into the representation and model. In particular, we extend the representation with 'contextual features', investigate the combination with delta features, and investigate and compare size-independent with size-dependent representations.

The previous sections also showed that there are few studies on signature verification based on hidden Markov models. In this thesis, the minimization of the verification error rate is most important. Our signature verification study demonstrates

that even professional forgers can hardly break the system. The study contributes a robust, time-normalized scoring technique, investigates a novel way of determining writer-specific thresholds, and investigates the use of pen-tilt, pressure, and contextual features in the representation.

1.5 Thesis outline

The main objective of the present thesis is to study the recognition and verification of on-line handwriting by employing hidden Markov models. This leads to an approach where we study handwriting recognition with the goal of including handwriting-specific and contextual knowledge to improve recognition and verification. Therefore, we discuss the properties of handwriting in Chapter 2, followed by a corresponding mathematical framework presented in Chapter 3. Chapter 4 discusses techniques to integrate knowledge of handwriting into representation and model using the novel framework of 'contextual features', the common contextual hidden Markov models and other techniques. The use of handwriting for verification purposes is explored in Chapter 6. The issue of 'vulnerability to forgers' is explored and novel hardware and software is exploited to attack this problem. The approach is based on digitizing hardware, which accurately samples not only position and pressure but also pen-tilt, and hidden Markov models using contextual features. The experimental results of recognition and verification are discussed in Chapter 5 and 7, respectively. Finally, we present our conclusions in Chapter 8.

2

A Conceptual Model of Handwriting

This chapter discusses the properties and models of the reading and writing process of handwriting in the context of man-machine interaction. The reason for accumulating background knowledge about properties, concepts and models of handwriting generation and reading is that successful handwriting processing is only possible if the input, i.e., the handwriting, is thoroughly understood. In the context of speech recognition, this is also recognized by Lee [1988] who identifies four problems which hampered early speech recognition systems and which are also relevant to handwriting processing. This chapter addresses two of the problems in particular which are the lack of knowledge of the invariants and properties of speech and the lack of a good speech unit to model. These problems do not only apply to speech recognition but also to handwriting processing.

At first, we present a number of basic handwriting properties in Section 2.1. These properties originate in the process of handwriting generation which models are discussed in Section 2.2. Next, we present a number of models for the human reading process in Section 2.3. Because this thesis is concerned with on-line handwriting processing, we discuss the acquisition of handwriting with a digitizer device in Section 2.4 where the digitizer properties determine our perspective on the handwriting. Finally, we discuss the representation of handwriting in Section 2.5 which provides a link to the mathematical model in Chapter 3.

2.1 Properties

This section discusses properties of handwriting in general in Subsection 2.1.1 and signatures in particular in Subsection 2.1.2 which are exploited in a recognition or verification task.

2.1.1 Handwriting properties

In elementary school, we have learned how to write. Depending on the country and the amount of training, handwriting develops differently from person to person because we start customizing our handwriting. Despite the variations, we can read other people's handwriting because the basic characteristics are the same. Intuitively, we can identify the following general properties:

- The handwriting of each individual is unique.

Although every handwriting is unique, we are able to read the majority of other people's handwriting written with the same character set. Because each individual is unique, an individual's neural and muscle system is also unique which makes the handwriting generated by that individual unique. In addition, each individual has a different history and experience which also affects the handwriting style. Handwriting examples which show this variability are shown in Figure 2.1.

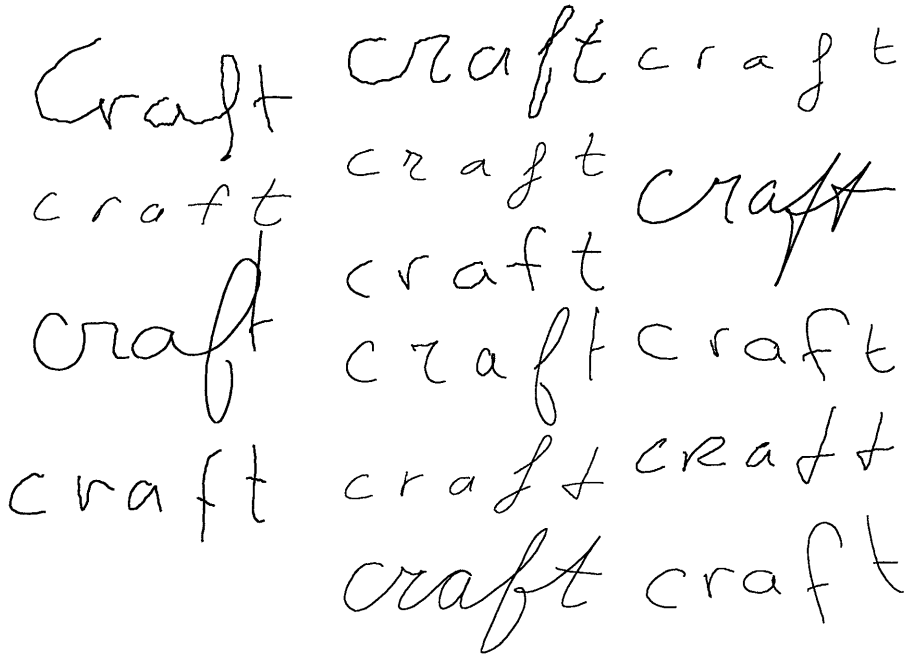


Figure 2.1. Inter-person handwriting variability.

- The handwriting of each individual is not consistent.

The handwriting of an individual varies in time. All written characters are somewhat different. Apart from small differences, some persons use several styles for the same characters, e.g., a mix of cursive and discrete style. The fact that it is not possible to exactly reproduce a piece of handwriting is an important feature in signature verification. If a signature is exactly the same as another original, some form of artificial reproduction has occurred.

Handwriting also varies as a result of writing speed. A rule of thumb is: the faster the writing, the less readable it is. In extreme cases, the handwritten scribbles are legible only to the writer himself because he is the only one who knows the context necessary to decipher the scribbles.

- The type of handwriting depends on nationality.

Although the Latin character set is used to write in a lot of different countries, the handwriting techniques and styles taught at elementary school are not always the same. An example is given in Figure 2.2, in which a common Dutch and a French '8' are compared. The different starting and end points are clearly visible.

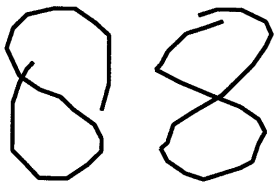


Figure 2.2. The digit '8'. Left: Dutch style '8'. Right: French style '8'.

- The shape of handwriting depends on the neighboring characters.

Besides different nationalities as a source of character shape variation of isolated characters, the character shapes also depend on the neighboring characters and the ligatures between characters in a character sequence. This is especially true in the case of cursive handwriting. Some countries tried to standardize these effects as occurred in the Netherlands by introducing the standard NEN2296 [1958].

Because ligatures are absent between *words*, the inter-words contextual effects on handwriting are minimal. This is an important difference with respect to speech where the effect of co-articulation changes word boundaries between words.

- Handwriting is overspecified.

Given a piece of handwriting, especially cursive handwriting, a handwriting recognition system is not always able to find the characters composing the word without a-priori knowledge of the word content. In fact, random parts of a written word without context can be interpreted in many ways.

Figure 2.3 shows an example of the handwritten word 'and'. The complete word has a unique interpretation. Most individuals recognize this word as 'and' without problems. However, a recognition system which analyzes the word step-by-step will test many parts of this word. Because these word parts are taken out of their context, their interpretation is ambiguous. Figure 2.3 shows that certain parts of 'and' can be interpreted as an 'i' or 'l' character. The phenomenon that a word with given interpretation and constituting characters can be interpreted in other ways and other character sequences by splitting the word in a sequence of handwritten parts, which are ambiguous if taken out of their context, is called overspecification.

Figure 2.3. Overspecified handwriting: 'and' which also contains an 'i' and 'l' character.

2.1.2 Signatures

Signatures have been used for centuries as a mean of authentication on official documents, and for signing papers and cheques. This subsection briefly discusses a number of a-priori properties, in particular *legibility* and *variability*.

Legibility. The legibility and status of signatures is effected by a country and its legal system. In North America, the law demands a signature to be a readable version of the name. In Europe a personal symbol is often used for a signature, but the fast writing of one's name is also used. The personal symbol is usually a symbolization of the name. Figure 2.4 shows an example of a European style signature. Since a symbolic representation of a person's name is often shorter than the full name, the number of strokes and writing time is on the average shorter.

Sato & Kogure [1982] and Yoshimura, Kato, Matsuda & Yashimura [1991] mention that Japanese and Chinese people are not used to writing a signature since this is not part of their identification system. If necessary, they sign by writing their name written in Kanji or Chinese characters, respectively. Illiterate people have a problem when they are asked to write a signature, but they are sometimes allowed to sign with a number of crosses.

Variability. [Plamondon & Lorette, 1990] discuss a number of reasons why a signature shows inconsistencies. The *age* of an individual is important. Young

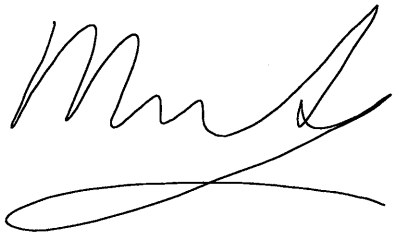


Figure 2.4. An illegible 'European style' signature.

people develop their own signature and the shape changes considerably over time. In contrast, the signature of an adult undergo few changes. Fairhurst, Cowley & Sweeney [1994] showed that the time of day influences the quality of a signature. The *mental state* of an individual can modify the appearance of signatures and increase their variability. It is known that some persons develop several different versions of their own signature. If, for example they have to place their signature often, they may use an abbreviated version of the signature or the initials only. *Physical conditions* like illness, a broken arm, extreme fatigue, flue, alcohol, and medication affect the writing process. A right-handed individual with a broken right arm will have to sign left-handed, which results in a wriggly signature. *Practical conditions* like the writing position, the kind of pen and the writing surface can affect a signature. The friction of the surface determines the pressure used for writing. *Marriage* is another factor. Some women change their signature after marriage from a signature based on their maiden name to one based on their husband's name, or to a combination of both.

2.2 Generation

After discussing basic properties and making some observations on handwriting, we continue with a discussion of some current views and models on how handwriting is generated. This information provides valuable hints about the units of writing. It may be possible to use a compact description of these units as representation. The model of handwriting generation covers the whole process from 'planning to write' to 'generate an ink trace'.

Handwriting is a learned process and not a trick that people can perform by instinct. The motor program that controls handwriting is trained in years of intensive practice at elementary and secondary schools. Handwriting is a sequence of pen strokes accumulating to a sequence of scribbles, characters and words.

These strokes and scribbles are either *pen-down* or *pen-up* movements. A pen-down movement is defined as the pen movement where the pen is placed on the paper to generate an ink trace. This is in contrast to a pen-up movement, where the

pen moves above the paper. We assume that writing starts with a pen-down movement followed by an alternating sequence of pen-up and pen-down movements.

In the literature, the term 'stroke' is often ambiguous because it is used to describe both elementary writing units and complete pen-down movements. In this thesis, we assume that a *stroke* is an elementary pen-down movement (to be defined later in this section) and a *scribble* is equivalent to a complete pen-down movement bounded by two pen-up movements. Therefore, a sequence of pen-down strokes constitute a scribble. A sequence of scribbles accumulates to meaningful units such as characters and words.

The term 'unit' is used to indicate a piece of handwriting like a stroke, scribble, character, or even word. The differences between these types of units is the duration and boundary conditions of the handwriting movement.

Handwriting cannot be generated infinitely fast. In the past, Tappert, Suen & Wakahara [1990] measured an average *writing speed* of about 1.5-2.5 English characters per second and a peak rate of 5-10 characters per second for simple shapes. Another result is that the down strokes of characters are more invariant than other strokes. Thomassen & Van Galen [1996] measured about 100-150ms per basic stroke which leads to 6-10 strokes per second. This corresponds with frequency components of 3-5 Hz in handwriting. Indeed, a frequency analysis over a large database of handwritings shows a small peak at 5Hz. The study by Worthington, Chainer, Wilford & Gundersen [1985] on signatures finds a range of 2-12Hz containing most action. Schomaker [1993] and Worthington et al. [1985] find that the highest frequencies in handwriting are about 20Hz-30Hz. This *bandwidth* limitation makes it possible to estimate a minimum sample frequency of 60pps, which is an absolute minimum. Tappert et al. [1986] suggests at least 100pps as sampling rate.

2.2.1 Motor models

Teulings & Schomaker [1992] consider fast handwriting as a *ballistic movement* of the pen-tip, which is defined in general terms as a movement without any instantaneous position feedback. Such movements are planned in advance and are not interrupted once started. This means that the movement trajectory is defined at the beginning of a stroke. Since planning such a movement requires a certain skill, only a trained motor program can generate these movements. Other examples of ballistic movements are piano playing, fast typewriting and riding on a bicycle.

There are several theories about the structure of a motor program. These motor models fall into two categories, the *top-down* models and the *bottom-up* models [Plamondon & Maarse, 1989].

According to Plamondon & Maarse [1989], the *top-down* approach is developed as a conceptual model for studying the application of the motor program to,

e.g., training and producing movement sequences. A black box contains the neural actions and muscle and limb movements. In contrast, the *bottom-up* approach includes physical or mathematical models which explain and predict the production of handwriting, i.e., the trajectory of the pen-tip.

The goals of the different motor models are the same because they try to explain how a 'thought' is translated into a spatial ink trace. This explanation should include limits to the timing of strokes implying certain constraints on the velocity.

Plamondon & Maarse [1989] and Plamondon, Alimi, Yergeau & Leclerc [1993] argue that bottom-up models can explain a number of handwriting characteristics. Note that the bottom-up models are normally divided in two categories [Plamondon & Maarse, 1989] which are *dynamic models* and *kinematic models*, respectively.

While the dynamic models assume that there is a direct relation between the writing trajectory and the properties of the neural and muscle systems, the kinematic models explain writing as a planned sequence of movements executed by the muscle system of the hand. An interesting aspect of the kinematic models is that they also explain why humans can write with their feet (after practice) or can write with chalk on a blackboard.

The difference between these two models becomes obvious if we take a view at the role of the muscles and limbs in the generation of handwriting. The dynamic models assume that muscles and limbs *cause* the shape of handwriting while kinematic models assume that muscles and limbs are only a *means* or implementation to produce handwriting. Below we discuss both models in more detail.

Dynamic models. The dynamic models include the classic example studied by Hollerbach [1981], who models handwriting with a combined, orthogonal muscle oscillation. Given the combined oscillation in x and y direction, a steady displacement in x direction and frequency and phase differences, the handwriting shape is modeled. The amplitude of the oscillations models the size of the handwriting. This model is sometimes referred to as the oscillation model.

Another dynamic model is a hierarchical model for handwriting generation which is discussed by Schomaker [1993] and Thomassen & Van Galen [1996]. In short, it assumes that handwriting generation process includes

- the translation of a thought into a grammatical and lexical message;
- the translation of this message into a sequence of allographs;
- the conversion of allographs into commands for muscles and limbs.

The model proposed by Schomaker [1993] considers handwriting as a sequence of *basic strokes* or *segments* and not as an oscillation. Evidence against an *oscillation*

model includes the fact that simple repeated patterns like 'elele' are difficult to produce without errors for more than two seconds.

Kinematic models. Kinematic models have the convenient property that they are invariant to starting point, inclination and size [Plamondon, Alimi, Yergeau & Leclerc, 1993]. The kinematic models include the Delta LogNormal model [Guerfali & Plamondon, 1995], and the earlier LogNormal model [Plamondon, Alimi, Yergeau & Leclerc, 1993]. The DeltaLogNormal model describes handwriting as a sequence of overlapping basic strokes. These basic strokes are modeled with only four parameters which are start direction, relative length to previous stroke, curvature and relative starting time. The basic strokes are similar to the stroke units proposed by Schomaker [1993], except that the strokes in the study of Guerfali & Plamondon [1995] are allowed to overlap. The Delta LogNormal explains not only the *shape* of handwriting, but also the *velocity* profile of the pen-tip movement.

2.2.2 Consequences

Plamondon & Maarse [1989] concluded that velocity is an important parameter to control the motor model in order to achieve the desired spatial output. They regard handwriting as a sequence of basic strokes with minor activity at the beginning and end of a stroke. The strokes are a natural unit-of-writing.

Little activity at stroke boundaries fits nicely with the proposal of Schomaker [1993] and Teulings & Schomaker [1992] to take the zero crossings of the y velocity as basic stroke boundaries. Hollerbach [1981] notes that the point of y velocity inversion is important because shape and amplitude manipulation at that point give different corner shapes. Therefore, the points of y velocity inversion provide a natural segmentation of handwriting into segments which is used to define a representation in Section 2.5.

The fact that signatures are considered 'fast handwriting' [Herbst & Liu, 1977] is also important. In the context of motor models, this suggests that a signature is a sequence of ballistic movements which is produced by a specifically trained motor model. If such a highly trained motor model is necessary to produce a signature then it will be very writer-dependent and hard to simulate.

2.3 Reading

Although the human reading process has been studied for more than 100 years, there is no conclusive theory about how the recognition of words during the reading process works exactly. Originally, two extreme views have dominated the research: analytical theories which assume that the letters in a word induce a word image which results in a recognized word and global theories that suggest that entire words are recognized as a unit. There is evidence supporting both models [Taylor

& Taylor, 1983; Bouwhuis, 1979] which has led to the development of hybrid models. A summary of views on off-line recognition is given by Senior [1994].

There is less controversy about the clues which are important in the reading process. Basically, they are the visual clues derived from character and word shape together with contextual clues. Bouma [1971] studied visual features in character recognition and confusion. Taylor & Taylor [1983] indicate that the recognition of complete words in reading is more important than that of single characters. They even describe the reading process as a three-step process:

- complete words are processed in about 50-100ms;
- a more detailed analysis about 50ms after word presentation on a letter basis, working from word boundaries towards the center of the word;
- a slow but complete one-by-one character scan.

Lexical and other contextual knowledge play an important role in the reading process. More familiar words are recognized more easily than unknown words. Bouwhuis [1979] extensively discusses word recognition theories, including some models in which visual and contextual input is combined to recognize a word.

We conclude that context is of prime importance in correctly interpreting the visual input during reading. The visual input of handwriting parts only is easily confused.

2.4 Acquisition

This section discusses the conversion of handwriting into electronic signals. To this end we discuss the properties of the employed digitizer and the nature of the resulting handwriting signals.

2.4.1 Digitizer

Writing on paper results in an image. This image contains static information about handwriting. The dynamic information such as pressure, tilt, and timing is lost. However, forensic document examiners and document analysis software can derive some timing information from the image only as explained by Doermann [1993] and Guberman & Rozentsveig [1976]. This involves microscopic analysis of the ink trace and knowledge about the usual order of pen strokes. If we write on paper using a special instrumented stylus, both static and dynamic information becomes available. An example by Crane & Ostrem [1983] shows the use of a pen with displacement meters to measure the acceleration of the writing. Another possibility is to use a digitizer tablet as a writing surface in order to collect the dynamic information as explained in Section 1.1.

We employ a transparent digitizer for on-line handwriting processing. Our study on handwriting recognition assumes that the digitizer samples a sequence of (x, y) coordinates for all pen-down movements. The signature verification study explores the benefits of additional, writer-specific information like pressure and pen-tilt. We assume that the input for the signature verification task is a sequence of $(x, y, pressure, tilt)$ coordinates for both pen-up and pen-down movements.

The transparent digitizer employed in this thesis is called PAID (Philips Advanced Interactive Display), which is a Philips proprietary tablet platform for capturing handwriting and signatures [Duwaer, 1993]. PAID consists of an LCD plus orthogonal sensors for pen and finger input sampling $(x, y, pressure, tilt)$ with up to 200 pps and optional filtering. Pressure and tilt in x and y directions are measured in 64 levels. Tilt measurements are only available for the pen input. This functionality is implemented with a sensor plate plus dedicated IC (UCA1000) [Quinnell, 1995], which translates pen and finger measurements into coordinates on the serial bus. The tablet is connected via RS-232 port to a PC or workstation with (Pen)Windows or Unix/X-Windows.

The spatial resolution of the PAID is 4096 times 4096 coordinates independent of the actual tablet size. While the *resolution* indicates the minimum distance the pen-tip has to move in order to generate a new coordinate, the *accuracy* denotes the absolute difference between measured and actual position. Spatial accuracy on an example transparent digitizer is $250\mu m$ while the resolution is $40\mu m$.

The measured pressure is the axial pen force (APF) while the exerted pressure is the normal pen force (NPF) as shown in Figure 2.5. If the pen is held horizontally and the normal pen force is larger than zero then the axial pen force equals zero. If the pen is held vertically then we measure the condition $APF = NPF$. We measure the inclination or pen-tilt as two independent signals in x and y direction as shown in Figure 2.5.

The sampled coordinate stream is equidistant in time and consists of tuples $(x, y, p, \theta_x, \theta_y)$ which contain the following data:

- $x(t)$ the x-coordinate sampled at time t;
- $y(t)$ the y-coordinate sampled at time t;
- $p(t)$ the axial pen force at time t;
- $\theta_x(t)$ the pen-tilt mapped in the X-Z plane at time t;
- $\theta_y(t)$ the pen-tilt mapped in the Y-Z plane at time t.

It is possible to compute the first and second differentials of the (x, y) stream resulting in velocity, acceleration, delta pressure and delta tilt measurements.

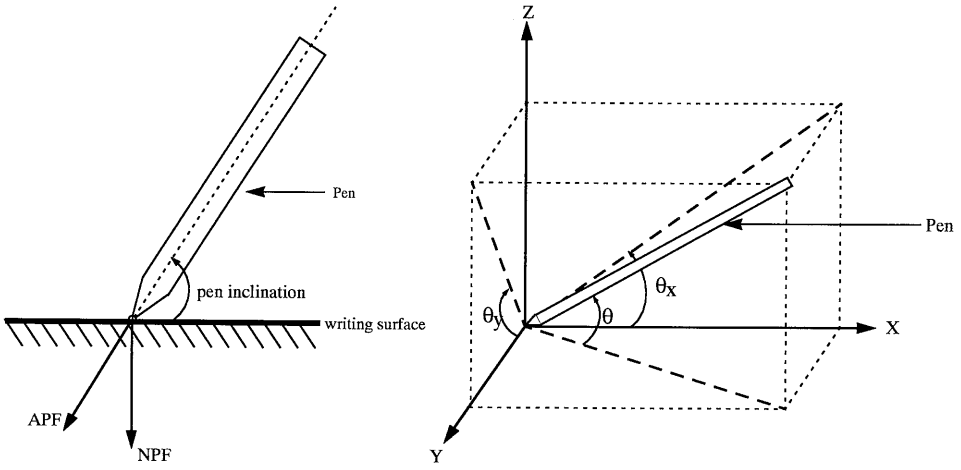


Figure 2.5. Left: Pen-pressure decomposition. Right: Pen-tilt decomposition.

2.4.2 Handwriting signals

A digitizer tablet enables us to collect different aspects of handwriting compared with pen-and-paper handwriting only. Instead of the static image on paper, we can measure a coordinate stream from which we can derive an approximation of the usual static image. In addition, we can measure handwriting properties which are normally not or hardly visible.

We briefly discuss the perception of handwriting through the eye of a transparent digitizer. The digitizer perceives a number of measurable signals illustrated with help of the handwritten word 'and'.

Electronic Ink. The handwriting is sampled as a sequence of pairs of x and y coordinates and appears on the display as 'electronic ink'. Figure 2.6 shows the written word in the form of the sampled coordinates only. The sample rate is about 120 pps. The usual static image of the word is obtained by interpolating between the sampled points.

Figure 2.6. The sampled signal representing 'and'.

Pen-up. One of the differences with respect to writing on paper is that the PAID tablet samples continuously in time not only *on* the writing surface but also above it. Therefore, not only the ink 'on the paper' is measured but also the movements in an adjustable area of about 1 cm above the tablet are measured. The pen movements which generate ink are called *pen-down* movements in contrast to the *pen-up* movements above the tablet. An example is given in Figure 2.7 where the left image shows only the points at which the pressure exceeded a certain threshold value. This corresponds to the behavior of a normal pen which generates ink only if the pressure is sufficiently high. The right image in Figure 2.7 also shows the movement above the tablet. The pen-down scribbles are drawn in solid while the pen-up part is dotted.

This type of measurement is used to find the 'delayed-strokes' as discussed in Subsection 1.1.1. If these movements in the air are stable, they are a potential source of extra information in a signature verification system.

Figure 2.7. The pen-up signal of the word 'and'. Left: without pen-up samples. Right: with pen-up samples.

Velocity. The velocity signal is obtained as the first derivative of the (x,y) signal. Because the (x,y) sampling introduces quantization errors and spatial filters are used to smooth the (x,y) signal, noise is introduced into the velocity signal, which is smoothed with the aid of extra filters.

The velocity signal of the example word 'and' is plotted in Figure 2.8. The velocity during the pen-up signal has been set to zero. The speed units on the vertical axis in Figure 2.8 are [coordinates/sec] while the horizontal axis denotes the sample numbers. To convert this speed to [cm/sec], the absolute dimensions of the tablet are needed.

There is an obvious relation between long and short strokes with high and low velocity, respectively. Also note that points of high curvature correspond to points of low velocity. Important are the velocity inversions in the y direction. As already mentioned in Section 2.2, the timing of these significant points inside the handwriting signal is robust and can be used to obtain a natural segmentation of

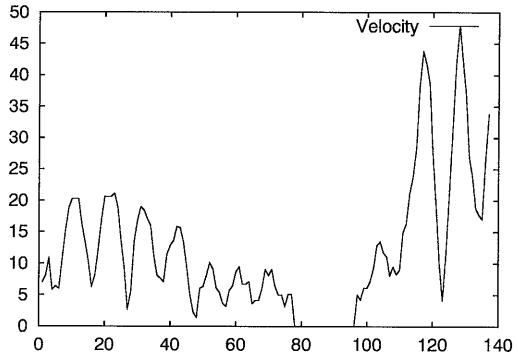


Figure 2.8. The unfiltered velocity signal of 'and'.

the handwriting [Schomaker, 1990]. Also remember that ballistic movements are difficult to imitate, i.e., the velocity signal is difficult to imitate.

Pressure. The pressure signal of the example word is given in Figure 2.9 where the pressure is plotted on the vertical axis and the sample number on the horizontal axis. Pressure is measured in 64 levels which are linearly divided over a pressure range of 200 gram. Alternatively, logarithmic pressure mapping is used.

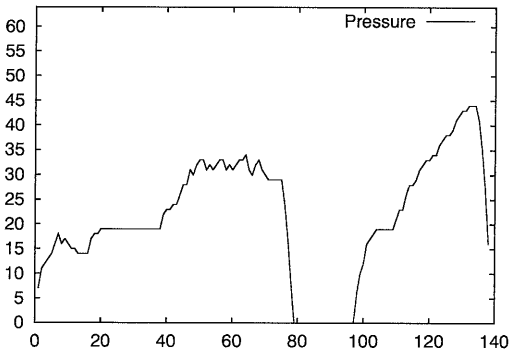


Figure 2.9. The unfiltered pressure signal of 'and'.

When the pressure drops below an adjustable threshold, no ink is generated and we assume that the pen is being lifted from the writing surface. This allows us to differentiate pen-down and pen-up strokes.

Tilt. Figure 2.10 shows the signals for the pen-tilt in x and y direction. Basically, pen-tilt or pen-inclination is the angle the pen makes with respect to the digitizer surface as a result of the way the pen is held in the hand. The digitizer employed in our experiments interleaves the sampling of tilt in x and y direction, which is the

reason that these two signals represent only half the (x,y) sampling speed. Tilt in x and y direction have a range of [-90;90] degrees divided into 64 levels, where zero corresponds to -90^o and 63 to $+90^o$.

Left- and right-handed people can easily be distinguished on the basis of this information. If the tilt in x direction is in the range [0;31] we assume that the writer is left-handed. In other cases, a right-handed writer is assumed. Since every individual has a specific way of holding a pen, this is writer-dependent information which is useful in the signature verification system.

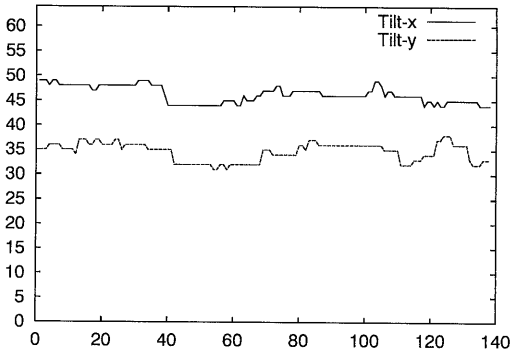


Figure 2.10. The unfiltered tilt signal of 'and'.

2.5 Representation

The properties and models of handwriting presented in this chapter provide a wealth of clues for successful handwriting recognition. In order to bridge the gap between the conceptual model and the formal hidden Markov model that is presented in the next chapter, the handwriting is split into a sequence of units (see Section 2.2) which are processed by the hidden Markov model. The representation determines how the unit is defined.

The representation has to exploit a number of lessons learned in the previous sections. Firstly, the statistical model processes a sequence of handwriting units. Secondly, overspecification prevents early decisions on the identity of characters and words. The final decision on the interpretation is postponed until all the relevant information is available for interpretation. Thirdly, it is not possible to decide beforehand, based on local handwriting clues, which characters are in the handwriting scribble sequence.

In the literature, the term *segmentation* is often used to describe the explicit process of splitting the handwriting into a sequence of characters. In a recognition system based on hidden Markov models, the segmentation is an implicit process of

the recognition where a *time-alignment* of the input data gives the assignment of data parts to characters.

The preprocessing maps the sampled handwriting input to the actual representation in two steps: *grouping* or *blocking* followed by *feature extraction*. Overall, this results in a sequence of units described by feature vectors $O = o_1, o_2, ..., o_T$, which are processed by the hidden Markov model.

1. A sequence of adjacent samples is grouped into a processable unit. This process is called grouping or blocking. A more general term for a handwriting unit is an *item*.
2. To describe the set of grouped samples, a set of features is computed which results in a feature extraction step which results in a *feature vector* representing a part of the handwriting.

Because the grouping process is similar for both recognition and verification, several options are discussed here. In contrast to that, the feature extraction process differs for recognition and verification which is why they are discussed separately in the relevant chapters.

2.5.1 Grouping

The basic issues in selecting the most appropriate unit of writing as discussed in Section 2.2 are resolution, information content, and invariance. The unit of handwriting is chosen depending on the application.

First, the number of samples in a unit is a result of a tradeoff between information content and resolution. If we group only one or two samples in a unit then the resolution is high and only few features are needed to accurately describe the unit. If all data is grouped into one unit then the resolution is very low and all information is contained in one unit. An alternative way to improve the resolution is to allow an overlap in time or space of the handwriting unit. A 50% spatial overlap effectively doubles the resolution.

Second, it is important to realize that the grouping is influenced not only by the number of samples per unit but also by the choice of the boundary points. Example choices are a fixed number of points per unit or invariant boundary points according to the velocity criteria in Subsection 2.2.2. Below we give a list of unit options with increasing resolution and decreasing complexity.

- All handwriting samples are grouped and processed together. In the context of signature verification, this is sometimes called the *parametric approach*. A large number of features is necessary to describe the handwriting part because the information content is high.

- All samples of a pen-down or pen-up *scribble* are grouped together. For some problems like signature verification this is a natural grouping. A disadvantage is that a pen-down scribble may be as large as a complete signature which means that some scribbles represent most of the handwriting while others are small and irrelevant.
- All samples of pen-down or pen-up *strokes* are grouped together. This natural segmentation has been explained in Subsection 2.2.1. Strokes are smaller than scribbles. They are also invariant and equally distributed over the whole handwriting. Therefore, the number of features required to describe a stroke is smaller compared to a scribble.
- Alternatively, we can decide to group a fixed number of samples in a unit. Such a unit is called a *frame*. If the sampling is continuous in time then the unit covers a fixed time window. If the sampling is continuous in space then the frames covers a fixed length of the handwriting.
- If each unit contains only one sample then only few features are required to describe the units. In the context of signature verification, the approach is referred to as the *functional approach*.

3

A Theory of Hidden Markov Models

Pattern recognition is a classification process which assigns *labels* or *categories* to objects of interest. These objects are generally called *patterns*. A decision rule determines the label of a given input pattern. In some applications, the input patterns come from a well-structured space whereas in other cases the input is noisy and ambiguous. While humans perform a number of pattern classifications (like handwriting recognition) almost effortless, machines often encounter serious difficulties.

There are two fundamentally different approaches to pattern recognition: *structural* recognition and *statistical* recognition. The structural or syntactic approach exploits the structure of an input pattern by building a model of its primitive parts, e.g., an annotated tree or graph. In contrast, statistical or stochastic models describe the input patterns by a number of *features*. The model is built directly from example data (*data-driven* learning) and results in a set of estimated probability distributions describing the input patterns. The classification or labeling is a selection of the most likely model.

In this thesis we focus on statistical pattern recognition. A sequence of input *items* X represent an input pattern and each item X is described with a set of L features $X = (x_1,...,x_L)$ that is often called a feature vector. The output consists of one label from a set $\Omega = \{\omega_1,...,\omega_i,...,\omega_C\}$ of C labels or categories, where

each output label ω_i is described by a model λ_i. Therefore, the classification is a function $f : \mathbb{R}^L \rightarrow \Omega$.

We assume that there is a prior probability $P(\omega_i)$ of each label. In addition, there is a class-conditional distribution $P(X|\omega_i)$ which describes the conditional probability of an item X given a label ω_i. In case of continuous values of X we use the probability densities $p(X)$ and $p(X|\omega_i)$ instead of the probabilities $P(X)$ and $P(X|\omega_i)$. To establish a decision on the identity of an item X, we use the Bayesian decision rule for minimizing the error rate.

Definition 3.1 (Bayesian decision rule). Given are an input item X and a set Ω of C possible labels. Then the decision with minimum error rate is called the Bayesian decision rule which is

$$\bar{\omega} = \text{argmax}_{\omega} P(\omega_i|X).$$

□

This rule is the central element in the classification system that we develop in this thesis. Bayes' rule on conditional probabilities is used as implementation and defines how to combine prior probabilities to a posterior probability $P(\omega_i|X)$.

Theorem 3.1 (Bayes' rule). *Given an input item X, a number of possible labels ω_i, the class-conditional distributions $P(X|\omega_i)$, and the prior probability $P(X)$ of an input item X. Then the conditional probabilities are related by*

$$P(\omega_i|X) = \frac{P(X|\omega_i) \cdot P(\omega_i)}{P(X)}.$$

□

We employ Bayes' rule and implement the Bayesian decision rule as $\bar{\omega} = \text{argmax}_{\omega} P(X|\omega_i)P(\omega_i)$ because the probability $P(X)$ is independent of $\bar{\omega}$. Because the posterior probabilities $P(\omega_i|X)$ depend on the unknown quantities $P(\omega_i)$ and $P(X|\omega_i)$, we have to estimate these quantities. The amount of available information determines the difficulty of this estimation. We call the estimation process *training* and estimate $P(\omega_i)$ and $P(X|\omega_i)$ on the basis of example data. We use Maximum Likelihood (ML) training to implement the training. The process which evaluates the posterior probabilities $P(\omega_i|X)$ is called *evaluation*.

Depending on the knowledge about the class-conditional distributions, we use a *parametric* or a *non-parametric* estimation technique. [Huang, Ariki & Jack, 1990]. If the form of the class-conditional densities $P(X|\omega_i)$ is known then a parametric technique is used, e.g., a single Gaussian density. Alternatively, we use a mixture of densities with known structure to model $P(X|\omega_i)$ because such a mixture can model arbitrary functions.

Unlike Rabiner & Juang [1993] and Lee [1988], the evaluation algorithms for single labels and label sequences are presented in a uniform notation to provide a better understanding of differences and improvements. The presentation of training techniques includes the discussion of both Forward-Backward and Viterbi training in relation to discrete and continuous hidden Markov models.

3.1 Markov Models

A Markov Model (MM) is a stochastic model which describes the probability of a state based on the probabilities of previous states. In a discrete time Markov model at time t, where t is the time index, we observe a state q_t. The order of the model describes how many previous states contribute to the state transition probability. From here, we assume that each Markov model is a first-order Markov model which has a state transition probability $a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$.

Definition 3.2 (Markov model). A Markov model is a 3-tuple $\lambda = (A, \pi, S)$ with the following properties

- S is a set of states $S = \{S_1, S_2, ..., S_N\}$, where N is the number of states. The state at time t is called q_t;
- A is a state transition matrix $[a]_{N \times N}$ which contains the transition probabilities a_{ij} and $i = 1...N$, $j = 1...N$. The transition probability from state S_i to state S_j is called $a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$, while the self transition probability is denoted as a_{ii}. The sum of the outgoing transition probabilities is $\sum_{j=1}^{N} a_{ij} = 1$;
- π is the set of *initial* probabilities $\pi = \{\pi_i | i = 1, ..., N\}$ where $\pi_i = P(q_1 = S_i)$;
- $P(q|\lambda)$ is the probability of an observed state sequence q of length T for a given model λ which is computed as

$$P(q|\lambda) = \pi_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}}.$$

□

An example Markov model $\lambda = (A, \pi, S)$ with three states and state transition matrix

$$A = \begin{pmatrix} 0.1 & 0.8 & 0.1 \\ 0.2 & 0.2 & 0.6 \\ 0.1 & 0.4 & 0.5 \end{pmatrix}$$

is given in Figure 3.1.

Markov models are often used to describe chains of events where every event corresponds to a clearly observable state in the model. An example is a 26-state

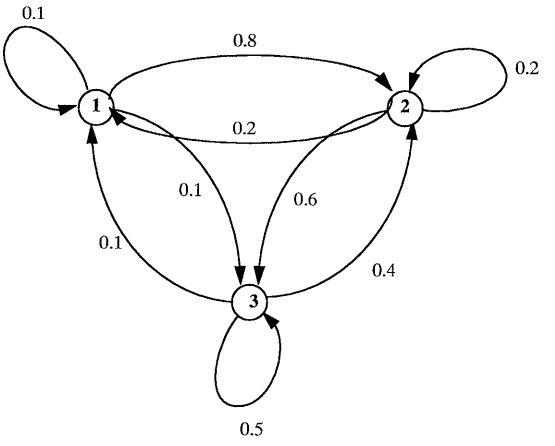


Figure 3.1. An example of a 3-state Markov model.

character transition model with states $\{a,...,z\}$ [Huang et al., 1990] where the transition probabilities are determined by a given vocabulary.

3.2 Hidden Markov Models

A hidden Markov model (HMM) is a stochastic model used to classify an input pattern. In context of the hidden Markov model, we refer to items X observed at time t as *observations* o_t. In the experiments in Chapter 4 to 7, we assume a continuous type of observations, i.e., the observations are feature vectors with dimension L from $\mathbb{R}^L$. In contrast to a Markov model where each observation corresponds to a physical event, a hidden Markov model associates each observation with a probability density function. This probability density function models a *hidden* stochastic process, hence the name hidden Markov Model. We assume a first-order model. In this thesis, we discuss a first-order hidden Markov model as a stochastic model for handwriting recognition and verification.

Assumption 3.1 (The output independency). *It is assumed that the probability or probability density of an observation in the current state depends only on the current and not on the previous states and observations, i.e.,* $P(o_t|o_{t-1},...,o_1;q_t,q_{t-1},...,q_1) = P(o_t|q_t)$ *and* $p(o_t|o_{t-1},...,o_1;q_t,q_{t-1},...,q_1) = p(o_t|q_t)$. □

Definition 3.3 (Observation signal). The sampled observation signal O is a sequence of feature vectors as in $O = \{o_1, o_2,..., o_T\}$. □

An informal definition of the goal of the classification system is to map the signal O onto a piece of text or a binary decision.

Definition 3.4 (Hidden Markov model). A hidden Markov model is a 4-tuple $\lambda = (A, B, \pi, S)$ with the following properties

- S is the set of states $S = \{S_1, S_2, ..., S_N\}$, where N is the number of states. The state at time t is called $q_t \in S$;
- A is a transition matrix $[a]_{N \times N}$ which contains the transition probabilities a_{ij} and $i = 1...N$, $j = 1...N$. The transition probability from state S_i to state S_j is called $a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$ and the constraints are $\forall i, j : 0 \leq a_{ij} \leq 1$ and $\forall i : \sum_j a_{ij} = 1$. The self transition probability is notated as a_{ii};
- B is the set of probability density functions $\{b_j(o_t)\}$ which models for each state the probability density of o_t in that state denoted as $b_j(o_t) = p(o_t | q_t = S_j)$ with the constraint $\int b_j(o_t) do_t = 1, 1 \leq j \leq N$;
- π is the set of *initial* probabilities $\pi = \{\pi_i | i = 1, ..., N\}$ where $\pi_i = P(q_1 = S_i)$ and the constraint $\forall i : \sum_i \pi_i = 1$;
- $p(O, q|\lambda)$ is the probability density of a state sequence q and observation sequence O for a given model λ where $p(O, q|\lambda)$ is computed as

$$p(O, q|\lambda) = \pi_{q_1} \cdot \prod_{t=1}^{T-1} a_{q_t q_{t+1}} \cdot \prod_{t=1}^{T} b_{q_t}(o_t).$$

□

The application of a hidden Markov model in a classifier introduces three basic questions [Rabiner & Juang, 1993] which are discussed in detail in Section 3.4. Given an observation sequence O of length T and a hidden Markov model λ we want to answer the following problems [Lee, 1988]:

1.	Evaluation:	How do we compute the probability density $p(O|\lambda)$?
2.	Explanation:	How do we compute a corresponding state sequence q of length T which explains the observations O in some optimal way?
3.	Training:	How do we compute the parameters of the model λ to maximize $p(O|\lambda)$?

3.3 Model Structure

In order to refine the hidden Markov model structure, a number of variations of the transition matrix and emission probability density function are specified. More specific structure variations like tied transitions, tied states, and state duration are discussed elsewhere, e.g., by Rabiner & Juang [1993].

3.3.1 Transitions

The transition matrix A in the hidden Markov model λ defines the possible paths through the model. As an example, the next figure shows a fully connected (ergodic), five-state hidden Markov model compared with a partly connected hidden Markov model.

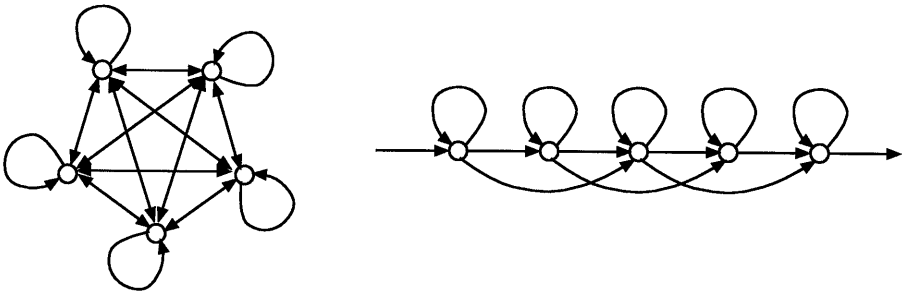


Figure 3.2. Left: fully connected hidden Markov model. Right: left-to-right hidden Markov model.

The partly connected hidden Markov model has a special structure which reflects the time-flow while the data is being processed. This model type is also called *left-to-right model* or Bakis model. Every state has three outgoing transitions called the *self, next* and *skip* transition which results in the following transition matrix A for the five-state example. The *next* transition will usually be the most likely transition to reflect time-flow.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} \\ 0 & 0 & 0 & a_{44} & a_{45} \\ 0 & 0 & 0 & 0 & a_{55} \end{pmatrix}$$

Examples of the use of left-to-right models in the context of speech recognition, handwriting recognition and signature verification are given by Ney et al. [1994], Nathan et al. [1995] and Yang [1995], respectively. Other transition structures for speech and handwriting recognition are used by Lee [1988] and Bunke et al. [1995], respectively.

3.3.2 Emission probability densities

In this thesis, we assume a continuous type of hidden Markov model with continuous observations and probability density function $b_j(o_t)$. The term 'hidden Markov model' is used as a synonym for 'continuous hidden Markov model'. The probability density function gives the probability density $p(o_t|q_t = S_j) = b_j(o_t)$

of an observed feature vector o_t at time t in state S_j. Example density functions used to model $b_j(o_t)$ are Gaussian density functions, a mixture of Gaussian density functions [Rabiner & Juang, 1993] or a mixture of Laplacian density functions [Ney et al., 1994]. An example with a mixture of K Gaussian densities is $b_j(o_t) = p(o_t|q_t) = \sum_{k=1}^{K} w_k \mathcal{G}(o_t, \mu_k, U_k)$ where $\mathcal{G}$ represents the Gaussian density specified by mean μ_k of the k^{th} density and covariance matrix U_k of the k^{th} density. The factor w_k is a weighting factor which models the contributions of the individual densities to the mixture density with constraint $\sum_{k=1}^{K} w_k = 1$.

The advantage of a continuous model for $b_j(o_t)$, specifically the mixture of Gaussian densities, is that we can model arbitrary probability distributions. The disadvantage of the continuous model is that it contains many parameters like mean, covariance and mixture weights which have to be estimated accurately. Models can be simplified with assumptions like a diagonal covariance. Basically, the use of enough training material to estimate all model parameters assures the correct modeling and behavior of the continuous models. Possible consequences of the lack of enough training data [Lee, 1988] include a singular covariance matrix, variance estimation errors, and zero variance.

Note that the different clusters of densities can be seen as cluster of *allophones* or *allographs*, i.e., the same phoneme or character but pronounced or written differently. Loosely speaking, a mixture of densities is the natural way of modeling these allophones and allographs.

Alternatively, a quantized version of $b_j(o_t)$ in Definition 3.4 is used. Such a hidden Markov model with a quantized emission probability is called a discrete hidden Markov model [Lee, 1988; Rabiner & Juang, 1993]. While the discrete model uses a probability distribution $b'_j(v_k)$ and computes the probability $P(O|\lambda)$, the continuous model uses a probability density function $b_j(o_t)$ and computes a probability density $p(O|\lambda)$.

Definition 3.5 (Discrete hidden Markov model). Given the definition of a hidden Markov model in Definition 3.4. Then a discrete hidden Markov model is a 5-tuple $\lambda = (A, B, \pi, S, V)$ where $B = \{b'_j(v_k)\}$, V a set of K output symbols or prototype vectors $\{v_1, ..., v_K\}$, a quantized emission probability function $b'_j(v_{k^*}) = P(k^* = \text{argmin}_k d(o_t, v_k)|q_t = S_j)$, the function $d(x,y)$ a distance function and the constraint $\sum_k b'_j(v_k) = 1$. □

Figure 3.3 shows a graphical representation of the probability space for both discrete and continuous hidden Markov models. For a given observation o_t, we compute the nearest prototype vector v_k (each indicated by a cross in Figure 3.3) to compute the emission probability $b'_j(v_k)$. Because each prototype vector has an associated probability, the nearest prototype vector v_k directly determines the emission probability $b'_j(v_k)$. Typically, 256 prototype vectors v_k are used to model a

state. The advantage of discrete modeling is that the evaluation is fast because the model of B is essentially a table of N states times K output symbols. Disadvantage is that the modeling is coarse due to quantization.

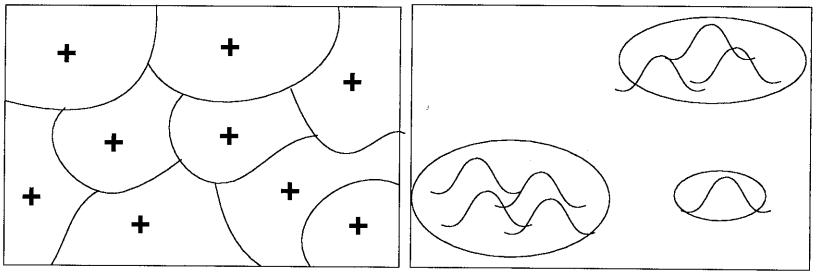


Figure 3.3. Hidden Markov model state space. Left: discrete model where each '+' indicates a prototype vector. Right: continuous model with a number of Gaussian densities.

In handwriting recognition, Bellegarda et al. [1995] compare continuous and discrete hidden Markov models in a character recognition task. The performance is similar but the discrete hidden Markov model uses 15-20 times less processing time. Nathan et al. [1995], Chen et al. [1993], and Bellegarda et al. [1993] explore continuous hidden Markov models. Huang, Ariki & Jack [1990] and Rabiner & Juang [1993] discuss hybrid models.

3.4 Evaluation

This section discusses the algorithms involved in the evaluation and explanation of a hidden Markov model. These are two of three basic problems in the theory of hidden Markov models defined in Section 3.2.

The basic question in *evaluation* is how to compute the probability density $p(O|\lambda)$ for a given observation sequence O and model λ. Subsection 3.4.1 discusses an exact algorithm while Subsection 3.4.2 discusses an approximate algorithm. Both solutions involve the use of state sequences.

Definition 3.6 (State sequence). Given a hidden Markov model λ and an observation signal O containing T observations o_t. Then a sequence of T states $q = (q_1,...,q_T)$ is called a state sequence which describes for each observation $o_t \in O$ the corresponding state q_t at time t. □

To represent the process graphically, the hidden Markov model states and the time steps are combined in a lattice representation as in Figure 3.4. An arbitrary path through the lattice corresponds with a state sequence q.

Definition 3.7 (Observation sequence probability density). Given a hidden Markov model λ and an observation sequence O. Then the probability density of

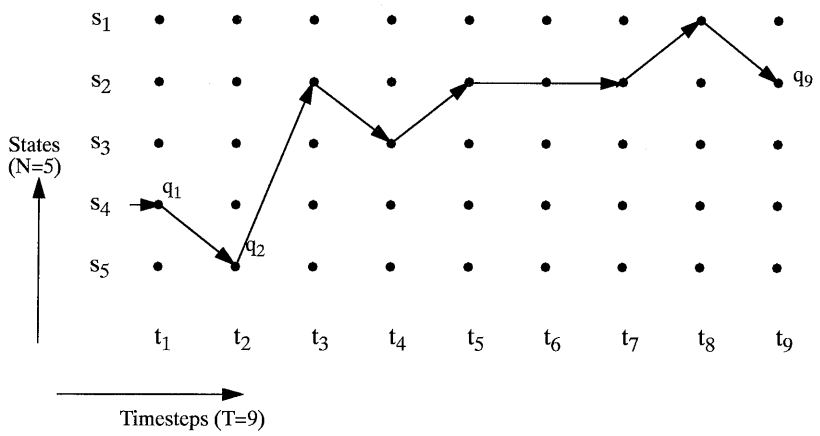


Figure 3.4. One path through the lattice of hidden Markov model classification.

the observation sequence O is defined as $P = p(O|\lambda)$ which is the sum of $p(O, q|\lambda)$ over all possible state sequences q in λ. □

For a given observation sequence O and a hidden Markov model λ this accumulates to

$$\begin{aligned} p(O|\lambda) &= \Sigma_q p(O, q|\lambda) \\ &= \Sigma_q p(O|q, \lambda) P(q|\lambda) \\ &= \Sigma_q [\pi_{q_1} \Pi_{t=1}^{T-1} a_{q_t q_{t+1}}] \cdot [\Pi_{t=1}^{T} b_{q_t}(o_t)]. \end{aligned}$$

Because the number of possible paths through the lattice is N^T, a straightforward computation is prohibitively complex. Therefore, smarter search algorithms are needed to compute $p(O|\lambda)$.

While the exact solution to $p(O|\lambda)$ computes the joint probability density over all paths q, the approximate solution aims at the computation of the best path q^* where the best path is defined as the most likely state sequence q given O and λ.

Formally, this means that the exact algorithm computes

$$p(O|\lambda) = \sum_q p(O, q|\lambda),$$

while the approximate algorithm computes

$$p(O|\lambda) \approx p(O, q^*|\lambda) = \max_q p(O, q|\lambda),$$

where $q^* = \text{argmax}_q$ denotes the best and most likely possible state sequence. The approximate solution is often called the *maximum approximation*.

Efficient computation of the exact and approximate solution is possible if we extract a recurrency relation and apply dynamic programming (DP) [Bellman, 1957; Rabiner & Juang, 1993].

3.4.1 The Forward Algorithm

The *Forward* scoring algorithm [Lee, 1988; Rabiner & Juang, 1993] is an algorithm to solve the exact computation of $p(O|\lambda)$ for a generic hidden Markov model with N states and a given observation sequence O of length T. The auxiliary variable $\alpha_t(j) = p(o_1...o_t, q_t = S_j|\lambda)$ is introduced to specify the probability density of a path of t observations ending in state S_j and having observed $O = o_1,..., o_t$.

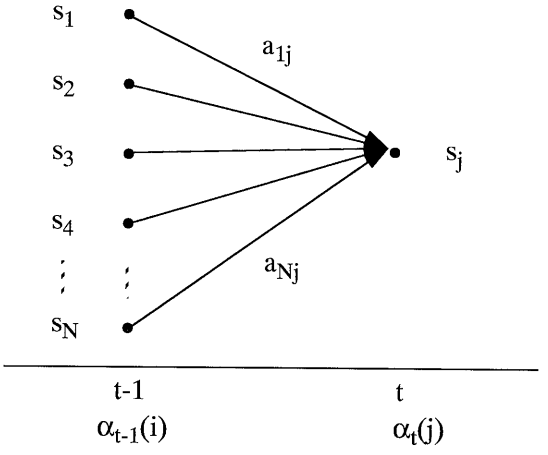


Figure 3.5. Graphical representation of auxiliary variable α.

The variable $\alpha_t(j)$ allows us to define a recurrency relation which describes the new value of α as a combination of all paths from previous states S_i to the new state S_j. This is represented graphically in Figure 3.5 and formally in the recurrency relation in Definition 3.8.

Definition 3.8 (Forward recurrency relation). Given a hidden Markov model λ, with transition matrix A and emission probability densities B, and an observation sequence O. Then $\alpha_t(j)$ is computed at time t from all states S_i at time $t-1$ with the Forward recurrency relation as

$$\alpha_t(j) = [\sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij}] \cdot b_j(o_t).$$

□

The *Forward* algorithm is formalized in Figure 3.6. The core of the algorithm is the recurrency relation in Definition 3.8. The recurrency is initialized in a first step in Figure 3.6 with the initial state probabilities times the emission probability densities of o_1. After processing all observations o_t, the algorithm terminates. The final step is to add the probability densities $\alpha_T(j)$ in every state. This sum gives the total probability density $p(O|\lambda)$.

Step 1. Initialization

$$\alpha_1(j) = \pi_j b_j(o_1)$$

where $1 \leq j \leq N$

Step 2. Recurrency

$$\alpha_t(j) = [\sum_{i=1}^{N} \alpha_{t-1}(i) \cdot a_{ij}] \cdot b_j(o_t)$$

where $1 \leq j \leq N$ and $2 \leq t \leq T$

Step 3. Termination

$$P = p(O|\lambda) = \sum_{j=1}^{N} \alpha_T(j)$$

Figure 3.6. The Forward algorithm.

A counterpart of the Forward recurrency relation and algorithm is the *Backward* recurrency relation and algorithm. The probability density of the tail of O starting in state S_j is $\beta_t(j) = p(o_{t+1},...,o_T|q_t = S_j, \lambda)$ which is efficiently computed based on Definition 3.9. Note that the initial condition of the Backward algorithm is $\beta_T(j) = 1$. This Backward probability density is not used in any evaluation algorithm but is used in Section 3.5 to train a hidden Markov model.

Definition 3.9 (Backward recurrency relation). Given a hidden Markov model λ, with transition matrix A, emission probability densities B, and an observation sequence O. Then $\beta_t(i)$ is computed with a recurrency relation called the Backward recurrency relation from all states S_j at time $t+1$ as

$$\beta_t(i) = [\sum_{j=1}^{N} a_{ij} \cdot b_j(o_{t+1}) \cdot \beta_{t+1}(j)].$$

□

3.4.2 The Viterbi Algorithm

In contrast to the exact solution for $p(O|\lambda)$, we can also compute an approximate solution. The probability density of the best path, i.e., the most probable path, through the lattice in Figure 3.4 is computed for a generic hidden Markov model with N states and a given observation sequence O of length T. An auxiliary variable

$$\alpha_t(j) = \max_{q_1,\cdots,q_{t-1}} p(o_1...o_t, q_1...q_{t-1}, q_t = S_j|\lambda)$$

is introduced which specifies the path with maximum probability of t observations ending in state S_j and leading through the states $q_1,...,q_t$. Note that we use the same variable $\alpha_t(j)$ as in the discussion of the Forward algorithm. The reason is

that we want to emphasize the similarities of the Forward and Viterbi algorithms.

The recurrency relation which defines $\alpha_t(j)$ in Definition 3.10 expresses the new value of α based on the *best* path from all previous states S_i to the new state S_j. Compared with the Forward recurrency in Definition 3.8, only the summation is changed to a maximum operation. Hence the alternative description *maximum approximation* for this relation. In the context of a time-synchronous algorithm which computes a best-path approximation to $p(O|\lambda)$ where all paths end at time t, this is called the Viterbi algorithm as explored by Forney [1973] and shown in Figure 3.7.

Definition 3.10 (Viterbi recurrency relation). Given a hidden Markov model λ, with transition matrix A, emission probability densities B, and an observation sequence O. Then $\alpha_t(j)$ is computed with a recurrency relation called the Viterbi recurrency relation from all states S_i at time $t-1$ as

$$\alpha_t(j) = [\max_{i=1...N} \alpha_{t-1}(i) \cdot a_{ij}] \cdot b_j(o_t).$$

□

The Viterbi algorithm is formalized in Figure 3.7. The core of the algorithm is the recurrency relation in Definition 3.10. The recurrency is initialized in step 1 in Figure 3.7 with the initial state probabilities times the emission probability densities of o_1. This is identical to the Forward algorithm in Figure 3.6. After processing all observations o_t, the algorithm terminates. The final step is to select the state S_j which gives the maximum of the probability densities $P^* = \max_{j=1...N} \alpha_T(j)$. This maximum P^* approximates the probability density $p(O|\lambda)$.

Step 1. Initialization

$$\alpha_1(j) = \pi_j b_j(o_1)$$

where $1 \leq j \leq N$

Step 2. Recurrency

$$\alpha_t(j) = [\max_{i=1...N} \alpha_{t-1}(i) \cdot a_{ij}] \cdot b_j(o_t)$$

where $1 \leq j \leq N$ and $2 \leq t \leq T$

Step 3. Termination

$$P^* = p(O, q^*|\lambda) = \max_{j=1...N} \alpha_T(j)$$

Figure 3.7. The Viterbi algorithm.

To extract the actual best state sequence q^*, i.e., to solve the *explanation* problem (see Section 3.2), the algorithm is extended with an array v per timestep which keeps track of the best states for every timestep t. These extensions are added to the Viterbi algorithm of Figure 3.7 resulting in the Viterbi algorithm with backtrace

administration of Figure 3.8. One extra step is needed to obtain the best state sequence q^* from the best-path administration $v_t(j)$. The extra step is step 4 in Figure 3.8. After the most likely state at time T is found, the time-indices t are traversed backwards to complete the best state sequence q^*.

Step 1. Initialization

$$\alpha_1(j) = \pi_j b_j(o_1)$$

$$v_1(j) = 0$$

where $1 \leq j \leq N$

Step 2. Recurrency

$$\alpha_t(j) = [\max_{i=1..N} \alpha_{t-1}(i) \cdot a_{ij}] \cdot b_j(o_t)$$

$$v_t(j) = \text{argmax}_{i=1..N}[\alpha_{t-1}(i) \cdot a_{ij}]$$

where $1 \leq j \leq N$ and $2 \leq t \leq T$

Step 3. Termination

$$P^* = \max_{j=1..N} \alpha_T(j)$$

$$q_T^* = \text{argmax}_{j=1..N} \alpha_T(j)$$

Step 4. Backtrace

$$q_t^* = v_t(q_{t+1}^*)$$

where $1 \leq t \leq T-1$

Figure 3.8. The Viterbi algorithm plus backtrace administration.

3.4.3 Evaluation enhancements

Two important notes have to be made about the previously discussed algorithms. First, the presented Forward and Viterbi algorithms are suited for a hidden Markov model with a generic structure. If the structure of the hidden Markov model is known beforehand, we can speed up the algorithm by exploiting its structure.

If the structure is a left-to-right model with three previous states as explained in Subsection 3.3.1, then the recurrency relation of the Viterbi algorithm changes to allow only the transitions from three previous states to the current state and assumes the following form

$$\alpha_t(j) = [\max_{i=0,1,2} \alpha_{t-1}(i') \cdot a_{i'j}] \cdot b_j(o_t) \text{ where } i' = \max(1, j-i) \text{ and } j = 1..N.$$

Additionally, the left-to-right model implies the additional constraints that the start-state is $q_1 = S_1$ and the end-state is $q_T = S_N$.

Second, the continuous multiplication of probabilities can result in a floating point underflow. Therefore, the probability density P is often replaced with the

loglikelihood $\tilde{P} = -\log(P)$. Additional advantage is that all multiplications become additions. This leads to Figure 3.9 which shows the Viterbi algorithm from Figure 3.8 plus the adaptations for a left-to-right model and loglikelihood computation.

Step 1. Initialization

$$\tilde{\alpha}_1(j) = \tilde{\pi}_j + \tilde{b}_j(o_1)$$

$$v_1(j) = 0$$

where $1 \leq j \leq N$ and

$$\tilde{\pi}_j = \begin{cases} 0 & \text{if } j = 1 \\ \infty & \text{if } j \neq 1 \end{cases}$$

Step 2. Recurrency

$$\tilde{\alpha}_t(j) = [\min_{i=0,1,2} \tilde{\alpha}_{t-1}(i') + \tilde{a}_{i'j}] + \tilde{b}_j(o_t)$$

$$v_t(j) = \text{argmin}_{i=0,1,2}[\tilde{\alpha}_{t-1}(i') + \tilde{a}_{i'j}]$$

where $1 \leq j \leq N$ and $2 \leq t \leq T$ and $i' = \max(1, j - i)$

Step 3. Termination

$$\tilde{P}^* = \tilde{\alpha}_T(N)$$

$$q_T^* = N$$

Step 4. Backtrace

$$q_t^* = v_t(q_{t+1}^*)$$

where $1 \leq t \leq T - 1$

Figure 3.9. The Viterbi algorithm for left-to-right models using loglikelihood.

3.5 Training

Reliable training of the hidden Markov model is the most difficult of the three basic hidden Markov model problems [Lee, 1988]. Goal of the training is to derive a hidden Markov model which describes the temporal and spectral characteristics of the input signal. This is achieved with a model which maximizes the probability density $p(O|\lambda)$ given an observation sequence. The difficulty of training is due to the fact that there is no exact solution but only an iterative solution to maximize $p(O|\lambda)$. The iterative training is the Maximum Likelihood (ML) training for which the theoretical framework was laid by Baum [1972]. More recent discussions of hidden Markov model training are given by Rabiner & Juang [1993] and Levinson, Rabiner & Sondhi [1983] and Lee [1988].

First, we outline a training system for hidden Markov models in Subsection 3.5.1 to place all components in context. Subsection 3.5.2 summarizes the reestimation of the hidden Markov model. Subsection 3.5.3 and Subsection 3.5.4 provide an exact and approximate solution to the training problem, respectively, similar to the Forward and Viterbi evaluation algorithms. A sketchy proof of the training convergence is given in Subsection 3.5.6.

3.5.1 Training overview

The training process of a hidden Markov model is an iterative process that proceeds along the lines of the Estimation-Maximization (EM) training shown in Figure 3.10. After an initialization of the hidden Markov model, the two step process of

- time-alignment of observations O with the states of S,
- reestimation of model parameters based on the data distribution over the model states,

is repeated until some stop-criterion is met. The training converges to a local optimum. Since the training is data-driven, a database is needed that contains enough samples of each label. In an example by Picone [1990], three to nine iterations were enough to capture most information in the database. Guyon, Makhoul, Schwartz & Vapnik [1996] give a reasonably accurate estimation of the size of training and test database, together with a confidence estimate of the recognizer performance.

First, the hidden Markov model parameters are initialized. Although the training convergence is guaranteed (see Subsection 3.5.6), the training converges to a local optimum only. Therefore, the initial choice of the parameters is important. A reasonable choice is to use a linear time-alignment of the observation sequence O with the states S of model λ [Ney et al., 1994].

Second, the supervised training consisting of time-alignment and parameter reestimation is executed. Subsection 3.5.3 and Subsection 3.5.4 discuss the use of the Forward-Backward and Viterbi algorithm for time-alignment in combination with reestimation for discrete and continuous type hidden Markov model. Subsection 3.5.5 discusses the adaptations to the training, based on the Viterbi algorithm, of the continuous hidden Markov model in order to use mixture densities to model the emission probability densities.

Some authors suggest an optional third step which is corrective training [Picone, 1990; Rabiner & Juang, 1993; Bahl, Brown, De Souza & Mercer, 1988]. Corrective training aims at improving the recognition accuracy by reducing the probability of incorrect hypotheses and increasing the probability of the correct hypotheses.

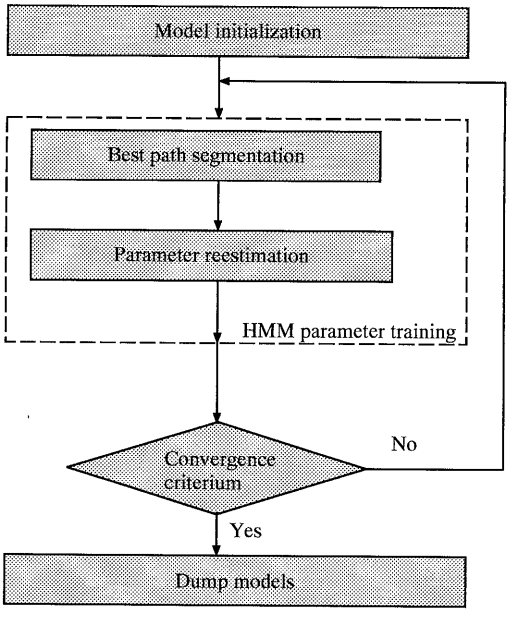


Figure 3.10. Estimation-Maximization training.

3.5.2 Reestimation

The reestimation goal of the training is to reestimate a model λ leading to an improved, more likely model $\bar{\lambda}$. The iterative Estimation-Maximization procedure reestimates the hidden Markov model parameters of λ on each iteration and is formally described as $f : O, \lambda \rightarrow \bar{\lambda}$ where $\bar{P} = p(O|\bar{\lambda}) \geq p(O|\lambda) = P$ and $p(O|\bar{\lambda})$ is maximized. This is one of the three fundamental problems introduced in Section 3.2.

To achieve the above objective, we have to reestimate the hidden Markov model parameters of the model λ based on the observation sequence O. Generalization to a set of observation sequences is straightforward. We have to reestimate for every state in λ the following parameters.

- The transition probability a_{ij} which describes the probability for a transition from state S_i to state S_j.
- The emission probability density $b_j(o_t)$ which describes the probability density of the observation o_t in state S_j.
- The initial state probability π_i which describes the probability that the first state of the best state sequence q^* is state S_i.

We distinguish between two types of time-alignments: an *exact* way and an *approximate* way. This is similar to our discussion of the probability evaluation where the

Forward algorithm provides the exact solution and the Viterbi algorithm gives an approximate one. Likewise, we can use the Forward-Backward (FB) algorithm or Baum-Welch algorithm for exact reestimation and the Viterbi algorithm for the approximate approach.

Because there are two time-alignments (Viterbi and Forward-Backward) and two types of hidden Markov model (discrete and continuous), four possible training combinations are considered. The combination of Viterbi time-alignment with the continuous type hidden Markov model will be used later for recognition and verification.

3.5.3 Forward-Backward training

The following set of auxiliary variables helps to define the reestimation [Levinson, Rabiner & Sondhi, 1983].

Definition 3.11 (Auxiliary functions Forward-Backward training). Given a hidden Markov model $\lambda = (A, B, \pi, S)$, an observation sequence O of length T, the probability density $P = p(O|\lambda)$, and the auxiliary variables $\alpha_t(i)$ and $\beta_{t+1}(j)$. Then we define the following functions.

1. The function $n_{ij}(S)$ denotes the expected number of transitions from state S_i to state S_j for a given O and λ and is given by

$$n_{ij}(S) = \frac{1}{P} \sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j).$$

2. The function $n_i(S)$ denotes the expected number of outgoing transitions from state S_i for a given O and λ and is given by

$$n_i(S) = \sum_{j=1}^{N} n_{ij}(S).$$

3. The function $\delta_k(o_t)$ is only used in the reestimation of discrete hidden Markov models (Definition 3.5) where $V_k = \{o_t | \forall_j d(v_k, o_t) \leq d(v_j, o_t)\}$ and

$$\delta_k(o_t) = \begin{cases} 1 & \text{if } o_t \in V_k \\ 0 & \text{if } o_t \notin V_k \end{cases}.$$

□

This leads to the following definitions of Forward-Backward reestimation formulas.

Definition 3.12 (Forward-Backward reestimation of a_{ij}). Given a hidden Markov model λ and the auxiliary variables from Definition 3.11. Then the

transition probability is reestimated as

$$\begin{aligned}\bar{a}_{ij} &= \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of outgoing transitions from state } S_i} \\ &= \frac{n_{ij}(S)}{n_i(S)}.\end{aligned}$$

□

Definition 3.13 (Forward-Backward reestimation of π_i). Given a hidden Markov model λ and the auxiliary variables from Definition 3.11. Then the initial probability of state S_i is reestimated as

$$\begin{aligned}\bar{\pi}_i &= \text{expected probability of state } S_i \text{ at time } t = 1 \\ &= \tfrac{1}{P}\alpha_1(i)\beta_1(i).\end{aligned}$$

□

We distinguish between two separate reestimation procedures for $b_j(o_t)$ and $b_j'(v_k)$, respectively, because the internal structures of discrete and continuous hidden Markov models are slightly different. The reestimation of discrete hidden Markov models involves the counting of output symbols $o_t \in V_k$. For the continuous case we assume that the emission probability density is modeled using one Gaussian density $b_j(o_t) = \mathcal{G}(o_t, \mu_j, U_j)$ where μ and U are the mean and covariance, respectively, of the density to be reestimated.

Definition 3.14 (Forward-Backward reestimation of $b_j'(v_k)$ (discrete)). Given a discrete hidden Markov model λ and an observation sequence O. Then the emission probability distribution $b_j'(v_k)$ of state S_j is reestimated as

$$\begin{aligned}\bar{b}_j'(v_k) &= \frac{\text{expected number of times in state } S_j \text{ observing } o_t \in V_k \text{ given } O}{\text{expected number of times in state } S_j \text{ given } O} \\ &= \frac{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)\delta_k(o_t)}{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)}.\end{aligned}$$

□

Definition 3.15 (Forward-Backward reestimation of $b_j(o_t)$ (continuous)). Given a continuous hidden Markov model λ with one Gaussian density and an observation sequence O. Then the emission probability density function $b_j(o_t)$ of state S_j is reestimated as

$$\bar{b}_j(o_t) = \mathcal{G}(o_t, \bar{\mu}_j, \bar{U}_j) \text{ where } \begin{cases} \bar{\mu}_j = & \frac{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)o_t}{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)} \\ \bar{U}_j = & \frac{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)(o_t - \bar{\mu}_i)(o_t - \bar{\mu}_i)'}{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)}. \end{cases}$$

□

3.5.4 Viterbi training

Viterbi training is simpler and more efficient than the exact Forward-Backward training presented in the previous subsection. We can take advantage of the fact that the Viterbi algorithm produces a best state sequence q^* with maximum likelihood as presented in Figure 3.9. Because q^* is a time-alignment of O with the states S, q^* tells us exactly which observation o_t is assigned to which state $q_t = S_j$ and which transitions are taken. This reduces the training procedure to efficient administration and counting. The following set of variables helps to explain the Viterbi reestimation [Levinson et al., 1983].

Definition 3.16 (Auxiliary functions Viterbi training). Given a hidden Markov model $\lambda = (A, B, \pi, S)$, an observation sequence O of length T and a state sequence q of length T which contains the observed states $q_t = S_i$ for every o_t. Then we define the following functions.

1. The function $n_{ij}(S)$ denotes the number of transitions from state S_i to state S_j for a given O and λ.
2. The function $n_i(S) = \sum_{j=1}^{N} n_{ij}(S)$ denotes the number of outgoing transitions from state S_i for a given O and λ.
3. The function $m_j(S)$ denotes the number of observations o_t in state S_j.
4. The function $r_i(S)$ equals 1 in the initial state $q_1 = S_i$ and otherwise 0.

□

Definition 3.17 (Viterbi reestimation of a_{ij}). Given a hidden Markov model λ, an observation sequence O, a state sequence q of length T which contains the observed states $q_t = S_i$ for every o_t, and the auxiliary functions from Definition 3.16. Then the transition probability is reestimated as

$$\begin{aligned} \bar{a}_{ij} &= \frac{\text{observed number of transitions from state } S_i \text{ to state } S_j}{\text{observed number of outgoing transitions from state } S_i} \\ &= \frac{n_{ij}(S)}{n_i(S)}. \end{aligned}$$

□

Definition 3.18 (Viterbi reestimation of π_i). Given a hidden Markov model λ, an observation sequence O, a state sequence q of length T which contains the observed states $q_t = S_i$ for every o_t, and the auxiliary variables from Definition 3.16. Then the initial probability of state S_i is reestimated as

$$\begin{aligned} \bar{\pi}_i &= \text{expected probability of state } S_i \text{ at time } t = 1 \\ &= \frac{r_i(S)}{\sum_{i=1}^{N} r_i(S)}. \end{aligned}$$

□

Again, we distinguish between the reestimation of $b_j(o_t)$ for a discrete and continuous hidden Markov model.

Definition 3.19 (Viterbi reestimation of $b'_j(v_k)$ (discrete)). Given a discrete hidden Markov model λ, an observation sequence O, a state sequence q of length T which contains the observed states $q_t = S_j$ for every o_t, and the function $\delta_k(o_t)$ from Definition 3.11. Then the emission probability $b'_j(v_k)$ of state S_j is reestimated as

$$\begin{aligned}\bar{b}'_j(v_k) &= \frac{\text{observed number of times in state } S_j \text{ and observing observation } o_t \in V_k}{\text{observed number of times in state } S_j} \\ &= \frac{1}{m_j(S)} \sum_{t=1}^{T} \delta_k(o_t).\end{aligned}$$

□

For the continuous case, one Gaussian density $b_j(o_t) = \mathcal{G}(o_t, \mu_j, U_j)$ is assumed where μ and U are the mean and covariance, respectively, of the density to be reestimated.

Definition 3.20 (Viterbi reestimation of $b_j(o_t)$ (continuous)). Given a continuous hidden Markov model λ with one Gaussian density, an observation sequence O, and a state sequence q of length T which contains the observed states $q_t = S_j$ for every o_t. Then the emission probability density function $b_j(o_t)$ of state S_j is reestimated as

$$\bar{b}_j(o_t) = \mathcal{G}(o_t, \bar{\mu}_j, \bar{U}_j) \text{ where } \begin{cases} \bar{\mu}_j = & \frac{1}{m_j(S)} \sum_{t=1 \wedge \{q_t = S_j\}}^{T} o_t \\ \bar{U}_j = & \frac{1}{m_j(S)} \sum_{t=1 \wedge \{q_t = S_j\}}^{T} \}(o_t - \bar{\mu}_j)(o_t - \bar{\mu}_j)'. \end{cases}$$

□

3.5.5 Mixture densities

In this thesis we will use the Viterbi algorithm for both training and recognition. Because hidden Markov model states often model several aspects of the data, e.g., allophones and allographs, one Gaussian density is only a crude approximation of the real distribution of observations in state S_j. Therefore, we employ mixture densities. The emission probability density function $b_j(o_t)$ is modeled as a mixture of K Gaussian functions $b_j(o_t) = \sum_k w_{jk} \mathcal{G}(o_t, \bar{\mu}_{jk}, \bar{U}_{jk})$ with $\sum_k w_{jk} = 1$. In the context of the Viterbi algorithm, we use

$$b_j(o_t) = \max_k w_{jk} \mathcal{G}(o_t, \bar{\mu}_{jk}, \bar{U}_{jk})$$

as the maximum approximation.

To train this mixture density, we have to refine the Viterbi reestimation. While the time-alignment remains the same, the estimation of $b_j(o_t)$ needs a clustering step to determine K clusters to be modeled with Gaussian densities [Rabiner & Juang, 1993] [Ney et al., 1994]. The k-Means clustering algorithm [Duda & Hart,

1973] is often employed to determine the cluster centroids. The mixture gains w_{jk} are estimated by counting the number of elements in each cluster and computing the relative contribution.

Given a clustering of the observations o_t into K sets $\mathcal{O}_k$ and the auxiliary function $m_{jk}(S)$ which denotes the number of observations o_t in density k of state S_j where $m_j(S) = \sum_k m_{jk}(S)$, this leads to the following reestimation of the emission probability density.

Definition 3.21 (Viterbi mixture density reestimation of $b_j(o_t)$ (continuous)). Given a continuous hidden Markov model λ, a mixture of Gaussian densities describing the emission probability densities $b_j(o_t)$, an observation sequence O, and a state sequence q. Then the emission probability density function $b_j(o_t)$ in state S_j is reestimated as

$$\bar{b}_j(o_t) = \max_k \; w_{jk} \cdot \mathcal{G}(o_t, \bar{\mu}_{jk}, \bar{U}_{jk})$$

$$\text{where} \begin{cases} \bar{\mu}_{jk} = & \frac{1}{m_{jk}(q)} \sum_{t=1 \wedge \{q_t = S_j\} \wedge \{o_t \in \mathcal{O}_k\}}^{T} o_t \\ \bar{U}_{jk} = & \frac{1}{m_{jk}(q)} \sum_{t=1 \wedge \{q_t = S_j\} \wedge \{o_t \in \mathcal{O}_k\}}^{T} (o_t - \bar{\mu}_{jk})(o_t - \bar{\mu}_{jk})'. \end{cases}$$

□

3.5.6 Training convergence

In this subsection, we outline a 'proof-of-convergence' of the hidden Markov model training and reestimation process. The original proof is from Baum [1972]. Here, we follow the proof as explained by Levinson, Rabiner & Sondhi [1983].

Given is a model $\lambda = (A, B, \pi, S)$ with probability density $P = p(O|\lambda)$ for a given observation sequence O. Then the reestimated model is $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi}, S)$ with probability density $\bar{P} = p(O|\bar{\lambda})$. Our goal is to prove that the condition

$$p(O|\bar{\lambda}) \geq p(O|\lambda)$$

holds.

First, we define a set of auxiliary variables.

Definition 3.22 (Convergence proof auxiliary variables). Given a hidden Markov model λ, a reestimated model $\bar{\lambda}$, an observation sequence O, and a state sequence q. Then we define the following variables.

1. $u_q = p(O, q|\lambda)$, where u_q denotes the probability of the path q and an observation sequence O for a given model λ.
2. $v_q = p(O, q|\bar{\lambda})$, where v_q is similar to u_q except that it is the probability given the model $\bar{\lambda}$.
3. $\sum_q u_q = P$ and $\sum_q v_q = \bar{P}$ as explained in Subsection 3.5.2.

□

Next, we present two lemmas as formulated by Levinson, Rabiner & Sondhi [1983] which are needed to rewrite and reformulate the problem. Lemma 3.1 is also known as Jensen's inequality and is based on the concavity of the log function.

Lemma 3.1 (Log Lemma). *Let u_q be positive real numbers and v_q be non-negative real numbers such that $\sum_q v_q > 0$. Then from the concavity of the log function it follows that*

$$\log\left(\frac{\sum v_q}{\sum u_q}\right) \geq \frac{1}{\sum_q u_q} \cdot [\sum_q (u_q \log v_q - u_q \log u_q)].$$

□

Lemma 3.2 (Maximum). *If $c_i > 0$ holds, where $i = 1,...,N$, then subject to the constraint $\sum_i x_i = 1$, the function*

$$F(X) = \sum_i c_i \log x_i$$

attains its unique global maximum when

$$x_i = \frac{c_i}{\sum_i c_i}.$$

□

Then the proof starts with rewriting our problem $p(O|\bar{\lambda}) \geq p(O|\lambda)$ to

$$\log\left(\frac{p(O|\bar{\lambda})}{p(O|\lambda)}\right) \geq 0.$$

The problem is now in a form that allows the application of Lemma 3.1 and the problem is rewritten to

$$\log\left(\frac{p(O|\bar{\lambda})}{p(O|\lambda)}\right) \geq \frac{1}{p(O|\lambda)} \cdot [Q(\lambda, \bar{\lambda}) - Q(\lambda, \lambda)]$$

where the auxiliary function Q equals $Q(\lambda, \bar{\lambda}) = \sum_q u_q \log v_q$.

If we can show that $Q(\lambda, \bar{\lambda}) - Q(\lambda, \lambda)$ is always positive then the property is proven. Because we cannot prove directly $Q(\lambda, \bar{\lambda}) - Q(\lambda, \lambda) \geq 0$, we reshape this problem to a search for the maximum of the auxiliary function Q. If we can prove that $Q(\lambda, \bar{\lambda})$ has a maximum and we can compute this maximum then the proof is complete. This is the step where Lemma 3.2 is used. Lemma 3.2 tells us that Q has a unique maximum. Rewriting Q in three independent terms, which describe the initial, the transition, and the emission probability densities, and the application of Lemma 3.2 results in three expressions which describe the global maximum of Q. These three expressions correspond with the reestimation formula's of π, b

and a of the Forward-Backward training algorithm discussed in Subsection 3.5.3. Because the reestimation is a maximization, $Q(\lambda, \bar{\lambda})$ eventually reaches its unique maximum. Further improvement of the model parameters $\bar{\lambda}$ is not possible and the training stops.

3.6 Advanced topics

In the previous sections, we discussed the fundamental properties and algorithms in pattern recognition based on hidden Markov models. This section discusses some advanced techniques to improve and extend the classification. The topics to improve classification performance are the integration of relevant context knowledge into the hidden Markov models and language models. We extend the classification techniques with beam search which elaborates on the Viterbi algorithm.

First, we model the contextual effects of handwriting with contextual hidden Markov models in Subsection 3.6.1. Language structure is modeled with a statistical language model in Subsection 3.6.2. We present an efficient search process which integrates the knowledge sources of hidden Markov model and language model to recognize labels and label sequences in Subsection 3.6.3.

3.6.1 Contextual models

In the previous sections, we have discussed how to train and recognize hidden Markov models. All models represent one specific class of labels which are insensitive to their context. In this subsection, we introduce context-dependent models.

As shown in the previous chapter, the data to model is sometimes context dependent, e.g., the shape of written characters depends on the shape of the neighboring characters. Therefore, it is useful to make a specialized model for a pattern *given its context*. These context-dependent models, as opposed to the previously discussed context-independent models, provide more accurate models.

Definition 3.23 (Context-independent models). Given a hidden Markov model λ and a label c. Then a context-independent or context-free model is a model λ_c that depends only on the items representing label c. □

Definition 3.24 (Context-dependent models). Given a hidden Markov model λ, a label c and two additional labels l and r. Then a context-dependent or contextual model is a model $\lambda_{\{l\}c\{r\}}$ that depends on the items representing label c given the context of the labels l and r where l is a left context label and r a right context label. If a left or right context is missing then we obtain the models $\lambda_{\{\$\}c\{r\}}$ and $\lambda_{\{l\}c\{\$\}}$, respectively. Note that $\lambda_{\{\$\}c\{\$\}} = \lambda_c$. □

In the context of speech and handwriting recognition, we refer to models $\lambda_{\{l\}c\{r\}}$ as *triphones* and *trigraphs*, respectively. Models with left or right context only are

referred to as biphones and bigraphs.

Since the early work on triphones in speech recognition by Schwartz, Chow, Roucos, Krasner & Makhoul [1984], the use of triphones is well established now [Lee, 1988; Ney et al., 1994]. Schwartz et al. [1984] demonstrated that the use of contextual models can lead to a decrease in word error rate (WER) of up to 50% over context-independent models. The use of trigraphs in handwriting recognition based on hidden Markov models is much less explored. One of the few works on this topic is the study by Starner, Makhoul, Schwartz & Chou [1994].

An example of the difference between context-independent and contextual models for the word 'there' is given in Table 3.1.

Table 3.1. The word 'there' modeled with different types of models.

Model type	**Model implementation**
Context independent	$\lambda_t, \lambda_h, \lambda_e, \lambda_r, \lambda_e$
Right context	$\lambda_{\{\$\}t\{h\}}, \lambda_{\{\$\}h\{e\}}, \lambda_{\{\$\}e\{r\}}, \lambda_{\{\$\}r\{e\}}, \lambda_{\{\$\}e\{\$\}}$
Left context	$\lambda_{\{\$\}t\{\$\}}, \lambda_{\{t\}h\{\$\}}, \lambda_{\{h\}e\{\$\}}, \lambda_{\{e\}r\{\$\}}, \lambda_{\{r\}e\{\$\}}$
Left and right context	$\lambda_{\{\$\}t\{h\}}, \lambda_{\{t\}h\{e\}}, \lambda_{\{h\}e\{r\}}, \lambda_{\{e\}r\{e\}}, \lambda_{\{r\}e\{\$\}}$

The use of contextual models introduces an important tradeoff. The more models we use to describe the items, the less data will be available to train the models because the data has to be divided over many classes. So, if we have a large number of models, they cannot be trained accurately because of a lack of data. Given an alphabet with 26 symbols, the number of possible contextual models is $26^3 = 17576$ but, in practice, it will be possible to train only a fraction of the models due to the aforementioned tradeoff. The number of contextual models that can be effectively handled is usually between 100 and 1000. One approach is to train each model with at least 50 observations.

So far, we have defined the context of a contextual model in terms of labels from an alphabet. Alternatively, context categories or label classes can be explored. An example is a model with category of labels as in $\lambda_{\{g(l)\}c\{g(r)\}}$ where g is a function $g : \Omega \rightarrow \Omega'$ with $\Omega' \subset \Omega$. Other approaches include the automatic clustering of contextual models.

3.6.2 Language Model

The purpose of language modeling is to exploit language structure in order to minimize the ambiguity inherent in spoken and written text. In other words, a language model maximizes the available contextual knowledge and therefore most likely will improve the recognition performance. In case of a small-vocabulary recognition system, this can be achieved with a dialog structure. In the context of this thesis, a large-vocabulary recognition system will rely on a statistical language model and not on a formal grammar. The reason for this is that the statistical language model

is easily integrated with the hidden Markov model and the search algorithms. In the remainder of this subsection, we discuss the structure of such a statistical language model.

Definition 3.25 (Statistical language model). Given a word sequence $W = (w_1,...,w_Q)$ of length Q and the history $w_{i-1}...w_1$ of every word w_i. Then a statistical language model contains the probabilities $P(w_i|w_{i-1}...w_1)$ for every word w_i and computes the probability of the word sequence W as

$$P(W) = \prod_{i=1}^{Q} P(w_i|w_{i-1}...w_1).$$

□

Because infinite-length word histories are impractical, we limit the length of the history $w_{i-1}...w_1$ of word w_i to M. Such an approximate statistical language model is called an M-gram language model. The terms unigram, bigram and trigram are used to determine M-gram models where M is 1,2, and 3, respectively. A zerogram model uses word probabilities which are all the same, i.e., $P(w_i) = \frac{1}{Q}$. A unigram model assumes that $P(w_i) = P(w_i|w_{i-1}...w_1)$.

Definition 3.26 (M-gram language model). Given a word sequence $W = (w_1,...,w_Q)$ of length Q where each word w_i has a history of $M-1$ words with $M \geq 2$. Then an M-gram language model contains the probabilities $P(w_i|w_{i-1}...w_{\max(1,i-M+1)})$ for every word w_i and computes the probability of a word sequence W as

$$P(W) = \prod_{i=1}^{Q} P(w_i|w_{i-1}...w_{\max(1,i-M+1)}).$$

□

So far, we assumed that a text sequence is a sequence of words. Although this remains correct through this thesis, we can also assume that a text sequence is a sequence of characters. Both views are valid and constitute the same word sequence W. However, the probability of a character and word sequence is different which leads to a different probability $P(W)$. Therefore, we introduce word and character-based language models.

Definition 3.27 (Language model type). Given a word sequence $W = (w_1,...,w_i,...,w_{Q_w})$ of length Q_w where each word $w_i = (c_1,...,c_{l_i})$ is a sequence of l_i characters, a total number of characters in the text which equals $Q_c = \sum_{i=1}^{Q_w} l_i$, a history size $M-1$ where $M \geq 2$ for each word w_i. Then an M-gram, word-based language model contains the probabilities $P(w_i|w_{i-1}...w_{\max(1,i-M+1)})$

for every word w_i and computes the probability of a word sequence W as

$$P(W) = \prod_{i=1}^{Q_w} P(w_i | w_{i-1} ... w_{\max(1,i-M+1)})$$

while an M-gram, character-based language model contains the probabilities $P(c_i | c_{i-1} ... c_{\max(1,i-M+1)})$ for every character c_i and computes the probability of a word sequence W as

$$P(W) = \prod_{i=1}^{Q_c} P(c_i | c_{i-1} ... c_{\max(1,i-M+1)}).$$

□

In a large text corpus, word sequences are counted and used to estimate probabilities as in

$$P(w_i | W') = \frac{F(w_i, W')}{F(W')}$$

where the word sequence history equals $W' = (w_{i-1} ... w_{i-N+1})$ and the function F denotes the word sequence frequency.

The actual process of estimating the probabilities is more complex because some word frequencies are about zero. More details are given by Ney et al. [1994] for word-based language modeling while Guyon & Pereira [1995] present a variable length character-based language model.

The complexity of a language model is quantified in terms of *entropy* and *perplexity*. In general terms, the information content of a message is described by its entropy and is computed as demonstrated by Shannon [1951], Witten & Bell [1990] and Rabiner & Juang [1993]. The entropy is also the average number of bits to describe a label such as a character or word.

Definition 3.28 (Entropy). Given a word sequence $W = (w_1, ..., w_Q)$ of length Q and the probability of every word w_i given by a statistical language model. Then we compute the entropy of a sequence of words W as

$$H = -\sum_{i=1}^{Q} P(w_i) \log_2 P(w_i)$$

while an approximation is computed as

$$H = -(\frac{1}{Q}) \sum_{i=1}^{Q} \log_2 P(w_i)$$

for a typical long text sample. □

The perplexity can be interpreted as the average number of possible words following the current word. This can be formalized as follows.

Definition 3.29 (Perplexity). Given a word sequence $W = (w_1,...,w_Q)$ of length Q and a language model which defines the probability of the word sequence $P(W)$. Then the perplexity PP of a sequence of words W is defined as

$$PP = P(W)^{-\frac{1}{Q}}.$$

□

Perplexity and entropy of a text source are related as $PP = 2^H$. We combine Definition 3.26 with the conditional probability $P(w_i) = P(w_i|w_{i-1}...w_{\max(1,i-M+1)})$ to compute the perplexity and entropy of a word sequence W using an M-gram language model as $H = \log_2 PP = -\frac{1}{Q}\sum_{i=1}^{Q}\log_2 P(w_i|w_{i-1}...w_{\max(1,i-M+1)})$.

In the study of Witten & Bell [1990], the entropy and perplexity of the English language are discussed on the basis of the example of the Brown corpus. This corpus contains about one million words from 500 sources. It is written using 94 character symbols and a vocabulary of about 100,000 unique words. The entropy H and perplexity PP for some language models based on characters and words are given in Table 3.2.

Table 3.2. Entropies and perplexities of language models based on characters and words, respectively, of the Brown corpus for various M-gram models according to Witten & Bell 1990.

Language model type	**Measure**	**M-gram**				
		0	**1**	**2**	**3**	**4**
Character	H	6.55	4.47	3.59	2.92	2.33
	PP	94	22.16	12.88	7.57	5.03
Word	H	16.61	11.47	6.06	2.01	-
	PP	100K	2836.70	66.71	4.03	-

To put this into perspective, a handwriting recognition task based on a trigram character model has a perplexity of 7.57, i.e., given a current character, there is on average a choice of 7.57 characters for the character to follow. In case of the trigram word model, there are only 4.03 word candidates that can follow the current word, which is a significant reduction compared to the 100,000 possibilities for the zero-gram model which assumes equal probabilities for all words. Witten & Bell [1990] also note that the *character* based entropy for the same trigram *word* model is only 0.34, which corresponds to a character-based perplexity of 1.27 compared with 7.57 for the purely character-based model. This clearly shows that a word-based language model outperforms a character-based model at the cost of fixed vocabulary. A smaller perplexity typically leads to a better recognition performance. Therefore, a language model with low perplexity is preferable.

3.6.3 Beam search

The Viterbi algorithm described in Figure 3.9 handles the evaluation of a sequence of items representing *one* label given *one* model which results in $p(O|\lambda)$. This has to be generalized to a *multi* class *isolated* and *connected* class classification problem. For this we need the following definitions.

Definition 3.30 (Isolated class classification). Given a sequence of items $O = (o_1,...,o_T)$ and a set of N hidden Markov models $\Lambda = \{\lambda_1,...,\lambda_N\}$ representing M labels $\Omega = \{\omega_1,...,\omega_M\}$, determine the most likely label $\omega^* = \omega_{i^*}$ where $i^* = \text{argmax}_i p(O|\lambda_i)$. □

Definition 3.31 (Connected class classification). Given a sequence of items $O = (o_1,...,o_T)$ and a set of N hidden Markov models $\Lambda = \{\lambda_1,...,\lambda_N\}$ representing M labels $\Omega = \{\omega_1,...,\omega_M\}$, determine the most likely label sequence $\omega^* = (\omega_1,...,\omega_L)$ of unknown length L. □

The straightforward approach to isolated class classification is to handle all the models one by one to determine the best matching model. Although perfectly feasible, this is not the most efficient solution. For example, the number of states that have to be evaluated per timestep for a vocabulary of 20,000 words containing 50 states each is $O(10^6)$. For both isolated and connected word recognition, the one-stage *beam search* algorithm as proposed by Ney [1984] is an effective and efficient solution.

The approach with a one-stage beam search has a number of desirable properties. First, it enables us to handle input of arbitrary length representing either one or more labels. In handwriting recognition, this corresponds to the ability to recognize characters, words, and sentences using the same algorithm. Secondly, the beam search algorithm computes the best possible solution over all models while evaluating only a fraction of all states. Thirdly, a time-synchronous evaluation is possible, i.e., the approach is suitable for real-time recognition where the search proceeds every time new data becomes available. Fourthly, no explicit segmentation into characters is required. On the contrary, the segmentation of words and sentences into characters is a side-effect of the recognition. The process is illustrated in Figure 3.11.

It should be noted that beam search is a generic technique which is applied not only in speech or handwriting recognition but also in scheduling [Ow & Morton, 1988]. Alternative search algorithms include the best-first A^* search as presented by Paul [1991] and Level-Building as presented by Rabiner & Juang [1993], which is compared with one-stage beamsearch by Ney [1984]. Compared to three other algorithms, the one-stage beamsearch requires less storage and fewer computational resources [Ney, 1984].

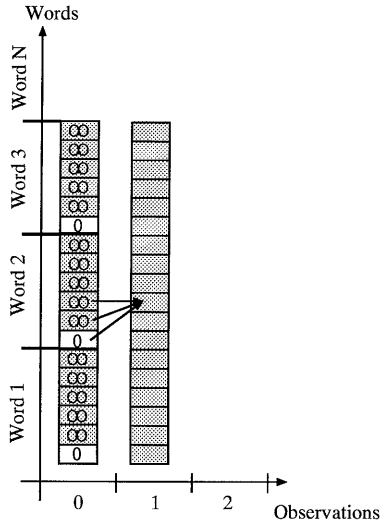


Figure 3.11. One-stage beam search.

Other search algorithms in connected handwriting recognition are often two-stage approaches like the algorithms presented by Weissman et al. [1994] and Schenkel et al. [1994] who construct a labeled graph where the labels are characters detected by a time delay neural network. The graph is searched with the aid of the Viterbi algorithm. Similarly, Manke & Bodenhausen [1994] compute the best path over the outputs of a time delay neural network using the Viterbi algorithm. A two-stage search where the first stage is a preselection of labels is presented by Nathan et al. [1995].

Isolated class recognition

Based on Definition 3.30 and the Viterbi algorithm in Figure 3.9, a beam search algorithm is defined for a set of Q words $W = \{w_1,...,w_k,...,w_Q\}$ where each word w_k contains N_k states. States and transitions are assumed to be word-specific. The best score in a timestep t in word k is denoted as $\alpha_t^*(j,k)$ where k is a word index and j a state index. The width of the beam is determined by the threshold τ. All states with scores lower than $\alpha_{t-1}^*(j,k)+\tau$ are active at time t and expanded, while all other states are pruned. Because the pruning threshold depends on the best values in the beam at time t, the beam is self-focusing. The recognized word has index k^*. Left-to-right hidden Markov models are assumed.

The above algorithm suggests that we compute loglikelihoods for all states of all words at every timestep t. The obvious improvement is to compute new loglikelihood values only for the active states (whose score is smaller than infinity)

Step 1. Initialization

$$\alpha_1(j,k) = \begin{cases} -\log(b_{jk}(o_1)) & \text{if } j = 1 \\ \infty & \text{if } j \neq 1 \end{cases}$$

where $1 \leq k \leq Q$ and $1 \leq j \leq N_k$

Step 2. Recurrency

$$\forall t, j, k \; \alpha_t(j,k) = \begin{cases} v = [\min_{i'} \alpha_{t-1}(i',k) - \log a_{i'jk}] - \log b_{jk}(o_t) & \text{if } v \leq \alpha^*_{t-1} + \tau \\ v = \infty & \text{if } v > \alpha^*_{t-1} + \tau \end{cases}$$

where $i' = \{\max(1,j), \max(1,j-1), \max(1,j-2)\}$ and $2 \leq t \leq T$

Step 3. Termination $P^* = \min_k \alpha_T(N_k,k)$ and $k^* = \text{argmin}_k \alpha_T(N_k,k)$

Figure 3.12. The beam search algorithm for isolated class recognition using left-to-right models and loglikelihood computation.

and the states that can be reached from the active states.

Another improvement is to detect duplicate states and compute a new value only once for every state. All words are character strings modeled with a sequence of character models and states. Such a search space layout is called a *linear dictionary* as opposed to a *tree dictionary* which efficiently reduces the number of duplicate states by sharing the prefixes of dictionary words.

Connected class recognition

Based on Definition 3.31 and the Viterbi algorithm in Figure 3.9, we design another variation of beam search which is also applicable to the recognition of label sequences. This approach needs a number of extensions compared to the algorithm of Figure 3.12.

First, one extra state is used at the beginning of each word. This *language state* is initialized at every time step with the probability of the language model for the word in question. Next, we have to track the best word hypothesis $k^*(t)$ in addition to the best state hypothesis $\alpha^*_t(j,k)$. The best word at time t serves as a reference for the language model. Finally, the best word at time t is tracked and stored in a traceback array, together with a pointer to its predecessor and the current time t. This allows the tracking of the best recognized word sequence at any time t. In algorithmic form this leads to the algorithm in Figure 3.13, where all likelihoods are again expressed in $-\log(P)$. Note that we assumed a bigram language model and included only the essential scoring but not the traceback administration for word sequence recovery [Ney, 1984].

Step 1. Initialization

$$\alpha_1(j,k) = \begin{cases} 0 & \text{if } j = 0 \\ \infty & \text{if } j \neq 0 \end{cases}$$

where $1 \leq k \leq Q$ and $0 \leq j \leq N_k$

Step 2. Recurrency

$$\forall t, j, k \quad \begin{array}{ll} \alpha_t(0,k) & = -\log p(w_k | w_{k^*(t-1)}) \\ \alpha_t(j,k) & = \begin{cases} v = [\min_{i'} \alpha_{t-1}(i',k) - \log a_{i'jk}] - \log b_{jk}(o_t) & \text{if } v \leq \alpha^*_{t-1} + \tau \\ v = \infty & \text{if } v > \alpha^*_{t-1} + \tau \end{cases} \end{array}$$

where $i' = \{\max(0,j), \max(0,j-1), \max(0,j-2)$ and $2 \leq t \leq T$ and $0 \leq j \leq N_k$

Step 3. Termination $P^* = \min_k \alpha_T(N_k,k)$ and $k^* = \text{argmin}_k \alpha_T(N_k,k)$

Figure 3.13. The beam search algorithm for connected class recognition.

3.7 Problem Definition

In the previous section, we have presented the theoretical framework of handwriting recognition and verification on the basis of hidden Markov models using a transformation of input patterns, represented by items, to class labels. This section defines the exact type of labels that are used and we also formally define the problem of handwriting recognition and verification. For this we need the following definitions.

Definition 3.32 (Alphabet). An alphabet $\mathcal{A}$ is a subset of ASCII symbols. The most frequently used alphabets are

- $\mathcal{A}_{\text{digits}} = \{0,1,...,9\}$
- $\mathcal{A}_{\text{lower}} = \{a,b,...,z\}$
- $\mathcal{A}_{\text{upper}} = \{A,B,...,Z\}$

A text $\mathcal{T}$ is defined as a sequence of alphabet symbols $\mathcal{T} \in \mathcal{A}^+$ where $\mathcal{A}^+$ represents the label sequences of non-zero length. □

Definition 3.33 (Binary decision). A binary decision results in a 'yes' or 'no' answer represented by $b \in B = \{0,1\}$. □

Based on these label definitions, we define the recognition and verification of input items.

Definition 3.34 (Handwriting recognition). Given an alphabet $\mathcal{A}$ represented by a set of C labels $\omega = \{\omega_1,...,\omega_C\}$ where $\omega_i \in \mathcal{A}$ and each label with model λ_i, and

a sequence of input items $X = x_1, x_2, ..., x_T$. Then we define handwriting recognition as the classification ω^* of X where ω^* is the most likely label sequence $a^* = (a_1, ..., a_l)$ of length l with variable l, given by $a^* = \text{argmax}_\omega p(\omega|X) = (a_1, ..., a_l)$ where $a^* \in \mathcal{A}^+$ and $p(a^*|X)$ is maximal. □

Note that the definition of handwriting recognition covers both isolated and connected class recognition. The recognition of a label string of length one is an isolated class recognition problem. In contrast, the recognition of a label string with length more than one is a connected class recognition problem. Finally, we define the problem of handwriting verification as follows.

Definition 3.35 (Handwriting verification). Given an individual i, a piece of handwriting described with a model λ_i, a threshold τ_i, a binary decision which is either 'reject' or 'accept', and a sequence of input items $X = x_1, x_2, ..., x_T$. Then we define handwriting verification as a binary decision problem where the input is accepted if the condition $-\log p(X|\lambda_i) \leq \tau_i$ holds. □

4

Handwriting recognition

In this chapter we discuss solution strategies for the handwriting recognition problem presented in Definition 3.34 where we defined handwriting recognition as a stochastic classification problem which transforms a sequence of items to one or more labels. To transform the formal definition into a handwriting recognition system, we have to make the labels, model and items more concrete. In the following chapters, we assume that labels correspond with characters. As defined in Definition 3.32, these characters are part of an alphabet $\mathcal{A}$. The recognition of isolated characters concentrates on digit, uppercase and lowercase labels. The recognition of label strings, such as words and sentences, concentrates on strings of lowercase characters.

In the context of hidden Markov models, we refer to the items in Definition 3.34 as observations which are represented by feature vectors. We study the structure of the representation in Section 4.1 with special attention on how to integrate handwriting-specific and contextual knowledge. A hidden Markov model is chosen as a model for each character. Several model structures and handwriting-specific modifications are discussed in Section 4.2. The algorithmic framework which performs the transformation of feature vectors to characters and character sequences is explained in Section 4.3. We conclude the chapter which the experimental framework in Section 4.4.

4.1 Representation

A crucial part of a hidden Markov model classifier is the representation which is the subject of discussion in this section. To start with, the determination of handwriting parts to compute feature vectors from is explained in Subsection 4.1.1 and Subsection 4.1.2. This is followed by the writing size (in)dependence of the handwriting parts. Next, we discuss the features of the baseline representation in Subsection 4.1.3. The remainder of the section discusses the modeling framework of handwriting-specific knowledge with additional features in Subsection 4.1.4, the realization of a number of contextual features in Subsection 4.1.5 and the application of linear discriminant analysis in Subsection 4.1.6.

4.1.1 Handwriting variability

In Chapter 2 we argued that handwriting cannot be reproduced 100%. The variability of handwriting signals depends on the writer and the digitizer and has to be compensated either through normalization or by means of a model. In this subsection, we discuss the alternatives and choices.

In the beginning of this chapter, we made Definition 3.34 more concrete and have chosen to model characters. Apart from the various character shapes, which are modeled by the hidden Markov model, the three main sources of variability in handwriting signals are writing speed, writing size, and, sampling rate.

Because a hidden Markov model is known to compensate the variability of a single variable well if the data source obeys the properties of a Markov process [Levinson & Roe, 1990], at least two of the three variability sources have to be compensated. Careful normalization is important because coarse heuristics can introduce errors which cannot be compensated for by the hidden Markov model. Note that speech recognition systems based on hidden Markov models model the acoustic variability plus the speaking speed.

To find a normalization approach which is best suited to work with a handwriting recognition system based on hidden Markov models, we take a look at some example approaches in the literature. In the context of character recognition, Guyon, Albrecht, LeCun, Denker & Hubbard [1991] found a considerable degree of writing speed variation, which they normalized using a spatially equidistant representation that left only the writing size as variability source. Schomaker & Plamondon [1990] segment the handwriting on the basis of velocity inversion points, which result in pieces of handwriting independent of any of the three variability sources. In the context of word recognition, Bellegarda, Nathan, Nahamoo & Bellegarda [1993] use spatial subsampling to obtain a representation independent of writing speed and tablet speed. Beigi et al. [1994] and Schenkel et al. [1995] show how to compensate writing size for handwritten words.

Given the example normalizations from the literature, this thesis investigates two promising normalization approaches. The first approach uses spatial subsampling to normalize the writing speed and sample rate of the handwriting. This leaves the writing size to be normalized. The second approach uses velocity inversion points which eliminates all three variability sources.

4.1.2 Frames and segments

Before we discuss the features in a feature vector, we have to define how to group sampled coordinates into blocks that we use to compute a feature vector. We define two alternative approaches.

In the case of *segments*, the block boundaries are defined by the condition that the vertical handwriting speed is zero ($v_y = 0$). This block definition corresponds to the stroke definition introduced in Section 2.2. In Subsection 2.2.2, it is argued that the location of these boundary points within a character is invariant with respect to the handwriting size. In combination with features which are independent of writing size, this results in a complete representation independent of writing size. In contrast, a *frame* consists of a fixed number of consecutive, spatially resampled points. The pen trajectory has to be resampled to obtain equidistant points and thus compensate for writing speed variations.

Note that the segments improve the scalability of the handwriting which makes an explicit size normalization, as demonstrated by Nathan et al. [1995] , Weissman et al. [1994] and Schenkel et al. [1995] redundant. The explicit size normalization is often based on the detection of the bounding box of the written word followed by a computation to derive the word body. This explicit size normalization is very suitable for isolated word recognition tasks in which all input is available before normalization and recognition is done after writing. For sentence recognition where writing and recognition take place simultaneously to achieve real-time recognition, the explicit size normalization is less suitable. As an alternative we investigate size-independent segments for character and word type input.

A slightly different definition of frames is used in the context of character recognition. This variation is called 'size-independent frames' or 'si-frames' to distinguish them from the standard frames. The difference is that we compute the stepsize between equidistant points in single characters to obtain 100 points per characters. This approach leads to a fixed number of frames per character regardless of its size.

4.1.3 Feature selection

In the previous subsection we discussed alternatives to the grouping of coordinates, i.e., the sizes and boundaries of parts of handwriting. The next question is how to transform the parts of handwriting into features.

The easiest way of transforming the digitizer coordinate stream to a feature vector stream is producing one feature vector per sample as proposed by Seni, Srihari & Nasrabadi [1994] and Starner, Makhoul, Schwartz & Chou [1994]. Seni et al. [1994] used four features per vector while Starner et al. [1994] used only two features containing the angles relative to previous samples. Feature vector examples with more components for larger pieces of handwriting pieces are given by Schomaker [1990] and Guyon, Albrecht, LeCun, Denker & Hubbard [1991].

Based on the studies by Schomaker [1990], Guyon et al. [1991] and Seni et al. [1994], we select a set of 13 features which we use to describe the frames and segments. The basic set of features comprises aspect ratio, curvature, start and end angle relative to the x-axis, three intermediate angles at equidistant points along the segment, and a pen-down feature. Note that all the angles with respect to the x-axis are denoted as θ while the intermediated angles based on three points in the piece of handwriting are denoted as ϕ. This accumulates to a feature vector for a single observation as shown in Figure 4.1 while Figure 4.2 clearly demonstrates the computed features.

$$o_t = \begin{pmatrix} f_{t_0} \\ \vdots \\ f_{t_{12}} \end{pmatrix} = \begin{pmatrix} r_a(t) \\ c(t) \\ \sin\theta_{\text{start}}(t) \\ \cos\theta_{\text{start}}(t) \\ \sin\phi_1(t) \\ \cos\phi_1(t) \\ \sin\phi_2(t) \\ \cos\phi_2(t) \\ \sin\phi_3(t) \\ \cos\phi_3(t) \\ \sin\theta_{\text{end}}(t) \\ \cos\theta_{\text{end}}(t) \\ p_d(t) \end{pmatrix}$$

Figure 4.1. Symbolic description of the basic feature vector.

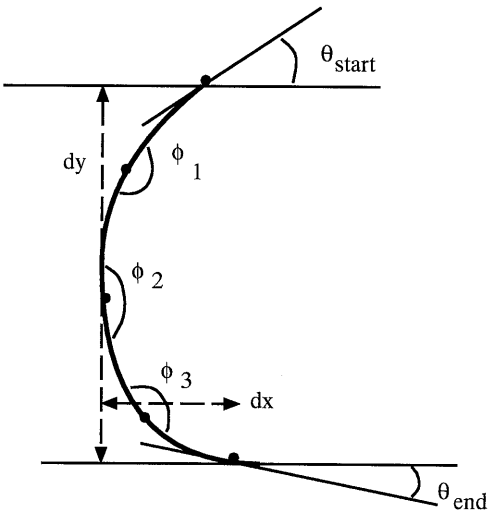


Figure 4.2. Graphical description of the basic feature vector.

There are five angles for representing a segment or frame. They are computed as described by Guyon et al. [1991] who takes advantage of the fact that the segment consists of a sequence of discrete samples where every three adjacent samples are used to measure an angle. Because each angle is represented by two components, sin and cos, this adds up to 10 angular features.

In addition to the 10 angular features in the feature vector in Figure 4.1, we

have three more features which are pen-down, curvature, and aspect ratio which are described at time t with the functions $p_d(t)$, $c(t)$, and $r_a(t)$, respectively. The pen-down feature is defined as

$$f_{t_{12}} = p_d(t) = \begin{cases} 1 & \text{if the pen-pressure is} > 0 \\ 0 & \text{otherwise} \end{cases}.$$

Figure 4.2 shows that dx and dy describe the width and height of the piece of handwriting. This leads to a definition of the aspect ratio equal to

$$f_{t_0} = r_a(t) = \frac{dx}{dx+dy}.$$

The curvature is defined as the relation between the path lengths of the samples of the piece of handwriting and the distance between the beginning and end point. The path length is defined as the sum of the distances between each adjacent coordinate pair in the piece of handwriting. For an observation o_t with N samples, we denote the path length of o_t with the function $pl(t) = \sum_{i=2}^{N}[(x(i)-x(i-1))^2+(y(i)-y(i-1))^2]^{0.5}$. The distance between start and end point is denoted as $d(t) = [(x(N)-x(1))^2+(y(N)-y(1))^2]^{0.5}$. This leads to a definition of the curvature of o_t equal to

$$f_{t_1} = c(t) = \frac{d(t)}{pl(t)}.$$

According to this definition, the real values of the features have a range of [0;1], except for the angles where the range is [-1;1].

Because not all digitizers support sampling above the tablet during a pen-up movement, it is possible that we do not have coordinates for the pen-up scribble to compute the angular and curvature features. Therefore, we decided to model all pen-up scribbles with a linear approximation based on the end point of the previous and the beginning of the next frame or segment which are pen-down movements per definition. Because start and end angles in the feature vector are the same, only the start angle is computed while the other angular components are zero. In case of a pen-up scribble, we assume that the curvature $c(t)$ is one and the pen-down feature $p_d(t)$ is zero.

4.1.4 Aggregate features

In the previous subsection, we introduced a basic feature vector in Figure 4.1 which represents the data in a frame or segment. Such a feature vector o_t is computed at every timestep t. Because it is known that feature events that describe consistent trends in handwriting over several points can improve recognition accuracy, we discuss some common techniques that compute additional features used to augment

the feature vector. In addition, we introduce a novel framework to describe the consistent trends in the handwriting.

The easiest technique is to splice several simple feature vectors into a larger feature vector. This is demonstrated in handwriting recognition by Bellegarda et al. [1994] and Starner et al. [1994]. Similar approaches are used in speech recognition [Rabiner & Juang, 1993; Lee, 1988] .

The addition of structural handwriting knowledge to a representation is not new. Manke & Bodenhausen [1994] and Kosmala, Rottland & Rigoll [1997] both use a 3x3 context bitmap which is a scaled-down version of an $N \times N$ bitmap containing an image of the environment of a feature vector. Seiler, Schenkel & Eggimann [1996] add features to describe structures in the handwriting, which are similar to loops. Schenkel et al. [1995] use the hat-feature to describe diacritical marks in the handwriting. A recent study by Liu, Liu & Dai [1996] explores multi-resolution features in an off-line, handwritten digit recognition task using a multilayer perceptron. Multi-resolution features are obtained from an image which is low-pass filtered and sampled at several different scales. The authors found an improvement over the baseline representation due to the new features.

In general, it is necessary to describe and integrate not only information which is local to one frame or segment, or global information spanning a complete word, but handwriting-specific information spanning several feature vectors.

This thesis explores a new framework to include structural knowledge into the representation of the hidden Markov model in Subsection 4.1.5 which bridges the gap between local information only and a global description of handwriting.

Delta features

One way to describe the dynamics of the signal is the use of delta features. This approach is adopted from speech recognition approaches [Rabiner & Juang, 1993; Lee, 1988]. Delta features are approximations to the derivatives of the observation vector with respect to time. One realization of delta features are straightforward, first-order differences of o_t with respect to other feature vectors. This leads to the following feature vector

$$o'_t = \begin{pmatrix} o_t \\ \frac{1}{2}(o_{t+1} - o_{t-1}) \end{pmatrix}.$$

Such a transformation doubles the size of the feature vector. If more feature vectors are combined then the feature vector size becomes three or more times as large. Usually, augmented delta features are computed from feature vectors in the range [t-2; t+2] but this also depends on the number of observations per second. It is impractical to compute delta features from more feature vectors due to high dimensionality of the combined feature vector.

Contextual features

In order to augment the feature vector with extra features describing the spatial dynamics of the handwriting signal, we developed the concept of *contextual* features. Instead of a mere combination of features as explained earlier, we employ functions on a variable set of previous and future feature vectors, and the corresponding original pieces of handwriting. As a consequence, delta features are a subclass of contextual features. Given a maximum time displacement d and a feature vector o_t this leads to an augmented feature vector o'_t with 13 basic features and K extra contextual features represented as f_1 to f_K.

$$o'_t = \begin{pmatrix} o_{t1} \\ \vdots \\ o_{t13} \\ f_1(o_{t-d},\ldots,o_t,\ldots,o_{t+d}) \\ \vdots \\ f_K(o_{t-d},\ldots,o_t,\ldots,o_{t+d}) \end{pmatrix}.$$

In contrast to the delta features, where the time displacements are often limited to [-2;2], the time displacement in this approach may be longer to catch longer-term structural relations. We assume that a positive time displacement d relates the current feature vector o_t to previous feature vectors up to o_{t-d} and future feature vectors up to o_{t+d}. Disadvantage of a relation with future feature vectors o_{t+d} is a possible interference with real-time recognition.

In this thesis, we use positive time displacements and relate the current feature vector to previous feature vectors only. Hence, we refer to the time displacement d as 'delay'. More precisely, we compute the contextual features from feature vectors in the range $[o_{t-d},\ldots,o_t]$ and refer to a feature vector o_{t-d} as 'delayed' feature vector.

At the boundaries of the scribble sequence, the contextual features $f_i(o_{t-d},\ldots,o_t)$ are not computed but replaced with 0. An alternative is a 'modulo' computation on the observation number $t-d$ which guarantees that the index $t-d$ always refers to a valid observation. The approach is flexible and allows the inclusion of longer-term information while limiting the feature vector size compared to delta features.

4.1.5 Realizations of contextual features

In the previous section, we introduced the concept of contextual features. These features are a mean to quantify structural relations, which may span several observations, in features of the representation of a hidden Markov model. In this section, we compute several types of contextual features on the basis of the framework pre-

sented in the previous section. In particular, we compute positional, size, overlap, and contour features which are used to augment the feature vector in Figure 4.1.

Positional features

The computation of angles based on the lines between adjacent samples is a common way of describing handwriting scribbles [Guyon et al., 1991; Seni et al., 1994; Schomaker, 1990; Yang, 1995]. In contrast, the angles of the lines between the center-of-gravities (cog) of successive frames or segments and the x-axis is structural knowledge which is a novel way to express spatial relationships between different observations. In comparison with delta features, this approach presents the available spatial information more explicitly. The approach is applicable to both segments and frames.

To compute the angle of the line between the centers-of-gravity of two feature vectors and the x-axis, we need a number of auxiliary variables. In particular, we need the definition of a 'center-of-gravity', the displacement between two centers-of-gravity and the distance between two centers-of-gravity.

For a feature vector o_t derived from a frame or segment composed of N_t coordinates (x,y), we compute the center-of-gravity $(x_{\text{cog}}, y_{\text{cog}})$ of the frame or segment as

$$x_{\text{cog}}(t) = \frac{\sum_i x_i}{N_t}, \quad y_{\text{cog}}(t) = \frac{\sum_i y_i}{N_t}.$$

The displacement between the centers-of-gravity of the current feature vector o_t and the delayed vector o_{t-d} is expressed as

$$\Delta x_{\text{cog}}(t,d) = x_{\text{cog}}(t) - x_{\text{cog}}(t-d), \quad \Delta y_{\text{cog}}(t,d) = y_{\text{cog}}(t) - y_{\text{cog}}(t-d).$$

The distance between the centers-of-gravity of the current vector o_t and delayed vector o_{t-d} is

$$d_{\text{cog}}(t,d) = \sqrt{(\Delta x_{\text{cog}}(t,d))^2 + (\Delta y_{\text{cog}}(t,d))^2}.$$

For each delay d, we compute the vector from the center-of-gravity of the current feature vector o_t towards the center-of-gravity of the delayed feature vector. We compute the angle of this vector with the x-axis. The computed angle is split into a sin and cos component which provide two positional or angular features f_{p_1} and f_{p_2} that express the location of a frame or segment relative to another frame or segment. We compute f_{p_1} (the sin component) and f_{p_2} (the cos component) as

$$f_{p_1}(t,d) = \frac{\Delta y_{\text{cog}}(t,d)}{d_{\text{cog}}(t,d)}, \quad f_{p_2}(t,d) = \frac{\Delta x_{\text{cog}}(t,d)}{d_{\text{cog}}(t,d)}.$$

Figure 4.3 shows an example of these angular features for the character 'h' containing four segments. We assume the existence of three delays d which values are 1, 2 and 3. We compute a center-of-gravity for each segment. Figure 4.3(A1) shows the position of these centers-of-gravity. For each of the three delays, we draw a vector from the (assumed) current segment number 4 to the delayed center-of-gravity as shown in Figure 4.3(A2). The sin and cos of the angle of these three vectors with the x-axis form a total six angular features.

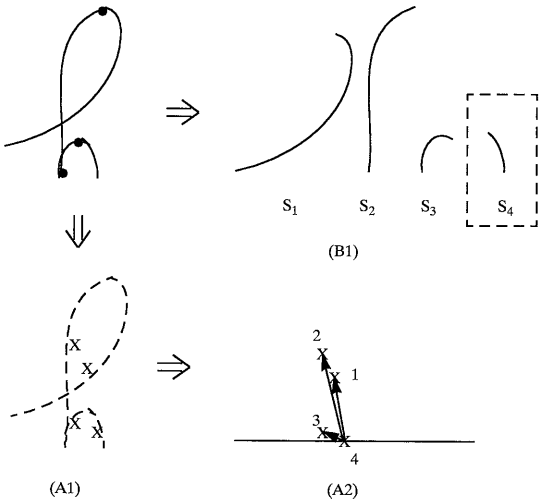


Figure 4.3. Examples of positional and size features.

Size features

In addition to the positional features discussed in the previous section, the use of size relations provides extra information. The size relations are useful only for segments because all frames contain the same number of samples, and the distance between samples is equal because of the spatial subsampling.

The idea is to relate the path lengths of current segments to previous segments in order to collect information about writing size changes in the representation. This approach partially compensates for the loss of size information resulting from the definition of segments which are size-independent.

For each delay d, the relative size of the current feature vector o_t is computed from a segment composed of N_t samples. This is implemented by computing the path length of every segment and comparing the path lengths. The path length of a segment composed of N_t samples is given by

$$pl(t) = \sum_{i=2}^{N_t} \sqrt{(x(i)-x(i-1))^2+(y(i)-y(i-1))^2}$$

The path length of the current feature vector o_t is compared with the path length of a delayed feature vector o_{t-d} with delay d as

$$f_s(t,d) = \frac{pl(t-d)}{pl(t)+pl(t-d)}$$

where f_s is the feature which expresses the size relation. The range of the feature value is $[0;1]$. If there is more than one delay then we compute the size feature for all delays and augment the feature vector with the resulting features.

An example is given in Figure 4.3(B1). The handwritten character 'h' has been split into four segments. We assume that the current segment number is four. We also assume that there are two delays d which are 1 and 3. In this situation, the feature vector o_4 can be augmented with two size features which are $f_s(4,3)$ and $f_s(4,1)$ resulting in two extra features.

Overlap features

Contextual features enable the computation of overlap relations similar to the hat-feature described by Weissman, Schenkel, Guyon, Nohl & Henderson [1994]. This allows us to express spatial relationships like a dot on an 'i' or a bar in a 't' or 'f' and add this structural information to the representation of the hidden Markov model. For example, a handwritten 'i' consists of two segments, a dot and a body segment. We know that one of the segments is placed above the body. If we are able to mark that segment then we have an extra feature which discriminates the dot from other segments. Note that we use the example of segment grouping, but the discussion in this section holds also for frames.

While the approach proposed by Weissman et al. [1994] detects and removes diacritical scribbles (like the dot on the 'i'), after which the data beneath the removed scribble is marked with the hat-feature, we employ the more general strategy of computing the overlap in x direction between subsequent feature vectors. The width of segments is used to determine the occurrence of an overlap and the relative overlap. The advantage of our approach is that this overlap provides data not only for diacritical marks but for all characters. The disadvantage is that the overlap of the 'i' dot with the 'i' body is stored in the dot itself, which is sometimes delayed.

To express the overlap relations, we need three auxiliary variables which describe the physical width of a segment, the actual overlap in x direction of segments and a boolean variable which indicates that an overlap takes place.

Given the current feature vector o_t which is computed from a segment containing N_t coordinates (x,y). We assume that $x_{\min t}$ is the leftmost x coordinate in the segment of feature vector o_t and $x_{\max t}$ is the rightmost coordinate. Then we can describe the physical width in x direction of the segment as $w(t) = (x_{\max t} - x_{\min t})$.

Now that we know the width of the segment used to compute o_t, we can detect overlaps in x direction between segments. If two segments occur at identical x-positions with identical width then the overlap is 100%. The auxiliary variable

$$r(t,d) = \frac{w(t-d)}{w(t)+w(t-d)}$$

indicates the amount of overlap between two segments with a value in the range [0;1]. An additional variable $o(t,d)$ holds a boolean value which indicates whether an overlap between o_t and o_{t-d} occurs.

$$o(t,d) = \begin{cases} 1 & \text{if the frames or segments at time } t \text{ and } t-d \\ & \text{overlap in } x \text{ direction.} \\ 0 & \text{otherwise.} \end{cases}$$

The combination of $w(t)$, $r(t,d)$, and $o(t,d)$ enables us to express the overlap relation in a feature f_o. For a given delay d this leads to

$$f_o(t,d) = \frac{1}{t-d+1} \sum_{i=t-d}^{t-1} o(t,i) \cdot r(t,i).$$

This single feature indicates if an overlap of segments occurs for a given delay d and all feature vectors o_{t-d} to o_t. The feature also quantifies the amount of overlap. No overlap means $f_o(t) = 0$ while 100% overlap results in $f_o(t) = 1$. The single feature $f_o(t)$ can be used to augment each feature vector o_t. To summarize, the overlap relation expresses an average overlap of the current frame or segment at time t with the previous d observed feature vectors.

Contour features

Several authors have shown the advantages of the contour information of a word in word recognition or pre-selection of viable word alternatives. Disadvantage of most approaches is that the complete word is needed before processing. Here, we present an on-the-fly computation of contour information based on contextual features.

Ho, Hull & Srihari [1992] explored a recognition algorithm for typed or printed words based on contour information only. Seni, Srihari & Nasrabadi [1994] used contour information based on ascenders and descenders to reduce the size of a dictionary as a pre-processing step for a large-vocabulary, cursive word recognition system. A similar approach is presented by Madhvanath & Srihari [1996] for off-line data. Caesar, Gloger & Mandler [1995] used contour information to estimate the baseline of off-line handwritten words. Similarly, Seiler, Schenkel & Eggimann [1996] used contour information to determine the word height for an explicit, size normalization step.

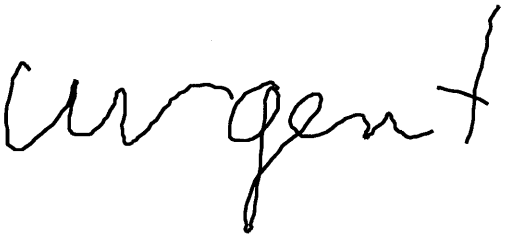


Figure 4.4. Handwritten word 'urgent'.

To add contour knowledge to the framework based on hidden Markov models, we need to find a way to compute a contour score which is continuously updated during the left-to-right processing of the input word.

The first option is to explicitly detect characters, determine the bounding boxes of the characters and score the relation between the size of the bounding boxes as contour transitions. This leads to a separate stream of features besides the basic feature vector developed above. As a consequence, a second hidden Markov model is needed and the recognition and training process involves the simultaneous processing on two different hidden Markov models for characters and contours, respectively.

The second option is to compute contour transitions on-the-fly and integrate the features describing the transition into the representation. This eliminates the need for a second stream of feature vectors and the training and the recognition process remains unchanged. In the example in Figure 4.5 the contour information is determined for the feature vector representing the left-most part of the 'g' in 'urgent'.

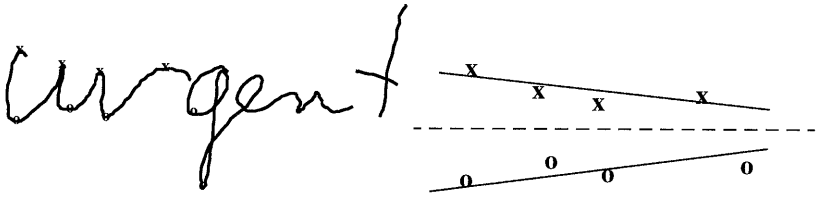


Figure 4.5. Left: segmentation point 'x' for the top contour and 'o' for the bottom contour. Right: regression line estimation based on segmentation points.

The contour features are computed for the segment representation only because the segmentation points indicated in Figure 4.5 are a natural mean for following the top and bottom contours. These segmentation points are the boundary points of the segments. The top and bottom contours are approximated with a regression line as shown in Figure 4.5.

The slope estimation is based on the boundary points of the previous d seg-

ments. We denote the slope of the top contour at time t as s_{tt} while the slope of the bottom contour is s_{bt}. Given the current feature vector o_t and a delay d, we describe the slope of the estimated top and bottom contour with two features $f_{c\text{top}}(t,d)$ and $f_{c\text{bottom}}(t,d)$. Therefore, each feature vector o_t is augmented with two additional features with range [-1;1].

$$f_{c\text{top}}(t,d) = \frac{\arctan(s_{tt}) \cdot 2}{\pi}, \quad f_{c\text{bottom}}(t,d) = \frac{\arctan(s_{bt}) \cdot 2}{\pi}$$

The minimum delay d is six segments which provides three points to compute the top and bottom contour where we define contour as the Least Mean Square (LMS) estimated or linear regression line through the top and bottom segmentation points. The more points we have to compute the regression line, the better the estimate. A delay of 10 segments seems reasonable.

A large delay d limits the use of the contour estimation to word and sentence recognition. Note that the delay is typically longer than one character. Therefore, there are contextual effects which means that the value of the two features $f_{c\text{top}}(t,d)$ and $f_{c\text{bottom}}(t,d)$ depends on the current character but also on the previous character(s).

Figure 4.6 shows an example of the contour signal for both top and bottom contour of the word 'urgent' in Figure 4.4. The delay d is 10 which means that the previous 10 segments are used to compute the contour features. The feature values for the top and bottom contour are plotted at each timestep t.

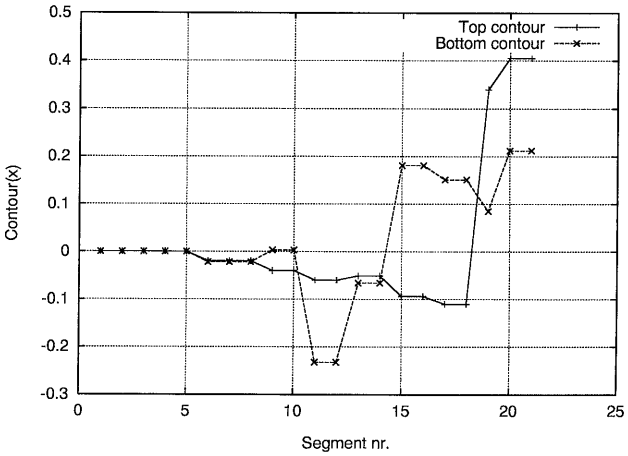


Figure 4.6. Contour signal for the word 'urgent'.

4.1.6 Linear Discriminant Analysis

The previous sections discussed the features in an observation o_t. We introduced techniques to augment feature vectors with additional features describing structural relations in handwriting. Because this can result in a high-dimensional feature vector, this section discusses a technique for reducing the size of the feature vector.

Linear discriminant analysis (LDA) [Fukunaga, 1990] is a well known statistical technique which is frequently used in statistical pattern classification for compressing the information contents, with respect to classification, of a feature vector by a linear transformation. This is especially important in combination with the augmented feature vectors presented above. In contrast to Principal component analysis (PCA), LDA takes into account that data originates from different classes.

Given a feature vector $x \in \mathbb{R}^L$ and a number of class labels $\{\omega_1,...,\omega_C\}$, then we search a linear transformation $f : \mathbb{R}^L \rightarrow \mathbb{R}^D$ where $D \leq L$ which results in a transformed feature vector $\tilde{x}$ with dimension D. To find the transformation, we have to compute the mean μ_i of all data in a class ω_i, the within-class covariance matrix U_w, the between-class covariance matrix U_b and the total covariance matrix U_t. In the following, we only use the within-class covariance U_w and the total covariance U_t.

In the situation where we train a classifier, the class labels are classes in the LDA sense, i.e., we take all feature vectors of the training data and divide the feature vectors into several classes. For example, if we assume that each state of a hidden Markov model is a class ω_i then we use all feature vectors which occur in a state i to compute a mean μ_i and a within-class covariance matrix U_w.

To compute the means μ_i and covariance matrices U_w and U_t, we do two passes over the training data. We assume that the training data contains C classes with a total of N feature vectors and that each class ω_i contains N_i feature vectors. In the first pass, we compute the average feature vector for each class i as μ_i and the average of the complete data set μ_t.

$$\mu_i = \frac{1}{N_i} \sum_{\{x_j \in \omega_i\}} x_j, \quad \mu_t = \frac{1}{N} \sum_j x_j$$

In the second pass, we compute the covariance matrices U_w and U_t.

$$U_w = \frac{1}{N} \sum_{i=1}^{C} \sum_{\{x_j \in \omega_i\}} (x_j - \mu_i)(x_j - \mu_i)', \quad U_t = \frac{1}{N} \sum_j (x_j - \mu_t)(x_j - \mu_t)'$$

Now that we have the within-class and total covariance matrices U_w and U_t, respectively, we define a performance index for the class separation as $J = tr(U_w^{-1} U_t)$. The larger the performance index J, the better the class separability [Duda & Hart,

1973]. The trace elements of $U_w^{-1}U_t$ indicate the discriminative value of the corresponding features in the feature vector before LDA transformation.

From the training data, we compute the LDA transformation matrix W with dimension $L \times D$ and $D \leq L$. This is done by solving the eigenvalue problem $U_w^{-1}U_t\Phi = \Phi\Lambda$ [Fukunaga, 1990] which results in a matrix W containing the eigenvectors which are sorted according to their eigenvalues. The matrix W containing the D eigenvectors with the largest eigenvalues is the LDA transformation matrix. The eigenvalues indicate the discriminative value of the transformed features.

Now that we have the LDA transformation matrix W, we use it to transform the feature vectors. LDA is a feature extraction process which transforms the feature vector $x \in \mathbb{R}^L$ according to $\tilde{x} = Wx$. This is a transformation of the form $x \in \mathbb{R}^L$ to $\tilde{x} \in \mathbb{R}^D$ where $D \leq L$. The LDA transform matrix W is the matrix which maximizes the performance index J.

The result of the LDA transformation is a feature vector $\tilde{x} = Wx$ which is decorrelated, ordered, and maximally compact. The first property is advantageous, because in our hidden Markov models we employ diagonal covariance matrices as discussed in Section 4.2. The second property means that the features are ordered according to decreasing eigenvalues. The first features, i.e., those with the largest eigenvalues, contribute most to class separability. Finally, the third property states that for any subset of features 1 to D the sum of the eigenvalues is maximum. More specifically this means that no other subset of the same number of D features or linear combinations thereof has a larger sum.

LDA has been employed in the training and recognition process similar to that described by Haeb-Umbach & Ney [1992] for speech recognition. Training is carried out in three steps:

- first an ordinary training is carried out. This results in a time-alignment, i.e., a class label for each feature vector. Note that we define the classes in the LDA sense to be hidden Markov model states;
- next, the within and total class scatter matrices are computed and from them the LDA transformation is obtained by solving an eigenvalue problem [Fukunaga, 1990];
- finally, a completely new training is conducted on the LDA-transformed feature vectors. The dimension of the transformed feature vector can optionally be reduced by discarding the least important rows of the LDA transformation matrix.

LDA is preferred over Principal component analysis [Bellegarda et al., 1994] because LDA takes into account that the items originate not from one but from several classes. Although LDA is routinely applied in speech recognition [Haeb-Umbach

& Ney, 1992] , there are only few examples of the application of LDA in handwriting recognition based on hidden Markov models. An example is the work by Stampa, Caesar, Gloger, Kaltenmeier & Mandler [1996] who apply LDA to off-line, isolated, handwritten digit recognition and who report 91% correctly recognized ZIP codes. PCA is used by Kassel [1995] and Bellegarda et al. [1994] for character recognition based on hidden Markov models and by Cho, Lee & Kim [1995] for word recognition based on hidden Markov models.

Lindwurm, Breuer & Kreuzer [1996] employ discriminant analysis in the context of an off-line, multi-expert, handprinted recognition system leading to 79.1% correct for alphanumerics and 95.2%, 91% and 87.3% for digits, upper and lowercase data, respectively.

Except for Haeb-Umbach & Ney [1992], who quote a 18% relative improvement for a specific test due to LDA, the above studies do not give the relative improvement due to LDA or PCA.

In the context of this thesis, a state of a hidden Markov model is used as class definition for LDA. Hence, the LDA tries to transform the data in order to ease the separation of the hidden Markov model states. Chapter 5 will quantify the benefits of LDA in the context of handwriting recognition on the basis of hidden Markov model.

4.2 Models

In the previous chapter, we presented several hidden Markov model structures and gave a formal definition of the handwriting recognition in Definition 3.34. In the beginning of this chapter, we made the problem definition more concrete and chose hidden Markov models to model characters. This section discusses how to exploit handwriting-specific knowledge in combination with hidden Markov models.

The section starts with a discussion of the basic structure of a hidden Markov model in Subsection 4.2.1. This is followed by a discussion on the modifications to the hidden Markov model in Subsection 4.2.2 and Subsection 4.2.3 in order to accommodate for different styles of handwriting input. Language models in the context of handwriting recognition are discussed in Subsection 4.2.4.

4.2.1 Structure

The basic theory of hidden Markov models has been discussed in Chapter 3. This subsection discusses the parameter choices resulting in a model which serves as an experimental basis for the experiments of Chapter 5. The general objective is to obtain a robust hidden Markov model which allows the integration of handwriting-specific knowledge into representation or model structure. The objective is *not* to tune the model structure to specific characters or circumstances.

The basic model is a left-to-right (Bakis) model with loop, forward and skip transitions between the states. The number of states depends on the expected number of observations. All the allographs are modeled in a single model which means that there is one hidden Markov model per character of the alphabet that is considered. The transition matrix of each hidden Markov model is estimated during the training. The observation probability densities are continuous mixtures of Gaussian densities with density-specific, diagonal covariance matrices. The diagonal covariance components have a lower bound of 0.25 times the state-specific covariance to prevent estimation errors due to lack of data.

4.2.2 Model improvements

In the previous sections, a left-to-right hidden Markov model was assumed as a character model and a word model is obtained by concatenation of character hidden Markov models. This basic approach remains the same. Nevertheless, we know that it is possible to integrate handwriting-specific knowledge into the model structure to obtain a more accurate character model. An example is the state duration modeling by Senior [1994], the 'backspace' state by Starner et al. [1994], and the ligature model by Cho et al. [1995]. Our study concentrates on ligature and backspace modeling.

First, the difference between discrete and mixed-style handwriting is used to modify the structure of the hidden Markov model describing a word. It can be argued that the main difference is that discrete handwriting does not contain ligatures. All the characters are separated by a pen-up movement. In contrast, cursive style implies a pen-down ligature.

It is possible to extend a handwritten word model, based on a sequence of character models, by 'pause' models between the characters. This is shown in Figure 4.7. In the case of discrete input, the pause model is a one-state model which is used to exclusively model pen-up ligatures. In the case of pure cursive input, the pause model contains only pen-down ligatures. In the case of mixed input, either case is possible. We will use the notation */pause/* to refer to the handwriting realization instead of the abstract model.

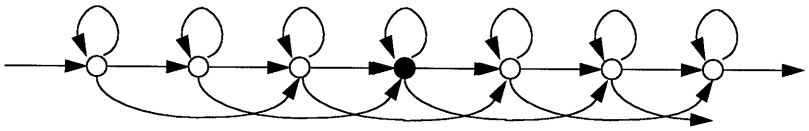


Figure 4.7. A hidden Markov model of a word which includes an intermediate 'pause' model which is colored black.

Like trigraph modeling, which is explained in Subsection 4.2.3, a pause model is an attempt to isolate the variability of character transitions. Consequently, the

context-free character models will contain fewer variable shapes.

Cho, Lee & Kim [1995] also use pauses in off-line word recognition and showed that the recognition accuracy improved by 20% to 45%, depending on the dictionary size. Their approach is to use a number of pause models describing categories of character transitions depending on the neighboring characters. This approach can be described as the use of a 'contextual pause model'. In contrast to their approach, we model *all* ligatures with the same pause model. This is based on the assumption that the number of ligature types is limited. Pen-up ligatures are approximated with a single linear line. Pen-down ligatures will normally contain one or two segments, depending on whether they are diagonal or horizontal. The number of frames per pen-down ligature is variable.

Second, diacriticals sometimes disturb the handwriting when part of the diacritical is delayed and written at the end of a word. We can model this effect of diacritical marks if we assume that diacriticals occur immediately after the character body (which is correct) or at the end of the word (in a delayed case). An additional state at the end of a word models this behavior. In this thesis, we chose the pause state to be this extra state at the end of a word. A similar approach is used by Starner et al. [1994], who introduced the term 'backspace state' to refer to the extra state at the end of a word. However, the benefit of this state was not compared with a situation without backspace model.

4.2.3 Contextual models

An examination of the example word in Figure 4.4 immediately shows that the shape of a character like an 'r' or 'e' is affected by the shape of the neighbor characters. This was already discussed in Section 2.1. To model these effects, N-graphs can be used which are allographs given a certain context. We define the context as a number of characters written previously or after the character under examination. An N-graph with N equal to 2 is referred to as *bigraph*. Such a model is a character model with one left or right context character. A *trigraph* is a model which spans three characters, i.e., both a left and right contextual character are given.

Starner, Makhoul, Schwartz & Chou [1994] and Kosmala, Rottland & Rigoll [1997] demonstrated that a trigraph based recognition system works fine in a writer-dependent test. The use of triphones (or recently even quinphones) is common in speech recognition as presented by Rabiner & Juang [1993], Ney et al. [1994], Lee [1988], and Schwartz et al. [1984]. An alternative approach is to use ligature models as discussed in the previous subsection.

In this study, we limit the contextual models to bigraph (left-context) and trigraph models. The example word in Figure 4.4 is modeled with context-

independent models as

$$/u/,/r/,/g/,/e/,/n/,/t/.$$

With trigraphs, this results in

$$\{\#\}/u/\{r\},\{u\}/r/\{g\},\{r\}/g/\{e\},\{g\}/e/\{n\},\{e\}/n/\{t\},\{n\}/t/\{\$\},$$

where # and $ are the markers for word start and word end, respectively. The middle character of the example trigraphs in /c/ notation refers to the context-dependent models while the other characters, indicated with $\{x\}$, provide context. This also implies that a dictionary on basis of contextual models is different compared to a context-free dictionary.

The set of bigraphs and trigraphs used to model the handwritten characters is computed on the basis of the labels of the training data, the expected labels in an application domain, and the number of training examples. Contextual models are calculated on the condition that enough observations are given for a model. In practice, that means that sufficient training data is required for every contextual model [Kosmala et al., 1997; Lee, 1988]. We used a minimum of 30 samples to train a model.

The algorithm to select trigraphs and bigraphs is simple. First, we parse all the training text and calculate the number of occurrences of characters, bigraphs and trigraphs. Next, we parse the training text again and add contextual models if certain conditions are met. We add a given trigraph to the set of models if its count exceeds a threshold. Bigraphs are added if their count exceeds the same threshold and if the bigraph is not part of another trigraph.

4.2.4 Language models

A language model as defined in Subsection 3.6.2 is an extra context source and therefore has the potential to improve recognition performance. In this thesis we concentrate on two types of language models which are *character* and *word*-based as defined in Definition 3.27. Word-based language models are routinely used in speech recognition for dictation or other tasks. The application of word-based language models to handwriting recognition is straightforward.

Character-based models have been used in handwriting recognition but there are few studies of such a language model in the context of hidden Markov models. The current study compares unigram with bigram language models and character with word-based language models.

Character-based language models are easier to train than word-based language models because the number of classes is smaller. Their advantage is that they allow handwriting recognition with an unlimited vocabulary. Their fundamental flaw is that a character unit provides less context than a word unit. An example comparison

of a character and word-based language model is given in Subsection 3.6.2.

This implies that a word-based language model is more accurate than a character-based model. However, a word-based language model fails if a sentence contains an unknown word which is not in the dictionary. A character-based language model handles any sentence or text as long as the basic characters are known.

As an example, the probability of the sentence part 'a fistful of' is computed for some language models in Table 4.1, where '#' indicates the beginning of the sentence.

Table 4.1. Computation of a sentence probability given some language models.

Unigram	Word	$P(W) = P(a) \cdot P(fistful) \cdot P(of)$
Bigram	Word	$P(W) = P(a\|\#) \cdot P(fistful\|a) \cdot P(of\|fistful)$
Unigram	Char	$P(W) = P(a) \cdot P(f) \cdot P(i) \cdot P(s) \cdot P(t)...$
Bigram	Char	$P(W) = P(a\|\#) \cdot P(f\|a) \cdot P(i\|f) \cdot P(s\|i) \cdot P(t\|s)...$

An excellent example of a word-based grammar in handwriting recognition based on hidden Markov models is given by Starner, Makhoul, Schwartz & Chou [1994] who achieve a word error rate of 3%-5% for writer-dependent recognition with a 25,000 words vocabulary.

There are few examples of earlier applications of character-based, statistical language models. Guyon & Pereira [1995] study a character-based language model with a variable context size of up to six characters. Unfortunately, no recognition experiments are done. Kassel [1995] uses both unigram and bigram grammars based on characters with perplexities of 36.0 and 11.3, respectively, and achieved a 27% error reduction for a 62 class recognition task. Kundu, He & Bahl [1989] use both first and second-order hidden Markov models in off-line, handwritten word recognition and measure 92.5% correct words. An example of an analytical language model, comprising plain M-grams, in a non hidden Markov model environment is given by Hull & Srihari [1982].

4.3 Algorithmic aspects

In order to combine the abstract definition of handwriting recognition in Definition 3.34 with the previously discussed representations and models into a working classifier, a number of processing techniques and algorithms are needed. First, we discuss the transformation of handwriting samples to feature vectors in Subsection 4.3.1. The next step is to employ the feature vectors in a training algorithm in Subsection 4.3.2 to compute the hidden Markov models for each character. The recognition uses the computed character models to transform the feature vectors into computer readable text in Subsection 4.3.3.

4.3.1 Preprocessing

The on-line handwriting is sampled as a time-equidistant signal at a speed between 100pps and 200pps. Currently, only the x and y components of the sampled signal are used to determine the feature vectors whose components are normalized to the values in the range [-1;1] for angles and [0;1] otherwise. An averaging filter with window size five is applied to smooth the signal. To obtain frame-based feature vectors, the handwriting is resampled to a spatially equidistant signal after which sequences of adjacent points are used to compute feature vectors. Segment feature vectors are obtained without spatial resampling. Only velocity inversions are computed and used as segment boundaries. The scribbles associated with both frames and segments overlap 50% spatially. Note that pen-down and pen-up samples are processed differently. While the pen-down data is handled as explained above, the pen-up data always results in one frame or segment. The procedure is summarized in Figure 4.8.

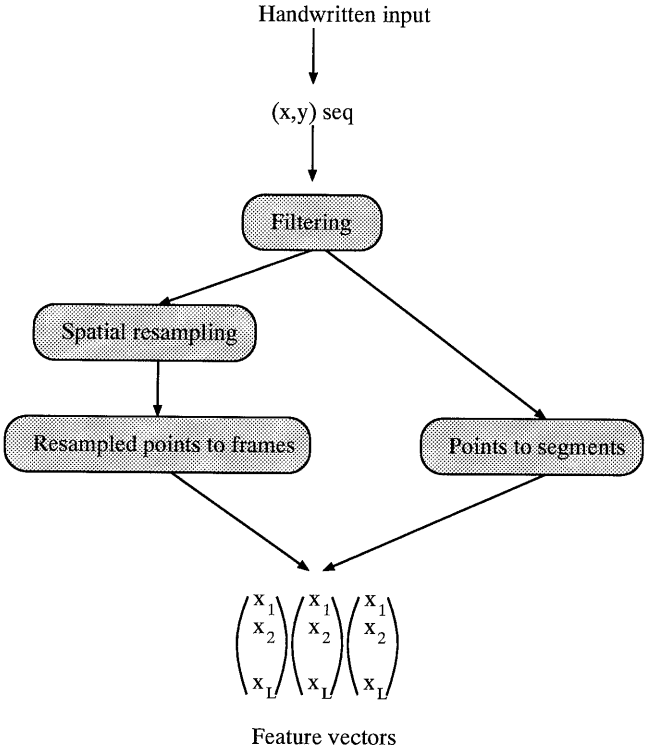


Figure 4.8. Handwriting signal preprocessing.

Handwriting has some properties which are considered to be unwanted distortions in a handwriting recognition task. Although some handwriting preprocessors ex-

plicitly compensate these distortions, the preprocessor presented here ignores them and models the variations with the hidden Markov model.

First, the handwriting is not necessarily written horizontally. The corresponding *slope* can be detected and compensated, as demonstrated by Senior [1994] and Caesar, Gloger & Mandler [1995] . Secondly, the writing is possibly *slanted* which means that the basic strokes are not vertical. A left slant is often observed in the case of left-handed writers and a right slant in the case of right-handed persons. It is possible to detect and compensate the slant, as explained by Bozinovic & Srihari [1989] . However, the technique is not robust. Thirdly, *writing size* is a factor which is sometimes handled by the preprocessor, especially in character and word recognition in which the complete input is available before recognition starts. Various techniques can be used as demonstrated by Beigi, Nathan, Clary & Subrahmonia [1994] and Seiler et al. [1996] . Fourthly, *diacriticals* can be handled by the preprocessor. If a dot on an 'i' or bar of a 't' is delayed until the end of a word then this data is optionally removed and converted to a feature. This approach is presented by Schenkel et al. [1995] and Nathan et al. [1995] .

4.3.2 Training

In Section 3.5 we concentrated on the theory of training hidden Markov models. The current section focuses on the implementation of the theory. We employ the maximum likelihood criterion and apply the Viterbi approximation [Rabiner & Juang, 1993] in the training algorithm to estimate the parameters of a continuous hidden Markov model. All likelihood computations are log-scaled to prevent underflow. Gaussian densities are used to model the probability density in states.

Initially, the training creates 26 context-independent, lowercase character models. Alternatively, 26 uppercase or 10 digits or any combination of character classes is possible. All allographs of a character are mapped to a single character model. An additional pause model is created to model pen-up parts between adjacent characters and ligatures as explained in Subsection 4.2.2.

The training procedure is outlined in Figure 4.9. Characters, words or sentences are processed. All text is modeled as a sequence of character models, with an optional pause model between the characters, resulting in a linear state sequence. Except for the first iteration, in which observations and states are aligned linearly, the Viterbi approximation is used to align the observations from the preprocessor with the state sequence. We constrain the first observation to the first state and the last observation to the last state. This is achieved by assigning a probability of one to π_1 and zero to the π_i for $i > 0$.

The training algorithm presented in Figure 4.9 is slightly different compared to approaches in the literature. Although the path-estimation is identical to the approach in Ney et al. [1994], the computation of the mixture densities is different.

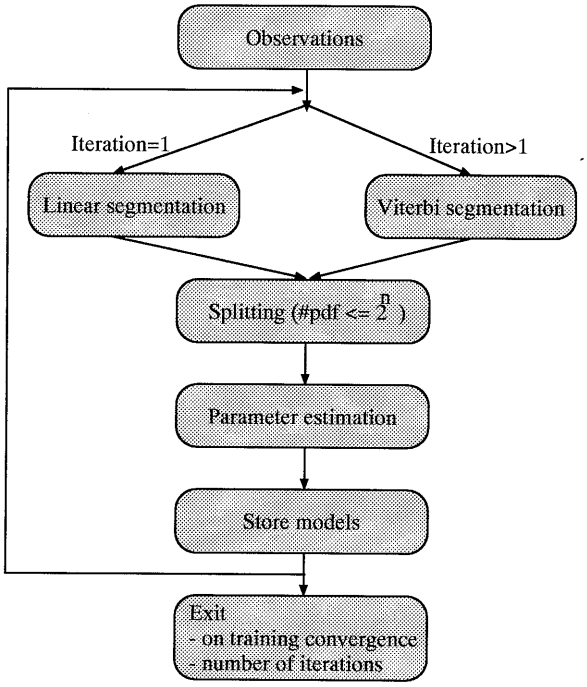


Figure 4.9. Training process of a hidden Markov model.

We do not increase the maximum number of mixture densities per iteration but fix the maximum number of densities per state, e.g. 32, and use up to this maximum number of densities in every iteration.

In addition to the context-independent (CI) training, a training with LDA transformation and/or contextual character models is possible. The training of the context-dependent (CD) models is similar to the LDA training explained in Subsection 4.1.6. The contextual models are initialized with the context-independent models. Next, the Viterbi training is again applied and the first iteration uses the same time-alignment as the last iteration with context-independent training. The stacking of training procedures is summarized in Figure 4.10.

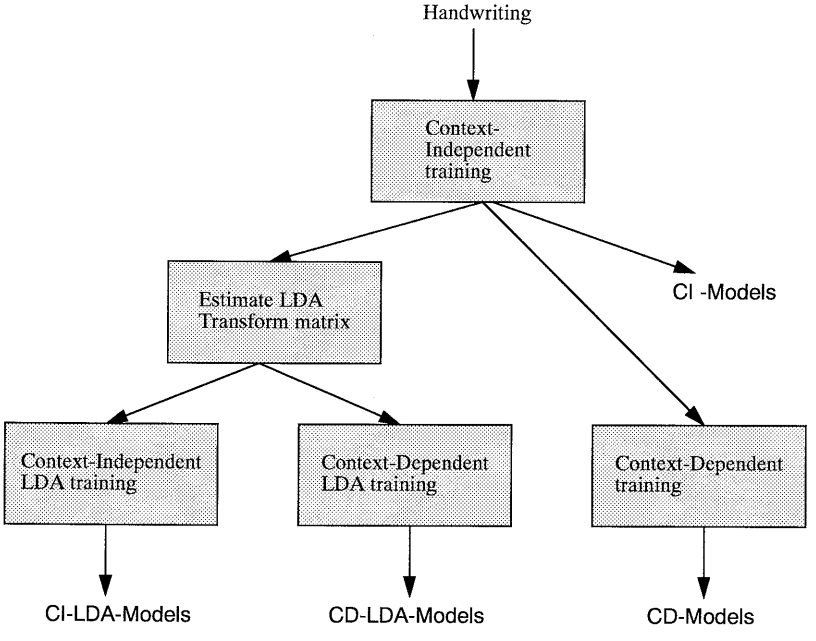


Figure 4.10. The combination of various training processes for hidden Markov models.

4.3.3 Recognition

The basic algorithms for handwriting recognition based on hidden Markov models have been discussed in Section 3.4 and Subsection 3.6.3. This subsection considers some implementation oriented aspects.

Recognition is based on the one-stage beam search algorithm discussed by Ney [1984] and Ney et al. [1994] . All knowledge sources, i.e., the pen signal, the dictionary, and, for some experiments, a language model, are applied simultaneous, thus avoiding premature decisions. Hypotheses pruning is applied for efficiency.

For each timestep, the $\log(b(o_t))$ computations are carried out in a preprocessing step.

In the context of speech recognition, the one-stage beam stage is often called 'time-synchronous' because of the time-equidistant sampling. In handwriting recognition based on hidden Markov models, it makes sense to call the one-stage beam search a frame of segment synchronous approach. For every newly generated frame or segment, a new observation or feature vector is computed and tested against the state array formed by the dictionary. A writing speed of two characters per second and an average of 10 frames per character results in about 20 observations per second for frames. In case of segments, we assume an average of five segments per characters which results in about 10 observations per second.

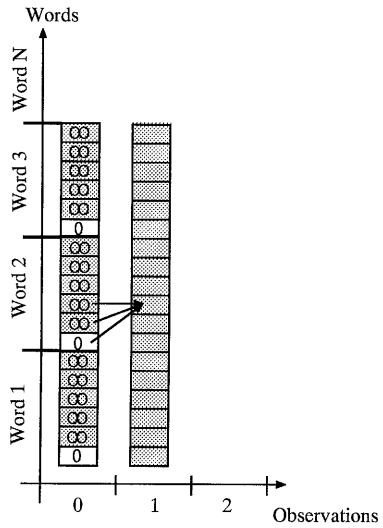


Figure 4.11. Schematic beam search for word recognition based on hidden Markov models.

The state space is schematically represented in Figure 4.11, identical to Figure 3.11, where a linear word dictionary forms a state array on the y-axis. On every timestep on the x-axis, one observation is processed for all relevant states.

The context information provided by a dictionary is always mapped on the state array on the y-axis. However, there are several possibilities. The simplest is a linear dictionary where the states of all characters of all words are placed in the state array. A more efficient approach is to share common word parts in a tree-organized dictionary [Haeb-Umbach & Ney, 1994].

Additionally, we can choose whether we compute only the active states or all states. Examples of such fast search techniques are given by Manke et al. [1996]

and Ratzlaff et al. [1996], where character duration modeling is used as additional pruning constraint.

If no new data is generated then we select the most likely terminal state with the lowest loglikelihood score. Sentence recognition involves the tracking of sequences of best words as explained by Ney [1984]. There is no difference between character and word-based language models other than that a character-based language model requires a vocabulary of only 26 words representing the lowercase characters.

The next chapter discusses character and word recognition experiments with a tree-organized dictionary and a state-cache of active states. The sentence recognition experiments are done with a linear dictionary.

4.4 Experimental framework

Although the empirical results are discussed in Chapter 5, two issues are briefly discussed here because they relate to the recognition system in general. These issues are error counting in Subsection 4.4.1 and coding in Subsection 4.4.2.

4.4.1 Error counting

It is easy to compute character and word error rates which is just a matter of counting recognition errors and dividing with the total number of tested labels. However, two important issues related to scoring have to be resolved.

The first is the scoring of sentence recognition results. In contrast to the scoring of word and character recognition results, sentence level scoring is complicated by occurrence of 'insert' and 'delete' errors in addition to 'substitution' errors. Some authors have argued that insertions are no errors, but merely undesired.

In this thesis, we compute the Levenshtein distance [Sankoff & Kruskal, 1983] between the original and recognized text and use it as the number of errors in a sentence. The Levenshtein distance is a common metric to compare strings and corresponds with the minimum sum of substitute, insertion, and deletion actions to transform the recognized text into the original text. A dynamic programming algorithm computes the Levenshtein distance on the basis of a comparison between the word sequence of the recognition result and the word sequence of the expected, correct sentence. Similar error rate definitions are used by Lee [1988] and Kassel [1995].

$$\text{ErrorRate} = \text{E} = \frac{\text{LevenshteinDistance}}{\text{CorrectSentenceLength}}.$$

The accuracy is defined as

$$\text{WordAccuracy} = 100 \cdot (1 - \text{E}).$$

The second issue is the significance of recognition results. In word recognition, the test set is clearly defined and obtained from 10 individuals. This means that we can compute 10 individual, mean word error rates and compute a between-writer standard deviation σ.

No standard deviation is computed for character recognition tasks. Instead, the framework of Guyon, Makhoul, Schwartz & Vapnik [1996] provides a qualitative guideline for testing whether enough test samples are available in relation to the empirical error rate. The reason for this is that the size of the test set should be inversely proportional to the measured error rate E. For a test set of statistically independent samples it can be guaranteed with a confidence level of 95% that the real error rate $\tilde{E}$ will not be worse than $1.25E$ where E is the measured, mean error rate. A guideline is to use a test set size $n = \frac{100}{E}$. This means that a test set of 10,000 independent samples is enough to test a recognizer with measured error rate $E = 0.01$.

4.4.2 Code description

The implementation of the recognition system is split into two parts. A first module is the preprocessor which reads Unipen datafiles [Guyon et al., 1994] and transforms them into preprocessed datafiles. All handwriting data, including any on-site collected datafiles, are stored in Unipen format. The preprocessor is written in ANSI-C and is built from 11,000 lines of code of which 4,000 lines are Unipen-uplib code.

Second, the training and recognition module reads the preprocessed files and transforms them into either handwriting models or a recognized output text. This software has been built in C++ [Stroustrup, 1991; Stroustrup, 1994], may be compiled with several compilers (g++, CC/MipsPro, VC++) and works on several platforms such as HP-UX, SGI/Irix, Linux and WinNT. The size of the complete system is 32,000 lines of code containing 64 classes resulting in an executable of approximately 350 KByte for Intel CPU and roughly 700-1000 KByte for HP-PA-Risc and SGI/Mips R10000 depending on CPU, options and inlining.

Eight different types of recognition (Linear and Tree dictionary, Word and Sentence recognition) are modeled using a state array containing a beam of hypotheses. These classes all inherit from 'Beam'. There are nine iterators through beams and observation containers. 'Word' and 'Character' are classes which inherit from 'Ink'. The components of a hidden Markov model ('State', 'Pdf') are separate classes. 'Frame' and 'Segment' are classes which inherit from 'Observation'. All preprocessed data is contained in an 'ObservationRepository'. 'Dictionary' is a base class for two different types of dictionary trees.

5

Recognition Experiments

This chapter discusses the empirical results obtained by applying the theoretical framework of Chapter 4 to an extensive handwriting recognition study. Section 5.1 discusses the employed datasets. Section 5.2 presents the parameters of the hidden Markov model. Sections 5.3 to 5.5 present results for different representations and models in handwritten character, word and sentence recognition tasks.

5.1 Data

Over the past 20 years, research on proprietary datasets often resulted in high recognition rates which could not be reproduced in the field. Real progress has been made in the algorithmic aspects, which has allowed more robust processing of more complex and variable input. Over the years, datasets have become larger and more diverse, culminating in the Unipen database as discussed by Guyon, Schomaker, Plamondon, Liberman & Janet [1994], which contains handwriting samples from more than 2700 writers resulting in more than five million handwritten characters. The present study explores a general approach to handwriting recognition. We test the algorithms and models not only on characters, words, and sentences but also on different styles of handwriting.

Parameter selection is based on datasets not employed in the recognition phase. While most data was collected at Philips sites, we also used some data from release

7 of the Unipen training data [Guyon et al., 1994]. Unipen development or test data was not available for experiments. Part of the character recognition tests were carried out using data from the Unipen training data section 1[abc], which was split randomly into 70:30 parts to enable recognition tests. The word training data set contains 7000 mixed style words collected on-site and about 3000 words from section 6 of the Unipen data. The dataset properties used in this study are summarized in Table 5.1 which also shows how datasets were split into non-overlapping training and recognition parts.

Important dataset parameters are set size, alphabet size, digitizer and sample rate and number of writers. Basically, the more writers, nationalities and digitizers, the more difficult the recognition task. All recognition results are expressed in character error rate (CHER) and word error rate (WER).

Table 5.1. The datasets used in the recognition experiments. We use the following abbreviations: P=Philips, U=Unipen, D=Discrete data, M=Mixed-style data.

Dataset Name	**#Writers**	**#Symbols**	**Sample rate (pps)**	**Data Source**	**Train set size**	**Test set size**
Character data						
Digits1	>50	10	60-170	P	1301	555
Digits2	566	10	60-200	U	10515	4506
Upper	1355	26	60-200	U	17945	7691
Lower	2082	26	60-200	U	42280	18120
Lower2	10	26	160	P	-	1040
Word data						
Mixed1	10	26	160	P	-	500
Mixed2	10	26	160	P	-	2483
Mixed.train	>50	26	60-200	P/U	10357	-
Sentence data						
Sentence1	10	26(D)	160	P	-	500
Sentence2	10	26(M)	160	P	-	500

It is important to note that the sets *Lower2*, *Mixed1*, and *Sentence2* were written by the same 10 writers. This enabled us to test and compare the performance for a fixed set of writers with character, word, and sentence data. *Mixed2* contains the hand-segmented words from the sentences in *Sentence2* which enables us to compare the word error rates for word against sentence context.

The average length of the handwritten words in *Mixed1* and *Mixed2* is 6.5 and 4.7 characters, respectively. Based on this data, we conduct two recognition experiments for each dataset using vocabularies with 200 and 20,000 words, respectively. The 200 words dictionary of *Mixed1*, which uniformly covers all the characters in

the alphabet, has an average word length of 6.5 characters while the 20,000 words vocabulary has an average word length of 8.1 characters. The dictionaries which include all words of *Mixed2* have an average word length of 5.0 and 8.1 characters for the 200 and 20,000 words dictionary, respectively

The texts of *Sentence1* and *Sentence2* are adapted titles from the 'New Scientist' journal with an average length of five words. Figure 5.1 shows samples from *Mixed1* and *Mixed2*.

The Unipen character data shows much variation as a result of the different sources, digitizers, and writer nationalities. The dataset *Lower2* contains lowercase data from a less variable source and serves as comparison.

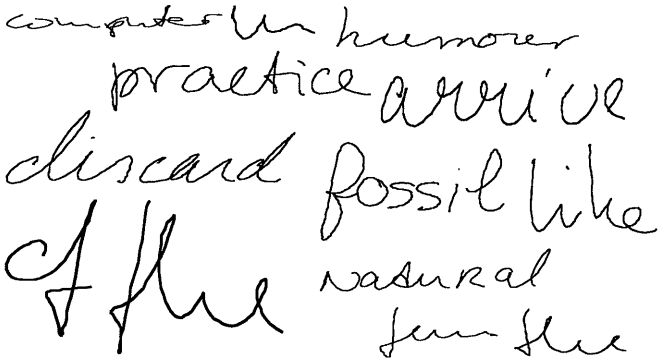


Figure 5.1. Data samples from test set *Mixed1* and *Mixed2*.

5.2 Parameter selection

To tune the hidden Markov models of the recognition system we must choose appropriate values for the parameters that are introduced in the models. Some parameters are chosen based on a-priori knowledge while others are chosen based on preliminary experiments.

Chosen parameters refer to *filtering*, *transition estimation*, *frame width*, *spatial overlap*, and *diagonal covariance lower bound*. *Filtering* of the handwriting signals is done with a moving average filter using a window width of five samples. This is done to remove quantization noise. We estimate the *transition probabilities* of the hidden Markov model during the training. The frame representation uses frames of 20 spatially equidistant points and aims at an average of nine frames per character. The number of segments of course depends on the data. Both frames and segments *overlap* spatially by 50%. Reason for this is to generate more feature vectors per character. This is especially beneficial for the segments because the number of segments per character is limited. To ensure the robust estimation of the density-specific *diagonal covariance*, we compute the covariance of both densities

and states and use 0.25 times the state-specific covariance as a lower threshold for the density-specific covariance.

Preliminary experiments determine the number of *states* per character and the number of *densities* per state. As a start, we determine the number of states for each label as $\#states = 0.75 \cdot \#observations$. This leads to a first estimate of seven and three states per character for frame and segment blocking, respectively. Note that the hidden Markov model topology, i.e., the number of states and the allowed transitions, determines a minimum number of observations to traverse the model. This minimum number of observations is $floor(\#states/2) + 1$.

First, we determine the parameters for character recognition. The number of states and densities for both frames and segments is determined using dataset *Digits1*. We use size-independent frames, i.e., the size dependency of the frames is eliminated by rescaling each character (see Subsection 4.1.2). We compute three positional features with time delays one, two and four as explained in Subsection 4.1.5. Diagonal covariance is assumed. The test results are presented in Table 5.2. As a result, we choose 6 states and 32 densities for character recognition tasks on the basis of a representation with size-independent frames. We use the same number of states for each character.

Table 5.2. The character error rate [%] of a digit recognition test with size-independent frames.

States	**Densities**			
	2	8	32	64
1	15.0	8.3	4.9	4.1
3	8.6	4.1	3.4	3.6
6	5.2	3.1	2.7	3.2
9	4.7	3.6	3.6	3.6

Table 5.3 shows the results of a similar experiment based on a representation using segments. A maximum of 32 densities per state is used. As result of this test, the length of the hidden Markov model is chosen equal to three states in a character recognition task. Note that the result of the hidden Markov model with four states suffers of the problem described above of not sufficient observations to process the model.

Next, we fixed the *word* recognition parameters at seven and five states for the representations based on frames and segments, respectively. This means that the hidden Markov models for word recognition are longer compared to the models used in character recognition. Reason is that we expect comparatively more observations due to the ligatures in cursive or mixed-style words. The same number of states is used for each character with a maximum of 32 densities per state.

Table 5.3. The character error rate [%] of a digit recognition test with representation based on segments and a maximum of 32 densities per state.

States	Densities
1	9.0
2	9.9
3	9.0
4	21.3

5.3 Character recognition

Subsections 5.3.1 to 5.3.7 explore the effect of various improvements of the representation and model. The objective is not to build the ultimate character recognition system but to test the benefits of representation and model enhancements in the context of character (and later word) recognition. The experiments start with the baseline representation of Subsection 4.1.3 and end with a recognition test on an alphanumeric task with 62 label classes using a full-fledged representation of up to 42 features.

5.3.1 Comparison of the basic representation

In the baseline experiment we employ size-independent frames, as discussed in Subsection 4.1.2, in order to obtain almost exactly 100 spatially equidistant points per character, resulting in nine overlapping frames. This eliminates variations due to a different number of observations per character, and should result in the best possible performance. The number of segments depends on the character complexity. This setup allow us to derive baseline results for three types of data, which serve as references for the other experiments.

Table 5.4. Baseline performance (in [%] character errors) for representations based on size-independent frames and segments.

Representation	Data Set	Baseline 13 features
si-frames	Digit2	5.7
	Upper	19.8
	Lower	22.3
segments	Digit2	14.4
	Upper	44.1
	Lower	45.2

The results of Table 5.4 show that the size-independent frames (si-frames) perform much better than the representation based on segments. The difference in character error rate is a factor of 2-3 for all the datasets.

Next, the baseline feature vector is augmented in two ways using delta and contextual features, as explained in Subsection 4.1.4. Because we have not yet determined the 'optimal' set of contextual features, a working set of three angular features with delays 1,2, and 4 is used.

Compared to the baseline representation, which uses local information to compute the feature vectors, both delta and contextual features model information spanning several observations. Because of the inclusion of extra information, the use of augmented feature vectors results in a higher performance.

Table 5.5. Comparison of augmented feature vectors for si-frames and segments in [%] character error rate.

Representation	**Data set**	**Delta 26 features**	**Contextual (1,2,4) 19 features**
si-frames	Digit2	4.1	4.2
	Upper	12.7	11.2
	Lower	15.2	14.9
segments	Digit2	9.3	8.5
	Upper	35.8	28.9
	Lower	37.0	30.9

Given the result of the augmented representations, we conclude that both representations with delta and contextual features are superior to the baseline representation. Compared to the baseline, the relative reduction of the character error rate is 25%-40% for the representation based on contextual features. The results for the representation with contextual features are slightly better compared to the results of the representation with delta features.

5.3.2 Size-independent versus average scale frames

The tests discussed in the previous subsections were executed with size-independent frames, i.e., the size dependency of the frames is eliminated by rescaling each written character. In the context of a word or sentence recognizer this is not possible because we do not know where the characters are in the data. Therefore, the results of the previous section are compared with frames scaled to the file-specific, average size of the training data in order to determine their performance difference. The tests are carried out for data set *Digit2* only. A lower performance is expected because the preprocessing uses less information about the handwritten input.

Table 5.6. Size-independent frames versus average scale frames. Results in [%] character error rate.

Representation	Baseline 13 features
si-frames	5.7
frames	21.2

Table 5.6 shows that the average frame representation results in almost four times more errors for dataset *Digit2*. A closer look at the errors and data reveals the cause of the problem. Basically, the scaling to average size results in few feature vectors for simple digits like a '1', which is often written as a vertical bar. In some cases, only three frames are produced. As explained in Subsection 4.1.2, these are not enough frames to process the hidden Markov model. It turns out that about 25% of the errors of Table 5.6 are caused by this problem. Because the same problem occurs with uppercase and lowercase data, these tests were skipped.

5.3.3 Covariance

While the previous tests concentrated on representation improvements, the current experiment compares three variations of Gaussian densities. So far, we assumed that the covariance matrix of the hidden Markov model is modeled as a *diagonal* and density-specific vector. Given N densities and feature vectors of dimension L, this results in a hidden Markov model with $2NL$ parameters.

The potential benefit of a *full covariance* model is that a more exact model will lead to better results if sufficient training data is available. The drawback is that such a model has much more parameters to be estimated compared to the diagonal covariance. Instead of $2NL$ parameters, a density-specific, full-covariance model contains $NL+NL^2$ parameters.

As an alternative, *tied covariance* is a way to reduce the model size. A tied covariance model uses a single covariance matrix or vector for all states and densities. A tied, diagonal covariance model contains only $(N+1)L$ parameters whereas a diagonal, density-specific covariance model uses $2NL$ parameters. The advantage of the tied covariance model is that its size is smaller at the cost of lower performance. Because of the fewer parameters, there is always sufficient data to reliably estimate the (co)variance of the tied covariance model.

The purpose of this experiment is to quantify the performance difference for the three types of models. It is expected that the most accurate model, i.e., full-covariance, performs best as long as we provide sufficient training data.

We compute all results with a baseline representation augmented with six contextual features (angular features with delays 1,2, and 4) using size-independent

frames. We use six states per character and 32 densities for each hidden Markov model.

Table 5.7. A comparison of three Gaussian (co)variance implementations. Results are given as [%] character error rate.

Data set	Diagonal	Full	Tied diagonal
Digit2	4.2	4.2	5.1
Upper	11.2	10.1	15.4
Lower	14.9	13.9	17.5

The results in Table 5.7 show that the best results for all datasets are obtained using the density-specific, full covariance model. This model yields an improvement of the character error rate of 0%-10% compared to the diagonal density-specific covariance model. However, it was explained above that this model contains much more parameters. The tied covariance model leads to an increase in character error rate of 17.5%-37.5%, which is a relatively high increase. Therefore, the remaining tests are carried out with the diagonal, density-specific covariance as this seems to be the best balance between performance and model size.

5.3.4 LDA transformation

In Subsection 4.1.6 we discussed the theoretical foundation of Linear Discriminant Analysis (LDA) and its application to handwriting recognition. This subsection presents two experiments to explore the use of LDA to reduce the character error rates and to investigate the relation between feature vector size after LDA transformation and character error rate.

In the first experiment, the performance improvement is measured while the size of the transformed feature vector remains the same before and after the transformation. The LDA transformation matrix is dataset specific. The results are summarized in Table 5.8.

Table 5.8. Recognition results with si-frames and LDA transformation in character error rate [%]. The relative improvement of the character error rate [%] compared with the untransformed data is given between brackets.

Dataset	Baseline 13 features	Delta 26 features	Contextual (1,2,4) 19 features
Digit2	4.7 (17.5)	3.5 (14.6)	3.1 (26.2)
Upper	19.0 (4.0)	14.0 (−10.2)	9.8 (12.5)
Lower	18.7 (16.1)	13.6 (10.5)	12.5 (16.1)

Table 5.8 shows that the improvement of the character error rate over the original

results presented in Subsection 5.3.1 is independent of the representation and is highest for digits, followed by lowercase and uppercase data. The character error rate decreases in all cases except for uppercase data in combination with delta representation. The average decrease in character error rate due to LDA is 19.4% for digits and 15.0% for lowercase data. The results obtained for the uppercase data seem to be different, which has to be investigated.

In a second experiment, the feature vector size is reduced by the LDA transformation to find out how much relevant information is contained in the transformed features. Therefore, the feature vector with contextual features is chosen as a start representation from which transformed features are stepwise removed. Table 5.9 summarizes the result with size-independent frames using the feature vector augmented with contextual features. It is assumed that a similar effect is possible for the other representations.

Table 5.9. Recognition results with si-frames and LDA transformation for feature vectors of different sizes. Results in character error rate [%].

Dataset	**Number of features**					
	19	16	12	8	2	0
Digits2	3.1	3.1	3.1	3.1	21.7	90
Upper	9.8	8.6	9.2	11.4	46.4	96.2
Lower	12.5	11.8	11.9	13.9	49.4	96.2

Table 5.9 shows that the feature vector size can be halved with no or only a slight increase of the character error rate. The table also shows that the character error rates for lowercase (5% relative decrease) and uppercase data (12% relative decrease) reach a minimum using 16 features. The error rate reduction is similar to the reduction achieved in the word recognition experiments by Dolfing & Haeb-Umbach [1997].

Additionally, Table 5.9 shows that a representation of only two features leads to roughly 80% correctly recognized digits and almost 50% correctly recognized characters. This is a surprising observation because the number of features is quite small. In contrast, the last column of Table 5.9 contains the character error rate for guessing (random selection), which is a lot worse.

5.3.5 LDA feature analysis

In contrast to the previous subsection which used the LDA transformation to achieve performance improvement, this subsection will concentrate on the use of LDA to compare the relative importance of features with special attention for contextual features.

So far, we only used feature vectors which are augmented with three angular

features for the delays 1,2, and 4. However, there are more types of contextual features which are useful in character recognition.

In the experiment discussed below, we determine which contextual features have a high discriminative value. The representation based on size-independent frames is augmented with 16 contextual features based on the delays d with value one to eight and angular features. The trace elements of the matrix $U_w^{-1}U_t$, used to compute the LDA transformation matrix as discussed in Subsection 4.1.6, are used to rank the features according to their discriminative value. The segment representation is similarly augmented with 15 extra features derived from the delays d with value one to five and using angular and size features. This results in 16 and 15 additional features for frames and segments, respectively. Other contextual features are useful for word and sentence recognition only and are discussed later.

The experiment is conducted with the datasets *Digit2* and *Lower* and two representations. The frame-based results are presented in Table 5.10 and the segment-based results in Table 5.11. For the sake of clarity, the baseline features as discussed in Subsection 4.1.3 are printed in italics.

Based on the results presented in Table 5.3.1, which show that a representation containing contextual features significantly decreases the character error rate, we expect that the discriminative value of the contextual features is large compared to the baseline features. Additionally, a representation based on a selection of the most discriminative features is tested.

In the case of size-independent frames, Table 5.10 clearly shows that delayed features have a higher discriminative value compared to the 13 baseline features. A close look at the results of the segment-based representation reveals that the angles of delay one and two, together with the length score, are most useful for discrimination.

After we have measured the discriminative values of the feature components, we select a subset of contextual features for an additional experiment. In Table 5.5 and Table 5.8, we used the three delays with value 1,2, and 4. If we choose a subset of contextual features with a similar number of delays, but selected on basis of their large discriminative values, then we should be able to improve the recognition results compared to Subsection 5.3.1 because more appropriate contextual features are used. We choose the delays 1,3,6 and 7 for size-independent frames and compute angular features. We choose the delays 1,2, and 4 for segments with angular and size relations.

New models are built and recognition tests are carried out using the contextual features chosen above. The results are summarized in Table 5.12. This table shows that the results without LDA are slightly better compared to Table 5.5. The results with LDA transformation for size-independent frames are poorer compared to Table 5.8. This indicates that a LDA transformation of angular features with

Table 5.10. Ranked feature set for size-independent frames.

Digit2		**Lower**	
Value	Feature name	Value	Feature name
2.91233	DelayedAngle-7-sin	2.46595	DelayedAngle-1-cos
2.80732	DelayedAngle-6-cos	2.05793	DelayedAngle-6-cos
2.33747	DelayedAngle-1-cos	1.9298	DelayedAngle-7-sin
2.21584	DelayedAngle-3-cos	1.89103	DelayedAngle-2-cos
2.1182	DelayedAngle-6-sin	1.85207	DelayedAngle-7-cos
1.84113	DelayedAngle-4-sin	1.58909	DelayedAngle-3-cos
1.79766	DelayedAngle-7-cos	1.58061	DelayedAngle-6-sin
1.71249	DelayedAngle-4-cos	1.56904	*EndAngle-cos*
1.65836	*EndAngle-sin*	1.52141	DelayedAngle-4-cos
1.56414	*Phi1-sin*	1.49844	*EndAngle-sin*
1.49045	DelayedAngle-1-sin	1.44179	DelayedAngle-1-sin
1.48935	DelayedAngle-2-cos	1.42977	DelayedAngle-8-cos
1.48438	*Phi3-sin*	1.41653	*Curvature*
1.48393	*Curvature*	1.40783	*StartAngle-cos*
1.45034	DelayedAngle-5-cos	1.39527	DelayedAngle-5-cos
1.44033	*EndAngle-cos*	1.36884	*AspectRatio*
1.43737	DelayedAngle-3-sin	1.34754	*Phi3-sin*
1.3696	*StartAngle-cos*	1.3261	*Phi1-sin*
1.32947	*AspectRatio*	1.29071	DelayedAngle-4-sin
1.23635	*StartAngle-sin*	1.2857	DelayedAngle-3-sin
1.22629	DelayedAngle-8-sin	1.27067	DelayedAngle-8-sin
1.20388	DelayedAngle-2-sin	1.24001	DelayedAngle-5-sin
1.20132	DelayedAngle-5-sin	1.18124	*PenDown*
1.16272	*Phi2-sin*	1.17912	DelayedAngle-2-sin
1.10006	*PenDown*	1.15014	*StartAngle-sin*
1.08633	*Phi2-cos*	1.12472	*Phi2-cos*
1.01717	*Phi3-cos*	1.09036	*Phi1-cos*
1.01456	*Phi1-cos*	1.08317	*Phi3-cos*
≈ 1	DelayedAngle-8-cos	1.06337	*Phi2-sin*

Table 5.11. Ranked feature set for segments.

Digit2		**Lower**	
Value	Feature name	Value	Feature name
1.83667	DelayedLength-1	2.77742	DelayedLength-1
1.47304	DelayedLength-2	2.00185	DelayedAngle-1-cos
1.42182	DelayedAngle-2-cos	1.78699	DelayedLength-2
1.3895	DelayedLength-4	1.40618	DelayedAngle-2-cos
1.37518	*Phi3-sin*	1.3682	*EndAngle-cos*
1.27913	DelayedAngle-1-sin	1.35018	DelayedAngle-2-sin
1.27674	DelayedAngle-4-sin	1.34301	DelayedAngle-1-sin
1.22026	*Phi1-sin*	1.33331	*Phi2-sin*
1.21226	*Curvature*	1.25331	*Phi3-sin*
1.21188	*EndAngle-cos*	1.2409	DelayedLength-3
1.15227	DelayedAngle-1-cos	1.24068	*Phi1-sin*
1.14375	*Phi2-sin*	1.23484	*AspectRatio*
1.13711	*StartAngle-cos*	1.21158	*Curvature*
1.12317	DelayedLength-3	1.18632	*StartAngle-cos*
1.12082	DelayedAngle-4-cos	1.18056	DelayedLength-4
1.1124	DelayedLength-5	1.17757	DelayedAngle-3-sin
1.1	DelayedAngle-5-sin	1.14821	DelayedAngle-3-cos
1.09646	*Phi1-cos*	1.13185	*Phi2-cos*
1.08627	*Phi2-cos*	1.11566	*StartAngle-sin*
1.08516	*StartAngle-sin*	1.11258	*Phi3-cos*
1.08293	DelayedAngle-5-cos	1.11105	*PenDown*
1.0781	DelayedAngle-3-sin	1.07263	*EndAngle-sin*
1.07462	*EndAngle-sin*	1.07231	DelayedAngle-4-cos
1.04457	DelayedAngle-3-cos	1.06611	DelayedAngle-4-sin
1.03733	*AspectRatio*	1.0544	*Phi1-cos*
1.02739	*PenDown*	1.01587	DelayedLength-5
1.01433	*Phi3-cos*	1.0142	DelayedAngle-5-cos
≈ 1	DelayedAngle-2-sin	1.0037	DelayedAngle-5-sin

Table 5.12. Recognition results of representations augmented with a subset of the most discriminative contextual features. Results are given with and without LDA in character error rate [%].

Dataset	si-frames		segments	
	No LDA	LDA	No LDA	LDA
Digit2	4.7	3.2	7.7	7.2
Upper	10.7	10.0	27.8	27.7
Lower	14.3	15.3	28.9	26.1

size-independent frames with delays 1,2, and 4 produces better recognition results compared to a LDA transformation of angular features with delays 1,3,6 and 7.

5.3.6 Combined delta and contextual features

In Subsection 5.3.1, we investigated the effect of delta and contextual features. In Subsection 5.3.4, we explored the benefits of the LDA transformation. The question remains whether the effect of these improvements can be stacked to achieve an even better recognition system. Therefore, the subject of this subsection is to combine delta and contextual features with an optional LDA transformation. The representation based on size-independent frames is used together with in combination with four delays (1,3,6,7) and angular features resulting in a feature vector of $2 \cdot (13 + 4 \cdot 2) = 42$ features.

The information modeled with delta and contextual features is different. While the contextual features capture explicit, structural knowledge, the delta features model information using derivatives of the observations. Because of the different information content, combination is useful to achieve a further reduction of the character error rate.

Table 5.13. Results for combined delta and contextual features with and without LDA transform in [%] character error rate. The relative character error rate improvement of the LDA transformed feature vector with only 30 features compared to the untransformed vector in column one is given between brackets.

Dataset	si-frames		
	42 features	42 feat. + LDA	30 feat.+ LDA
Digit2	3.1	3.1	2.8 (9.7%)
Upper	8.4	8.9	7.9 (6.0%)
Lower	12.2	11.7	10.8 (11.5%)

A comparison of the experimental results of Table 5.13 with the results in Table 5.5 for size-independent frames and contextual features shows that stacking the delta

and contextual features reduces the character error rate by 18% - 25%. Those cases where the original and transformed number of features remain the same do not yield an improvement, e.g, the second column in Table 5.13. However, Table 5.13 shows that the use of an LDA transformation to reduce the feature vector size does result in an improvement for all datasets. With an LDA transformed feature vector of 30 components, an average of 9% improvement is achieved compared to the full-size, untransformed representation of column one of Table 5.13.

Table 5.13 is based on the best scored characters. Because the correctness of the best hypothesis of a classifier is only part of its characteristic, we extend the description with the correctness of one of n best hypotheses. Figure 5.2 plots the top-n scores for the size-independent frames in combination with the LDA transformation and a feature vector size of 30 features after transformation. Figure 5.2 shows that the performance of the classifier rapidly approaches more than 98% correct if more than the four best hypotheses are taken into account. This means that, in combination with extra context sources, we can substantially improve the effectiveness of the classifier.

Finally, the confusion matrices for each of the tested datasets are given in the Tables 5.14, 5.15, and 5.16. The confusion matrix compares the expected, correct recognition result with the best hypothesis of the classifier. We concentrate on the incorrect classifications, i.e., we have removed the correct classifications from the diagonals of Table 5.14, 5.15, and 5.16.

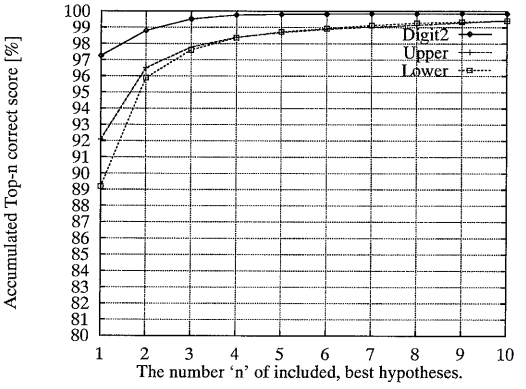


Figure 5.2. Top-n character score [%] versus the number of n best hypotheses. The Results are given for the LDA transformed representation (30 features) based on the combined delta and contextual features.

5.3.7 Alphanumeric recognition

In the previous subsections, representations and modeling issues have been studied based on the results for three separate datasets. This subsection discusses what hap-

Table 5.14. Digit confusion matrix. Correct labels vertically, recognized labels horizontally. Dark areas indicate entries with high confusion. The text of the entry with the highest confusion is inverted.

	0	1	2	3	4	5	6	7	8	9
0		2		1	1	1	3	2	5	5
1	2		1	2	4	1		6		
2								3		
3			1			3		1	4	
4	2	12				1		5	1	1
5		1							4	1
6	2	1			5					
7	4	5	3							
8	4	2		1						
9	7				4				3	

pens if these three datasets have to be recognized simultaneously, which increases the number of label classes to 10+26+26=62. The increase in label classes also leads to an increase of potential confusions because similar or identical allographs are sometimes used for different symbols!

We estimate the expected extra confusion based on our a-priori knowledge of character allographs. At least two characters have the same allograph with three different meanings. These are the allographs /o/ and /l/ which are used for three labels each that are (o,O,0) and (1,i,l), respectively. In other words, without context knowledge, the correct classification of these characters is extremely difficult or impossible.

There are 12 other allographs (/c/, /k/, /m/, /n/, /p/, /s/, /u/, /v/, /w/, /x/, /y/, /z/) for which uppercase and lowercase allographs can be identical. Under the assumption that for 50% of all allographs there exists a form that might introduce confusion, and that 50% of these lead to a failure in recognition, we expect a character error rate of at least

$$\frac{((2\cdot 3)+(12\cdot 2))\cdot \frac{1}{2}\cdot \frac{1}{2}}{62} = 12.1\%.$$

In other words, the writer-independent character error rate is at least 12%. This is a rough estimate but indicates that writer-independent, alphanumeric recognition is far from trivial.

In the experiment reported in Table 5.17, the representation with delta plus contextual features used in Table 5.13 is applied to the alphanumeric recognition task. The hidden Markov models of 62 characters are used simultaneously in the

Table 5.15. Uppercase confusion matrix. Correct labels vertically, recognized labels horizontally. Dark areas indicate entries with high confusion. The text of the entry with the highest confusion is inverted.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A						4	16	16	7							1			2	1	1			1		
B	1			2		1			1	2				4		1		4	2						1	
C					1	1	1		1			6														
D	1		1					1	1						1	4		1								
E						22	20	1	9		2	3								3	1	1		1		1
F				1	2			7	10	2	1		2				1	5	3	2				1		
G			2		1	1			3	1					5		1		1						1	
H	4					6			8		2			2				1		1						
I			1	1	4	6		1		10	1	8					2		3	2		1		1	1	1
J				1					2				1				1			4						
K			3		1	2		2	5								1	1					1		1	
L			2		1	1			3																1	
M						2		6	2					1												
N	2					1		7	5				5					2					7	1		
O			2				2			3							7			1	5					
P				13					3															1		
Q					1		1		2			1			4											
R	4	1		5		1			3		4			1		2										1
S	3				1		1			4	1	2													1	
T				1	2	8			24	11		1							1					1	4	
U	1				2						1		3		3						38				2	
V								1	1						1					1	5			1	1	
W	1					2		2	2			1	1	4							1					
X					2				6	1	1						1			3					2	
Y					1	3	2		9	1		1	1	1			1			2				2		1
Z		2			1	2			1	1		1					1	1	1							

Table 5.16. Lowercase confusion matrix. Correct labels vertically, recognized labels horizontally. Dark areas indicate entries with high confusion. The text of the entry with the highest confusion is inverted.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a			7	15	5	2	4	8	1	1	5		1	4	17		12	7	1	1	55	1	3	4	1	17
b						3	1	7	4		4	5			1	6			4			1			2	1
c	1				10	1			6			1			2			3	2					3		
d	4		2			1	1		2	7	1	5		1	5		2			3	12			1		
e	3		61	1		4			9		1	69				4		1		7				7		2
f		2			1		2	5	1	4		1				4	4	2		15					3	
g						1			1	5	1						10	1	13						14	1
h	1	11				2			6		7	9		21		1		3		5	4	1				
i	1		17		1	1	1			3		99		2	1	3	3	8		14	5	1		1	2	1
j							3		2									1		2	1			1	3	2
k		9	9	1		3		37	6			5		6		1		6		6	5			2		
l			15	2	13	5		1	21	1								1		6		1		2	2	
m		2						1	1		3			11									5			
n	6	2						47	2		12		42			4		15		1	22	7	16	1		
o	22	2	15	1	2	1	1		4	6						1	2	1	5	1	8	14	1			2
p					1			3	2		7	3	6					7	3	4		1		2		
q	8						6					1			1	4									3	1
r		1	2	1	7	5	2	4	8	3	5	2		39		8	1		2	8	1	53		8	2	13
s	1	1	1			3	5	1	4	6		1		1	1			3		3					6	4
t		7			1	45		3	20	3	1	25				4		3			1	2		14	8	2
u	9	1		4					2		1		5	14	13					1		67	9		2	
v					1	1					1	1	1	2	1		1	7		1	4		1	5		
w	1								1		2		3	1		1					8	3		1		
x	1				1				1		1	3	2	2						2			1		2	
y							21		5	6		6			1	1	1		1	8	2			6		4
z			1			1	9			1									3							

alphanumeric classifier. The classifier is applied to three separate datasets. The corresponding results are shown in Table 5.17. The average performance over all three datasets is reported in the 'combined' row. Note that the LDA transformation matrix is computed based on data from all 62 classes.

Table 5.17. Results for the alphanumeric recognizer with combined delta and contextual features.

Dataset	**si-frames**	
	No LDA	**LDA, 30 feat.**
Digit2	19.3	16.3
Upper	24.4	24.9
Lower	29.3	24.1
Combined	26.6	23.1

Table 5.17 shows an average character error rate of 26.6% without LDA, and 23.1% with LDA. This can be compared with the character error rate reported by Kassel [1995], who found character error rates between 22.8% and 19.0% for a similar, 62 class problem using a hidden Markov model classifier.

Our results have been obtained on a part of the Unipen dataset while Kassel [1995] uses an on-site collected dataset. Because one of the prime objectives of the Unipen dataset is to collect data from various writers, sources and nationalities, it is likely that the Unipen dataset is more diverse.

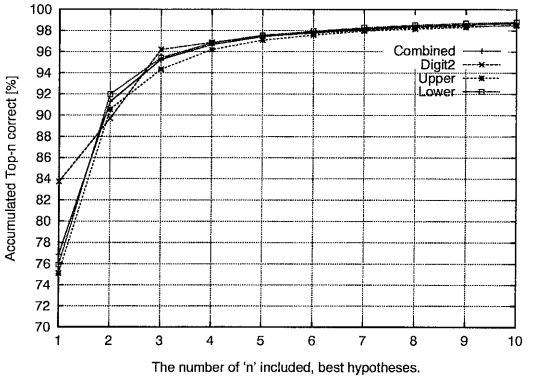


Figure 5.3. Top-n alphanumerics correct [%] versus number of character hypotheses.

Table 5.17 characterizes the hidden Markov model classifier with the correctness of the most likely hypothesis. We extend the characterization of the classifier with the top-n results on each dataset presented in Figure 5.3. This figure shows the number of times that the correct classification is part of the n most likely hypotheses.

5.3.8 Discussion

After the extensive testing discussed in the previous subsections, the remaining question is how the measured performance compares with the performance of classifier systems reported in the literature. To this end, we first summarize the human performance and our best results. The second part of this subsection is somewhat speculative and discusses our results in relation with other character recognition results.

LaLomia [1994] studied the character error rate that humans find acceptable for a computer recognizing handwriting and found an upperbound of 3% error rate. In Section 1.4, a quote from the literature referred to a human character error rate of 4.4% - 3.2% for handprinted, discretized characters. Kassel [1995] measured a human character error rate of 18.3%-15.9% (in word context) and 23.5% -21.1% (without context) on a 62 class, alphanumeric recognition task.

We assess the significance of the achieved recognition results on basis of the arguments in Subsection 4.4.1. Character error rates between 2.8% and roughly 30% are obtained for different tests. Given the properties of the training and test data as explained in Section 5.1, we can argue that even the best results (2.8%) with smallest test set (4506 digits) yield reliable results.

Table 5.18. On-line character recognition results from the literature. All results given in character error rate (CHER). The following abbreviations are used: D=Digits, L=Lowercase, U=Uppercase, S=Special symbols, MS=Microsoft, WI=Writer-independent, WD=Writer-dependent. (*)= 2.3% for digits only.

Author(s)	Classes	Type	#Writers	CHER
Kassel [1995]	62 (DUL)	WI	150	19.0 - 22.8
Bellegarda et al. [1994]	81 (DULS)	WI	8+4	22.7
Guyon et al. [1991]	36 (DU)	WI	250	3.4 (*)
Chang et al. [1994]	26 (L)	WI	16	9 (MS), 5 (CIC)
Yang [1995]	10 (D), 26(L)	WD	3	6.7 (D), 8.5 (L)
Tappert [1991]	44 (DUS)	WD	9	2.8

Our results have been obtained using a part of the Unipen training dataset, which contains diverse data from different nationalities, digitizers and persons. Although no direct results are available for comparison, we expect higher error rates compared to experiments reported in the literature because of the highly diverse data. The best writer-independent character recognition results obtained for the data of 500 to 2000 writers achieved in our study show a character error rate of 2.8% for the digit data, 7.9% for the uppercase data, 10.8% for the lowercase data (all Table 5.13), and 23.1% for all data using the combined, alphanumeric recognizer in Table 5.17. A character error rate result of 9.8% was achieved on *Lower2*. In

future, results on the Unipen development and test data will become available enabling meaningful comparisons.

The comparison of recognition results with results from the literature as shown in Table 5.18 is a hazardous task because of differences in data, digitizers and other factors. Therefore, the next paragraphs are somewhat speculative.

The obtained error rates on alphanumeric data are a bit higher than those reported by Kassel [1995] and Bellegarda, Bellegarda, Nahamoo & Nathan [1994]. We tested with 62 classes whereas Bellegarda et al. [1994] used 81 character classes. It should be noted that the used training and test data originates from 10 to 100 times more different writers. Compared to Guyon, Albrecht, LeCun, Denker & Hubbard [1991], the error rate on digit data is slightly worse. It is interesting that our results for uppercase data (7.9%) is far worse than the reported 3.4% they reported for digits plus uppercase data. Fujisaki, Beigi, Tappert, Ukelson & Wolf [1992] reported writer-independent results for discrete handwriting, obtained with a dedicated handprint recognizer, where the character error rate ranges from 2.8% for digits to 9.3% in an 82 class recognition task. They found that 50% of the errors are due to case and shape confusions. While the digit error rate of our generic system compares fine, the alphanumeric results are worse.

Because the number of writers in our experiments is 10 to 100 times larger than reported in other tests, it is likely that the number of character allographs has also increased. If we assume an increase proportional to $\sqrt{\#writers}$, than we have 3 to 10 times more different character allographs to recognize. The number of potential confusable characters increases accordingly. Even if the number of additional character allographs is overestimated, the number of character confusions increases. This effect of an enlarged train and test set raises the question whether this is a good way of testing a classifier. The minor character error rate difference between the diverse *Lower* dataset and the more homogeneous *Lower2* dataset is an indication that adding more training data might not improve the classifier performance. Without additional experiments, we can only speculate on the cause of this problem, i.e., too much similar allographs, a failing representation or other causes. A pragmatic approach would be to cluster the training and test data, e.g., according to country.

The character recognition results have been obtained with a hidden Markov model classifier and data-driven training. No special heuristics are tuned to the data for optimal recognition. Additionally, the classifier is the same, though trained on different data, compared to the classifier for hidden Markov model word recognition in the next section. In this light, it is remarkable that such a classifier achieves recognition results comparable or slightly worse to literature results.

5.4 Word recognition

This section explores representation and model improvements for handwritten word recognition. Without lack of generality, we may concentrate on the recognition of mixed-style words. This writing style is the most difficult to decipher. The objectives are to employ handwriting-specific knowledge into representation and modeling, to compare different techniques and to improve recognition results.

First, we explore an alternative approach to explicit scaling of handwritten words in Subsection 5.4.1. Next, representation improvements are introduced in Subsection 5.4.2 to Subsection 5.4.5. These improvements include the use of delta and contextual features similar to those in the character recognition studies. However, the modeling on the basis of contextual features is extended to take advantage of the larger word context. Finally, the hidden Markov model structure and models are modified in order to take advantage of prior handwriting knowledge. This is discussed in Subsection 5.4.6 and Subsection 5.4.7.

5.4.1 Scalability

Next to writing speed and sampling rate, writing size is often explicitly normalized as demonstrated by Nathan, Beigi, Subrahmonia, Clary & Maruyama [1995], Beigi, Nathan, Clary & Subrahmonia [1994] and Weissman, Schenkel, Guyon, Nohl & Henderson [1994]. A bounding box of a handwritten word or word body is determined and used to normalize the writing size. This approach is very suitable for isolated word recognition tasks in which all the input is available before normalization, but it is less suitable for sentence recognition in which writing and recognition take place simultaneously. As an alternative to the normalization of writing size, we investigated size-independent representations.

As explained in Subsection 4.1.3, the feature vectors for frames and segments contain 13 low-level, size-independent features. In addition, contextual features are employed. We use four angular features, which are explained in Subsection 4.1.5, obtained with different delays for frames and segments. We expect the segment representation to be writing size invariant because the boundary points of the segments, i.e., points based on velocity inversions, are invariant to handwriting size (see Section 2.2).

In order to investigate writing size dependence, we asked 10 writers to write a set of 50 words in four different sizes with scales 0.5, 1, 2, and 4 where scale 1 corresponds with normal writing size for most writers. The other scales indicate relative multiplication factors. Note that the test set *Mixed1* contains the scale 1 data. Writers were instructed to write in lowercase but unconstrained otherwise. The resulting set of four times 500 words is represented by either frames or segments. The experiments are done with both a 200 and 20,000 word vocabulary.

The hidden Markov models for characters contain six and five states for frames and segments representation, respectively.

Table 5.19. Comparison of frames and segments for four different writing sizes. The table shows the word error rate in [%]. Left: the experiment with a 200 word dictionary. Right: the experiment with a 20,000 word dictionary.

Repr.	**Writing size**			
	0.5	1	2	4
Frame	17.9	1.0	31.2	97.3
Segment	3.2	3.2	2.0	2.9

Repr.	**Writing size**			
	0.5	1	2	4
Frame	44.5	9.8	66.3	100.0
Segment	17.5	16.7	14.7	18.2

The results presented in Table 5.19 clearly show that the segments are essentially independent of the writing size while frames show a better peak performance, i.e., only 1% word error rate (WER) for the normal size data. The peak performance difference is noticeable. Table 5.19 only shows the mean word error rate over all written words and writers. If we compute the word error rate for each writer, then we find that the standard deviation of the writer-specific error rates with segment-based representation is roughly 2% for the 200 word vocabulary experiment and 10% for the experiment with a 20,000 word vocabulary.

Next, the data in Table 5.19 can be used to infer the requirements for a frame-based, explicit size normalization intended to outperform the segment representation. We can draw two graphs on the basis of the results of Table 5.19 for the experiment with 20,000 word vocabulary. These graphs show the relation between writing size and word error rate for frames and segments, respectively. This allows the determination of the two intersection points of the frames and segments graphs. The intersection points will indicate a word error rate of approximately 17%. With linear interpolation, we derive the corresponding writing sizes which are 0.9 and 1.13. In other words, if we can guarantee that the explicit size normalization scales the word to a size x in the range $0.9 \leq x \leq 1.13$, then the frame-based representation will work better compared to the segment-based representation for the 20,000 word vocabulary experiment in Table 5.19. Because we do not have such a technique, most experiments discussed in the next sections use a segment-based representation.

Finally, Table 5.19 shows that the word error rate for frames becomes 100% if the writing size becomes four times the normal size. This is not surprising since there was no such data in the training set.

5.4.2 Delta and contextual features

In Subsection 5.3.1, we studied the advantages of delta and contextual features for character recognition tasks. This subsection explores the advantages of contextual features in a word recognition task. It is expected that the augmented feature

vectors outperforms the baseline representation also in a word context.

Based on the *Mixed1* data, the word error rate of a baseline representation with 13 components is compared to feature vectors which included delta features (baseline + 13 deltas) and contextual features (baseline + 6 angular features), respectively. The six angular features are constructed as explained in Subsection 4.1.5 from three angles with the delays 1,2, and 4, representing positional relations.

Table 5.20. The word error rates in [%] for segment type of feature vectors for a 200 word and a 20,000 word dictionaries. The mean error rate is indicated as μ while the between-writer standard deviation is indicated as σ.

Vocabulary size	**Number of features in representation**					
	13 Baseline		**13+6 Contextual =19**		**13+13 Delta =26**	
	μ	σ	μ	σ	μ	σ
200 W	7.5	7.0	2.4	2.3	2.7	2.0
20,000 W	28.0	17.1	13.3	10.7	18.1	11.7

Table 5.20 shows that the augmented feature vector clearly outperforms the baseline representation, and that contextual features perform better than delta features. In the case of the contextual features, fewer additional vector components are required to attain the performance improvement. The relative word error rate reduction for the 20,000 word vocabulary test is 52.5% for the contextual features and 35.4% for the delta features. This means that the additional six contextual features halved the word error rate!

Compared to the character-based experiment in Subsection 5.3.1, the relative error rate reduction is higher. The measured word error rate with a 20,000 word vocabulary compares well with the literature results by Manke et al. [1995], Schenkel et al. [1995] and Nathan et al. [1995]. We measured a between-writer standard deviation of $\sigma = 10.7$. This also compares well to the experiment of Schenkel et al. [1995] where word error rates between 5% and 40% were reported.

5.4.3 Linear Discriminant Analysis

We investigated two aspects of the LDA transformation. First, the performance improvement due to the LDA transformation is investigated while the feature vector size before and after transformation remains the same. A similar improvement as in the character recognition experiment with LDA transformation (Subsection 5.3.4) is expected. Second, the error rate as a function of the feature vector size after transformation is investigated.

First, Table 5.21 summarizes recognition results with LDA in an otherwise unchanged experiment. Comparison of the results in this table with the results

without LDA in Table 5.20 reveals a clear improvement.

Table 5.21. The word error rates [%] for segment type of LDA transformed feature vector for a 200 word and a 20,000 word dictionary. The mean error rate is indicated as μ while the between-writer standard deviation is indicated as σ. Full dimension of LDA transformed feature vector.

Vocabulary size	Number of features before LDA					
	13 Baseline		13+6 Contextual =19		13+13 Delta =26	
	μ	σ	μ	σ	μ	σ
200 W	3.7	3.1	2.2	1.6	2.6	2.1
20,000 W	19.0	13.8	12.7	8.5	13.7	10.2

Second, Figure 5.4 shows the relation between the number of features remaining after transformation and the word error rate. For both delta and contextual features, a minimum word error rate is reached after about four features have been dropped which corresponded to 10%-20% of the feature vector. This result is similar to Table 5.9 for character recognition with the LDA transformation. The best result corresponds to a 11.2% word error rate for the transformed feature vector with contextual features.

A relative reduction in word error rate of 12% is achieved compared to the LDA transformed result with full feature vector. Compared to the result before transformation, the total gain obtained by LDA in case of the representation with contextual features is a word error rate reduction from 13.3% to 11.2% (16% relative reduction) which compares well with a similar experiment in speech recognition [Haeb-Umbach & Ney, 1992].

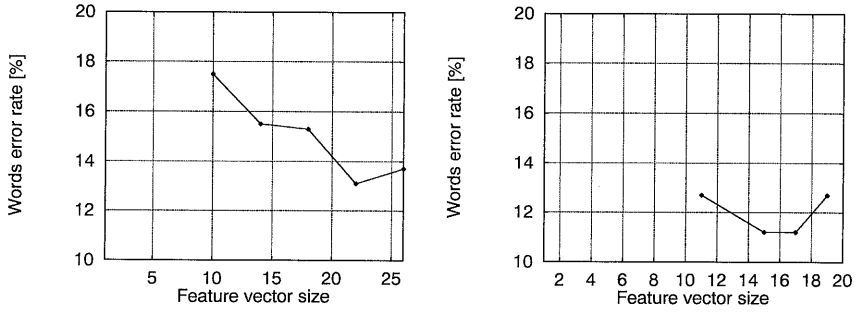


Figure 5.4. Word error rate [%] versus feature vector size after the LDA. A 20,000 words vocabulary is used. Left: delta features prior to LDA. Right: contextual features prior to LDA.

5.4.4 Contextual features analysis

After the determination of the performance benefits due to LDA, we use LDA to test the discriminative information of contextual features as explained in Subsection 4.1.6. We earlier used the same procedure in Subsection 5.3.5 to rank the features in a character recognition context and found that the contextual features have a higher discriminative value compared to the baseline features.

In addition to the earlier positional or angular features, we model *size*, *contour*, and *overlap* relations on basis of the contextual features, and integrate them into the representation. Both frame and segment-based representations are augmented with additional features.

The experiment is conducted for both frames and segments. All contextual features discussed in Subsection 4.1.5 are used to train a model based on the *Mixed-train* training data. The theory of LDA is used to compare the discriminative information of each feature using the hidden Markov model states as class definition. As explained in the previous chapter and by Fukunaga [1990], the trace elements of the matrix $U_w^{-1}U_t$, are used as an indication of discriminative value.

In the frame-based representation, we use eight delays to augment the representation with angular (8x2) and overlap (1) features resulting in 17 extra features. This gives a feature vector of 13 + 17 = 30 features. The size feature is omitted since the length of all frames is by definition the same. The segment-based representation is augmented with 6 delays resulting in angular (6x2), size (6x1), contour (2), and overlap (1) features. This results in a feature vector of 13 + 21 = 34 features. The maximum delay is chosen just longer than the number of states of a character. Hence, the features also model contextual effects. We refer to these feature vector realizations as the *All-delay* representations.

The experiments result in a feature ranking based on their discriminative value presented in Table 5.22. The baseline features as discussed in Subsection 4.1.3 are printed in italics. First, we observe that the contextual features generally have a higher discriminatory value than the 13 baseline features. Second, the most discriminative contextual features are the angular/positional features which relate the relative positions of the center-of-gravities of feature vectors. Third, the *PenDown* feature is scored best in both representations based on frames and segments. This is in contrast to the similar experiment in character recognition context in Subsection 5.3.5. The difference is attributable to the fact that we have a lot of discrete-style, handwritten words in the training set. Fourth, the example of the segment representation shows that the contextual *size* features are roughly equally discriminative compared to the *angular* features. Unfortunately, the more complex *contour* and *overlap* features have only moderate discriminative value.

The results of a recognition test in which the *All-delay* representation is used

Table 5.22. Significance of (contextual) features in the original space based on the LDA transformation matrix.

Segments		**Frames**	
Value	Feature name	Value	Feature name
1.47475	*PenDown*	1.74517	*PenDown*
1.38932	DelayedAngle-1-cos	1.5935	DelayedAngle-2-cos
1.36003	DelayedLength-1	1.57086	DelayedAngle-1-cos
1.29699	DelayedLength-2	1.42215	DelayedAngle-1-sin
1.27216	DelayedAngle-1-sin	1.39946	*EndAngle-cos*
1.25464	*EndAngle-cos*	1.34718	DelayedAngle-8-cos
1.25082	DelayedAngle-2-cos	1.32532	DelayedAngle-3-sin
1.24088	DelayedAngle-4-cos	1.3162	DelayedAngle-3-cos
1.21928	DelayedAngle-4-sin	1.31514	*EndAngle-sin*
1.21493	DelayedLength-3	1.31064	Overlap
1.21254	*Phi3-sin*	1.30495	*Curvature*
1.20445	Overlap	1.28736	DelayedAngle-8-sin
1.17843	DelayedAngle-3-cos	1.27246	DelayedAngle-5-sin
1.17494	Contour-top	1.26926	DelayedAngle-4-cos
1.17031	DelayedAngle-3-sin	1.25112	DelayedAngle-6-cos
1.16256	DelayedAngle-6-cos	1.21134	DelayedAngle-5-cos
1.15904	DelayedAngle-5-sin	1.2007	DelayedAngle-4-sin
1.15631	DelayedAngle-2-sin	1.19738	*Phi3-sin*
1.14972	*Curvature*	1.1784	*Phi1-sin*
1.12948	DelayedAngle-6-sin	1.14727	DelayedAngle-6-sin
1.12732	*Phi1-sin*	1.14544	DelayedAngle-7-cos
1.10453	DelayedLength-4	1.1427	*AspectRatio*
1.09407	DelayedAngle-5-cos	1.14119	DelayedAngle-2-sin
1.08201	*StartAngle-cos*	1.11643	DelayedAngle-7-sin
1.07813	*AspectRatio*	1.10391	*StartAngle-cos*
1.07111	*EndAngle-sin*	1.08495	*StartAngle-sin*
1.06999	*Phi2-sin*	1.06843	*Phi2-sin*
1.06907	DelayedLength-6	1.00046	*Phi3-cos*
1.0666	DelayedLength-5	≈ 1	*Phi2-cos*
1.05701	*Phi1-cos*	≈ 1	*Phi1-cos*
1.05231	Contour-bottom		
1.04627	*Phi2-cos*		
1.02937	*Phi3-cos*		
1.02907	*StartAngle-sin*		

with both frames and segments on the *Mixed1* test set are presented in Table 5.23. Although these results are comparable with Subsection 5.4.2 where we used fewer delayed features, this comparison is of limited value. The comparison shows a word error rate for frames that is somewhat higher than before and a word error rate for segments which is a little better. This is most likely due to the increased number of features which complicates the training process. More training iterations probably have to be carried out until the training converges.

Table 5.23. Results in word error rate [%] with a representation based on all the contextual features. The results are obtained with both frames and segments using test set *Mixed1*. The mean error rate is indicated as μ while the between-writer standard deviation is indicated as σ.

Representation	**Vocabulary size**			
	200		**20,000**	
	μ	σ	μ	σ
Frames	1.6	1.5	10.0	9.6
Segments	2.4	2.6	14.7	8.9

5.4.5 Word recognition with delta and contextual features

The previous subsections investigated whether a feature vector augmented with either delta or contextual features improves the recognition of handwritten words. This subsection investigates whether the combination of delta and contextual features brings an additional improvement in a word recognition context similar to Subsection 5.3.6 in a character context.

After checking the discriminative values of all features, the segment representation is augmented with not only angular features but also with size features for the delays 1,2, and 4. This representation is called *Contextual-Angle-Length-(1,2,4)* and contains a total of 22 features. In an additional experiment, this representation is combined with delta features to investigate whether the improvement due to delta and contextual features can be stacked. This representation contains 44 features and is called *Delta-Contextual-Angle-Length-(1,2,4)*. Finally, an additional training with the LDA transformation is carried out to obtain an additional improvement. The results with augmented representations are summarized in Table 5.24.

First, we observe that the *Contextual-Angle-Length-(1,2,4)* representation does not give an improvement over *Contextual-Angle-(1,2,4)*. Only the standard deviation is smaller. Second, the stacked delta plus contextual representation leads to an improvement of the word error rate from 13.3% to 12.5%. The LDA transformation combined with the removal of 10%-20% of the features (see Subsection 5.4.3)

Table 5.24. Recognition results in word error rates [%] with segments for various, augmented representations. All the tests are carried out with the *Mixed1* test set and vocabularies of 200 and 20,000 words. The mean error rate is indicated as μ while the between-writer standard deviation is indicated as σ. Abbreviations: CAL = Contextual-Angle-Length. DCAL = Delta-Contextual-Angle-Length.

Representation	**Vocabulary size and LDA**					
	200, no LDA		**20K, no LDA**		**20K, LDA**	
	μ	σ	μ	σ	μ	σ
Baseline	7.5	7.0	28.0	17.1	19.0 (13 feat)	13.8
CA-(1,2,4)	2.4	2.3	13.3	10.7	12.7 (19 feat) 11.2 (15 feat)	8.5
Delta	2.7	2.0	18.1	11.7	13.7 (26 feat)	10.2
All-delay	2.4	2.6	14.7	8.9	-	-
CAL-(1,2,4)	2.7	2.3	13.3	8.2	-	-
DCAL-(1,2,4)	2.0	2.2	12.5	9.3	11.8 (32 feat)	6.7

further improves the word error rate from 12.5% to 11.8%. This is almost as good as the word error rate of 11.2% obtained for *Contextual-Angle-(1,2,4)* with LDA and 15 features. Third, it is interesting that the combined, untransformed *DCAL-(1,2,4)* performs better than *Contextual-Angle-(1,2,4)* but the situation is reversed after using the LDA transformation and feature vector size reduction. The most likely explanation of this behavior is that 32 is not the optimal number of features for the LDA transformed *DCAL-(1,2,4)* representation. It is likely that, instead of removing 10%-20% of the features based on earlier experimental results, a more complete evaluation of the relation between error rate and number of features after the LDA transformation could find an even lower error rate.

In summary, we showed that representation improvements in a word recognition test using a vocabulary of 20,000 words yield a total improvement of the word error rate from 28.0% to 11.2% which is a 60% error-rate reduction.

5.4.6 Pause and backspace

In the previous subsections, we did not pay attention to the exact structure of the hidden Markov model. We have been using a left-to-right model where concatenated character models form a word model. However, a simple concatenation of hidden Markov models is only a coarse model of a handwritten, mixed-style word. Possible improvements have been discussed in Subsection 4.2.2.

This subsection investigates the effect of the 'pause' and 'backspace' models in combination with the data sets *Mixed1* and *Mixed2*. The vocabulary sizes in the recognition experiments range from 200 to 20,000 words. We use a segment-based

representation with angular features and the delays 1,2, and 4. It is expected that the 'pause' model reduces the overall word error rate as argued by Cho et al. [1995]. The effect of the 'backspace' model is not predictable due to lack of reference material.

Table 5.25. Recognition results in word error rate[%] using models extended with a pause and/or backspace (BS) model. The mean error rate is indicated as μ while the between-writer standard deviation is indicated as σ. Top: The results for the *Mixed1* test set. Bottom: the results for the *Mixed2* data set.

Model	**Vocabulary size**							
	200				20,000			
	No BS		BS		No BS		BS	
	μ	σ	μ	σ	μ	σ	μ	σ
No pause	2.7	2.7	3.5	3.1	16.5	11.2	19.0	14.5
Pause	2.0	1.8	2.4	1.9	13.9	10.2	15.1	10.4

Model	**Vocabulary size**							
	200				20,000			
	No BS		BS		No BS		BS	
	μ	σ	μ	σ	μ	σ	μ	σ
No pause	11.0	6.7	16.6	7.5	26.9	11.7	34.0	12.3
Pause	8.7	5.5	14.1	6.7	23.2	11.6	31.3	12.7

The results of the experiment are summarized in Table 5.25. The model configuration 'pause-no-backspace' always produces the best results. This is exactly the configuration used in the experiments throughout the previous subsections. The relative reduction in word error rate compared to 'no-pause-no-backspace' models is 20%-25% for a 200 word vocabulary and around 15% for a 20,000 word vocabulary. This result is in line with Cho et al. [1995] although we measure a lower improvement in word error rate.

The use of a 'pause' but no 'backspace' model yields the best recognition results. The interpretation of this result is that the ligature shapes vary strongly. If the ligatures are trained together with the character models, the ligatures add an extra source of variation to the character shapes, i.e., the character models get 'polluted' with ligature noise. The */pause/* model separates the ligature shape variations from the character shape variations. This leads to more accurate character models and a more accurate recognition. Another way of separating the ligature variations from the character variations is the use of trigraphs as discussed in the next subsection.

The use of the 'backspace' model always results in a performance degradation. If we compare the results of the 'pause-no-backspace' configuration with 'pause-

backspace', the relative increase of the word error rate is 20%-60% for a 200 word vocabulary and 10%-35% for the 20,000 word vocabulary.

There are a number of possible reasons for the failure of the 'backspace' model. First, our assumption to use */pause/* also for */backspace/* is a potential oversimplification because ligatures might have different shapes compared to the dots and horizontal bars of diacriticals at the end of a word. Second, we have used the 'backspace' model at the end of every word. The use of a 'backspace' model only at the end of words which contain a diacritical mark will probably give better results. These topics are beyond the scope of this thesis but are recommended for further study.

Finally, we observe that the recognition of the dataset *Mixed2* is generally less good compared to *Mixed1*. This is due to the average word length in the datasets. A short word simply provides less context compared to longer words and the more context, the better the recognition result. A similar effect was observed by Schenkel et al. [1995].

5.4.7 Contextual models

It is known from previous work presented in Chapter 2 that the shape of characters and ligatures depends on the neighbor characters. Therefore, it makes sense to model the contextual differences in order to gain accuracy. Preferably, complete word models should be trained but this is not practical due to lack of training data and the shear number of resulting models which scales linearly with dictionary size. An alternative is to train contextual character models.

In speech recognition, it is common to train triphone models. Example studies like Lee [1988] show that the use of triphones reduces the error rate in user-independent sentence recognition up to 50%. Starner, Makhoul, Schwartz & Chou [1994] employ the handwriting equivalent of triphones, trigraphs, in a writer-dependent study on the recognition of on-line, handwritten sentences. Excellent results are achieved. However, the study does not compare contextual with context-free models and therefore, the benefit due to contextual models remains unclear. More recently, Kosmala, Rottland & Rigoll [1997] applied trigraphs to a writer-dependent, on-line handwritten word recognition task and achieved an error-rate reduction of 50% and 35% with 1000 and 30,000 words vocabularies, respectively.

In the context of this thesis, we compare context-free and contextual trigraph models in a user-independent recognition task. A segment-based representation is used which employs angular features with three delays 1,2, and 4. There are no 'pause' models between characters. The training procedure starts with 10 reestimation iterations using a context-free, 'no-pause-no-backspace' model of 26 characters obtained in Subsection 5.4.6. After that, training continues with another 10 iterations and a varying number of contextual models. The set of contextual models

includes both trigraphs and bigraphs. The bigraphs depend only on the left character. A specific contextual model is included if we have more than a threshold number of examples in the training set as summarized in Table 5.26. For example, the set of 147 contextual models includes 18 trigraphs, 100 bigraphs and 26 unigraphs or context-free models. In addition, we have start-word and end-word markers as discussed in Subsection 4.2.3 and do not use a pause model. Both datasets *Mixed1* and *Mixed2* are used as a test set.

Table 5.26. The minimum number of training samples for each model and the corresponding number of contextual models.

Number of training samples	Number of models
-	26
300	66
200	147
100	284
60	476
30	878

Table 5.27. Effect on word error rate [%] of an increasing number of contextual models. Results are obtained with 200 and 20,000 word vocabulary and two data sets.

Models		Dataset and vocabulary size			
#models	#densities	Mixed1		Mixed2	
		200	20K	200	20K
26	3009	2.7	16.5	11.0	26.9
26	3946	2.9	13.7	8.1	23.7
66	8979	2.0	13.5	5.8	21.3
147	17394	1.8	11.4	6.5	20.7
284	25763	2.2	13.5	6.8	21.1
476	31964	3.1	18.8	7.3	23.0
878	35314	4.3	21.7	8.7	27.8

First, it is observed that the additional 10 training iterations reduce the word error rate even with a constant number of models. This is shown in Table 5.27 as the difference between the first and second line of results with 26 models.

Second, the table shows that the word error rate does not decrease monotonously with the number of models but instead reaches a minimum word error rate and increases again. This effect is expected and caused due to the bal-

ance between more specific models and available training data per model. Lee [1988] called this the 'trainability versus specificity' issue. Kosmala, Rottland & Rigoll [1997] observed a similar effect with a lowest word error rate at 25 trigraph models. The between-writer standard deviation with dataset *Mixed1* for a 200 and 20,000 word vocabulary is $\sigma = 2.0$ and $\sigma = 7.7$, respectively. The best result is reached with the use of 147 contextual models. The between-writer standard deviation of the results with dataset *Mixed2* and 147 contextual models is $\sigma = 4.0$ and $\sigma = 9.3$, respectively, with a 200 and 20,000 word vocabulary.

The error-reduction in the experiment with 147 contextual models and a 20,000 word vocabulary is 16.8% and 12.7% for the data sets *Mixed1* and *Mixed2*, respectively. The experiment with a 200 word vocabulary achieves an error-rate reduction of 38% and 19.2% with *Mixed1* and *Mixed2*, respectively.

5.4.8 Discussion

As in character recognition, the question is how to interpret the achieved results. The results of the conducted experiments are compared with similar experiments in the literature and human performance measured on similar recognition tasks.

The first indication of human performance is given in Section 1.4 where 72% correctly recognized cursive words are quoted in absence of context. In a small experiment, Schomaker [1994] employs up to 20 humans to judge handwritten words which results in two conclusions. First, neatly handprinted words are recognized well (98%). Second, cursive style words are much more difficult to recognize. Recognition rates between 88% and 54% correct words are measured without any vocabulary knowledge. The 88% correct words is measured for neat cursive handwritten words while the 54% is measured for sloppy, fast cursive words. These results give a reasonable indication on what to expect of a computer recognizing handwriting.

Comparing the recognition results with results from the literature is a hazardous task because of different data sets, writers and other factors. The presented recognition results in Table 5.28 on off-line handwritten data provide background information while the on-line results serve as comparison material. Main result is that the techniques employed in this thesis lead to a state-of-the-art handwriting recognition engine. We obtained all results with a hidden Markov model classifier, data-driven training and one-stage beam search. No special heuristics are used to tune the data for optimal recognition.

A word error rate of 10%-20% with a 20,000 word vocabulary is standard for a mixed-style word recognition system. However, the between-writer standard deviation is large. In our case, the between-writer standard deviation is $\sigma = 8.5$ for a word error rate of 11.2%. This is also observed by Schenkel et al. [1994] who reported word error rates between 5% and 40% for nice and sloppy writing, respec-

Table 5.28. On-line word recognition results in word error rates (WER) [%] from literature with cursive and unconstrained handwritten data (most lowercase). Abbreviations: WI=Writer-independent, WD=Writer-dependent, Off=Off-line recognition, On=On-line recognition, and Voc.=Vocabulary.

Author(s)	Voc.	Type	#Writers	WER
Cho et al. [1995]	10K	WI/Off/HMM	?	29
Seni and Cohen [1994]	21K	WI/Off/TDNN	9	37.6
		WD/Off/TDNN	20	8.4
Bunke et al. [1995]	150	WD/Off/HMM	5	2
Nathan et al. [1995]	21K	WI/On/HMM	25	18.9
Schenkel et al. [1994]	20K	WI/On/TDNN	59	20
Manke et al. [1995]	20K	WI/On/TDNN	80	8.6
Dolfing and Haeb-Umbach [1997]	20K	WI/On/HMM	10	11.2

tively. A similar variation is caused by the word length of the handwritten words. With a vocabulary of 20,000 words, a word error rate of 11.2% is measured on dataset *Mixed1* and a word error rate of 20.7% is measured for the short, sloppy words of *Mixed2*. Even the results on this worst case dataset compare fine with Table 5.28.

More results on comparable problems include the work by Schomaker & Teulings [1992] who reported a 88% character recognition rate for a writer-independent system with neat, cursive handwriting without vocabulary. Tappert [1982] presents an early result for a writer-dependent, cursive recognizer which achieved 95% correct characters based on a dynamic programming technique.

An interesting observation is that the employed techniques in Nathan et al. [1995], Schenkel et al. [1994], Manke et al. [1995], and Dolfing & Haeb-Umbach [1997] are complementary to a certain extent. We could speculate that the introduction of contextual features to the systems of Nathan et al. [1995], Schenkel et al. [1994], Manke et al. [1995] gives an improved error rate while duration modeling, allograph clustering and other techniques can improve the system of Dolfing & Haeb-Umbach [1997].

Looking forward to the next section, we can already conclude that only the use of more context knowledge in whatever form will boost the recognition to an acceptable level with a vocabulary of 20,000 words. At the same time, research should focus on a more robust recognition or fast user adaptation to reduce both word error rate and deviation in a writer-dependent situation. The empirical prediction of the recognition performance presented by Cortes [1995] provides an approach for a meta-search to find the best combination of representation and model.

5.5 Sentence Recognition

This section explores techniques for sentence recognition. We model each character with a hidden Markov model. In the previous sections, we used beam search for isolated word recognition. The current section uses connected class recognition, i.e., we do not recognize one label from a dictionary but a character or word sequence. The classification is based on a one-stage beam as presented in Figure 3.13. The beam search determines the optimal label sequence and is guided by the hidden Markov models and a language model. Note that the segmentation of a sentence into words or characters is implicit rather than explicit as in the system used by Mahadevan & Srihari [1996].

As explained in the previous sector, the language model is the context source which limits the number of interpretations. Therefore the effect of several language models is explored. The three main objectives can be summarized as follows:

- Investigate the use of an M-gram, character-based language model for unlimited vocabulary recognition.
- Test and compare character and word-based language models.
- Investigate the effect of language models on different types of handwritten sentences.

5.5.1 Character-based language models

The objective of this experiment is to investigate the effect of an M-gram statistical language model on sentence recognition. We concentrate on a character-based language model which models a handwritten text as a sequence of characters rather than words. This perspective enables the recognition of handwritten text with unlimited vocabulary. A disadvantage is that we loose the information about word boundaries.

The experiment compares zerogram (M=0), unigram (M=1) and bigram (M=2) models on handwritten sentences in discrete style, represented by frames and segments, respectively. The test set is *Sentence1*. Although a character-based recognition of sentences is possible without language model, the language model improves the recognition accuracy. An exact prediction is difficult because there is not much comparable material. Kassel [1995] found an improvement from 71.7% to 79.3% correct characters for a handwritten sentence recognition task with a bigram model (PP=11.3) and 62 character models. This corresponds to a reduction in error rate with more than 25%.

Given the *Sentence1* data set as described in Section 5.1, representations based on segments and frames are used to represent the sentences. The frame-based representation uses eight delays with angular and overlap features. Because this is identical to the representation in Subsection 5.4.4, the hidden Markov models

computed in that task are used for recognition. The segment representation employs three delays (1,2,4), and angular features as discussed in Subsection 5.4.2. Only 26 lowercase characters are modeled. Note that the models are trained on mixed-style words while the test material consists of discrete-style data.

Two character-based language models are computed using the Linux HOWTO material which contains about $7.0 \cdot 10^6$ characters. We used $4.2 \cdot 10^6$ lowercase characters to compute unigram probabilities and $3.9 \cdot 10^6$ bigram examples to compute a set of bigram probabilities.

The experimental results are summarized in Table 5.29. The use of the bigram model produces the best recognition results. Compared to the recognition without language model, the reduction of the error rate is around 15%. The between-writer standard deviation is roughly 8%-10%. It is not clear why the unigram result for frames is better when compared to the bigram result.

Table 5.29. The effect of character-based language models in sentence recognition. Results are in character error rates [%].

Representation	***M*-gram**		
	0 (*PP*=26)	**1 (*PP*=18.1)**	**2 (*PP*=12.3)**
Frames	25.2	20.1	20.9
Segments	24.1	22.3	20.7

The difference in relative error reduction between our results and those reported by [Kassel, 1995] can be explained from the difference in perplexity (Subsection 3.6.2). While Kassel [1995] achieved about 25% error reduction when comparing the recognition results with zero-gram (*PP*=63) and bigram (*PP*=11.3) language model, we achieve an error rate reduction of 15% while reducing the perplexity from 26 to 12.3 for the experiments with zero-gram and bigram language model, respectively.

Although the character-based recognition performance improves due to the language model, the achieved performance as shown in Table 5.30 is not good enough for practical use. These results can be improved by adding more context information.

Table 5.30. Example results of character-based sentence recognition using a bigram language model. The observed character error rate is about 20%.

Recognition Result	**Expected Result**
a f i *c* t f u l o f f a *r* c i n a t i n g f a c t s	a f i s t f u l o f f a s c i n a t i n g f a c t s
a *m* a t *v* r a l w a y w i *x* h w e e *c l* s	a n a t u r a l w a y w i t h w e e d s
a f r i c n w i d d o *y* s f e a *w* c a t s	a f r i c a n w i l d d o g s f e a r c a t s
a l t h e f *v* n *c* f *x* h e f e s t i v a l	a l l t h e f u n o f t h e f e s t i v a l

5.5.2 Character-based versus word-based language models

In Subsection 3.6.2, we concluded that a word-based, M-gram language model provides more context than a character-based model of the same order M. Therefore, we compare the effect of the use of a character and word-based language model. In contrast to the previous subsection, a word-based unigram and bigram language model is employed in the recognition of the *Sentence1* test data.

The use of a word-based language model is standard practice in speech recognition but not in handwriting recognition. Starner, Makhoul, Schwartz & Chou [1994] use a bigram language model with vocabulary size of 25.595 words in the recognition of handwritten sentences. These handwritten sentences are taken from the Wall Street Journal corpus used in continuous speech recognition and discussed by Paul & Baker [1992]. The written sentences are purely cursive, not unconstrained, and the hidden Markov models are writer-dependent. In contrast, all our tests are writer-independent.

Compared to Starner et al. [1994], the scope of the current experiment is limited. Although the vocabulary size goes up to 20,000 words, the language model is dedicated to the expected sentences in the test set. The derived language models have a very low perplexity which can be useful in restricted domains like check reading. Therefore, the word error rate derived in the current experiment is merely an indication or lower-bound on the expected sentence recognition error rates.

The handwritten data is represented with the same segment representation as in the previous subsection. The hidden Markov models for lowercase characters are again trained on mixed-style word data.

Table 5.31. Sentence recognition results using a character and word-based language model (M-gram) with perplexity PP and different vocabulary sizes. Results in character error rate (CHER) [%] and word error rate (WER) [%].

Error rate	**Vocabulary size and language model.**						
	26			**200**		**20,000**	
M-gram	0	1	2	1	2	1	2
PP	26	18.5	12.5	88.0	4.1	88.0	4.1
CHER	24.1	22.3	20.7	7.2	1.6	7.2	2.4
WER				17.7	2.6	17.7	4.1

First, we observe that the character error rates of the experiment with a bigram, word-based language model (1.6% and 2.4%) is roughly a factor ten smaller compared to the character error rate of the bigram, character-based language model (20.7%). The reason for this large improvement is that the word-based language model describes the sentence text very accurately due to its closed vocabulary, i.e., the word-based language model is a better context model than the character-based

language model. However, the price for this improvement is that only words from a dictionary are recognized while the character-based language model allows the recognition of an unlimited vocabulary.

Second, we learn that the word error rate for a 20,000 word vocabulary and bigram grammar has dropped to 4.1%. The word error rate with a unigram language model is 17.7%. The difference between these word error rates is caused by the improved context modeling of the bigram language model.

Table 5.32. Effect of a bigram language model with a 20,000 word vocabulary and both context-free and contextual hidden Markov models with the between-writer standard deviation σ in brackets.

Error rate	Model type	
	Context-free	Contextual
CHER	2.4	1.8
WER	4.1 (3.9)	2.6 (2.9)

Finally, the previous test with bigram language model and a 20,000 word vocabulary is repeated. We exchanged the used hidden Markov models for the contextual models (147 models) as tested in Table 5.27. This results in an additional improvement of the word error rate of 25%-35% with otherwise unchanged parameters. The resulting word error rate of 2.6% can be roughly read as 1-out-of-40 written words incorrect. The between-writer standard deviation is $\sigma \approx 2.9\%$ for the contextual results. This error rate is very low but remember that the language model is estimated on the sentences in the test set.

Table 5.33. Examples of sentence recognition results with a word-based language model. The observed word error rate is about 2.6%.

Recognition Result	Expected Result
a fistful of *fit situations*	a fistful of fascinating facts
a natural way with weeds	a natural way with weeds
african wild dogs fear cats	african wild dogs fear cats
all the fun of the festival	all the fun of the festival

5.5.3 Discrete versus mixed-style input

The sentence recognition experiments in the previous subsections use handwritten, discrete-style sentences. The recognition of the sentences is based on hidden Markov models for 26 lowercase characters trained on mixed-style data. Because handwritten sentences are normally written unconstrained, we compare the previous results with results on an extra, more ambiguous test set *Sentence2* which contains the same text as before but written mixed-style by different writers.

The experiment is conducted with a segment representation and a vocabulary size of 20,000 words. We employ 26 hidden Markov models, i.e., one for each lowercase character, and an additional pause model [Dolfing, 1998]. The bigram language model is the same as in the previous subsection, i.e., estimated on the text of the sentences in the test set.

Table 5.34. Sentence recognition results in word error rate [%] using unigram and bigram language model with 20,000 vocabulary and context-free and contextual models, respectively. Between-writer standard deviation in brackets.

Data set	**Model type**		
	Context-free		**Contextual**
M-gram	1 (PP = 88.0)	2 (PP = 4.1)	2 (PP= 4.1)
Sentence1 (discrete)	17.7 (13.9)	4.1 (3.9)	2.6 (2.9)
Sentence2 (unconstrained)	17.3 (12.8)	3.6 (3.5)	0.6 (0.6)

Although the word error rates for the unconstrained sentences are somewhat lower compared to the discrete data, there is no significant difference. For this experiment, the best observed writer-independent, sentence recognition result with a 20,000 word vocabulary is a word error rate of 0.6%. Again, this is with a dedicated language model.

Compared to the word error rate of 20.7% in Table 5.25 on the handsegmented, unconstrained data *Mixed2* of the same sentences using the same contextual hidden Markov models, the word error rate of 0.6% is an improvement with a factor 34.5 despite the additional ambiguity of unknown word boundaries in the sentences.

6

Signature Verification

In this chapter we discuss on-line signature verification based on hidden Markov models. In Definition 3.35, we defined handwriting verification as a stochastic classification problem. To transform the formal definition into a signature verification system, we have to make the type of handwriting, labels, model and items more concrete. In this chapter, we concentrate on signatures as a subclass of all possible handwritings and on signature verification as a subclass of the handwriting verification in Definition 3.35. The formal Definition 3.35 uses two possible labels which correspond with a 'reject' and 'accept' decision on the handwritten signature. In the context of hidden Markov models, we refer to the items in Definition 3.35 as observations which are represented by feature vectors.

In Section 6.1, we give a brief discussion of important issues in signature verification such as a system overview, forgery types, algorithms, and approaches presented in the literature. We study the representation in Section 6.2 with special attention on how to exploit the novel digitizer hardware and handwriting-specific knowledge. A hidden Markov model is chosen as a model for each signature. Several model structures and modifications are discussed in Section 6.3 where we concentrate on the differences with the hidden Markov models used in handwriting recognition. The experimental results are discussed in Chapter 7.

6.1 Introduction

In this section, we start with an overview of the signature verification system in Subsection 6.1.1 which is based on Definition 3.35. Afterwards, we introduce the type of forgeries which are used in this thesis and the literature in Subsection 6.1.2. This is followed by a forensic perspective on genuine and forged signatures in Subsection 6.1.3. Next, we summarize a number of common models and algorithms in on-line signature verification in Subsection 6.1.4. Finally, Subsection 6.1.5 uses the information of the previous sections to summarize and comment on a number of important results presented in the literature.

6.1.1 System overview

Figure 6.1 presents an overview of the process of signature verification on the basis of hidden Markov models. First, an individual writes a signature on a digitizer. The properties of this digitizer are discussed in Subsection 2.4.1. This digitizer samples the signature at equidistant points in time. At each timestep t, the digitizer samples the x and y coordinates, the pressure p, the tilt θ_x in x direction and the tilt θ_y in y direction. Because of the equidistant sampling in time, we can use the x and y coordinates to derive a velocity v and acceleration a signal. Writing and sampling of the signature is done simultaneously. The sampling of the signature results in a stream of coordinates $(x, y, p, \theta_x, \theta_y)$.

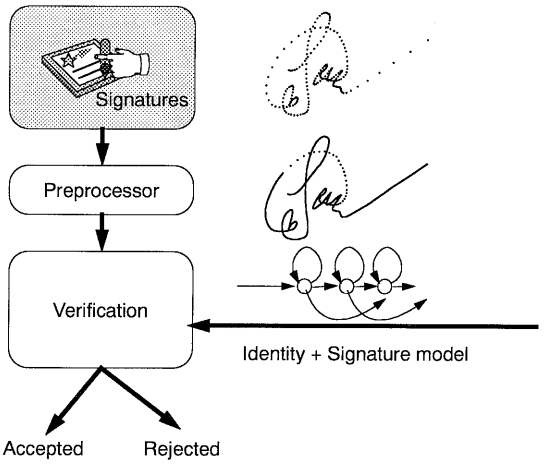


Figure 6.1. Overview signature verification system.

The sampled signature is processed by the preprocessor which transforms the stream of coordinates $(x, y, p, \theta_x, \theta_y)$ into a stream of feature vectors. These feature vectors are the input to the verification process. Because we assume that we know which individual writes the signature, we compare the stream of feature vectors with a hidden Markov model of the signature of this individual. The verifica-

tion process results in a binary decision where the written signature is accepted as genuine or rejected as a forgery.

The signature preprocessing is very similar to the handwriting preprocessing in Section 4.3. Main difference is that the digitizer samples a stream of $(x, y, p, \theta_x, \theta_y)$ coordinates [Dolfing & Van Oosterhout, 1996; Dolfing, Van Oosterhout & Aarts, 1998] instead of (x, y) coordinates as used in handwriting recognition. Because we also derive velocity and acceleration signals, this implies some additional filtering of speed, acceleration and pressure signals.

An important choice in signature preprocessing is the choice of the representation. This involves the choice of a grouping procedure and feature vector. The grouping or blocking of coordinates presented in Subsection 2.5.1 determines which sequence of coordinates is used to compute a feature vector. Here, we choose to divide signatures into segments as discussed in Subsection 4.1.2 and Figure 4.8. This grouping choice corresponds exactly with the size-independent segments used in handwriting recognition in Chapter 4. While the grouping is discussed in Section 6.2, the choice of the feature vector is discussed in Chapter 7.

Furthermore, we choose to model a signature with a left-to-right hidden Markov model. This model is trained from several example signatures in the same way as a word or character model is trained. The verification is a simplified recognition because we rely on a given identity and compare the written signature with only one hidden Markov model. The comparison is implemented with the Viterbi algorithm as presented in Figure 3.9 and results in a computed loglikelihood which is compared with a threshold value to complete the verification.

6.1.2 Forgeries

The quality of the forgeries has a large effect on the false acceptance (FA), false rejection (FR) and equal-error (EE) rates, where the equal-error rate is found at the threshold where FA equals FR. Therefore, the type of forgeries used in the experiments should be specified. It is known that the difference in false acceptance rate between simple and skilled forgeries can be an order of magnitude. In this thesis, we discriminate between four types of forgeries. These are zero-effort, home-improved, over-the-shoulder and professional forgeries. The difference between these forgery types is that they show an increasing similarity with the original signature because the forger has access to an increasing amount of information about the genuine signature. The professional forgeries look the most like the genuine signatures. Next, we take a closer look at each forgery type.

Zero-effort forgery. A zero-effort forgery is not really a forgery but a random scribble or signature of another individual. Given a database of signatures and an individual whose identity is verified based on a signature, we exploit the signatures

of all other individuals in the database as 'forgeries'. This experimental setup is a test to determine whether the signature verification system can reject more or less random scribbles. Compared to the use of home-improved or over-the-shoulder forgeries, the measured equal-error rate with zero-effort forgeries should be significantly lower.

Home-improved forgery. The category of home-improved forgeries includes forgeries which are made when a forger has a paper copy of the genuine signature and has ample opportunity to practice the signature at home. After a lot of practice, a forger might be able to produce a forged signature which is reasonably similar to the genuine signature. It is important to note that the imitation is based only on the *static* image of the original signature.

Over-the-shoulder forgery. The category of over-the-shoulder forgeries contains the forgeries where the forger is present while the genuine signature is written. The forger learns not only the spatial image but also the *dynamic* properties of the signature by observation of the signing process. The knowledge of both spatial and dynamic characteristics allows a forger to forge a signature which is very similar to the genuine signature. Both the over-the-shoulder and the home-improved forgery are examples of amateur forgeries. In the literature, over-the-shoulder and home-improved forgeries are not distinguished. Instead, they are all referred to as amateur, semi-skilled or even skilled forgeries [Plamondon & Lorette, 1989].

Professional forgery. Finally, there exist forgeries which are produced by individuals who have professional expertise in handwriting analysis. Normally, these professionals have no more experience than other individuals with copying the dynamic signature information. Examples are document examiners or forensic scientists. Because of their experience, they know how to discriminate between genuine and forged signatures. Hence, they are able to circumvent obvious problems and exploit their knowledge to produce high quality, spatial forgeries. These forgeries are named professional forgeries.

After this overview of forgery types, we present a number of approaches from the literature which use the discussed forgery types. Yang, Widjaja & Prasad [1995] describe a signature verification system based on hidden Markov models where the error rate is measured using zero-effort forgeries. Crane & Ostrem [1983], Lee, Berger & Aviczer [1996], Sato & Kogure [1982], Worthington, Chainer, Wilford & Gundersen [1985] and Yoshimura, Kato, Matsuda & Yashimura [1991] all use skilled forgeries mainly based on an image of the original signature. Although the verification system employs dynamic and spatial information, the forgeries are based on spatial information only.

There are few approaches where dynamic information is provided to the forgers. Lee et al. [1996] provides the forgers with the average writing time of a genuine signature to indicate the writing speed. Crane and Ostrem [1983] provides the forgers with a video tape of the original signing process. Plamondon [1994a] uses acoustic feedback of the genuine signature, generated by the pen while writing, to convey dynamic information to forgers.

The available forgeries do not only effect the classifier testing but also classifier design as demonstrated by Lee et al. [1996] who conclude that forgeries are not essential but helpful to improve performance.

6.1.3 Forensic perspective

In Chapter 4 and Chapter 5 we have learned that more context knowledge in the representation, trigraphs and language model improves the recognition accuracy. The same principle applies to signature verification. The more relevant information is extracted from the signature and employed in representation and model, the better the verification works. Because the digitizer in an on-line signature verification system extracts additional data signals, see Section 1.1 and Subsection 2.4.1, on-line signature verification generally leads to better results compared to off-line verification [Plamondon & Lorette, 1989]. Bromley, Bentz, Bottou, Guyon, LeCun, Moore, Sackinger & Shah [1993] show that there are situations where computerized on-line verification systems can outperform unskilled humans on skilled forgeries.

Traditionally, we use signatures as an authentication mean, e.g., in banking affairs on cheques and in business on documents. While expert document examiners judge whether the signature is genuine or forged based on given related writing material, the check reading at a bank counter normally depends on an employee untrained for detecting signature forgeries. Wilkinson, Pender & Goodman [1991] argue that this is no problem since most forgeries are 'casual' forgeries. Nevertheless, the question of human performance on signature verification compared to an off-line signature verification system is raised.

Found & Rogers [1995] compared three forensic examiners with a computer system to verify off-line signatures. The task was to classify signatures into one of three complexity classes which are easy, moderately easy and complex. Computers and experts agreed on 73.5% of the classifications. Plamondon & Lorette [1989] comment on the relation between document examiners and off-line signature verification systems as follows

> "According to evaluation by expert document analysts, skilled forgery detection would not be efficient with off-line systems, as long as dynamic and static features are not exhaustively extracted from a written specimen."

Because the digitizer employed in this thesis extracts both spatial and dynamic characteristics of a written signature, we conclude that this is a viable starting point for on-line signature verification and allows the detection of skilled forgeries.

6.1.4 Algorithms

A variety of algorithms is used for signature verification of which a subset is compared by Parizeau & Plamondon [1990]. Common to all algorithms is the existence of one or more reference models which are compared against a written signature. Important is the difference between a linear and non-linear comparison of a written signature with the model.

Simple models use exactly one signature as reference, use a linear alignment and compute the Euclidean distance between reference and written signature. Other approaches average a number of signatures to derive a reference model and combine this with a Euclidean distance as demonstrated by Crane & Ostrem [1983]. Advantage of this approach is the processing speed but disadvantage is that a linear comparison between reference and input signature is a simplification.

A more sophisticated approach is dynamic time warping (DTW) which is a dynamic programming algorithm to achieve a non-linear alignment between reference and written signature. This approach was first used in speech recognition [Sakoe & Chiba, 1978] and later introduced into signature verification [Sato & Kogure, 1982; Wirtz, 1995].

The distance between reference and input signature is typically computed with the Viterbi algorithm as discussed in Subsection 3.4.2. Signature verification systems discussed by Wirtz [1995] and Yashimura & Yoshimura [1992] use dynamic time warping with not only x and y signals. While Wirtz [1995] demonstrates dynamic time warping over x, y and pressure signals, Yashimura & Yoshimura [1992] include the velocity signal in dynamic time warping.

Compared to the simple models which use one or an average of genuine signatures as reference model, a statistical model is more sophisticated. Examples are given by Hastie, Kishon, Clark & Fan [1992], Nelson, Turin & Hastie [1994] and Clark, Hastie & Kishon [1990]. Based on a number of genuine signatures, the mean and variance of the signature are computed. Schmidt & Olschewski [1995] show an example of a Kohonen type neural network in signature verification with a false rejection rate of 15% and a false acceptance rate of 17.5%. Doux & Milgram [1995] briefly describe an on-line signature verification system based on dynamic time warping and a neural network where the best result is a false acceptance rate of 18% and a false rejection rate of 3.3% achieved with skilled forgeries. Leclerc & Plamondon [1994] summarize other approaches using neural networks for both static and dynamic signature verification.

Another type of statistical model is the hidden Markov model of which the

application to signature verification is explored in this thesis and by Dolfing, Van Oosterhout & Aarts [1998]. Two similar approaches are discussed by Mohankrishnan, Paulik & Khalil [1993a] and Yang, Widjaja & Prasad [1995]. Although hidden Markov models are rarely used in signature verification, they are regularly employed in speaker or voice verification with good results [Naik, Netsch & Doddington, 1989; Naik, 1990; Naik, 1994].

6.1.5 Literature

In general, the comparison of signature verification approaches is difficult due to the different equipment, algorithms, models and databases. There is no standardized benchmark or database. Additionally, the type and quality of forgeries used to estimate the verification accuracy is different for every approach. This leads to false acceptance, false rejection and equal-error rates which make no sense without information about other system parameters.

Signature variability is discussed by Plamondon & Lorette [1990] and Fairhurst, Cowley & Sweeney [1994] who discuss inconsistencies and changes which depend on time and situation. Factors which affect a signature shape are age, mental and physical condition, practical conditions and time-of-day. We also know that people in Japan and China are not used to write a signature [Sato & Kogure, 1982; Yoshimura, Kato, Matsuda & Yashimura, 1991] and simply use their name if necessary. Additionally, we have indications that the writing time of a European signature is shorter compared to an American signature because American signatures are required to be legible.

A classic overview of signature verification approaches is given by Plamondon & Lorette [1989] and is updated by Leclerc & Plamondon [1994]. Because we have already discussed the equipment used in this thesis, the signature verification system, forgery types and typical algorithms found in the literature, we can now give a brief overview of relevant results reported in the literature in Table 6.1 which gives a general idea of state-of-the-art performance and the kind of databases and algorithms used.

The column 'Reference' in Table 6.1 indicates the information sources in the literature. 'Database size' indicates the number of genuine and forged signatures in the database used for testing. The column 'Distance' indicates the algorithm to compare the written signature with the reference(s) model. The multistage algorithm implements the binary decision of Definition 3.35 in a sequence of several stages. The idea is that the verification ends as soon as one of the stages can make a reliable decision.

The column 'Threshold' shows the type of threshold and corresponding implementation. Different threshold types are *fixed*, *factor* and *personal*. Some verification systems use one fixed threshold for all signatures. Because there is a wide

Table 6.1. Overview of signature verification results reported in the literature.

Reference	Database size	Distance	Threshold	Error rate
Crane and Ostrem [1983]	5800	Euclidean	Personal	EE 1.5%
Fairhurst et al. [1994]	8500	Euclidean	Personal	FR 0.85% (per Session)
				FR 1.8% (per Signature)
Hastie et al. [1992]	100	Statistical model	Personal	-
Lee et al. [1996]	10,000	Euclidean	Fixed	EE 2.5%
Plamondon [1994a]	460	Multistage	Personal	EE 0.5%
Sato and Kogure [1982]	440	DTW	Personal	FR 1.8% FA 0%
Taguchi et al. [1989]	200	Multistage	Factor	FR 6.7% FA 0%
Worthington et al. [1985]	28,000	Regional correlation	Personal	FR 0.19% FA 0.56% (per Session)
				FR 1.77% FA 0.28% (per Signature)
Yang et al. [1995]	500	hidden Markov model	Personal	FR 4.44% FA 1.78%
Yoshimura et al. [1991]	2200	DTW	Factor	EE 1%

variety of signatures, this does not always work. Other possibilities are to compute a personal threshold either automatically or 'by hand'. An intermediate solution is to relate the signature model to unseen test signatures via a multiplication factor or additional offset [Yoshimura, Kato, Matsuda & Yashimura, 1991; Liu, 1978]. Thresholds in the context of hidden Markov model signature verification are discussed in Subsection 6.3.2.

The experiment of Worthington et al. [1985] provides a landmark because of the large number of involved signatures, the fact that these results were obtained in a field test similar to Crane & Ostrem [1983], and the low error rates. Despite the fact that forgers who wrote accepted forgeries got a financial reward, the session error rate, i.e., the error rate after a number of trials of the individual to make a successful signature, is very low. However, in this thesis we concentrate on signature error rates and do not compute session error rates.

6.2 Representation

In Chapter 4, we discussed the representation for handwriting recognition based on hidden Markov models. This resulted in a representation which describes spatial information in handwriting, i.e., the shape of handwriting parts. Therefore, the representation is a feature vector which contains spatial information. A number of writer and digitizer dependent variables like writing speed or sample rate are normalized before feature extraction.

In comparison with that, signature verification depends on writer-dependent information. The writer-dependent information includes not only the shape of the signature but also the dynamics of the signing process. Examples are writing speed and the exerted pressure while writing. Therefore, the representation has to include not only spatial information but also dynamic information.

In order to design a representation for signature verification based on hidden Markov models, we discuss grouping and feature extraction. The grouping is discussed in Subsection 6.2.1. The process of feature extraction is basically the same as discussed for handwriting recognition in Subsection 4.3.1. The difference is that signature verification uses additional features describing the handwriting dynamics which are discussed in Subsection 6.2.2. The framework of contextual features is applied to the dynamic features in Subsection 6.2.3. In addition to that, the LDA transformation (see Subsection 4.1.6) is exploited in the context of signature verification in Subsection 6.2.4.

6.2.1 Grouping

The purpose of grouping is already discussed in Subsection 2.5.1. In the context of on-line signature verification, we have to define sequences of coordinates, sampled

equidistant in time by the digitizer, which are used to compute feature vectors. Typically, the number of coordinates for each group is chosen to balance resolution and information content. We present the grouping choice used for the signature verification experiments in this thesis after we have discussed a number of approaches presented in the literature.

Traditionally, on-line signature verification approaches have been classified into two categories based on their grouping approach [Plamondon & Lorette, 1989]: the *functional* and the *parametric* approach. While the functional approaches compares the sampled signals with a reference signal set, the parametric approach extracts a set of features from the sampled signature and matches the features against a reference model.

The parametric approach is demonstrated by Plamondon & Lorette [1989] and handles the signature in one piece. A set of features is extracted to describe the signature. Other examples are given by Crane & Ostrem [1983], Beatson [1985], Taguchi, Kiriyama, Tanaka & Fujii [1989], Mital, Hin & Long [1987], Fairhurst & Brittan [1994], and Lee, Berger & Aviczer [1996].

An alternative approach is a scribble-based segmentation which splits the signature in scribbles before processing. A number of features is computed for each pen-up or pen-down scribble [Dimauro, Impedovo & Pirlo, 1992]. Instead of scribbles, we can group coordinates into strokes as defined in Subsection 2.2.1 This is demonstrated by Plamondon [1994a] and provides a more fine-grain signature description. Compared to the scribble-based approach, this approach will generate more feature vectors to describe the signature because a scribble can contain several strokes.

The functional approach computes features for every coordinate. Comparison of a written signature with a reference model is possible with techniques like dynamic time warping. Worthington, Chainer, Wilford & Gundersen [1985] and Zimmermann & Varady [1985] present examples of this approach. To normalize the number of coordinates per signature, a resampling is often performed as demonstrated by Sato & Kogure [1982], Lam & Kamins [1989], Yashimura & Yoshimura [1992], and Yang, Widjaja & Prasad [1995].

As mentioned in Section 6.1, the grouping procedure in this thesis for on-line signature verification is based on elementary strokes bounded by the condition that the velocity in y direction is zero. This corresponds with the segments of Subsection 4.1.2. The objective of this grouping choice is a writing size independent representation which is also insensitive to writing speed and sample rate variations. Additionally, the study by Dolfing & Haeb-Umbach [1997] demonstrates that the choice of segments produces good results in on-line handwriting recognition with hidden Markov models, i.e., the use of segments leads to feature vectors which contain information relevant to the recognition of characters and words. Because

signatures can be compared with fast handwritten words [Herbst & Liu, 1977], we assume that the same grouping as in on-line handwriting recognition is a valid starting point for our studies.

6.2.2 Features

Subsection 4.1.3 discussed a set of spatial features which accurately describes the spatial image of a segment. Because a signature contains both spatial and dynamic information as discussed in Chapter 2, extra features are necessary to capture the dynamic information. We concentrate on the use of pen-up data, velocity, pressure and pen-inclination or pen-tilt information. Based on the digitizer specification and the sampled signals, a baseline representation for dynamic information is chosen and described in Subsection 7.3.1. The dynamic features are normalized to the interval $[0;1]$ by either linear scaling, a cut-off function with fixed threshold or a soft-threshold with an *atan* function. Linear scaling is applied to pressure and tilt which have 64 levels. Because we use segments and the features are writing size independent, the complete representation is writing size independent.

Pen-up

One of the differences of on-line signature verification with off-line verification is that the PAID tablet samples continuously in time not only *on* the writing surface but also *above* it. Therefore, not only the ink 'on the paper' is measured but also the movements in an adjustable area of about 1 cm above the tablet. An example containing both pen-up and pen-down strokes is given in Figure 6.2 where the pen-up and pen-down movements are drawn dotted and solid, respectively.

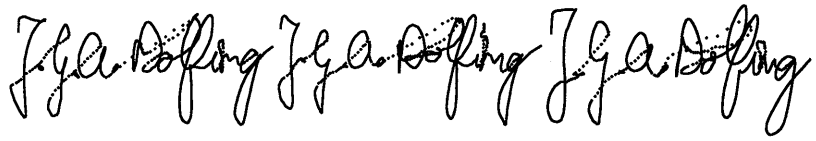


Figure 6.2. Pen-up information in signature.

The comparison of the original signatures in Figure 6.2 with the home-improved and over-the-shoulder forgeries in Figure 6.3 shows significant differences. First, the loop of pen-up coordinates before the original 'D' character is missing in the forgeries. Second, the dot on the 'i' is originally made immediately after the 'i' body while the forgers place the dot after the 'n' and at the end of the word, respectively.

Because these examples (and others) show that the pen-up data is writer-dependent, we do not strip the pen-up samples from the signature but model them just like the pen-down data. This is in contrast to our handwriting recognition approach where the pen-up samples are removed from the signal.

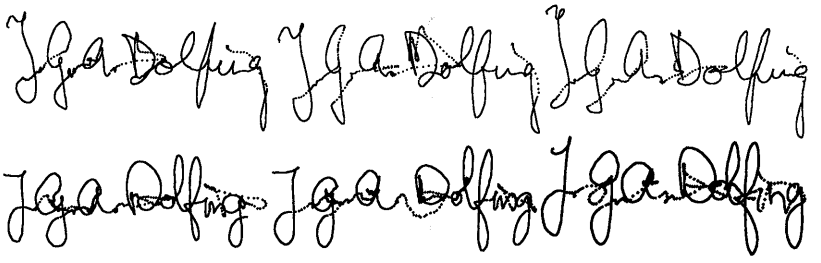


Figure 6.3. Pen-up information in two types of forgeries. Top: home-improved forgeries. Bottom: over-the-shoulder forgeries.

Velocity

Velocity is an important parameter in handwriting generation and verification. This is supported by numerous studies by, e.g., Plamondon, Alimi, Yergeau & Leclerc [1993], Plamondon [1993], and Schomaker & Plamondon [1990]. Figure 6.4 illustrates the fact that the velocity profile is writer-dependent and hard to reproduce. The image of the genuine signature which results in this velocity profile looks like an 'o' plus extra horizontal movement. Although the spatial image of the signature is simple, a forgery looks different in the velocity domain [Van Galen & Van Gemmert, 1995]. Especially the peak velocity is important in this example.

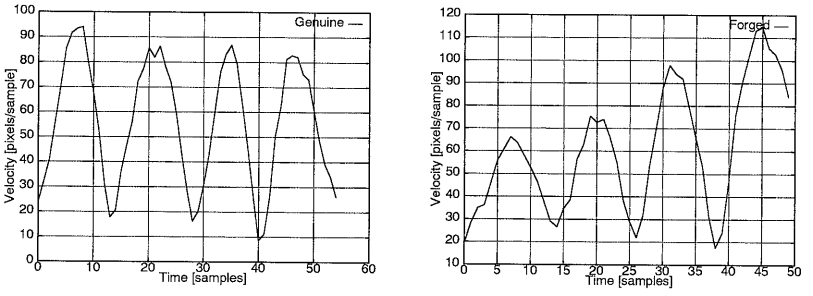


Figure 6.4. Velocity profile of genuine and forged signature. Left: genuine signature. Right: forged signature.

For each segment, we compute the features $v_{\text{begin}}, v_{\text{end}}, v_{\text{max}}$ and v_{avg}. These features represent the velocity at a segment begin, at segment end, the maximum velocity in the segment and the average velocity of the segment. Because of the segment definition where the boundaries are defined by the condition $v_y = 0$, the features v_{begin} and v_{end} only model the velocity v_x at the segment borders. A cut-off function with threshold $v_{\text{threshold}}$ is used to normalize the velocity features. The threshold $v_{\text{threshold}}$ is determined with a histogram of all velocities v in the signature database for which holds $v \leq v_{\text{threshold}}$ for 95% of all velocities. This leads to the

feature

$$f_{v_{\text{begin}}} = \begin{cases} \frac{v_{\text{begin}}}{v_{\text{threshold}}} & \text{if } v_{\text{begin}} \leq v_{\text{threshold}} \\ 1 & \text{if } v_{\text{begin}} > v_{\text{threshold}} \end{cases}$$

and similar for the other velocity features.

Pressure

The availability of pressure measurements is a fundamental difference between on-line and off-line signature verification. It is known from studies by Plamondon & Lorette [1989] and Schomaker [1990] that there are basically three groups of people with repeatable, more or less repeatable and not repeatable pressure profiles. Studies by Wirtz [1995], Zimmermann & Varady [1985], Sato & Kogure [1982] and Schmidt & Olschewski [1995] employ pressure in signature verification. Figure 6.5 illustrates the pressure profile for the signature in Figure 6.2.

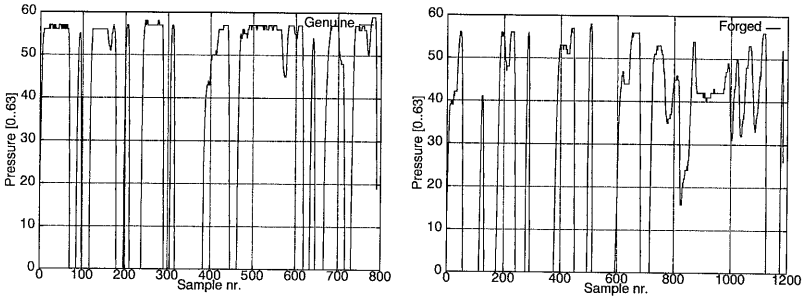


Figure 6.5. Pressure profile of a genuine and forged signature. Left: genuine signature. Right: forged signature.

Similar to velocity features, we compute the features $p_{\text{begin}}, p_{\text{end}}, p_{\text{max}}, p_{\text{avg}}$ and some combined features. All pressure features are normalized by a division with 64 to obtain a feature with range $[0;1]$.

Tilt

Pen-tilt or pen-inclination describes the angle of the pen with respect to the digitizer surface as explained in Figure 2.5. This angle is caused by the way the writer holds the pen. Left and right handed people are easily discriminated based on this information. Because every individual has a different pen-grip, this is writer-dependent information which is useful in signature verification. Taguchi, Kiriyama, Tanaka & Fujii [1989] study pen-tilt and conclude that it is an important characteristic. In contrast to Taguchi et al. [1989] who only study the pen-angle with the writing surface, the current study explores the benefits of independent tilt in x and y direction with a state-of-the-art digitizer.

Figure 6.6 illustrates the tilt profile for the example signature in Figure 6.2. For the genuine signature, we measure a pen-tilt of approximately 45 and 35 in x and

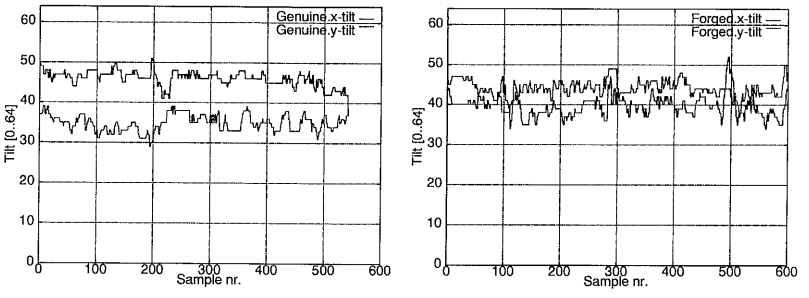


Figure 6.6. Tilt profile of a genuine and forged signature. Left: genuine signature. Right: forged signature.

y direction, respectively. The tilt of the forged signature is approximately 40 and 55 for x and y direction, respectively. These pen-tilt levels remain almost constant during the writing process of the signature.

Similar to pressure and velocity, we compute the features $\theta_{\text{begin}}, \theta_{\text{end}}, \theta_{\text{max}}, \theta_{\text{avg}}$ independent for x and y direction. All tilt features are normalized by a division with 64, because there are 64 tilt levels, to obtain a feature in the range $[0;1]$.

6.2.3 Aggregate features

Subsection 4.1.4 discussed how to augment a feature vector with delta and contextual features in order to capture long-term trends in handwriting to improve handwriting recognition. Except for the feature set, the difference between hidden Markov model based handwriting recognition and signature verification is small. Therefore, we apply the augmenting techniques also to signature verification.

The framework of contextual features allows us to define new features which capture the longer term trends of dynamic information. For example, Figure 6.4 shows the velocity profiles of a genuine and a forged signature where the velocity peaks are the main difference. Since the peak velocity $v_{\text{max}}(k)$ is already computed for every segment k, a contextual feature

$$f_{\text{delayedVelocity}} = \frac{v_{\text{max}}(k-t)}{v_{\text{max}}(k) + v_{\text{max}}(k-t)}$$

for current segment k and delay t is easily defined. Such a feature is similar to the size feature in Subsection 4.1.5. In the same way, contextual features for pressure, acceleration and tilt relations can be modeled based on average, maximum and other features.

Finally, we have to realize that both the spatial and dynamic information in a signature play an important role. The combination of a spatial feature vector (Subsection 4.1.3) with the previously discussed dynamic features allows simultaneous

modeling of all properties. Combined with the framework of delta and contextual features to model long term trends, a powerful signature representation in the context of hidden Markov models is possible. The merits of individual features are studied in Chapter 7.

6.2.4 Signature verification with LDA

The theory and application of linear discriminant analysis in handwriting recognition has been discussed earlier in Subsection 4.1.6. This subsection discusses how to employ LDA in the context of signature verification.

First, the objective is to use LDA for easier discrimination between genuine and forged signatures. Second, the objective is to exploit the statistics of a signature and its hidden Markov model to compare and automatically select features. Third, the objective of LDA is to improve the feature vector compactness.

The first objective is achieved with the naive approach of a class definition in the LDA sense of 'genuine' and 'forgery' signatures. This means that a representative set of forgeries is needed for each genuine signature. An LDA transformation is subsequently used to transform the input for optimal class discrimination. Approaches based on a model of the forgeries (without LDA) are described by Plamondon [1994b] and include examples of the use of neural networks in signature verification by Bromley, Bentz, Bottou, Guyon, LeCun, Moore, Sackinger & Shah [1993]. An approach based on synthetic discriminant functions (SDF) is demonstrated by Wilkinson, Pender & Goodman [1991].

The fundamental flaw of all signature verification approaches with a two class approach (genuine and forged) and on basis of a complete forgery model is the lack of realistic forgeries during training. It is not practical to enroll forgeries for every individual at the same time as the genuine signature. Neither can we guarantee that sufficient forgeries have been enrolled to estimate the class of forged signatures reliably.

As a solution, we adopt an approach where LDA is based only on original signature data. This is achieved with a different class definition, i.e., we adopt every state of the signature model as a different class. This is the same approach as in Subsection 4.1.6 for handwriting recognition. The rationale behind this approach is that the LDA transformation produces a more specific model with an improved state discrimination. This simply means that a forgery has to be more accurate to be falsely accepted. Note that this approach emphasizes the fit of observations in the hidden Markov model states, i.e., the probability distribution function $b(o_t)$.

The second objective is achieved by removing features from the feature vector. This is possible because after the LDA transformation, the features are ordered according to their eigenvalues. Those with a large eigenvalue contribute most to class discrimination. Because the LDA transformation is computed from the gen-

uine signatures from one writer, the transform is writer-dependent and the selected features are *automatically* the most discriminative, personalized features.

In contrast to the approach by Crane & Ostrem [1983] or Lee, Berger & Aviczer [1996] who search the original feature space to find the best feature set, we use an approach based on LDA to automatically select features in the transformed feature space. However, the objective of Crane & Ostrem [1983] is to directly minimize the false acceptance and false rejection rate while the objective of the LDA feature selection approach is to enable better class discrimination. Based on our experience with LDA and handwriting recognition from Subsection 4.1.6, it is expected that LDA also reduces the false acceptance and false rejection rates.

6.3 Model

In the introduction of this chapter and the system overview of Section 6.1, it is already mentioned that a left-to-right hidden Markov model is used to model a signature. This is similar to Subsection 4.2.1 where the hidden Markov model structure for characters in handwriting recognition is explained. The exact structure of the hidden Markov model for signature verification is discussed in Subsection 6.3.1. In addition to handwriting recognition based on hidden Markov models, signature verification requires the use of a threshold to implement the binary decision of Definition 3.35. We discuss the threshold alternatives in Subsection 6.3.2. Signature verification also requires the comparison of one written signature with one signature model. This is normally done with the Viterbi algorithm. A normalization of the computed loglikelihood with respect to signature length is discussed in Subsection 6.3.3.

6.3.1 Structure

We choose to model each signature with a single hidden Markov model and concentrate on the model structure and the number of parameters. The basic structure of the hidden Markov model is identical to the hidden Markov model used for a character as presented in Subsection 4.2.1. The number of states depends on the number of observation vectors per signature. No explicit normalization of the input signal [Sato & Kogure, 1982; Lee, Berger & Aviczer, 1996] is necessary.

Once the hidden Markov model structure is defined then the reference model is computed based on a number of example signatures and the techniques in Subsection 4.3.2. The simplicity of the hidden Markov model is that reference model construction is *automatic* and *data-driven*. A possible disadvantage is that sufficient data is needed to compute a valid model.

Because the signature is never the same, we have to model several variations which can include extra underlines, dots or other variations. Instead of using sev-

eral reference signatures, i.e., several hidden Markov models, we use a hidden Markov model which allows more than one density per state. If the individual indeed uses several types of signatures then the training procedure for hidden Markov models will automatically use up to four densities per state to model these variations.

In contrast to that, the construction of a reference model in other approaches is often more complicated. Yoshimura, Kato, Matsuda & Yashimura [1991] select two or three representative signatures from a set of genuine signatures and propose a clustering techniques to achieve this. Liu [1978] also selects a group of representative signatures.

The size of the hidden Markov model in number of parameters is computed as

$$s(\lambda) = N \cdot (K \cdot 2) \cdot L + 1 + 3 \cdot N$$

where N is the number of states, L is the feature vector dimension and K the number of densities per state. The factor two is because of the density-specific covariance. For example, an individual who writes his or her name with an assumed number of 15 characters where each character generates about three segments (Chapter 2) with a maximum number of four densities and 13 features will generate $15 \cdot 3 \cdot 2 = 90$ segments. The additional factor 2 is because of the assumption that segments spatially overlap for 50% which leads to twice the number of segments. In a worst case scenario, this leads to a model with 90 states with a total size of $s(\lambda) = 90 \cdot (4 \cdot 2) \cdot 13 + 1 + 3 \cdot 90 = 9631$ parameters.

If the first objective is not performance but model size then the model size is easily reduced with techniques from Chapter 4. The use of tied covariances reduces the number of parameters with almost a factor two but causes a performance loss as discussed in Subsection 5.3.3. An LDA transformation reduces the number of features. In an extreme case, the number of states is fixed to one state, which works well in character recognition [Bellegarda, Bellegarda, Nahamoo & Nathan, 1994], and the model size is reduced with a factor N.

6.3.2 Thresholds

In Chapter 3, we formally defined the hidden Markov model. This was followed by a definition of signature verification in Definition 3.35. Because the main difference between handwriting recognition based on hidden Markov models and signature verification is the employed threshold, this subsection concentrates on thresholds in signature verification.

The choice and the determination of a threshold τ is essential for the performance of the verification system. The threshold separates the genuine from the forged signatures. In the context of this thesis, the threshold is a loglikelihood limit. We assume that we have training, validation and test signatures for every

individual i in the database. For each individual, the average loglikelihood of train, validate and test signatures is denoted l_{train}, l_{validate} and l_{test}, respectively. Figure 6.7 explains graphically the role of a threshold τ and provides a clue to automatic threshold determination.

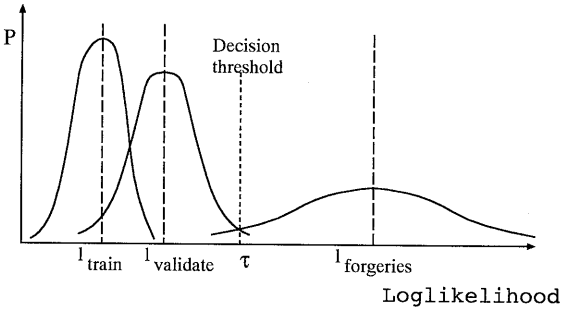


Figure 6.7. Relation between the average loglikelihood of training, validation, and forged signatures and a threshold which separates genuine and forged signatures.

Subsection 6.1.5 and Yang [1995] already mentioned the use of fixed, factorized and personalized thresholds. Therefore, we implement the following threshold types in signature verification with an increasing number of free parameters: uniform τ_u, adaptive τ_{at} and adaptive personalized τ_{apt}. The threshold is used to balance the equal-error rate and to customize the verification system for either low false acceptance or low false rejection. The effect of different thresholds is studied in detail in Chapter 7.

The *uniform* threshold $\tau_u = c$ implements a uniform strategy for all models. One fixed, loglikelihood value is chosen as a threshold for all individuals.

More adaptation is achieved with an *adaptive threshold.* Given the factor d which is equal for all individuals, we present two examples of adaptive thresholds. The first example is a threshold which depends on l_{train} and also on the standard deviation of the loglikelihood of the training signatures. The idea is that we should be able to compute a more accurate threshold $\tau_{at} = l_{\text{train}} + d \cdot \sigma_{\text{train}}$ when the signature exhibits few variations.

The second example achieves adaptation by means of additional signature examples. The mean loglikelihood l_{validate} is used. The idea is that only few validation signatures are available. Therefore, there is not enough data to estimate the standard deviation σ_{validate} reliably. This leads to the adaptive threshold: $\tau_{at} = l_{\text{validate}} + d$.

The previous adaptive threshold τ_{at} contains a factor d constant for all individuals. The objective of *adaptive personalized threshold* τ_{apt} is to adapt the threshold to writer-specific variations by adapting d to an individual i. Examples are $\tau_{apt} = l_{\text{train}} + d_i \cdot \sigma_{\text{train}}$ and $\tau_{apt} = l_{\text{validate}} + d_i$.

6.3.3 Signature scoring

In the previous sections, we assumed that the Viterbi algorithm computes the log-likelihood of the preprocessed signature O given the model λ which results in

$$l_{\text{std}}(O,\lambda) = -\log p(O|\lambda) = -[\log \pi_{q_1} + \sum_{t=1}^{T-1} \log a_{q_t q_{t+1}} + \sum_{t=1}^{T} \log b_{q_t}(o_t)].$$

This calculation is used in speech and handwriting recognition to score the log-likelihood of spoken and written data and is used by Yang [1995] in the context of hidden Markov model signature verification.

The transition probability $a_{q_t q_{t+1}}$ is essential in speech and handwriting recognition to compensate speed and size input variations. Most of these variations originate in the requirement of writer-independent recognition. Because signatures are inherent writer-dependent, the variation in writing time will be smaller compared to the writing time of characters or words in a writer-independent recognition system. Therefore, time-alignment for signature verification is less important.

In addition, the way LDA is used points to an alternative cost calculation based on the probability distribution function only which amounts to

$$l_{\text{avg}}(O,\lambda) = -\frac{1}{T} \sum_{t=1}^{T} \log b_{q_t}(o_t).$$

Such a cost function neglects the time-alignment of input with hidden Markov model and concentrates on the match between data and states. In other words, this loglikelihood calculation normalizes the loglikelihood with respect to the signature writing time. The next chapter will compare both likelihood scores experimentally.

7

Verification Experiments

This chapter discusses the empirical results of an extensive on-line, signature verification study of which the theoretical framework is discussed in Chapter 6.

The main objectives of this chapter are to explore the benefits of new digitizer technology in combination with a verification system based on hidden Markov models. This includes detailed questions about the benefits of pressure and pen-tilt measurements, linear discriminant analysis and the different contributions of spatial and dynamic information.

Another objective is to carefully test the reliability and vulnerability of the signature verification system. This is tested by using different types of signature forgeries, signature properties, thresholds as defined in Definition 3.35, and a forensic perspective. All the verification results are expressed in false acceptance (FA) and false rejection (FR) rates. The threshold at which false acceptance and false rejection become equal results in the equal-error rate (EE).

Section 7.1 discusses the on-line signature database and digitizer settings. Section 7.2 fixes a number of parameters of the hidden Markov model and validates the working of the classifier on a separate dataset. Section 7.3 explores the signature representations and models and in Section 7.4 we compare our results with results presented in the literature.

7.1 Data

There are two reasons to collect our own signature databases. First, the lack of standard databases, with the exception of the CADIX dataset [Yoshimura, Kato, Matsuda & Yashimura, 1991] which includes neither pressure nor pen-tilt. Second, we explore the benefits of the PAID hardware in combination with a signature verification system based on hidden Markov models. The collected databases contain the enrolled data 'as is' because extensive cleaning or exclusion of unwanted data leads to biased results.

Table 7.1 summarizes the properties of the two data sets. *Signature1* is used for parameter testing and is collected on a digitizer with no ability to measure pen-tilt. The data of *Signature2* is used in the experiments discussed in Section 7.3 and includes tilt signals, as demonstrated in Figure 6.6. The data of *Signature2* includes tilt signals in x and y direction, respectively, which are interleaved, resulting in a halved sample rate for the pen-tilt signals. The pressure and tilt signals include 64 levels each. The chosen digitizer settings enable us to sample the pen-tip trajectory in an area of about one cm above the writing surface.

Table 7.1. Hardware settings of the PAID digitizer during the collection of the two data sets. 'pps' means 'points per second'.

Properties	**Database**	
	Signature1	*Signature2*
Dataset size	600	4770
Number of writers	10	51
Digitizer	PAID	PAID
Sample rate	180 pps	120 pps
Pressure levels	64	64
Tilt levels	-	64
Tilt sample rate	-	60 pps

The *Signature1* dataset is collected from 10 individuals who each contribute 30 signatures and 30 *over-the-shoulder* forgeries. This results in a database of 300 genuine and 300 forged signatures. The data is collected during two writing sessions.

The data acquisition procedure involves a signer and a forger. While the signer is seated at a table, the forger is looking over his/her shoulder to capture both the image and the dynamic information of the written signature. Next, the forger produces the forgeries by imitating the movements of the original signer. This procedure is repeated with the signer and forger switching roles.

The *Signature2* dataset contains data from 51 individuals who each contributed 30 genuine signatures, 30 *over-the-shoulder* (SH) forgeries, and 30 *home-improved*

(HI) forgeries. Four forensic experts of the Rijswijk National Forensic Laboratories each contributed 60 *professional* (PR) forgeries.

This large database contains a total of 1530 genuine signatures and 3000 amateur forgeries, which are divided in 1470 over-the-shoulder forgeries and 1530 home-improved forgeries, and 240 professional forgeries. The genuine signatures and amateur forgeries are collected in three to five sessions over a period of three weeks. The individuals (45 male and 6 female) are employees of Philips Nat.Lab. and Eindhoven University of Technology. Each individual is given a paper copy of an unknown signature which was to be practiced at home to obtain the home-improved forgeries. The data acquisition procedure is identical to that of *Signature1*, except for the additional home-improved forgeries, which are written at the end of the session.

It is important to mention that the signature database is split in parts for training and testing purposes, respectively. For each individual, we use 15 genuine signatures, drawn at random from the 30 genuine signatures, to train a hidden Markov model. The 15 other genuine signatures are used for testing purposes. This separation between train and test data is maintained during all experiments. However, we do not always mention this separation. The mean loglikelihood of the 15 training signatures is denoted as l_{train} while the standard deviation is denoted as σ_{train} (see also Subsection 6.3.2).

Figure 7.1 shows a number of examples. The three signatures labeled (a) are genuine signatures, the (b) column contains over-the-shoulder forgeries while the (c) column contains three home-improved forgeries. The genuine signatures display a natural variation which simplifies forging.

Figure 7.1. Example signatures from the *Signature2* dataset.

The professional forgeries are made by forensic experts specialized in document examination. These experts normally deal with off-line data and have the expertise to detect handwritten, forged signatures. Although it is unclear whether they are able to invert this knowledge and produce high-quality forgeries that could be used to test the on-line verification system, they are the most likely to produce high-quality forgeries.

First, all signatures are classified into three complexity classes of signatures that are easy, moderately easy and difficult to forge. Next, each expert selected six signatures (two from each complexity group) which they would like to forge. Because the experts knew that dynamic information is important to the verification system, they selected signatures written in a writing style similar to their own. Because some choose the same targets, the signatures of a total of 20 writers are forged. After practicing for a while, ten copies of each signature are written on the digitizer, which results in 240 professional forgeries.

7.2 Parameters

The initial set of hidden Markov model parameters is chosen on the basis of our experience with handwriting recognition. This parameter set is tested with the *Signature1* dataset.

The chosen parameters are filtering, transition probabilities and the lower bound of the diagonal covariance. Filtering of the handwriting signals is done with a moving average filter and a window width of 5 samples to remove quantization noise. The transition probabilities of the hidden Markov model are estimated during the training. The segment representation is used without overlap. The number of segments depends on the data. Density-specific, diagonal covariance is used with a lower bound of 0.25 times the state-specific covariance. The number of states per signature is fixed to $\#states = 0.8 \cdot \#observations$ (Section 5.2) while the number of densities per states is limited to four. Unlike the handwriting recognition in Section 5.2, we used a multiplication factor of 0.8 to determine the number of states per signature because a preliminary experiment indicated that this works slightly better for signatures than 0.75.

Table 7.2. Preliminary experiment with two representations and two types of forgeries. Results are given in equal-error rates [%].

Representation	Amateur forgeries	Zero-effort forgeries
`Spatial`	9.72	2.99
`Dynamic(9)`	1.42	0.60

In the experiment to verify the working of this parameter set with the *Signature1* dataset, we chose for each individual a random subset of 15 genuine signatures

which are used to train the hidden Markov model. The other 15 signatures are used for testing. The results of this experiment are presented in Table 7.2. The structure of the `Spatial` and `Dynamic`(9) representations is explained later in Subsection 7.3.1. The forgery types are discussed in Subsection 6.1.2.

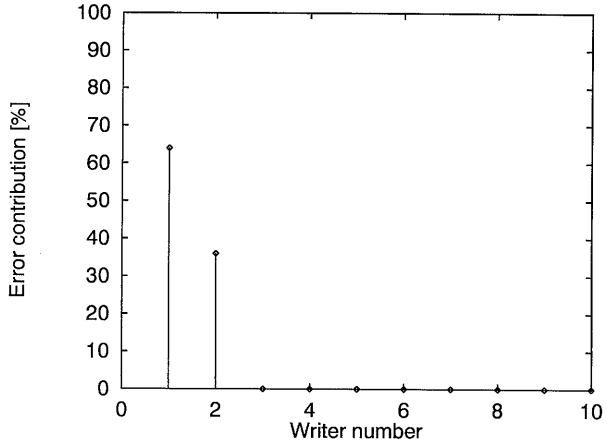


Figure 7.2. Distribution of the errors of the 10 writers of dataset *Signature1* using the `Dynamic`(9) representation. The writers are sorted according to their contribution to the error rate.

Next, we determine which writers contribute how many errors to the total error rate. The `Dynamic`(9) representation yields an equal-error rate of 1.42% with amateur forgeries. Figure 7.2 shows how the errors of the 10 writers of *Signature1* are distributed. The signatures of eight persons are perfectly verified which implies that only two persons are responsible for the errors. We conclude that our initial parameter set works fine and is a sound basis for further experiments.

7.3 Experiments

The following subsections discuss various features, representations and model improvements. In our formal problem definition in Definition 3.35, we used a threshold to distinguish between genuine and forged signatures. In the first part of the present section up to Subsection 7.3.6, we neglect the problem of threshold determination and manually choose a writer-specific threshold which results in a writer-specific equal-error rate. In the second part of this section, we automatically determine a single threshold, which is either personal or global, for all individuals as part of a full-fledged signature verification system.

After the discussion of features in signature verification based on hidden Markov models in Subsection 6.2.2, we concentrate on the structure of the feature vectors in Subsection 7.3.1. Subsection 7.3.2 compares the contribution of

pressure and pen-tilt in the original feature space as extension to a baseline representation. Subsection 7.3.3 compares two representations based on spatial and dynamic features after which Subsection 7.3.4 compares several segmentations. Next, a comparison between two thresholds is made in Subsection 7.3.5, followed by a comparison between scoring techniques in Subsection 7.3.6. The automatic determination of the thresholds is studied in Subsection 7.3.7. Subsection 7.3.8 conducts a study in greater detail of the individual feature contributions. Subsection 7.3.9 explores the consequences of the writing time of a signature. Finally, we compare a forensic perspective on signatures with the results of the on-line verification system in Subsection 7.3.10.

7.3.1 Feature vectors

In this subsection, we summarize the used feature vectors employed in signature verification. Only the feature vectors with dynamic features are summarized because the `Spatial` feature vector is discussed in Subsection 4.1.3. All the features are normalized in the interval $[0; 1]$. We first explain the baseline feature vector, after which we summarize the augmented feature vectors.

The baseline feature vector is called `Dynamic(6)` and contains the following features, discussed in Subsection 6.2.2, where v denotes the velocity and a the acceleration.

$$\begin{pmatrix} v_{\text{begin}} \\ v_{\text{end}} \\ v_{\text{max}} \\ v_{\text{avg}} \\ a_{\text{max}} \\ a_{\text{min}} \end{pmatrix}$$

The features v_{begin} and v_{end} are the velocities at the beginning and end of the segment. The feature v_{max} contains the maximal velocity in a segment while v_{avg} contains the average velocity. The features a_{max} and a_{min} contain the maximum and minimum accelerations in a segment.

Next, we concentrate on the augmented feature vectors. They are used to test the information content of the features that augment the baseline feature vector. Table 7.3 summarizes the features added to the baseline feature vector. The features p_{max} and p_{min} are the maximum and minimum pressure in a segment. In the case of a pen-up segment, both pressure features are zero. Delta pressure is modeled by two features for each feature vector which are Δp_{max} and Δp_{min}. We compute the signed delta pressure as the difference between two consecutive pressure samples, i.e., at each timestep t, we compute $\Delta p(t) = p(t) - p(t-1)$. After the grouping, we determine the maximum and minimum values of the part of delta pressure signal that is used to compute the feature vector.

Table 7.3. Signature verification feature vectors. All representations contain the baseline feature vector `Dynamic(6)` plus the additional features described in the table.

Representation	Additional features
`Dynamic(9)`	$p_{\max}, p_{\min}, \Delta p_{\max} - \Delta p_{\min}$
`Dynamic(Pressure)`	$p_{\max}, p_{\min}$
`Dynamic(Delta pressure)`	$\Delta p_{\max}, \Delta p_{\min}$
`Dynamic(Tilt-x)`	$\theta_{x_{\max}}, \theta_{x_{\min}}$
`Dynamic(Tilt-y)`	$\theta_{y_{\max}}, \theta_{y_{\min}}$
`Dynamic(Delta tilt-x)`	$\Delta\theta_{x_{\max}}, \Delta\theta_{x_{\min}}$
`Dynamic(Tilt)`	$2 \cdot \theta_{x_{\max}} - \theta_{x_{\min}}, 2 \cdot \theta_{y_{\max}} - \theta_{y_{\min}}$
`Dynamic(14)`	$p_{\max}, p_{\min}, \Delta p_{\max}, \Delta p_{\min}, \theta_{x_{\max}}$ $\theta_{x_{\min}}, \theta_{y_{\max}}, \theta_{y_{\min}}$

In addition, there is a set of tilt features where $\theta_{x_{\max}}$ is the maximal tilt value in x direction in a segment and $\theta_{x_{\min}}$ is the corresponding minimum tilt value. The $\theta_{y_{\max}}$ and $\theta_{y_{\min}}$ features are the corresponding values for tilt in y direction. Delta tilt is modeled with the feature $\Delta\theta_{x_{\max}}$, which is the maximal value for delta tilt in x direction in a segment. Similar features are computed for minimum delta tilt. Similar to the delta pressure computation, we compute the delta tilt as the difference between two consecutive tilt samples.

7.3.2 Pressure versus tilt

In this subsection, we explore and compare the effect of pressure and tilt in a signature verification system based on hidden Markov models. Because we know that pressure and tilt contain writer-dependent information (Subsection 6.2.2), we augment a baseline representation with additional features describing pressure and tilt. We expect the equal-error rate to drop when we use the augmented feature vectors.

The baseline experiment employs a baseline feature vector called `Dynamic(6)` which contains only velocity and acceleration features and was introduced in Subsection 7.3.1. The baseline results are summarized in Figure 7.3. As explained in Section 7.1, we use 15 genuine signatures to train a hidden Markov model λ_i for each individual i. The loglikelihoods of the same 15 signatures are used to determine the mean loglikelihood l_{train} and standard deviation σ_{train}. The x-axis models an increasing threshold value $\tau = l_{\text{train}} + x \cdot \sigma_{\text{train}}$. This threshold is writer-dependent because l_{train} and σ_{train} are writer-dependent. Figure 7.3 shows that the baseline experiment results in an equal-error rate of 12%. Note that the false acceptance curve (FA) in the figure is a combination of the false acceptance rate of home-improved (FA-HI) and over-the-shoulder forgeries (FA-SH).

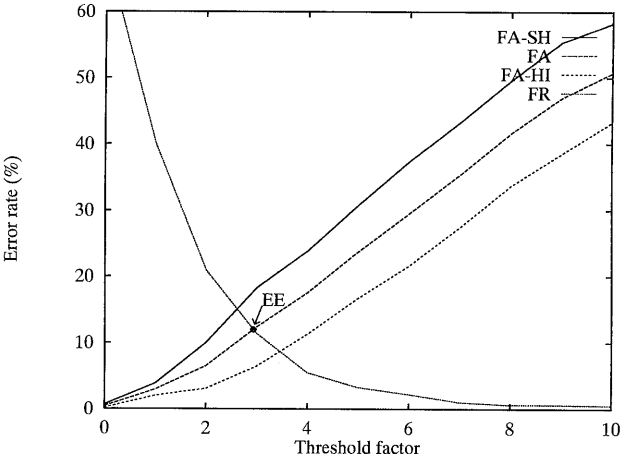


Figure 7.3. Error rates [%] for different thresholds with the `Dynamic(6)` feature vector where FA-SH is the false acceptance rate measured with only the over-the-shoulder forgeries, FA-HI is the false acceptance rate measured with the home-improved forgeries, and FA is measured with the combined set of forgeries.

Table 7.4. Equal-error rates [%] of the augmented signature representations where EE-SH is the equal-error rate measured with only the over-the-shoulder forgeries, EE-HI is the equal-error rate measured with the home-improved forgeries, and EE is measured with the combined set of forgeries. Improvement in EE is measured with respect to the `Dynamic(6)` representation.

Representation	**EE**	**EE-SH**	**EE-HI**	**Improvement**
`Dynamic`(6)	12	15.1	8.6	0
`Dynamic`(Pressure)	7.8	9.3	5.7	35%
`Dynamic`(Delta pressure)	10.8	12.4	8.7	10%
`Dynamic`(Tilt-x)	9.3	11.3	6.5	22%
`Dynamic`(Tilt-y)	9.7	11.6	7.4	19%
`Dynamic`(Delta tilt-x)	15.2	18.8	11.4	-27%
`Dynamic`(Tilt)	7.2	9.4	4.9	40%
`Dynamic`(14)	6.0	7.9	3.9	50%

Table 7.4 shows the equal-error rates using the baseline and augmented representations. The representations have been explained in Subsection 7.3.1. Most feature vectors contain eight features. The column 'Improvement' denotes the improvement of the equal-error rate over the baseline performance.

First, we note that the addition of the pressure features reduces the error rate by 35%. This improvement is explained by the fact that a pressure profile contains writer-dependent pressure information and, implicitly, the timing of pen scribbles. Another reason is the fact that a pen-up segment has zero pressure which enables the representation to model the timing of pen-up and pen-down segments.

Second, the delta-pressure features lead to a 10% improvement which is less than the improvement by pressure features. This indicates that delta-pressure contains less useful information than pressure. The pen-up and pen-down information encapsulated in the pressure profile is lost due to the delta operation.

Third, pen-tilt is continuously measured while the stylus is on or above the writing surface and contains information about an individual's pen-grip and whether the writer is left or right-handed. The tilt measurements do not include implicit information about pen-up and pen-down segments. While the addition of a tilt-x feature improves the performance by 22%, the tilt-y feature improves the performance by 19%. The combination of tilt-x and tilt-y features in the `Dynamic`(Tilt) feature vector is very successful. The error-rate reduction is 40%, which is almost the sum of the individual improvements. Because the error rate reduction due to pressure features is 35%, which is smaller than the reduction of 40% achieved with the tilt features, we have an indication that the combined tilt information is more important than pressure information. We validate this result later in Subsection 7.3.8. The delta tilt-x features lead to an increase of the error rates. Therefore, we have to assume that delta tilt does not contain useful information.

Last, the `Dynamic`(14) feature vector adds pressure, delta pressure, tilt-x and tilt-y to the baseline representation which results in a 50% error-rate reduction due to the combined features. However, this is only 10% more compared to the reduction with `Dynamic`(Tilt).

Similar to `Dynamic`(14), the `Dynamic`(13) representation combines the best features from previously used feature vectors. The `Dynamic`(Pressure) and the `Dynamic`(Tilt) are combined and extended with a delta pressure feature, an additional velocity feature representing the maximal velocity in relation to the average velocity and the number of samples per segment as an indication of segment size. We have chosen to use `Dynamic`(13) instead of `Dynamic`(14) in the next subsections for two reasons. First, because all feature vectors investigated in this subsection, except `Dynamic`(Delta tilt-x), resulted in an improvement of the equal-error rate. Hence, any combination of dynamic features in a feature vector like `Dynamic`(14), with either 13 or 14 features, will give a very good starting point

for further experiments. Second, because `Dynamic(13)` contains 13 features just like the `Spatial` feature vector which simplifies the implementation.

$$\texttt{Dynamic}(13) = \begin{pmatrix} v_{\text{begin}} \\ v_{\text{end}} \\ v_{\max} \\ v_{\text{avg}} \\ v_{\max} - v_{\text{avg}} \\ a_{\max} \\ a_{\min} \\ p_{\max} \\ p_{\min} \\ \Delta p_{\max} - \Delta p_{\min} \\ 2 \cdot \theta_{x_{\max}} - \theta_{x_{\min}} \\ 2 \cdot \theta_{y_{\max}} - \theta_{y_{\min}} \\ \text{number of samples} \end{pmatrix}$$

7.3.3 Spatial versus dynamic features

In this subsection, we compare the information content of features which model the spatial and dynamic information in a signature. The spatial features in the `Spatial` feature vector describe the signature in an off-line way, as discussed in Subsection 4.1.3, and is also used in the handwriting recognition experiments in Chapter 5. In contrast, the dynamic features in the `Dynamic(13)` representation, which has been chosen in the previous subsection as feature vector containing the dynamic information, fully exploits the dynamic signature properties.

Because we know from the literature [Plamondon & Lorette, 1989; Yoshimura & Yoshimura, 1994] that off-line signature verification is usually less reliable than on-line verification, we expect the verification based on the `Dynamic(13)` representation to outperform the `Spatial` feature vector. This is also suggested by the preliminary test in Table 7.2.

Figure 7.4 compares the verification results using the feature vectors `Spatial` and `Dynamic(13)` based on dataset *Signature2*. The experiment results in equal-error rates of 13.3% and 6.4% respectively, which shows that the use of dynamic features results in a more accurate verification.

7.3.4 Segmentation

After the comparison of different representations in the previous subsections, we now compare different segmentation techniques. The *standard* segmentation computes the segment boundaries based on the velocity inversions in y direction, i.e., the condition $v_y = 0$. The alternatives are *segment overlap* and $v_x v_y$ *segmentation*. The *segment overlap* segmentation requires the segments to have a spatial overlap

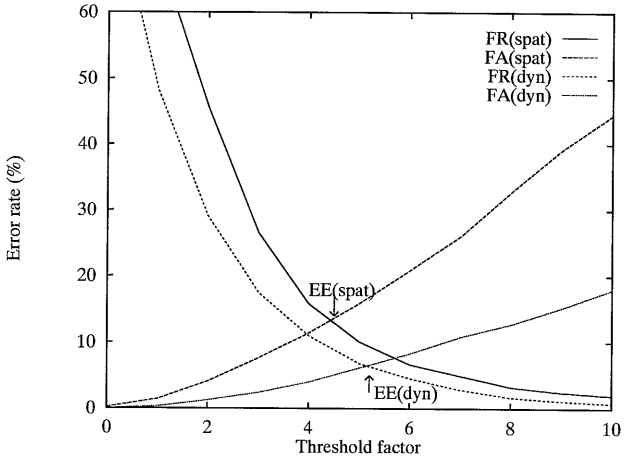


Figure 7.4. A comparison of the `Dynamic(13)` and `Spatial` feature vector. Error rates in [%].

of 50% which benefits include a better resolution because of the increased number of segments as explained in Subsection 2.5.1 and Section 5.2. The $v_x v_y$ *segmentation* determines the segment boundaries based on the velocity inversions in not only y but also x direction, i.e., the condition of either $v_y = 0$ or $v_x = 0$. Although this segmentation approach increases the resolution, the drawback is that only the position of the $v_y = 0$ is stable (Subsection 2.2.2) while the position of the $v_x = 0$ points is variable and writer-dependent.

Table 7.5 shows the average number of segments of each segmentation based on the *Signature2* dataset. The combination of *segment overlap* and $v_x v_y$ *segmentation* yields twice as many segments compared to the standard segmentation.

Table 7.5. Average number of observations per segmentation.

Number of observations	**Segmentation**	
	v_y	$v_x v_y$
Standard	26.30	32.04
Segment overlap	46.90	56.54

The question is which segmentation is the best basis for the verification system based on hidden Markov models. In our studies on handwriting recognition, we observed that the segment overlap improves the recognition performance. We also know that there are a number of short signatures which are segmented into only four or five segments when the standard segmentation is used. A larger number of segments with the same number of states per model results in more training data and a better signature model.

Table 7.6. Equal-error rates [%] for different signature segmentations. The improvement [%] relative to the performance of the standard segmentation is given between brackets.

Representation	**Segmentation**			
	Standard	v_x, v_y	**Overlap**	v_x, v_y **and Overlap**
`Dynamic(6)`	12.0	8.9 (25.8)	11.0 (8.3)	8.1 (32.5)
`Dynamic(13)`	6.4	5.8 (9.4)	5.0 (21.9)	5.4 (15.6)
`Spatial`	13.3	13.5 (-1.5)	10.8 (18.8)	10.9 (18.0)
Avg. improvement	-	(11.2)	(16.3)	(22.0)

In Table 7.6, we compare several segmentations techniques. We measure relative error-rate improvements of up to 32.5%. The effect of the segmentation depends strongly on the used representation. Because the average improvement for the combined $v_x v_y$ and overlap segmentation is best, we choose this combined segmentation as the basis for further studies in this chapter.

7.3.5 Threshold

It is known from the literature that a threshold function is an important part of a signature verification system [Plamondon & Lorette, 1989; Yang, 1995]. Therefore, we introduced the uniform, adaptive and adaptive personal thresholds in Subsection 6.3.2. The objective of this subsection is to compare the effect of an adaptive threshold τ_{at} with an adaptive personal threshold τ_{apt}. Additionally, we want to measure the best possible equal-error rate based on a manually determined, personal threshold which serves as a target while developing a procedure for automatic, adaptive threshold determination.

The best result with the `Dynamic(13)` representation is a 5.4% equal-error rate. We used the $v_x v_y$ plus additional *overlap* segmentation which performed best in the previous subsection. The threshold involved is τ_{at} which is adaptive because it uses the mean loglikelihood l_{train} of the training signatures of an individual as offset, combined with the standard deviation σ_{train} times a multiplication factor x. The threshold which minimizes the equal-error rate for all writers simultaneously is $\tau_{at} = l_{\text{train}} + 5.6 \cdot \sigma_{\text{train}}$, with multiplication factor $x = 5.6$, which is plotted in Figure 7.5 as a straight line.

Figure 7.5 plots the factor x in an adaptive personal threshold according to the formula $\tau_{apt} = l_{\text{train}} + x \cdot \sigma_{\text{train}}$. The writers on the x-axis are sorted according to an increasing number of segments in their signature.

We learn from Figure 7.5 that there is no clear relation between the multiplication factor of the standard deviation σ_{train} of the training signatures and the signature length in terms of the number of segments. This also means that the

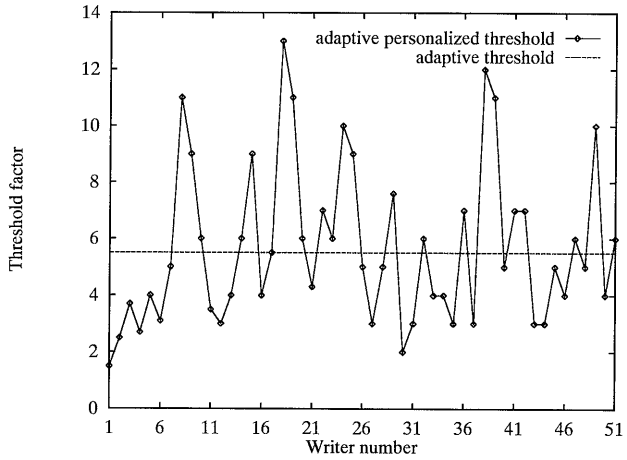


Figure 7.5. Comparison of the adaptive and the adaptive personal threshold.

quality of the signature models is diverse and indicates that the intra-class signature deviations are very different.

Table 7.7. Error rates [%] obtained with different thresholds.

Threshold	**Error rates**			
	FR	**FA**	**FA-SH**	**FA-HI**
τ_{at}	5.4	5.4	9.4	1.4
τ_{apt}	1.36	1.36	2.18	0.54

Table 7.7 compares the performance of an adaptive threshold τ_{at}, where the multiplication factor is $x = 5.6$, with that of the adaptive personal threshold τ_{apt} where x is determined such that the best personal equal-error rate is determined. The equal-error rate with an adaptive personal threshold is 1.36% which is 75% lower compared to the equal-error rate of 5.4% using an adaptive threshold. Note that the false acceptance rate for over-the-shoulder forgeries is four to six times the false acceptance rate of the home-improved forgeries.

On basis of the adaptive threshold τ_{at}, with multiplication factor $x = 5.6$ fixed for all individuals, we construct a procedure which automatically derives the threshold τ_{at} during the enrollment of a new individual. However, Table 7.7 shows that an adaptive personal threshold τ_{apt} results in a good equal-error rate of 1.36% and outperforms the automatic, adaptive threshold. Because we can not determine the τ_{apt} threshold automatically without the availability of forgeries for every signature, which is impractical in a consumer application, we concentrate on the adaptive threshold τ_{at} of which the multiplication factor is fixed at a best value for all signatures. Therefore, we conclude that we have to devise an algorithm for automatically finding the best threshold τ_{at}. This is studied in Subsection 7.3.7.

7.3.6 Score

We explore a number of extra techniques in attempts to achieve a better verification performance. First, we combine the features of the `Dynamic`(13) and `Spatial` representation with six contextual features. These are angular features with delay 1,2, and 3. This results in a representation which contains $13+13+6=32$ features called `Combined`(32). Our reason for doing this is that dynamic and spatial features provide complementary information while the contextual features have proven their benefits in handwriting recognition in Subsection 5.3.1. The combination of spatial and dynamic information was found to be important in a preliminary experiment where only the `Dynamic`(13) representation was used and a signature could be forged although it had no spatial similarity at all with the image of the genuine signature, just a similarity of dynamic properties.

Next, we apply an LDA transformation to each signature where the states of the hidden Markov model are the classes in the LDA sense. We modify the threshold τ_{at} to use l_{validate} instead of l_{train}. This is based on Figure 6.7 and explained in more detail in the next subsection.

The objective of the experiments in this subsection is to compare two scoring techniques. The first is the standard score $l_{\text{std}}(O,\lambda)$ which we used in the previous experiments. The second is the score $l_{\text{avg}}(O,\lambda)$, which is an average score over all states as explained in Subsection 6.3.3. This score normalizes the duration of the signature. Both scores are compared on the basis of the `Combined`(32) representation and the adaptive threshold τ_{at} of Subsection 7.3.5.

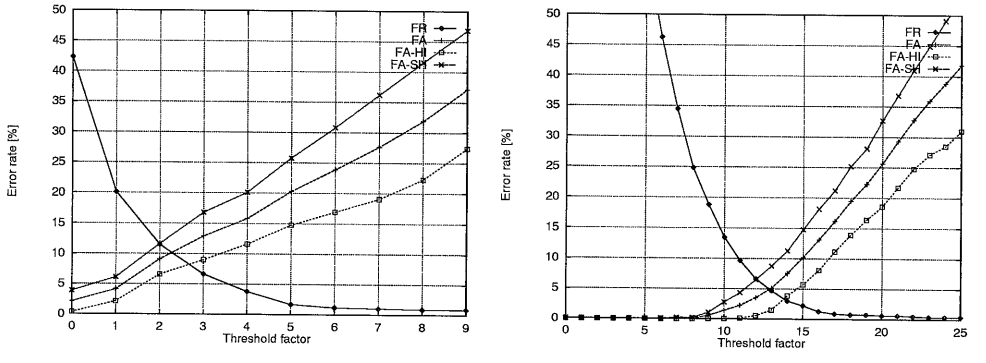


Figure 7.6. Comparison of two classifiers with the `Combined`(32) representation, adaptive thresholds and different scoring mechanisms. Left: the standard log-likelihood score $l_{\text{std}}(O,\lambda)$. Right: the normalized loglikelihood score $l_{\text{avg}}(O,\lambda)$.

Figure 7.6 shows that scoring with $l_{\text{std}}(O,\lambda)$ and $l_{\text{avg}}(O,\lambda)$ results in equal-error rates of 10.2% and 4.9%, respectively. Therefore, the normalized score $l_{\text{avg}}(O,\lambda)$ outperforms $l_{\text{std}}(O,\lambda)$ and $l_{\text{avg}}(O,\lambda)$ is used throughout the rest of the chapter.

7.3.7 Automatic threshold

In this subsection we compare three thresholds in a classifier based on the `Combined`(32) representation, the LDA transformation and the normalized score $l_{\text{avg}}(O,\lambda)$ explained in Subsection 6.3.3. The objective is to determine which threshold yields the lowest equal-error rate. In Subsection 7.3.5 we already showed that an adaptive personal threshold τ_{apt} outperforms an adaptive threshold τ_{at} by 75%, but the adaptive personal threshold was determined manually to realize the best possible equal-error rate for every writer.

The objective here is to use writer-specific information in combination with an *automatically* determined threshold to improve the verification performance without human interference during or after the enrollment. We investigate three thresholds on the basis of Figure 6.7 where the factor x is a variable multiplication or addition factor. The goal is to determine the best value for x, independent of the writer, in the three thresholds

$$\begin{aligned}
\tau_{\text{AdaptiveSigma}} &= l_{\text{validate}} + x \cdot \sigma_{\text{train}}, \\
\tau_{\text{Adaptive}} &= l_{\text{validate}} + x \cdot 2, \\
\tau_{\text{Fixed}} &= 82 + x.
\end{aligned}$$

The τ_{Fixed} threshold is a fixed threshold for all the signatures. The other two thresholds are based on the average likelihood l_{validate} of five validation signatures, in addition to the 15 training signatures, plus an offset which is either fixed (τ_{Adaptive}) or depends on the standard deviation of the training signatures ($\tau_{\text{AdaptiveSigma}}$).

Note that $\tau_{\text{AdaptiveSigma}}$ does not depend on the standard deviation of the validation signatures because earlier experiments showed that five validation signatures are not sufficient for a reliable estimation of the standard deviation. The value 82 in τ_{Fixed} is based on the average loglikelihood per state, i.e., $-\log b(o_t)$, as used in the previous subsection.

Figure 7.7 plots the false acceptance and false rejection curves obtained in the experiment to compare the thresholds τ_{Adaptive} and τ_{Fixed}. The equal-error rates are 1.9% and 2.2%, respectively. The earlier experiment with $\tau_{\text{AdaptiveSigma}}$ results in an equal-error rate of 4.9%, which is poorer than the results obtained with the other thresholds. The false acceptance and false rejection curves of the $\tau_{\text{AdaptiveSigma}}$ experiment are presented in the right figure of Figure 7.6. The equal-error rates of 1.9% and 2.2%, respectively, are the best results so far with a threshold which is determined completely automatically. The difference between the equal-error rates obtained in the experiments with τ_{Fixed} and τ_{Adaptive} is only 0.3%, which is remarkably small. When the threshold τ_{Adaptive} is determined manually (see Subsection 7.3.5) for each writer with the objective of a personal equal-error rate, we find an overall equal-error rate of 0.56%.

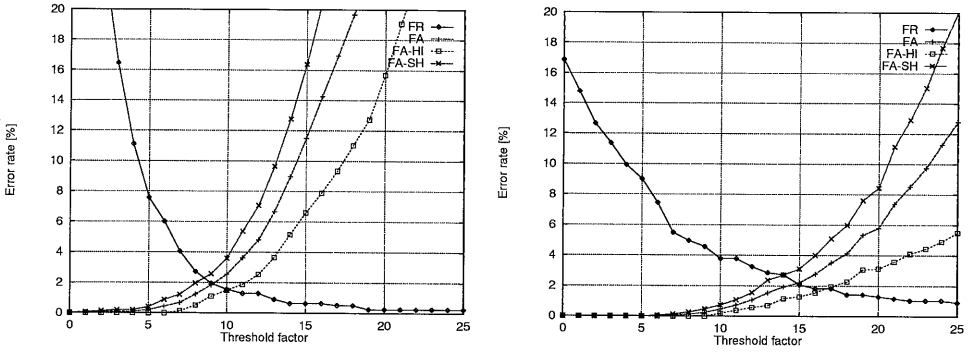


Figure 7.7. Comparison of two classifiers with two different adaptive thresholds and `Combined`(32) representation. Left: the classifier with the adaptive threshold τ_{Adaptive}. Right: the classifier with the fixed threshold τ_{Fixed}.

Figure 7.7 also shows that the over-the-shoulder forgeries are better forgeries than the home-improved forgeries which confirms Figure 7.3 where we found the same result. Analysis of the results of the τ_{Adaptive} experiment confirms that, at the equal-error point, we have a false acceptance of 2.58% for the over-the-shoulder forgeries and 1.11% for the home-improved forgeries. This means that the number of accepted over-the-shoulder forgeries is roughly twice the number of accepted home-improved forgeries.

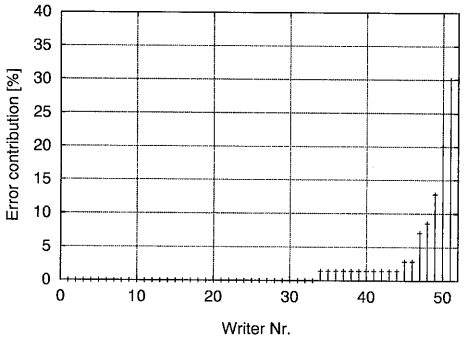


Figure 7.8. Distribution of the errors over all writers using `Combined`(32). The writers are sorted according to their contribution to the error rate.

Although Figure 7.7 gives a clear overview of the error distribution for all writers, it is not clear what the individual error contributions are. This is plotted in Figure 7.8 for the experiment with the τ_{Adaptive} threshold at the point of equal-error rate. It is obvious that two writers are responsible for 50% of all errors and five writers are responsible for 80% of all errors. The verification is perfect for 33 writers.

The conclusion is that it is possible to use an automatic, writer-specific threshold and that such a threshold results in an equal-error rate of 1.9%. The majority of all errors is caused by only a small number of writers. This is investigated in more detail in an experiment in one of the following subsections. For the sake of completeness, we mention the result obtained with an adaptive personal threshold, which is an equal-error rate of 0.56% with perfect verification of the signatures of 44 writers.

7.3.8 Feature analysis

Previous experiments in Subsection 7.3.2 showed that pressure and tilt features contain information that reduce the equal-error rate with up to 50%. In this subsection, we use the LDA transformation to compare the relative importance of features. The examined feature vector is the `Combined`(32) feature vector, as explained in Subsection 7.3.6, which contains spatial, dynamic and contextual features and enables a simultaneous comparison of all individual features.

The procedure employed is to compute the LDA transformation matrix for every signature model. This results in a feature ranking based on the features' discriminative value. The ranking is averaged over all the models which results in an average rank for each feature. The sorted list of the average feature ranks is summarized in Table 7.8, which is used as a qualitative indication of feature importance in this signature verification task.

Table 7.8. Feature ranking of all the features in the `Combined`(32) feature representation. Rank 0 is best, 31 is worst. *DeltaPresDiff* is a description of the feature Δ PresMax - Δ PresMin. *VelDiff* describes the feature $VelMax - VelMin$.

Feature	**Rank**	**Feature**	**Rank**	**Feature**	**Rank**
Pres-min	0	PenDown	12	DelayedAngle3-cos	2
Tilt-x	1	CosPhi-2	15	DelayedAngle3-sin	5
Vel-max	3	AspectRatio	16	DelayedAngle1-cos	9
Tilt-y	4	SinPhi-2	17	DelayedAngle1-sin	13
Nr-of-samples	6	SinPhi-1	20	DelayedAngle2-cos	14
Acc-min	7	Curve	22	DelayedAngle2-sin	15
Acc-max	8	CosPhi-3	24		
Vel-begin	10	CosPhi-1	25		
Vel-end	11	CosPhi-4	26		
VelDiff	17	SinPhi-3	27		
Pres-max	21	SinPhi-4	28		
Vel-avg	23	CosPhi-5	29		
DeltaPresDiff	31	SinPhi-5	30		
Avg.Rank	10.9	Avg.Rank	21.5	Avg.Rank	9.6

Table 7.8 shows that the average rank of the dynamic features is 10.9, that of the spatial features 21.5 and that of the contextual features 9.6. According to these figures, the dynamic features are more important than the spatial features. This is in line with the results in Subsection 7.3.3 where we examined the importance of features for a different type of experiment.

Because most features are computed on the basis of a *segment*, the features reflect the local, short-term structure of the signature. The global, long-term structure is not modeled with features but only implicitly by the number of states. An intermediate approach is to compute contextual features which describe signature parts spanning several segments and are able to capture the mid-term structure of the signature.

When we use Table 7.8 to compare the *mid-term* handwriting trends modeled with the aid of contextual features with *local* dynamic features, then we note that the contextual features are more important. Because it is difficult to devise features which capture the global structure of the signature in the hidden Markov model framework, it seems advisable to add more contextual features to describe more accurately the spatial and dynamic structure of the signature. Examples are discussed in Subsection 4.1.5 and Subsection 6.2.3.

7.3.9 Signature duration

In Subsection 7.3.7 we found that the automatic, adaptive threshold τ_{Adaptive} yields an equal-error rate of 1.9% while the fixed threshold τ_{Fixed} leads to an equal-error rate of 2.2%. These results serve as a starting point to the main question in this subsection, which is whether the duration of a signature affects the verification performance. The hypothesis is that a short signature contains less writer-dependent information, resulting in a greater probability of a forgery being accepted.

First, we plot the average duration of a signature of all the writers in Figure 7.9. The maximum writing time of a signature in this dataset is six seconds. Figure 7.9 shows that there are two signatures with average durations of less than a second. The durations of the other signatures is distributed across the interval from one to six seconds.

We then excluded all the signatures shorter than t seconds from the verification results. The result is plotted in Figure 7.10 for the adaptive and fixed threshold experiments. We used the equal-error rates of 1.9% and 2.2% for the aforementioned adaptive and fixed threshold, respectively, as a starting point which defines the total number of verification errors. Because we define the number of errors for each person as the sum of false acceptance and false rejection errors, which are not necessarily the same, only the results with the condition $t = 0$ are equal-error rates. With an increasing minimum signature duration t, more signatures are excluded and the total number of errors decreases.

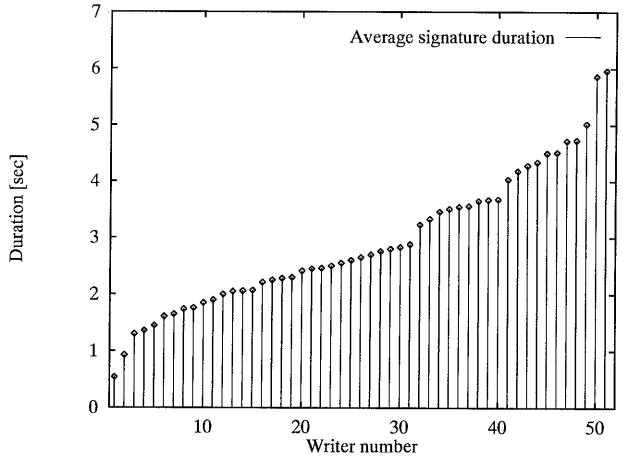


Figure 7.9. Average signature duration in seconds per writer.

When there is no correlation between signature duration and error contribution then the graph is a straight line with a constant, negative slope s. In view of our hypothesis that shorter signatures contain less information, we expect an error reduction much steeper than s for small values of the signature duration threshold t.

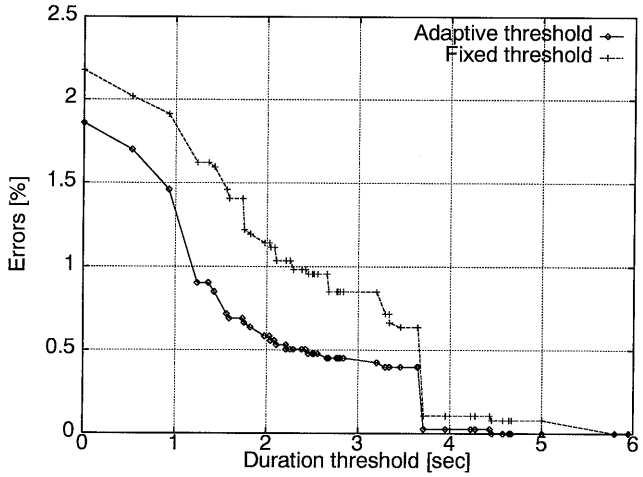


Figure 7.10. Signature duration threshold in seconds versus errors[%]. All signatures with a duration less than t seconds are excluded from the result.

Figure 7.10 shows that the error-rate of the adaptive threshold experiment is halved when we exclude the three shortest signatures with a threshold $t > 1.25$. With the same threshold, the error-rate of the fixed threshold experiment is reduced by 25%. These results correspond to an error-rate of 0.9% for the adaptive thresh-

old τ_{Adaptive}. However, the error-rate does not correspond to an equal-error rate. When the three shortest signatures with the $t > 1.25$ condition are excluded and the $l_{\text{avg}}(O, \lambda)$ score and threshold τ_{Adaptive} are used, an equal-error rate is found at $x = 9.4$, which corresponds to 1.0%. The corresponding FA-FR curves are plotted in Figure 7.11.

As an alternative to the equal-error rate, we provide the points were a 0% false acceptance or false rejection is reached in Figure 7.11. At the point of 0% false acceptance, we achieve a false rejection of 6.7%. At the point of 0% false rejection, we achieve a false acceptance of 7.1%.

Note that Figure 7.10 contains a signature with a duration of 3.7 seconds whose error contribution is much greater than that of the signatures with duration 3.6 or 3.9 seconds. Although hard to explain, the original data reveal that this individual's signature shows a high degree of variation in shape, which will affect the quality of the generated hidden Markov model.

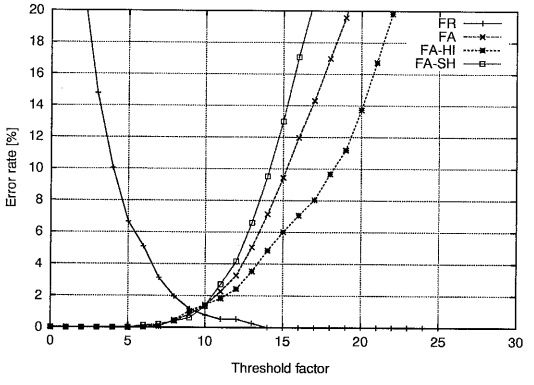


Figure 7.11. Classifier with a τ_{Adaptive} threshold. Three signatures with durations shorter than 1.25 seconds are excluded.

On the basis of the *Signature2* dataset, the hypothesis that shorter signatures contain less information is confirmed. However, there are only three writers with a signature of very short duration. The verification performance is easily improved if the enrollment procedure includes a test on minimum signature duration. When the duration of an enrolled signature is too short, the system could enroll a writers full name as an alternative.

7.3.10 Forensic perspective

This subsection concentrates on two questions related to forensic science. The first concerns the quality of professional forgeries produced by forensic handwriting experts. The second question is related to signature complexity, judged by forensic handwriting experts, in relation to the on-line verification performance with a

verification system based on hidden Markov models.

The professional forgeries are produced based on three genuine, written examples of every signature to forge. A difference with respect to the home-improved forgeries is that the amateurs used a *copy* of the three genuine signatures while the forensic experts worked with the original signatures.

Table 7.9. Error rates [%] obtained for a complete dataset with the adaptive threshold τ_{Adaptive}. The error rates are presented for each forgery type.

Representation	Equal-error rates		
	Average	Over-the-shoulder	Home-improved
Combined(32)	1.9	2.6	1.1

As a reminder, we summarize the best results in Table 7.9 which were obtained with the automatic, adaptive threshold in Figure 7.7. The representation with combined spatial, dynamic and contextual features is used. Segmentation is done with the combination of $v_x v_y$ and overlap segmentation.

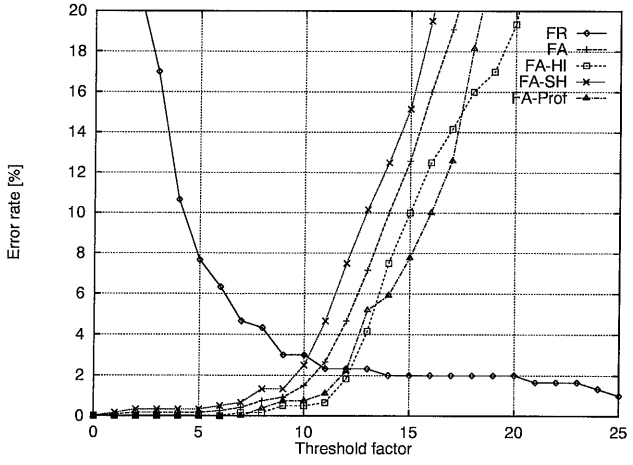


Figure 7.12. Error rates [%] for the signature subset of 20 writers for which professional forgeries are available.

Table 7.10. Equal-error rates [%] with the signature subset for which professional forgeries are available. The error rates are presented for each forgery type. 'Average' is the combined set of over-the-shoulder and home-improved forgeries.

Representation	Equal-error rates			
	Average	Professional	Over-the-shoulder	Home-improved
Combined(32)	2.45	2.33	2.88	2.33

Figure 7.12 shows the false acceptance and rejection curves separately for home-improved, over-the-shoulder and professional forgeries obtained with the same verification procedure as in Figure 7.6. The results are summarized in equal-error rates in Table 7.10. The equal-error rate called 'Average' shown in Table 7.10 is the equal-error rate determined with the combined set of home-improved and over-the-shoulder forgeries but is based on a subset of signatures and writers. This equal-error rate is higher than the equal-error rate on the complete signature database in Table 7.9. The difference is due to the different characteristics of the signatures chosen by the forensic experts.

The most important result in Table 7.10 is that there is no difference in equal-error rate between the home-improved forgeries and the professional forgeries. The over-the-shoulder forgeries made by amateurs on the basis of the dynamic data are still the best type of forgery. This result is explained by the fact that the professional forgers carefully study the original signatures. Every character is thoroughly studied and forged with great accuracy. This results in forgeries that resemble the original shape but not the dynamic profile. The result also indicates that it is difficult to extract the dynamic information from a genuine signature on paper in a limited time [Doermann, 1993].

These results indicate that the availability of dynamic information like writing velocity and pen-tilt is crucial for making good forgeries. Even forensic experts are unable to exploit genuine signatures on paper in order to 'beat' the on-line signature verification.

The second part of this subsection concentrates on signature complexity. The signatures in the *Signature2* database have been classified by the forensic experts as 'easy', 'moderately easy' or 'difficult' to forge. Table 7.11 summarizes the main properties of the three groups.

Table 7.11. Properties of the complexity groups.

Properties	**Signature complexity**			**Total**
	Easy	**Moderately easy**	**Difficult**	
Nr. of signatures	18	17	16	51
Avg. duration [sec]	2.68	2.94	3.13	2.92
Min/max duration [sec]	0.54/5.96	1.36/4.73	2.00/5.02	0.54/5.96

Table 7.12 shows the error rates for the signatures with different complexities. The 'easy' signatures have the highest equal-error rates. The decrease in equal-error rates with increasing signature complexity as judged by the forensic experts shows that the opinions of computer and forensic experts match well. This is supported by the findings of Subsection 7.3.9 where the three shortest signatures, responsible for 50% of the errors in the adaptive threshold experiment, are now classified as 'easy'.

Table 7.12. Equal error rates [%] of complexity groups versus threshold types in the classifier. The `Combined`(32) representation is used.

Threshold	Best (manual)	Adaptive	Fixed
Easy	2.2	3.0	4.0
Moderate	0.39	1.2	1.3
Difficult	0	0.2	0.4

Found & Rogers [1995] provide evidence that computer and forensic experts agree on the complexity of off-line signatures. This subsection provides a similar result for on-line signature verification although the complexity classification of genuine signatures was based only the signatures written on paper.

7.4 Discussion

We discuss the achieved results in relation to the performance reported in the literature which are summarized in Table 6.1. We summarize the achieved results and then concentrate on a comparison on basis of the use of hidden Markov models, the size of the signature database, the landmark of the best result presented in the literature and the type of employed forgeries.

An equal-error rate of 1.9% is achieved on a database of 51 writers, 1500 genuine signatures and more than 3000 skilled forgeries. With an automatic constraint that the signature duration should be longer than 1.25 seconds, this improves to an equal-error rate of 1.0%.

The number of approaches to signature verification based on hidden Markov models is very limited. One of the reported approaches by Yang, Widjaja & Prasad [1995] reports a false rejection rate of 4.44% and a false acceptance rate of 1.78% using zero-effort forgeries. Another study on basis of hidden Markov models by Mohankrishnan, Paulik & Khalil [1993b] reports equal-error rates of 2%-3% with zero-effort forgeries and a database of 1600 signatures from 16 writers. In comparison, we achieve an equal-error rate of 1.9% with skilled forgeries based on spatial and dynamic information of the genuine signatures.

Our study employs 4770 signatures. Compared with the study by Lee, Berger & Aviczer [1996] who achieved an equal-error rate of 2.5% with a larger database, we used more forgeries based on dynamic information and achieved a lower equal-error rate of 1.9%. However, our hidden Markov model is more complex.

The landmark study by Worthington, Chainer, Wilford & Gundersen [1985] used an even larger signature database and achieved a false rejection rate of 1.77% with a false acceptance of 0.28% per signature. This indicates a better result than the 1.9% reported in our study. However, we can reach an equal-error rate of 1.0% if the three persons with the shortest signatures (less than 1.25 sec) are excluded.

We can speculate that the subjects of Worthington et al. [1985] are mainly American. It was noted earlier that American signatures are on the average longer compared to European signatures. Therefore, an experiment with our hidden Markov model classifier and pure American signatures will probably show an improved result.

Another important factor in a comparison with literature results is the type of forgeries. The forgeries employed in the study by Worthington et al. [1985] are based on signature images which corresponds with our home-improved forgeries. If we only take home-improved forgeries into account then Figure 7.7 indicates an equal-error rate of 1.5%. If we choose a threshold to reach an almost zero false acceptance rate for home-improved forgeries then we find in Figure 7.7 that our study leads to a false rejection rate of 4.1% and a false acceptance rate of 0.13%.

In addition, Crane & Ostrem [1983] reached an equal-error rate of 1.5% with forgeries based on both spatial and dynamic data. This is slightly better than the 1.9% we achieved in this study. Yoshimura, Kato, Matsuda & Yashimura [1991] reached an equal-error rate of 1% using static forgeries and a dataset of 2200 signatures.

8

Conclusion

In this thesis we have discussed on-line handwriting recognition and verification based on hidden Markov models and techniques known from speech recognition. We have concentrated on representations and models and have demonstrated that good results can be obtained in on-line handwriting recognition and verification. We demonstrated large-vocabulary word recognition, character recognition and signature verification where results comparable with or better than those reported in the literature are achieved.

We have discussed a number of different representations in the context of handwriting recognition in order to address the goals of writing size independence, integration of knowledge about handwriting structure on the basis of the framework of contextual features and a robust, compact representation. We have shown that the writing size independence implicit in segment-based representations offers an attractive alternative to the usual word-based size normalization at the cost of a small decrease in peak performance. The tested representation alternatives, which include delta features, contextual features and LDA transformed features, each contribute a performance improvement. The combination of delta and contextual features yields an additional improvement. In the tested configuration for a segment-based recognizer with a 20,000 word vocabulary, the combined improvements yield a word error-rate reduction from 28% to 11.2%, which means that the error-rate is more than halved due to representation improvements alone.

Handwriting-specific knowledge has been integrated in the hidden Markov model by adding pause state models for ligature modeling, a backspace model to capture delayed strokes and contextual models to describe the effect of neighbor characters. While pause models improve the word recognition result by 15%-25% and contextual models led to a 12%-38% improvement, backspace models did not result in any improvement.

Because even humans are not able to recognize all handwritten words without context, an open-vocabulary recognition task will probably always be demanding. We recommend to combine open-vocabulary recognition with sentence recognition and the context of a language model in such a way that the language model models not only character sequences of complete words but also sequences across word boundaries. This would circumvent the problems of a character-based, open-vocabulary sentence recognition approach.

A suitable representation has proven to be vital for correct recognition. Although we can automatically select and personalize important features with LDA, the choice of the handwriting unit and number of samples per unit remains a manual choice. It is recommended to find a way of automatically determining and personalizing the necessary block-size and features based on the training data.

The on-line signature verification study discussed representations, features including pen-tilt, thresholds and a forensic perspective. Starting with an equal-error rate of 12%, the on-line signature verification based on hidden Markov models reached an equal-error rate of 1.9% and even 1.0% with a simple constraint on signature duration. This is tested with more than 4500 genuine and forged signatures by 51 writers. It is important to note that, in contrast to other researchers who used only forgeries based on static data (the image) in their signature verification studies, we provided the forgers with the opportunity to watch the signing process, which resulted in very good forgeries.

Although all the literature results are obtained using different databases, which complicates a meaningful comparison, our results are obtained using high-quality, skilled forgeries based on static and dynamic information of the genuine signature. The results indicate that we outperform other signature verification studies based on hidden Markov models and are comparable with or better than most of the results reported and summarized in Table 6.1.

We observed the fact that pen-tilt information is very important and compared it with pressure and other dynamic and spatial information. We showed that the error rate of on-line signature verification based on dynamic features outperforms that based on spatial features by a factor two. In addition, contextual features modeling mid-term, structural information were shown to outperform the dynamic features. The LDA transformation was used to personalize the feature vector. It was shown that signatures with a short duration contain little information and the exclusion

of the few shortest signatures halves the equal-error rate. An automatic thresholding technique based on hidden Markov models and the average loglikelihood was developed and its successful operation was demonstrated. We also demonstrated that the on-line signature verification system studied cannot be broken by forgeries made by forensic handwriting experts on basis of paper version of the genuine signatures. In addition, we showed that a forensic complexity classification of the genuine signatures into three classes (easy, moderately easy and difficult to forge) closely matches with the computer verification results.

We conclude that the framework of contextual features has the potential for additional improvement by modeling both short and mid-term spatial and dynamic information. The fact that forgeries based on static plus dynamic information are better than casual forgeries based on static information only leads to the recommendation to shield the direct viewing of the signing process in a signature verification product like a PIN code in an automatic teller machine. On the basis of the correlation between signature duration and error contribution, we recommend to include a signature duration threshold in signature verification products which excludes signatures shorter than 1.25 seconds. This is especially important for European signatures because in case of American signatures the legibility requirement leads to a longer average duration. If product requirements require the signature models to be stored in only a few bytes, the signature models are easily compressed by reducing and balancing the number of hidden Markov model states and densities in combination with a personalized, automatic feature selection method based on LDA to reduce the number of features.

We recommend to investigate improved techniques to automatically determine the writer-specific threshold because there is a gap between our current results of 1.9% equal-error rate and the best possible result of 0.56% based on a manually determined threshold. Next, we recommend to use LDA to select a writer-specific set of transformed features. Because of the success of the contextual features, based on spatial information, in handwriting recognition and verification, we recommend to introduce contextual features for dynamic information in signature verification.

Although we have not solved all the problems involved in handwriting recognition or verification, we may conclude that our approach based on hidden Markov models in combination with one-stage beamsearch provides an elegant, high-performance solution.

Bibliography

ABRAHAM, D., G. DOLAN, G. DOUBLE, AND J. STEVENS [1991], Transaction security system, *IBM system journal* **30**, 206–229.

ALI, F., AND T. PAVLIDIS [1977], Syntactic recognition of handwritten numerals, *IEEE Transactions on Systems, Man and Cybernetics* **7**, 537–541.

BAHL, L.R., P.F. BROWN, P.V. DE SOUZA, AND R.L. MERCER [1988], A new algorithm for the estimation of Hidden Markov Model Parameters, *International Conference of Acoustics, Speech and Signal processing*, IEEE, 493–496.

BAHL, L.R., F. JELINEK, AND R.L. MERCER [1983], A Maximum Likelihood Approach to Continuous Speech Recognition, *IEEE Transactions on pattern analysis and machine intelligence* **5**, 179–190.

BAUM, L.E. [1972], An inequality and associated maximization techniques in statistical estimation for probabilistic functions of Markov processes, *Inequalities* **3**, 1–8.

BEATSON, R. [1985], Signature dynamics in personal identification, *Congres modial de la protection et de la securite informatique et des communications*, 179–196.

BEIGI, H.S.M., K.S. NATHAN, G.J. CLARY, AND J. SUBRAHMONIA [1994], Size normalization in on-line unconstrained handwriting recognition, *International Conference on Image Processing*, 169–173.

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* **6**, 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, *International Conference of Acoustics, Speech and Signal processing*, 2631–2634.

BELLEGARDA, J.R., K.S. NATHAN, D. NAHAMOO, AND E.J. BELLEGARDA [1993], On-line handwriting recognition using continuous parameter hidden Markov models, *International Conference of Acoustics, Speech and Signal processing*, 121–124.

BELLMAN, R. [1957], *Dynamic programming*, Princeton university press.

BOES, U., G. FOGAROLI, G. MASLIN, H. KEIL, AND R.J. WHITROW [1990], *The paper interface (ESPRIT Project 295)* (Final Report ed.)., Technical report, EC, Konstanz, Germany.

BOSE, C.B., AND S. KUO [1992], Connected and degraded text recognition using hidden Markov model, *11th International Conference on Pattern Recognition*, IEEE, 116–119.

BOUMA, H. [1971], Visual Recognition of lower case letters, *Vision Research* **11**, 459–474.

BOUWHUIS, D.G. [1979], *Visual Recognition of Words*, Ph.D. thesis, Institute for Perception Research (IPO).

BOZINOVIC, R.M., AND S.N. SRIHARI [1989], Off-line cursive script word recognition, *IEEE Transactions on pattern analysis and machine intelligence* **11**, 68–83.

BROMLEY, J., J. BENTZ, L. BOTTOU, I. GUYON, Y. LECUN, C. MOORE, E. SACKINGER, AND R. SHAH [1993], Signature verification using a "siamese" time delay neural network, *International Journal of Pattern Recognition and Artificial Intelligence* **7**, 669–688.

BUNKE, M., M ROTH, AND G. SCHUKAT-TALAMAZINNI [1995], Off-line Cursive Handwriting Recognition using Hidden Markov Models, *Pattern recognition* **28**, 1399–1413.

CAESAR, T., J.M. GLOGER, AND E. MANDLER [1995], Estimating the Baseline for written Material, *3rd International Conference on Document Analysis and Recognition*, 382–385.

CHANG, L., ET AL. [1994], A comparison of Two Handwriting Recognizers for Pen-based Computers, *CASCON*, 364–371.

CHEN, M.Y., A. KUNDU, AND S.N. SRIHARI [1993], Handwritten word recognition using Continuous Density Variable Duration Hidden Markov Model, *International Conference of Acoustics, Speech and Signal processing*, IEEE, V105–V108.

CHEN, M.Y., A. KUNDU, AND J. ZHOU [1992], Off-line handwritten word recognition (HWR) using a single contextual hidden Markov model, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, IEEE Comput. Soc. Press, Los Alamitos, CA, USA, 669–672.

CHO, W., S.W. LEE, AND J.H. KIM [1995], Modeling and Recognition of Cursive Words with Hidden Markov Models, *Pattern recognition* **28**, 1941–1953.

CLARK, R.M., T. HASTIE, AND E. KISHON [1990], A model for comparing signatures, *IEEE International Conference on Systems, Man and Cybernetics*, IEEE 1, 326–330.

CORTES, C. [1995], *Prediction of generalization Ability in Learning Machines*, Ph.D. thesis, University of Rochester.

CRANE, H., AND J. OSTREM [1983], Automatic signature verification using a three-axis force-sensitive pen, *IEEE Transactions on Systems, Man and Cybernetics* **13**, 329–337.

DIMAURO, G., S. IMPEDOVO, AND G. PIRLO [1992], A stroke oriented approach to signature verification, in: S. Impedovo and J.C. Simon (eds.), *From pixels to features III: frontiers in handwriting recognition*, Elsevier science publishers b.v., 371–384.

DOERMANN, D.S. [1993], *Document Image Understanding: Integrating Recovery and Interpretation*, Ph.D. thesis, University of Maryland.

DOLFING, J.G.A. [1998], A comparison of ligature and contextual models for hidden Markov models based on-line handwriting recognition, *International Conference of Acoustics, Speech and Signal processing*, IEEE **2**, 1073–1076.

DOLFING, J.G.A., AND R. HAEB-UMBACH [1997], Signal Representations for Hidden Markov Model based on-line Handwriting Recognition, *International Conference of Acoustics, Speech and Signal processing*, IEEE **4**, 3385–3388.

DOLFING, J.G.A., AND J.J.G.M VAN OOSTERHOUT [1996], Analysis and Verification of on-line signatures, *Proceedings of the 5th European Conference for Police and Government handwriting experts*, The Hague.

DOLFING, J.G.A., J.J.G.M VAN OOSTERHOUT, AND E.H.L AARTS [1998], On-line Signature Verification with Hidden Markov Models, *14th International Conference on Pattern Recognition*, 2, 1309–1312.

DOUX, A., AND M. MILGRAM [1995], On-line signature verification using DTW and Neural Networks, *7th Biennial Conference of the International Graphonomics Society*, International Graphonomics Society 1, 46–47.

DUDA, R.O., AND P.E. HART [1973], *Pattern Classification and Scene Analysis*, Wiley, New York.

DUWAER, A.L. [1993], Data processing system with a touch screen and a digitizing tablet, both integrated in an input device, *United States Patent 5,231,381*.

EDELMAN, S., T. FLASH, AND S. ULLMAN [1990], Reading cursive handwriting by alignment of letter prototypes, *International Journal of Computer Vision* **5**, 303–331.

EVETT, L.J., C.J. WELLS, F.G. KEENAN, T. ROSE, AND R.J. WHITROW [1992], Using linguistic information to aid handwriting recognition, in: S. Impedovo and J.C. Simon (eds.), *From pixels to features III*, Frontiers in handwriting recognition, Elsevier Science Publisher B.V., 339–348.

FAIRHURST, M., AND P. BRITTAN [1994], An evaluation of parallel strategies for feature vector construction in automatic signature verificaton systems,

International Journal of Pattern Recognition and Artificial Intelligence **8**, 661–678.

FAIRHURST, M., K. COWLEY, AND E. SWEENEY [1994], *KAPPA automatic signature verification: Signature verification public trials and public survey on biometrics*, Technical report, British Technology group.

FORNEY, G.D. [1973], The Viterbi Algorithm, *Proceedings of the IEEE* **61**, 268–278.

FOUND, B., AND D. ROGERS [1995], Investigation of signature complexity for forensic purposes, *7th Biennial Conference of the International Graphonomics Society*, 52–53.

FUJISAKI, T., H.S.M. BEIGI, C.C. TAPPERT, M. UKELSON, AND C.G. WOLF [1992], Online recognition of unconstrained handprinting: a stroke-based system and its evaluation, in: S. Impedovo and J.C. Simon (eds.), *From pixels to features III*, Frontiers in handwriting recognition, Elsevier Science Publisher B.V., 297–312.

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**, 123–137.

FUKUNAGA, K. [1990], *Introduction to Statistical Pattern Recognition, Second Edition*, Academic Press, New York.

GALEN, G.P. VAN, AND A.W.A. VAN GEMMERT [1995], Dynamic features of mimicking another person's handwriting, *7th Biennial Conference of the International Graphonomics Society*, 158–159.

GLINSKI, S [1987], The graph search machine (GSM): a programmable processor for connected word speech recognition and other applications, *International Conference of Acoustics, Speech and Signal processing*, 519–522.

GUBERMAN, SH. A., AND V.V. ROZENTSVEIG [1976], Algorithm for the Recognition of Handwritten Text, *Avtomatika i Telemekhanika*, 122–129.

GUERFALI, W., AND R. PLAMONDON [1995], The Delta LogNormal Theory for the Generation and Modeling of Cursive Characters, *3rd International Conference on Document Analysis and Recognition*, 495–498.

GUYON, I., P. ALBRECHT, Y. LECUN, J. DENKER, AND W. HUBBARD [1991], Design of a neural network character recognizer for a touch terminal, *Pattern recognition* **24**, 105–119.

GUYON, I., J. MAKHOUL, R. SCHWARTZ, AND V.N. VAPNIK [1996], What size test set gives good error rate estimates, *5th International Workshop on Frontiers in Handwriting Recognition*, 313–316.

GUYON, I., AND F. PEREIRA [1995], Design of a linguistic postprocessor using variable memory length Markov models, *Conference handout ICDAR'95*,

–.

GUYON, I., L. SCHOMAKER, R. PLAMONDON, M. LIBERMAN, AND S. JANET [1994], UNIPEN project of on-line data exchange and recognizer benchmarks, *12th International Conference on Pattern Recognition*, 29–33.

GUYON, I., V.N. VAPNIK, B. BOSER, L. BOTTOU, AND S.A. SOLLA [1992], Structural Risk Minimization for Character Recognition, *Advances in Neural Information Processing Systems*, 471–479.

HAEB-UMBACH, R., AND H. NEY [1992], Linear Discriminant Analysis for Improved Large Vocabulary Continuous Speech Recognition, *International Conference of Acoustics, Speech and Signal processing*, 13–16.

HAEB-UMBACH, R., AND H. NEY [1994], Improvements in Beam Search for 10,000-Word Continuous-Speech Recognition, *IEEE Transactions on Speech and Audio processing* **2**, 353–356.

HASTIE, T., E. KISHON, R.M. CLARK, AND J. FAN [1991], A model for signature verification, *IEEE International Conference on Systems, Man and Cybernetics*, IEEE Conferences 1, 191–196.

HASTIE, T., E. KISHON, R.M. CLARK, AND J. FAN [1992], *A model for signature verification*, Technical report, AT&T Bell Laboratories.

HE, Y., M.Y. CHEN, AND A. KUNDU [1992], Handwritten word recognition using HMM with adaptive length Viterbi algorithm, *International Conference of Acoustics, Speech and Signal processing*, 153–156.

HERBST, N.M, AND C.N. LIU [1977], Automatic signature verification based on accelerometry, *IBM Journal Research & development* **21**, 245–253.

HO, T.K., J.J. HULL, AND S.N. SRIHARI [1992], A word shape analysis approach to lexicon based word recognition, *Pattern Recognition Letters* **13**, 821–826.

HOLLERBACH, J.M. [1981], An oscillation theory of handwriting, *Biological Cybernetics* **39**, 139–156.

HUANG, X.D., Y. ARIKI, AND M.A. JACK [1990], *Hidden markov models for speech recognition* (First ed.)., Edinburgh information technology series, Edinburgh university press, 22 George square, Edinburgh.

HULL, J.J., AND S.N. SRIHARI [1982], Experiments in text recognition with binary n-gram and Viterbi algorithm, *IEEE Transactions on pattern analysis and machine intelligence* **4**, 521–529.

JACOBS, T., AND A. SETLUR [1994], A field study of performance improvements in HMM-based speaker verification, *IEEE workshop on interactive technology for telecommunications applications*, IEEE 1, 121–124.

JELINEK, F. [1976], Continuous Speech Recognition by Statistical Methods, *Proceedings of the IEEE* **64**, 532–558.

KALTENMEIER, A., F. CLASS, P. REGEL-BRIETZMANN, T. CAESAR, J.M.

GLOGER, AND E. MANDLER [1993], Hidden Markov Models - A unified approach to recognition of spoken and written language, in: S.J.Poeppl (ed.), *Mustererkennung '93*, Informatik aktuell, Springer verlag, 191–198.

KASSEL, R.H. [1995], *A Comparison of Approaches to On-Line Handwritten Character Recognition*, Ph.D. thesis, MIT.

KIM, W.S., AND R.H. PARK [1996], Off-line recognition of Handwritten Korean and Alphanumeric Characters using Hidden Markov Models, *Pattern recognition* **29**, 845–858.

KOSMALA, A., J. ROTTLAND, AND G. RIGOLL [1997], An Investigation of the Use of Trigraphs for Large Vocabulary Cursive Handwriting Recognition, *International Conference of Acoustics, Speech and Signal processing*, 3373–3376.

KOVALEVSKY, V.A. [1980], *Image Pattern Recognition*, Springer-Verlag, New York.

KUNDU, A., AND P. BAHL [1988], Recognition of handwritten script: A hidden Markov model based approach, *International Conference of Acoustics, Speech and Signal processing*, IEEE, New York, NY USA, 928–931.

KUNDU, A., Y. HE, AND P. BAHL [1989], Recognition of handwritten word: first and second order hidden Markov model based approach, *Pattern recognition* **22**, 283–297.

LALOMIA, M.J. [1994], User-acceptance of handwritten recognition accuracy, *Companion proceedings of the CHI'94 conference on Human Factors in Computing Systems*, ACM, 107.

LAM, C., AND D. KAMINS [1989], Signature verification through spectral analysis, *Pattern recognition* **22**, 39–44.

LE CUN, Y. [1990], Handwritten ZIP Code Recognition with Multilayer Networks, *10th International Conference on Pattern Recognition*, IEEE, 35–40.

LE CUN, Y. [1993], On-line handwriting recognition with Neural Networks: spatial representation versus temporal representation, *6th International conference on handwriting and drawing*, IGS, 22–24.

LECLERC, F., AND R. PLAMONDON [1994], Automatic Signature Verification: The State of the Art - 1989-1993., *International Journal of Pattern Recognition and Artificial Intelligence* **8**, 643–660.

LEE, K.F. [1988], *Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System*, Ph.D. thesis, Carnegie-Mellon University.

LEE, K.F. [1989], *Automatic Speech recognition*, Kluwer Academic Publishers.

LEE, L., T. BERGER, AND E. AVICZER [1996], Reliable on-line human signature verification systems, *IEEE Transactions on pattern analysis and machine intelligence* **18**, 643–647.

LEEDHAM, C.G. [1990], Automatic recognition and transcription of Pitman's handwritten shorthand, in: R. Plamondon and C.G. Leedham (eds.), *Computer Processing of Handwriting*, 235–269.

LEVINSON, S.E., L.R. RABINER, AND M.M. SONDHI [1983], An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition, *The Bell System Technical Journal* **62**, 1035–1074.

LEVINSON, S.E., AND D.B. ROE [1990], A perspective on speech recognition, *IEEE Communications Magazine* **28**, 28–34.

LINDWURM, R., T. BREUER, AND K. KREUZER [1996], Multi Expert System for Handprinted Recognition, *5th International Workshop on Frontiers in Handwriting Recognition*, 125–129.

LIU, C.N. [1978], Reference design procedure for signature verification, *IBM Technical Disclosure Bulletin* **21**, 426–427.

LIU, C.L., Y.L LIU, AND R.W. DAI [1996], Multiresolution Statistical Feature and Structural Feature Extraction, *5th International Workshop on Frontiers in Handwriting Recognition*, 61–66.

MADHVANATH, S., AND S.N. SRIHARI [1996], Effective Reduction of Large Lexicons for Recognition of Offline Cursive script, *5th International Workshop on Frontiers in Handwriting Recognition*, 189–194.

MAHADEVAN, U., AND S.N. SRIHARI [1996], Hypothesis Generation for Word Separation in Handwritten Lines, *5th International Workshop on Frontiers in Handwriting Recognition*, 453–456.

MANKE, S., AND U. BODENHAUSEN [1994], A connectionist recognizer for on-line cursive handwriting recognition, *International Conference of Acoustics, Speech and Signal processing*, 633–636.

MANKE, S., M. FINKE, AND A. WAIBEL [1995], NPen++: A writer independent, large vocabulary on-line cursive recognition system, *International Conference on Document Analysis and Recognition*, 403–408.

MANKE, S., M. FINKE, AND A. WAIBEL [1996], A fast search technique for Large Vocabulary On-line Handwriting Recognition, *5th International Workshop on Frontiers in Handwriting Recognition*, 183–188.

MARTINET, J.F [1790], *Het vereenigd Nederland* (Second ed.)., Amsterdam.

MILLER, B. [1994], Vital signs of identity, *IEEE Spectrum* **31**, 22–30.

MITAL, D., C. HIN, AND W. LONG [1987], An on-line signature verification system, *IEEE International Conference on Systems, Man and Cybernetics*, IEEE 2, 837–841.

MOHANKRISHNAN, N., M. PAULIK, AND M. KHALIL [1993b], Issues pertaining to optimal performance of an on-line autoregressive model based signature verification system, *Proceedings of the Midwest Symposium on Circuits and*

Machines, IEEE 1, 677–681.

MOHANKRISHNAN, N., M. PAULIK, AND M. KHALIL [1993a], On-line signature verification using a nonstationary autoregressive model representation, *IEEE International Symposium on Circuits and Systems*, IEEE 3, 2303–2306.

MORI, S., C.Y. SUEN, AND K. YAMAMOTO [1992], Historical review of OCR research and development, *Proceedings of the IEEE* **80**, 1029–1058.

MORI, S., K. YAMAMOTO, AND M. YASUDA [1984], Research on machine recognition of handprinted characters, *IEEE Transactions on pattern analysis and machine intelligence* **6**, 386–405.

NAG, R., K.H. WONG, AND F. FALLSIDE [1986], Script recognition using Hidden Markov Models, *International Conference of Acoustics, Speech and Signal processing*, 2071–2074.

NAIK, J.M. [1990], Speaker verification: a tutorial, *IEEE Communications Magazine* **28**, 42–48.

NAIK, J.M. [1994], Speaker verification over the telephone network: Databases, algorithms and performance assessment, *ESCA workshop on automatic speaker recognition, identification and verification*, 31–38.

NAIK, J.M., L.P. NETSCH, AND G.R DODDINGTON [1989], Speaker Verification over long distance Telephone Lines, *International Conference of Acoustics, Speech and Signal processing* **1**, 524–527.

NATHAN, K.S., H.S.M. BEIGI, J. SUBRAHMONIA, G.J. CLARY, AND H. MARUYAMA [1995], Real-time on-line unconstrained handwriting recognition using statistical methods, *International Conference of Acoustics, Speech and Signal processing*, 2619–2622.

NELSON, W., W. TURIN, AND T. HASTIE [1994], Statistical methods for online signature verification, *International Journal of Pattern Recognition and Artificial Intelligence* **8**, 749–770.

NEN2296 [1958], *Handschrift voor het lager onderwijs: Schrijfletters en cijfers* (UDC 003.81: 372.51 ed.). Hoofdcommissie voor de normalisatie in Nederland (HCNN).

NEY, H. [1984], The use of a one-stage dynamic programming algorithm for connected word recognition, *IEEE Transactions on Acoustics, Speech and Signal processing* **32**, 263–271.

NEY, H., U. ESSEN, AND R. KNESER [1994], On structuring probabilistic dependences in stochastic language modelling, *Computer Speech and Language*, 1–38.

NEY, H., V. STEINBISS, R. HAEB-UMBACH, B.-H. TRAN, AND U. ESSEN [1994], An overview of the Philips research system for large vocabulary continuous speech recognition, *Int. Journal of Pattern Recognition and Ar-*

tificial Intelligence **8**, 33–70.

OH, S.C., J.Y. HA, AND J.H. KIM [1995], Context Dependent Search in interconnected Hidden Markov Model for unconstrained Handwriting Recognition, *Pattern recognition* **28**, 1693–1704.

OW, P.S., AND T.E. MORTON [1988], Filtered beam search in scheduling, *International Journal of Production Research* **26**, 35–62.

PARIZEAU, M., AND R. PLAMONDON [1990], A comparative analysis of regional correlation, dynamic time warping, and skeletal tree matching for signature verification, *IEEE Transactions on pattern analysis and machine intelligence* **12**, 710–717.

PAUL, D.B. [1991], Algorithms for an Optimal A* Search and Linearizing the Search in the Stack Decoder, *International Conference of Acoustics, Speech and Signal processing*, IEEE, 693–696.

PAUL, D.B., AND J.M. BAKER [1992], The Design for the Wall Street Journal-based CSR Corpus, *Speech and Natural Language Workshop*, DARPA, 357–362.

PAULIK, M., AND N. MOHANKRISHNAN [1993], A 1-d, sequence decomposition based, autoregressive hidden markov model for dynamic signature identification and verification, *Proceedings of the Midwest Symposium on Circuits and Machines*, 1, 138–141.

PICONE, J. [1990], Continuous speech recognition using hidden Markov models, *IEEE ASSP Magazine* **41**, 26–41.

PLAMONDON, R. [1993], Understanding stroke generation: a global neuromuscular approach, *International conference on handwriting and drawing*, 1–3.

PLAMONDON, R. [1994a], The design of an on-line signature verification system: From theory to practice, in: R. Plamondon (ed.), *Progress in automatic Signature verification*, 155–172.

PLAMONDON, R. (ed.) [1994b], *Progress in automatic Signature verification*, World Scientific, New York.

PLAMONDON, R., A. ALIMI, P. YERGEAU, AND F. LECLERC [1993], Modeling velocity profiles of rapid movements: a comparative study, *Biological Cybernetics* **69**, 119–128.

PLAMONDON, R., AND G. LORETTE [1989], Automatic signature verification and writer identification - the state of the art, *Pattern recognition* **22**, 107–131.

PLAMONDON, R., AND G. LORETTE [1990], Designing an automatic signature verifier, *Computer processing of handwriting*, World Scientific, 3–20.

PLAMONDON, R., AND F. MAARSE [1989], An evaluation of motor models of handwriting, *IEEE Transactions on Systems, Man and Cybernetics* **19**, 1060–1072.

PORT, O. [1996], Let your finger do the charging, *Business Week*, June 3, pp 43.

QUINNELL, R.A. [1995], Touchscreen Technology improves and extends its options, *EDN*, 52–63.

RABINER, L.R., AND B.H. JUANG [1993], *Fundamentals of speech recognition* (First ed.)., Prentice hall.

RATZLAFF, E.H., K.S. NATHAN, AND H. MARUYAMA [1996], Search issues in the IBM Large Vocabulary Unconstrained Handwriting Recognizer, *5th International Workshop on Frontiers in Handwriting Recognition*, 177–182.

RUBINE, D. [1991], *The automatic recognition of gestures*, Ph.D. thesis, School of Computer Science, Carnegie Mellon University.

SAKOE, H., AND S. CHIBA [1978], Dynamic programming algorithm optimization for spoken word recognition, *IEEE Transactions on Acoustics, Speech and Signal processing* **26**, 43–49.

SANKOFF, D., AND J.B. KRUSKAL [1983], *Time Warps, String Edits, And Macromolecules: The theory and practice of sequence comparison*, Addison-Wesley, Reading, Massachusetts.

SATO, Y., AND K. KOGURE [1982], Online signature verification based on shape, motion, and writing pressure, *International Conference on Pattern Recognition*, IEEE 2, 823–826.

SCHENKEL, M., I. GUYON, AND D. HENDERSON [1994], On-line Cursive Script Recognition using Time-delay Neural Networks and Hidden Markov Models, *International Conference of Acoustics, Speech and Signal processing*, 637–640.

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**, 215–223.

SCHMIDT, C., AND F. OLSCHEWSKI [1995], Signature verification using a self-organizing map - a connectionist approach based on dynamic feature of the signature, *7th Biennial Conference of the International Graphonomics Society*, 138–139.

SCHOMAKER, L. [1990], *Simulation and recognition of handwriting movements*, Ph.D. thesis, Nijmegen Institute for Cognition Research and Information Technology (NICI).

SCHOMAKER, L. [1993], Using stroke- or character based self-organizing maps in the recognition of on-line, connected cursive script, *Pattern recognition* **26**, 443–450.

SCHOMAKER, L. [1994], User-interface aspects in recognizing connected-cursive handwriting, *IEE Workshop*, 8/1–8/3.

SCHOMAKER, L., AND R. PLAMONDON [1990], The relation between Pen Force and Pen-Point Kinematics in handwriting, *Biological Cybernetics* **63**, 277–289.

SCHOMAKER, L., AND H.L. TEULINGS [1992], Stroke-versus character-based recognition of on-line, connected cursive script, *From Pixels to Features III:Frontiers in handwriting recognition*, 313–325.

SCHWARTZ, R., Y.L. CHOW, S. ROUCOS, R. KRASNER, AND J. MAKHOUL [1984], Improved Hidden Markov Modeling of Phonemes for continuous speech recognition, *International Conference of Acoustics, Speech and Signal processing*, IEEE, 35.6.1–35.6.4.

SEILER, R., M. SCHENKEL, AND F. EGGIMANN [1996], Cursive handwriting recognition: off-line versus on-line recognition, *5th International Workshop on Frontiers in Handwriting Recognition*, 23–28.

SENI, G., AND E. COHEN [1994], External word segmentation of off-line handwritten text lines, *Pattern recognition* **27**, 41–52.

SENI, G., R.K. SRIHARI, AND N. NASRABADI [1994], Large vocabulary recognition of On-line handwritten Cursive Words, *IEEE Transactions on pattern analysis and machine intelligence* **18**, 757–762.

SENIOR, A.W. [1994], *Off-line cursive Handwriting recognition using Recurrent neural networks*, Ph.D. thesis, University of Cambridge.

SENIOR, A.W., K.S. NATHAN, AND J. SUBRAHMONIA [1996], Duration modeling results for an on-line handwriting recognizer, *International Conference of Acoustics, Speech and Signal processing.*

SHANNON, G.E. [1951], Prediction and entropy of printed English, *Bell System Technical Journal* **30**, 50–64.

SICARD, E. [1992], An Efficient Method for the Recognition of Printed Music, *11th International Conference on Pattern Recognition*, 573–576.

SIMON, J.C. [1992], Off-line cursive word recognition, *Proceedings of the IEEE* **80**, 1150–1161.

STALLINGS, W.W. [1977], Chinese Character Recognition, in: K.S. Fu (ed.), *Syntactic Pattern Recognition, Applications*, Communication and Cybernetics, Springer Verlag, 95–121.

STAMPA, V., T. CAESAR, J.M. GLOGER, A. KALTENMEIER, AND E. MANDLER [1996], Recognizing Handwritten Numbers by Hidden Markov Models and Polynomial Classifiers, *5th International Workshop on Frontiers in Handwriting Recognition*, 41–46.

STARNER, T., J. MAKHOUL, R. SCHWARTZ, AND G. CHOU [1994], On-line cursive handwriting recognition using speech recognition methods, *International Conference of Acoustics, Speech and Signal processing*, 125–128.

STROUSTRUP, B. [1991], *The C++ programming language* (Second ed.)., Addison-Wesley, New York.

STROUSTRUP, B. [1994], *The Design and Evolution of C++*, Addison-Wesley, New York.

SUBRAHMONIA, J., K.S. NATHAN, AND M.P. PERRONE [1996], Writer dependent recognition of on-line unconstrained handwriting, *International Conference of Acoustics, Speech and Signal processing.*

SUEN, C.Y., M. BERTHOD, AND S. MORI [1980], Automatic recognition of handprinted characters - The state of the art, *Proceedings of the IEEE* **68**, 469–487.

TAGUCHI, H., K. KIRIYAMA, E. TANAKA, AND K. FUJII [1989], On-line recognition of handwritten signatures by feature extraction of pen-movements, *Systems and computers in Japan* **20**, 1–14.

TAPPERT, C.C. [1982], Cursive script recognition by elastic matching, *IBM J. Res. development* **26**, 765–771.

TAPPERT, C.C. [1991], Speed, Accuracy, and Flexibility Trade-Offs in On-line Character Recognition, *International Journal of Pattern recognition and Artificial intelligence* **5**, 79–95.

TAPPERT, C.C., A.S. FOX, J. KIM, S.E. LEVY, AND L.L. ZIMMERMAN [1986], Handwriting recognition on Transparent Tablet Over Flat Display, *SID Digest*, 308–312.

TAPPERT, C.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**, 787–808.

TAYLOR, I., AND M.M. TAYLOR [1983], *The psychology of reading*, Academic Press, New York.

TEULINGS, H.L., AND L. SCHOMAKER [1992], Invariant properties between stroke features in handwriting, *5th Biennial Conference of the International Graphonomics Society.*

THOMASSEN, A.J.W.M., AND G.P. VAN GALEN [1996], Temporal features of handwriting: Challenges for forensic analysis, *5th European conference of Government and Police Handwriting Experts.*

WARD, J., AND D. SCHULTZ [1993], Digitizer Renaissance, *Byte* **1**, 251–260.

WEISSMAN, H., M. SCHENKEL, I. GUYON, C. NOHL, AND D. HENDERSON [1994], Recognition-based segmentation of on-line run-on handprinted words: input vs output segmentation, *Pattern recognition* **27**, 405–420.

WILKINSON, T., D. PENDER, AND J. GOODMAN [1991], Use of synthetic discriminant functions for handwritten-signature verification, *Applied optics* **30**, 3345–3353.

WINKLER, H.J. [1996], HMM-based handwritten symbol recognition using on-line and off-line features, *International Conference of Acoustics, Speech and Signal processing*, 3438–3441.

WINKLER, H.J., AND M. LANG [1996], Symbol segmentation and recognition for understanding handwritten mathematical expressions, *5th International*

Workshop on Frontiers in Handwriting Recognition, 465–469.

WIRTZ, B. [1995], Stroke-based time warping for signature verification, *3rd International Conference on Document Analysis and Recognition*, 179–182.

WITTEN, I.H., AND T.C. BELL [1990], Source models for natural language text, *Int. Journal Man-Machine Studies* **32**, 545–579.

WOLMAN, A. [1992], Recognition of handwritten Music Notation, *International Computer Music Conference*, San Jose, 125–127.

WORTHINGTON, T., T. CHAINER, J. WILFORD, AND S. GUNDERSEN [1985], IBM dynamic signature verification, *Computer security*, 129–154.

WRIGHT, P.T. [1990], On-line recognition of handwriting, *GEC journal of research* **8**, 42–48.

YANG, L. [1995], *Processing and recognition of handwriting in multimedia environments*, Ph.D. thesis, Technische Universiteit Delft.

YANG, L., B. WIDJAJA, AND R. PRASAD [1995], Application of hidden Markov models for signature verification, *Pattern recognition* **28**, 161–170.

YASHIMURA, I., AND M. YOSHIMURA [1992], On-line signature verification incorporating the direction of pen movement- An experimental examination of the effectiveness, in: S. Impedovo and J.C. Simon (eds.), *From pixels to features III: frontiers in handwriting recognition*, Elsevier science publishers b.v., 353–361.

YOSHIMURA, I, AND M. YOSHIMURA [1994], Off-line verification of Japanese signatures after elimination of background patterns, in: R. Plamondon (ed.), *Progress in automatic Signature verification*, 53–68.

YOSHIMURA, M., Y. KATO, S. MATSUDA, AND I. YASHIMURA [1991], On-line signature verification incorporating the direction of pen movement, *IEICE Transactions* **74**, 2083–2092.

ZIMMERMANN, K., AND M. VARADY [1985], Handwriter identification from one-bit quantized pressure patterns, *Pattern recognition* **18**, 63–72.

Author Index

A

Aarts, E.H.L, 131, 135
Abraham, D., 5
Albrecht, P., 11, 64, 66, 70, 109, 110
Ali, F., 11
Alimi, A., 21, 22, 140
Ariki, Y., 32, 34, 38
Aviczer, E., 132, 133, 136, 138, 144, 171

B

Bahl, L.R., 12, 45
Bahl, P., 12, 82
Baker, J.M., 126
Baum, L.E., 44, 51
Beatson, R., 138
Beigi, H.S.M., 11–13, 36, 38, 59, 64, 65, 84, 110, 111, 113, 123
Bell, T.C., 10, 56, 57
Bellegarda, E.J., 12, 13, 38, 64, 68, 77, 78, 109, 110, 145
Bellegarda, J.R., 12, 13, 38, 64, 68, 77, 78, 109, 110, 145
Bellman, R., 11, 39
Bentz, J., 133, 143
Berger, T., 132, 133, 136, 138, 144, 171
Berthod, M., 10
Bodenhausen, U., 12, 59, 68
Boes, U., 11
Bose, C.B., 12
Boser, B., 11
Bottou, L., 11, 133, 143
Bouma, H., 23
Bouwhuis, D.G., 22, 23
Bozinovic, R.M., 11, 84
Breuer, T., 78
Brittan, P., 138
Bromley, J., 133, 143
Brown, P.F., 45
Bunke, M., 12, 36, 123

C

Caesar, T., 12, 73, 78, 84
Chainer, T., 20, 132, 136–138, 171, 172
Chang, L., 109
Chefalas, T.E., 11
Chen, F.R., 12
Chen, M.Y., 12, 38
Chiba, S., 134
Cho, W., 12, 78–80, 119, 123
Chou, G., 12, 54, 66, 68, 79, 80, 82, 120, 126
Chow, Y.L., 54, 80
Clark, R.M., 13, 134, 136
Clary, G.J., 12, 13, 36, 38, 59, 64, 65, 84, 111, 113, 123
Class, F., 12
Cohen, E., 123
Cortes, C., 123
Cowley, K., 6, 19, 135, 136
Crane, H., 23, 132–134, 136–138, 144, 172

D

Dai, R.W., 68

Denker, J., 11, 64, 66, 70, 109, 110
de Souza, P.V., 45
Dimauro, G., 138
Doddington, G.R, 13, 135
Doermann, D.S., 23, 170
Dolan, G., 5
Dolfing, J.G.A., 12, 13, 99, 123, 128, 131, 135, 138
Double, G., 5
Doux, A., 134
Duda, R.O., 50, 76
Duwaer, A.L., 3, 24

E

Edelman, S., 10
Eggimann, F., 12, 68, 73, 84
Essen, U., 36, 37, 45, 50, 54, 56, 80, 84, 86
Evett, L.J., 11

F

Fairhurst, M., 6, 19, 135, 136, 138
Fallside, F., 12
Fan, J., 13, 134, 136
Finke, M., 12, 87, 113, 123
Flash, T., 10
Fogaroli, G., 11
Forney, G.D., 42
Found, B., 133, 170
Fox, A.S., 10, 20
Fujii, K., 136, 138, 141
Fujisaki, T., 11, 110
Fukunaga, K., 76, 77, 115

G

Glinski, S, 12
Gloger, J.M., 12, 73, 78, 84
Goodman, J., 133, 143
Guberman, Sh.A., 11, 23
Guerfali, W., 22
Gundersen, S., 20, 132, 136–138, 171, 172
Guyon, I., 11, 12, 45, 56, 59, 64–66, 68, 70, 72, 82, 84, 89, 91, 92, 109–111, 113, 120, 122, 123, 133, 143

H

Ha, J.Y., 12
Haeb-Umbach, R., 12, 13, 36, 37, 45, 50, 54, 77, 78, 80, 84, 86, 87, 99, 114, 123, 138
Hart, P.E., 50, 76
Hastie, T., 13, 134, 136
He, Y., 12, 82
Henderson, D., 11, 12, 59, 64, 65, 68, 72, 84, 111, 113, 120, 122, 123
Herbst, N.M, 22, 139
Hin, C., 138
Ho, T.K., 73
Hollerbach, J.M., 21, 22
Huang, X.D., 32, 34, 38
Hubbard, W., 11, 64, 66, 70, 109, 110
Hull, J.J., 73, 82

I

Impedovo, S., 138

J

Jack, M.A., 32, 34, 38
Jacobs, T., 13
Janet, S., 89, 91, 92
Jelinek, F., 12
Juang, B.H., 10, 33, 35, 37–40, 44, 45, 50, 56, 58, 68, 80, 84

K

Kaltenmeier, A., 12, 78
Kamins, D., 138

Kassel, R.H., 78, 82, 88, 108–110, 124, 125
Kato, Y., 18, 132, 135–137, 145, 150, 172
Keenan, F.G., 11
Keil, H., 11
Khalil, M., 135, 171
Kim, J., 10, 11, 20
Kim, J.H., 12, 78–80, 119, 123
Kim, W.S., 12
Kiriyama, K., 136, 138, 141
Kishon, E., 13, 134, 136
Kneser, R., 56
Kogure, K., 18, 132, 134–136, 138, 141, 144
Kosmala, A., 13, 68, 80, 81, 120, 122
Kovalevsky, V.A., 11
Krasner, R., 54, 80
Kreuzer, K., 78
Kruskal, J.B., 88
Kundu, A., 12, 38, 82
Kuo, S., 12

L

LaLomia, M.J., 109
Lam, C., 138
Lang, M., 11
Leclerc, F., 10, 21, 22, 134, 135, 140
LeCun, Y., 11, 64, 66, 70, 109, 110, 133, 143
Lee, K.F., 10, 15, 33, 35–37, 40, 44, 54, 68, 80, 81, 88, 120, 122
Lee, L., 132, 133, 136, 138, 144, 171
Lee, S.W., 12, 78–80, 119, 123
Leedham, C.G., 11
Levinson, S.E., 44, 47, 49, 51, 52, 64
Levy, S.E., 10, 20
Liberman, M., 89, 91, 92
Lindwurm, R., 78
Liu, C.L., 68
Liu, C.N., 22, 137, 139, 145
Liu, Y.L, 68
Long, W., 138
Lorette, G., 10, 18, 132, 133, 135, 138, 141, 158, 160

M

Maarse, F., 20–22
Madhvanath, S., 73
Mahadevan, U., 124
Makhoul, J., 12, 45, 54, 66, 68, 79, 80, 82, 89, 120, 126
Mandler, E., 12, 73, 78, 84
Manke, S., 12, 59, 68, 87, 88, 113, 123
Martinet, J.F, 5
Maruyama, H., 12, 13, 36, 38, 59, 65, 84, 88, 111, 113, 123
Maslin, G., 11
Matsuda, S., 18, 132, 135–137, 145, 150, 172
Mercer, R.L., 12, 45
Milgram, M., 134
Miller, B., 5, 6
Mital, D., 138
Mohankrishnan, N., 13, 135, 171
Moore, C., 133, 143
Mori, S., 10
Morton, T.E., 58

N

Nag, R., 12
Nahamoo, D., 12, 13, 38, 64, 68, 77, 78, 109, 110, 145
Naik, J.M., 13, 135
Nasrabadi, N., 11, 66, 70, 73
Nathan, K.S., 12, 13, 36, 38, 59, 64, 65, 68, 77, 78, 84, 88, 109–111, 113, 123, 145
Nelson, W., 134

Netsch, L.P., 13, 135
Ney, H., 36, 37, 45, 50, 54, 56, 58, 60, 77, 78, 80, 84, 86–88, 114
Nohl, C., 11, 59, 65, 72, 111

O

Oh, S.C., 12
Olschewski, F., 134, 141
Ostrem, J., 23, 132–134, 136–138, 144, 172
Ow, P.S., 58

P

Parizeau, M., 134
Park, R.H., 12
Paul, D.B., 58, 126
Paulik, M., 13, 135, 171
Pavlidis, T., 11
Pender, D., 133, 143
Pereira, F., 11, 56, 82
Perrone, M.P., 13
Picone, J., 45
Pirlo, G., 138
Plamondon, R., 10, 18, 20–22, 64, 89, 91, 92, 132–136, 138, 140, 141, 143, 158, 160
Port, O., 2
Prasad, R., 13, 132, 135, 136, 138, 171

Q

Quinnell, R.A., 10, 24

R

Rabiner, L.R., 10, 33, 35, 37–40, 44, 45, 47, 49–52, 56, 58, 68, 80, 84
Ratzlaff, E.H., 12, 88
Regel-Brietzmann, P., 12
Rigoll, G., 13, 68, 80, 81, 120, 122
Roe, D.B., 64
Rogers, D., 133, 170
Rose, T., 11
Roth, M, 12, 36, 123
Rottland, J., 13, 68, 80, 81, 120, 122
Roucos, S., 54, 80
Rozentsveig, V.V., 11, 23
Rubine, D., 11

S

Sackinger, E., 133, 143
Sakoe, H., 134
Sankoff, D., 88
Sato, Y., 18, 132, 134–136, 138, 141, 144
Schenkel, M., 11, 12, 59, 64, 65, 68, 72, 73, 84, 111, 113, 120, 122, 123
Schmidt, C., 134, 141
Schomaker, L., 11, 20–22, 27, 64, 66, 70, 89, 91, 92, 122, 123, 140, 141
Schukat-Talamazinni, G., 12, 36, 123
Schultz, D., 10
Schwartz, R., 12, 45, 54, 66, 68, 79, 80, 82, 89, 120, 126
Seiler, R., 12, 68, 73, 84
Seni, G., 11, 66, 70, 73, 123
Senior, A.W., 10, 13, 23, 79, 84
Setlur, A., 13
Shah, R., 133, 143
Shannon, G.E., 10, 56
Sicard, E., 11
Simon, J.C., 11
Solla, S.A., 11
Sondhi, M.M., 44, 47, 49, 51, 52
Srihari, R.K., 11, 66, 70, 73
Srihari, S.N., 11, 38, 73, 82, 84, 124
Stallings, W.W., 11
Stampa, V., 78

Starner, T., 12, 54, 66, 68, 79, 80, 82, 120, 126
Steinbiss, V., 36, 37, 45, 50, 54, 80, 84, 86
Stevens, J., 5
Stroustrup, B., 89
Subrahmonia, J., 12, 13, 36, 38, 59, 64, 65, 84, 111, 113, 123
Suen, C.Y., 9, 10, 20
Sweeney, E., 6, 19, 135, 136

T

Taguchi, H., 136, 138, 141
Tanaka, E., 136, 138, 141
Tappert, C.C., 9–11, 20, 109, 110, 123
Taylor, I., 22, 23
Taylor, M.M., 22, 23
Teulings, H.L., 20, 22, 123
Thomassen, A.J.W.M., 20, 21
Tran, B.-H., 36, 37, 45, 50, 54, 80, 84, 86
Turin, W., 134

U

Ukelson, M., 11, 110
Ullman, S., 10

V

van Galen, G.P., 20, 21, 140
van Gemmert, A.W.A., 140
van Oosterhout, J.J.G.M, 131, 135
Vapnik, V.N., 11, 45, 89
Varady, M., 138, 141

W

Waibel, A., 12, 87, 113, 123
Wakahara, T., 9, 20
Ward, J., 10
Weissman, H., 11, 59, 65, 72, 111
Wells, C.J., 11
Whitrow, R.J., 11
Widjaja, B., 13, 132, 135, 136, 138, 171
Wilford, J., 20, 132, 136–138, 171, 172
Wilkinson, T., 133, 143
Winkler, H.J., 11, 12
Wirtz, B., 134, 141
Withgott, M., 12
Witten, I.H., 10, 56, 57
Wolf, C.G., 11, 110
Wolman, A., 11
Wong, K.H., 12
Worthington, T., 20, 132, 136–138, 171, 172
Wright, P.T., 11

Y

Yamamoto, K., 10
Yang, L., 13, 36, 70, 109, 132, 135, 136, 138, 146, 147, 160, 171
Yashimura, I., 18, 132, 134–138, 145, 150, 172
Yasuda, M., 10
Yergeau, P., 21, 22, 140
Yoshimura, I, 158
Yoshimura, M., 18, 132, 134–138, 145, 150, 158, 172

Z

Zhou, J., 12
Zimmerman, L.L., 10, 20
Zimmermann, K., 138, 141

Subject Index

A

Allograph, 4, 21, 79, 110
Alphabet, 1, 61, 63, 79

B

Bayes
  decision rule, 7, 32
  theorem, 32
Beam search, 58–60, 86
Biometrics, 5, 6

C

Confusion matrix, 104
Context, 4, 8–11, 53–57, 69, 80
Context dependent units, 80, 120
  trigraph, 80, 120–122
Covariance, 76, 78

D

Densities
  continuous, 36
  discrete, 37
  mixture, 50
Digitizer, 3, 10, 23–28
  sample speed, 3, 20, 64
  specification, 24
Dynamic programming, 11, 39, 134
Dynamic Time Warping, 134

E

Entropy, 56
Evaluation, 32

F

Feature, 139
  aggregate, 142
  angular, 70
  contextual, 142
  contour, 73
  hat, 72
  overlap, 72, 159
  pen-up, 139
  positional, 70
  pressure, 3, 27, 130, 141
  selection
    automatic, 144
  size, 71
  tilt, 3, 27, 130, 141
  vector, 65
  velocity, 26, 140
Feature vector, 32, 65, 131
  aggregate, 67
  contextual features, 69–75, 96, 103–104, 112–113, 115–117, 142–143, 165–166
  delta features, 68
  spliced, 68
Forensic science, 133
Frame, 65
  size-independent, 65, 95
Frames, 83

H

Handwriting, 4, 16–23
  baseline, 84
  diacritical, 84
  generation, 19
  information

dynamic, 3
static, 3
ligature, 17
model, 9
motor, 20–22
nationality, 17
off-line, 3, 11
on-line, 3, 11
overspecification, 18
reading, 22
recognition, 7, 10–13
unlimited vocabulary, 124
shape, 17
size, 64
slant, 84
slope, 84
speed, 20, 64
style, 4, 17
unit, 9, 19, 20, 22
variability, 4, 16, 64
verification, 7, 62
Hidden Markov model, 8, 31–62, 78
algorithm
Backward, 41
Forward, 40
Viterbi, 42
backspace state, 79, 118–120
beamsearch, 86
contextual, 53–54, 80–81, 120–122
covariance, 78
definition, 34–35
left-to-right, 36, 78, 79, 131, 144
ligature model, 79–80, 118–120
pause state, 79–80, 118–120
recognition, 12
implementation, 86, 89
signature, 144
size, 97, 145
threshold, 145, 160
automatic, 163
personal, 163
training, 44–53
convergence, 51
Forward-Backward, 47
implementation, 84
Viterbi, 49
transitions, 36
trigraph, 80
Human
acceptance, 109
performance, 10, 109
reading, 22

L

Language model, 11, 54–57, 81
Lda, 76–78
Ligature, 5, 17, 79
Linear discriminant analysis, 76–78, 143

M

Markov
chain, 33
model, 33–34
Motor model, 20–22

P

Pen
inclination, 155
pressure, 27, 155
robustness, 141
tilt, 27, 150, 155
Perplexity, 56–57, 125, 126
Preprocessor, 83
Principal component analysis, 76

R

Recognition, 61
formulas, 11
handwriting, 10

music, 11
sentences, 124
shorthand, 11
Representation, 9, 28–30, 64
blocking, 29–30, 159
feature extraction, 29
frame, 30, 65
grouping, 29–30, 131, 159
lda, 76–78
segment, 65

S

Scribble, 30
definition, 20
Search, 9
Segment, 65, 83
Segmentation, 29
signature, 159
Signature, 5, 18–19
complexity, 170
database, 135, 150
duration, 135, 144, 166
forgery
amateur, 132
home-improved, 151
over-the-shoulder, 151
professional, 132, 151, 169, 170
skilled, 132
type, 131, 135
functional approach, 138
grouping, 138
legibility, 18
length, 144
model
size, 145
structure, 144
parametric approach, 29, 138
reference, 144
reproduction, 17
score, 147, 162
segmentation, 159
size, 145
threshold, 137, 144, 145, 160
automatic, 163
variability, 18
verification, 13, 62, 131
features, 139
LDA, 143
writing time, 135
Speech, 10
recognition, 5, 8
verification, 13
Stroke, 22, 30
definition, 20
delayed, 5

T

Tied
covariance, 97, 145
Time alignment, 29, 49
Training, 32
Trigraph, 54, 80
Triphone, 54, 80

U

Unipen, 91

V

Viterbi, 41, 134
Vocabulary
unlimited, 82
Voice
verification, 13

W

WER, 54, 92, 112, 123, 126, 127
Word accuracy, 88
Word error rate, 54, 92, 112, 123, 126, 127

Samenvatting

In dit proefschrift behandelen we het probleem van on-line handschriftherkenning en verificatie. In tegenstelling tot een off-line benadering waar alleen statische informatie beschikbaar is, zoals een plaatje, gebruikt een on-line aanpak ook dynamische handschrift informatie zoals tijd, snelheid en druk. Onze aanpak is gebaseerd op het hidden Markov model (HMM), een statistisch model dat ook in de continue spraakherkenning wordt gebruikt.

Het doel van de handschriftherkenning is een schrijver-onafhankelijke herkenning van letters, woorden en zinnen. Verschillende representaties en modellen worden onderzocht waarbij geen extra eisen aan de schrijfstijl of layout worden opgelegd. Daarbij is de representatie van speciaal belang om schaalbaarheid, performance en compactheid te onderzoeken. Het concept van ‚contextuele features' wordt geïntroduceerd en toegepast om naast korte- ook middellange-termijn trends in handschriften te modelleren. Deze kenmerken kunnen het aantal herkenningsfouten halveren. Vergeleken met de ‚delta features' uit de spraakherkenning, gebruiken de ‚contextuele features' minder extra kenmerken. Schaalbaarheid wordt onderzocht door, in tegenstelling tot de gebruikelijke methode die de schrijfgrootte corrigeert na het schrijven van een complete letter of woord, een grootte-onafhankelijke representatie te onderzoeken die gelijktijdig schrijven, schalen en herkennen mogelijk maakt. Een uitgebreide numerieke studie bevat o.a. een woord-herkennings experiment met een vocabulair van 200 woorden (99% correcte herkenningen) en 20.000 woorden (88.8% correcte woorden) wat vergelijkbaar is met andere state-of-the-art resultaten.

De on-line handtekening verificatie combineert een nieuwe digitalizerings technologie (PAID = Philips Advanced Interactive Display) met het hidden Markov model om de toegevoegde waarde van dynamische informatie te onderzoeken. Verder worden verschillende representaties, modellen en drempelwaardes onderzocht. De kwetsbaarheid voor vervalste handtekeningen wordt onderzocht d.m.v. verschillende soorten vervalsingen van zowel amateurs als professionals. Daarvoor wordt een database met bijna 5000 handtekeningen gebruikt. Het punt van gelijke verhouding van het percentage geaccepteerde vervalsingen en verworpen originelen ligt tussen de 1% en 1.9%.

Curriculum Vitae

Hans Dolfing was born on August 20, 1966, in Amersfoort, the Netherlands. From 1985 to 1990 he studied computer science at the University of Twente in Enschede, the Netherlands. He graduated in December 1990, on the object-oriented design of an object allocation strategy in the distributed Sina system. His Master's thesis was written under the supervision of M. Aksit.

In September 1991, he joined the Philips Research Laboratories in Eindhoven. As a member of the PAID and ROSE project, he has been working on aspects of on-line handwriting recognition and verification with the objective of an user-friendly user-interface. In this work, he combines his interests in pattern recognition, object-oriented software engineering, high-performance computing, compilers and algorithms. In August 1997, he joined the Man-Machine Interfaces group at Philips Research Laboratories in Aachen, Germany, to work on speech recognition.

Stellingen

behorende bij het proefschrift

Handwriting Recognition and Verification

A Hidden Markov Approach

van

J.G.A. Dolfing

I

Feature vectoren in handschriftherkenning beschrijven vaak te veel ‚locale' informatie en te weinig ‚globale' informatie. (Dit Proefschrift, Hoofdstuk 5)

II

In de literatuur over handschriftherkenning is het gebruik van trainingsvoorbeelden ter verbetering van de herkennings nauwkeurigheid overschat, terwijl het gebruik van context ondergewaardeerd is.

III

In de literatuur over handtekening verificatie is het gebruik van druk overschat terwijl het gebruik van tilt ondergewaardeerd is. (Dit Proefschrift, pagina 156,165)

IV

Handschriftherkenning is gemakkelijker voor mensen dan voor computers. Voor verificatie is dit precies andersom.

V

Het toenemend gebruik van computers bevordert de schrijfvaardigheid niet. Daardoor zal handschriftherkenning steeds moeilijker worden.

VI

Het belang van de Open Software, zoals Linux en gcc, op software research en ontwikkeling wordt door bedrijven onderschat. Het belang van Microsoft op software research en ontwikkeling wordt overschat.

VII

Software engineers kunnen meer leren van ‚Programming Pearls' dan van ‚programming Perl'. (BENTLEY, J.L. [1986],*Programming Pearls*, Addison Wesley)

VIII

Er is vaak een omgekeerd-evenredig verband tussen software engineering en managment kennis. Deze mismatch kost het bedrijfsleven meer dan het Y2K probleem.

IX

In het kader van de oplopende hoeveelheid en complexiteit van landbouwquota is het aan te bevelen de hoeveelheid landbouwgebieden te verminderen en dit te combineren met een uitbreiding van Schiphol.

X

Spraakherkenners, handschriftherkenners en katten zijn natuurlijke vijanden van de muis.