Off-line Cursive Handwriting Recognition Using Synthetic Training Data

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Tamás Varga

aus Ungarn

Leiter der Arbeit: Prof. Dr. Horst Bunke
Institut für Informatik
und angewandte Mathematik,
Universität Bern

Off-line Cursive Handwriting Recognition Using Synthetic Training Data

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Tamás Varga

aus Ungarn

Leiter der Arbeit: Prof. Dr. Horst Bunke
Institut für Informatik
und angewandte Mathematik,
Universität Bern

Von der Philosophisch-naturwissenschaftlichen Fakultät angenommen.

Bern, den 31.1.2006

Der Dekan:

Prof. Dr. Paul Messerli

Abstract

The objective of this thesis is to investigate the generation and use of synthetic training data for off-line cursive handwriting recognition. It has been shown in many works before that the size and quality of the training data has a great impact on the performance of handwriting recognition systems. A general observation is that the more texts are used for training, the better recognition performance can be achieved.

In this work it is examined whether this observation holds if the training set is augmented by synthetically generated texts. The motivation is that augmenting the training set by computer generated text samples is much faster and cheaper than collecting additional human written samples.

For this purpose, two novel methods are presented for the generation of synthetic text lines. The first one is based on the geometrical perturbation of existing human written text line images. In the second method, handwritten text lines are synthesized from ASCII transcriptions, using templates of characters and the Delta LogNormal model of handwriting generation. To evaluate these two methods for synthetic training set expansion, the task of off-line cursive English handwritten text line recognition is considered.

In the last part of the thesis, a novel approach for text line segmentation into individual words is presented, in order to perform segmentation-based text line recognition, which gives new insights into the effects of synthetically expanded training sets.

Several configurations of the recognizer and the synthetic handwriting generation process are examined in the thesis. Based on the experimental results, it is concluded that the use of synthetic training data can lead to improved recognition performance of handwriting recognition systems.

Acknowledgments

The work presented in this thesis was supported by the Swiss National Science Foundation program "Interactive Multimodal Information Management (IM)2" in the Individual Project "Scene Analysis", as part of NCCR.

There are several people who contributed to my thesis, and I am very grateful to them for their help.

First of all, I thank Prof. Dr. Horst Bunke for supervising my work during the course of my PhD, and for his valuable advices from which I could benefit a lot.

I also thank Prof. Dr. János Csirik, under whose supervision I got my first insight into the field of handwriting recognition, and Prof. Dr. Hanspeter Bieri, for undertaking the role of the second examinator.

For the experiments, I used the handwriting recognition system developed by Dr. Urs-Viktor Marti. I am especially grateful to Dr. Simon Günter for his immense help during my research, as well as to Roman Bertolami, Marcus Liwicki and Andreas Schlapbach for proof-reading this thesis and suggesting many improvements on the content. Other colleagues who contributed to my thesis through their work, advices or discussions are Oliver Aeberhard, Dr. Stefan Fischer, Muriel Helmers, Dr. Christophe Irniger, Daniel Kilchhofer, Florence Lüthy, Michel Neuhaus, Daniel Schweri, Dr. Alessandro Vinciarelli, and Dr. Matthias Zimmermann — thank you very much.

Special thanks go to Susanne Thüler, whose invaluable help in the maze of administration made my life much easier here in Bern.

Finally, I would like to express my deep respect towards my parents who supported me throughout my studies.

Contents

Chapter 1

Introduction

It has always been an attractive challenge to build machines that are able to perform human activities, both physical and intellectual ones, especially after the advent of modern computers. As the years go by, computers become more powerful and the range of tasks they can be used for increases rapidly [36]. Considering the power of today's computers, it might be surprising that their ability at one of the most natural, typical and important human activity, namely reading, is still far away from humans' performance [15].

1.1 Handwriting Recognition

The ultimate aim of handwriting recognition research is to make computers able to read human written texts, with a performance comparable to or even better than that of humans. Reading means that the computer is given a piece of handwriting (e.g. a character, word, text line, or a whole page of text), and it provides the electronic transcription of that (e.g. in ASCII format). This problem has proved to be extremely difficult, and has been an active research topic for more than forty years [60, 62, 75, 95, 108].

Traditionally the field of handwriting recognition is divided into off-line and on-line recognition [87]. In off-line recognition, only the image of the handwriting is available for the computer, while in the on-line case temporal information such as pentip coordinates as a function of time is also available. Typical data acquisition devices for off-line and on-line recognition are scanners and digitizing tablets, respectively. Due to the lack of temporal information, off-line handwriting recognition is considered more difficult than on-line. Furthermore, it is also clear that the off-line case is the one that corresponds to the conventional reading task performed by humans.

The present thesis addresses the problem of off-line handwriting recognition. To avoid ambiguity, it is stated here that the following parts of the thesis are meant to deal with

off-line texts as well as off-line recognition, although some of the statements may be valid for the on-line case, too.

1.1.1 Off-line Recognition

There are several domains of off-line handwriting recognition, depending on the type of image to recognize. Among others one can consider the task of isolated character and digit recognition, word and numeric string recognition, text line recognition, or the recognition of whole handwritten pages.

Isolated character and digit recognition is usually considered as a pure pattern classification problem, where the true class of an input pattern has to be determined, given a finite number of candidate classes. That's why it is often used to evaluate the performance of novel pattern classification and feature extraction techniques [61]. The classification is based on features extracted from the input image. Numerous kinds of features have been proposed in the literature, such as zoning, projection histograms, various moments, contour profiles, contour-based descriptors, topological features like loops and endpoints, or the entire pixel matrix [110]. The extracted features are then used by a classifier to assign a class label to the input image. During the past decades, researchers have tried a huge repository of generic classifiers developed in the field of pattern recognition: template matching, k-nearest-neighbor classifier, Bayes classifier, structural matching techniques, neural networks, and many others. Extensive surveys are given in [4] and [75].

For word recognition, a straightforward idea would be to first segment the word image into isolated characters, and then perform character recognition, using techniques developed for that problem. However, there is an inherent difficulty with this approach, often referred to as Sayre's paradox [95]: "... the individual letters must be recognized as such before it can be determined where one inscription leaves off and another begins". Nevertheless, many segmentation techniques have been devised, but to reduce the risks of incorrect segmentation, segmentation and recognition are often coupled, e.g. by applying dynamic programming to find the correct character boundaries out of multiple candidates [63, 104]. Alternatively, there exist so-called holistic approaches that try to recognize a word as a whole, indivisible entity, without attempting any segmentation [64]. The recognition is usually based on features extracted from the word shape. The possible features include length, ascenders and descenders, holes and loops, dots, t-bars, a.s.o. Several psychological studies show that humans utilize such global features in reading. The application area of holistic methods is limited to the case when the number of possible word candidates, i.e. the size of the lexicon, is small, usually less than a hundred. One reason is that holistic features often prove to be too raw to differentiate among words of similar global shape.

In the recent years, Hidden Markov Model (HMM) based approaches have become dominant in word recognition [56, 115]. The word image is viewed as a sequence of observed

output values of an underlying stochastic process modeled by an HMM of that word. The word class whose HMM is the most likely to generate the input image is chosen as transcription. One reason of the popularity of this scheme is the well-established theoretical framework of HMMs, providing standard algorithms for training and recognition [88, 89]. The other reason is that they combine the advantages of the two previously mentioned approaches. That is, the difficult task of explicit segmentation is avoided, it is a byproduct of the recognition process (implicit segmentation). Moreover, HMMs can cope with large lexicons, because they are able to absorb huge variations of handwriting, including noise. Surveys on word recognition can be found in [102, 106, 115].

A relatively new field of interest is the recognition of general, unconstrained cursive handwritten text consisting of a sequence of words. It can be, for example, a text line or a whole page of text. Since usually the underlying language is the only constraint on the words that can occur in the text, typically a large lexicon is considered. Segmentation-based approaches first segment the text into individual words, and then try to recognize the extracted words [48, 71, 97]. To avoid the erroneous explicit segmentation of text lines into individual words, HMM-based approaches are also used to recognize whole lines of text [72, 116, 121]. These works make use of statistical language models [92], which can be easily integrated into the HMM framework.

For recent reviews on off-line handwriting recognition research the reader is referred to [4, 15, 54, 87].

1.1.2 Training of Recognizers

In order to be able to transcribe an input text image, the recognizer needs to store some knowledge about handwriting. Obviously, this has to include knowledge about the handwriting ink pattern, i.e. about the possible shapes of the different handwriting elements such as characters or words. Furthermore, other kind of knowledge concerning the task performed by the writer can also be stored. For example, if the task is to address a mail envelope, it is useful to know that the city name must match with the ZIP code.

In the most general sense, training is the process of supplying the recognizer with such knowledge about handwriting. However, since originally handwriting recognition has been viewed as basically being a pattern recognition problem, the term training usually relates to the acquisition of knowledge about the ink pattern, while other types of knowledge are termed as a-priori knowledge. Hereinafter, this terminology will be adopted.

The concrete details of the training method are dependent on the type of recognizer under consideration. Typically, properties of the handwriting ink pattern are extracted from a set of human written text samples called the training set. As a prevalent example of a-priori knowledge, for the task of general, unconstrained handwriting recognition, huge corpora containing different kinds of texts in electronic format, such as newspaper

articles, novels, biographies, etc., are used to obtain statistical linguistic knowledge of the underlying language.

1.1.3 Novel Trends of Development

Despite the existence of the numerous elaborated and mature recognition techniques outlined in Subsection 1.1.1, machines' reading performance is still considerably lower than that of humans. This inspired researchers to focus not only on the development of novel recognition algorithms, but also on the improvement of other aspects of handwriting recognition systems.

First, large and publicly available datasets of human written texts were collected [25, 34, 73]. Such databases enable better training of the recognizers, and also the comparison of their performances. Alternatively, to overcome the difficulties and inherent limitations of collecting a large number of human written samples, the possibility of generating synthetic handwriting is also a topic of investigation. Since this is the main focus of the thesis, the related work is presented in greater detail in Section 1.2.

Another approach which has become a very active and popular research area is the combination of classifiers [53, 91]. The basic idea is to use several different classifiers (experts) to classify an input pattern. The advantage of the approach is that errors made by an individual classifier can be corrected by the others, for example if we decide for the pattern class which is suggested by the majority of the recognizers. For further details see [55].

Nowadays it seems that human reading performance at general unconstrained texts cannot be achieved by using merely the information extracted from the ink pattern. There is a demand for shifting from the pattern recognition framework to a paradigm that emphasizes the utilization of much more a-priori knowledge [62]. According to the new framework described in [62], ambiguities occurring in difficult handwriting would be resolved mainly by applying the available linguistic knowledge, while using only general knowledge about the handwriting ink, namely its invariants common to a large variety of handwriting styles. This way the operation of the system would be transparent, so its errors could be analyzed and corrected more efficiently. Furthermore, the knowledge of such systems about handwriting ink is focused on the pertinent discriminative information, therefore it contains much less noise and useless information than that of contemporary recognizers trained on large databases of human written samples.

1.1.4 Applications

Handwriting recognition research is also greatly motivated by the possible commercial applications. So far, successful applications include areas where task specific knowledge

and constraints exist, comprising restrictions on the underlying lexicon [114]. Such applications are postal address reading [105], bank check processing [42], and form processing [28, 119].

Future applications aim at areas where no or only a few task specific constraints exist, with large lexicons involved. For example, the recognition of unconstrained texts like personal notes and communications, letters, faxes, or the automatic transcription of historical documents for building and indexing digital libraries [10].

However, when formulating our expectations towards contemporary and future handwriting recognition systems, we must take into consideration that humans also make mistakes at reading [14, 78, 109].

1.2 Synthetically Generated Text

The concept of synthetic text relates to both machine printed and handwritten documents. Synthesizing text means that real-world processes that affect the final outlook of a text are simulated by a computer program. For example, in the case of machine printed documents the printing and scanning defects, while in the case of handwriting the different writing instruments or the whole writing process can be modeled and simulated by the computer.

Synthetic texts can be generated in numerous different ways, and they have widespread uses in the field of document analysis and recognition. In the following, a brief overview is given. The approaches for both machine printed and handwritten synthetic text generation are presented, since they often have similar aims, and thus the findings and developments of one field can also affect and stimulate the other one and vice versa.

1.2.1 Improving and Evaluating Recognition Systems

The two main difficulties that contemporary text recognizers have to face are the degraded quality of document images as well as the great variation of the possible text styles [46, 76, 90]. The quality of document images usually degrades to various extent during printing, scanning, photocopying, and faxing. Style variation means that either different fonts might be used (machine printed text), or many individual writing styles can occur (handwritten text).

One way to alleviate the above mentioned problems is to train the recognizers using sets of text samples that are more representative to the specific recognition task under consideration. This idea is supported by two facts. First of all, every recognizer needs to be trained, because it has to be told how the different characters and/or words may look like. Furthermore, in the past decade researchers in the field of image pattern recognition

realized that the further improvement of recognition performance depends as much on the size and quality of the training data as on the underlying features and classification algorithms used [7]. As a rule of thumb says, the classifier that is trained on the most data wins.

A straightforward way to improve the training set is to collect more real-world text samples [25, 34, 73]. The effectiveness of this approach has been experimentally justified by numerous works in the literature, yielding higher recognition performance for increased training set sizes [16, 32, 93, 113]. Unfortunately, collecting real-world samples is a rather expensive and time consuming procedure, and truthing the collected data is error-prone [107, 117]. A possible solution to these drawbacks is to create text image databases automatically by generating synthetic data, which is cheap, fast, and far less error-prone. Furthermore, it enables the generation of much larger databases than those acquired by the conventional method. The main weakness of the synthetic approach is that the generated data may not be as representative as real-world data.

In machine printed OCR (Optical Character Recognition), especially when the possible fonts are a-priori known, the concept of representativeness of the training set can be approached from the side of document degradations. In [5, 22, 44], defects caused by the use of printing and imaging devices are explicitly modeled and applied to ideal input images (e.g. Postscript document) to generate realistic image populations. Such synthetic data can then be used to build huge and more representative training sets for document image recognition systems [6, 40, 70]. The ability of controlling the degree of degradation makes it also possible to carry out systematic design and evaluation of OCR systems [6, 9, 12, 39].

For handwriting recognition, no parameterized model of real-world image populations is available, due to the lack of mathematical models accounting for the enormous variations present in human handwriting. Nevertheless, several attempts to generate synthetic data for handwriting recognition systems are reported.

In [38], human written character tuples are used to build up synthetic text pages. Other approaches apply random perturbations on human written characters [16, 23, 35, 69, 74], or words [47, 98]. In [112], realistic off-line characters are generated from on-line patterns using different painting modes.

Generating synthetic handwriting does not necessarily require to use human written texts as a basis. In [29] and [111], characters are generated by perturbation of the structural description of character prototypes.

Those works where the application of synthetic training data yielded improved recognition performance over natural training data are related to the field of isolated character recognition [16, 23, 69, 74]. The natural training set was augmented by perturbed versions of its human written samples, and the larger training set enabled better training of the

recognizer. However, to the knowledge of the author, for the problem of general, cursive handwritten word and text line recognition, no similar results involving synthetically generated text images have been reported.

Finally, perturbation approaches can also be applied in the recognition phase, making the recognizer insensitive to small transformations or distortions of the image to be recognized [35, 47, 100].

1.2.2 Handwritten Notes and Communications

The use of handwriting has the ability to make a message or a letter look more natural and personal. One way to facilitate the input of such messages for electronic communication is to design methods that are able to generate handwriting-style texts, particularly in the style of a specific person.

Such methods have several possible applications. For example, using a word processor, editable handwritten messages could be inputted much faster directly from the keyboard. For pen-based computers, errors made by the user could be corrected automatically by substituting the erroneous part of text by its corrected version, using the same writing style.

In [33], texts showing a person's handwriting style are synthesized from a set of tuples of letters, collected previously from that person, by simply concatenating an appropriate series of static images of tuples together.

Learning-based approaches are presented in [18], [19], and [118], to generate Hangul characters, handwritten numerals, and cursive text, respectively, of a specific person's handwriting style. These methods need temporal (on-line) information to create a stochastic model of an individual style.

A method that is based on character prototypes instead of human written samples is presented in [57]. Korean characters are synthesized using templates of ideal characters, and a motor model of handwriting generation (see [85]) adapted to the characteristics of Korean script. The templates consist of strokes of predefined writing order. After the geometrical perturbation of a template, beta curvilinear velocity and pen-lifting profiles are generated for the strokes, which are overlapped in time. Finally, the character is drawn using the generated velocity and pen-lifting profiles.

One possible application of the method is to build handwriting-style fonts for word processors. On the other hand, the method can provide training data for handwriting recognizers. Although the generated characters look natural and represent various styles, they were not used for training purposes.

1.2.3 Reading-based CAPTCHAs

At present, there is a clear gap between the reading abilities of humans and machines. Particularly, humans are remarkably good at reading seriously degraded (e.g. deformed, occluded, or noisy) images of text, while modern OCR systems usually fail when facing such an image [8].

This observation can be used to design so-called CAPTCHAs (Completely Automatic Public Turing test to tell Computers and Humans Apart), to distinguish humans from computers [1, 2, 3]. The main application of CAPTCHAs is to prevent computer programs from automatic registration to publicly available services offered on the Internet. For example, this way spammers can be prevented from registering automatically thousands of free e-mail accounts for their fraudulent activities.

Several reading-based CAPTCHAs were proposed in the literature. All of them synthesize a degraded text image that is used to challenge the applicant to read it. The approval for the access to the required resource is then based on the correctness of the answer the applicant types in. The challenges may contain machine printed texts [1, 8, 11, 13, 17, 59, 101], or handwriting [94]. Reading-based CAPTCHAs that are already in industrial use include [1], [59], and [101].

1.3 Aim, Motivation, and Overview of the Thesis

The aim of this thesis is to investigate the generation and use of synthetic training data for off-line cursive handwritten text line recognition. For this purpose, two novel methods are presented for the generation of synthetic text lines. To evaluate these methods, the task of off-line cursive English handwritten text line recognition is considered. Printing and imaging defects are not addressed in this work, i.e. it is supposed that the variability of handwritten text images of the same textual content results from the individual handwriting styles as well as from the different writing instruments used.

The motivation is threefold. First, in the field of off-line isolated character recognition, synthetic training data has been used successfully to improve the performance of character classifiers, due to the increased size and variability of the training set. However, for the problem of cursive word and text line recognition, no similar results have been reported in the literature. Second, it has not yet been investigated how suitable is such synthetic handwriting generated from ideal input images (i.e. character prototypes) using a motor model of handwriting generation, for the training of recognition systems. The idea of this approach is analogous to that of the field of machine printed OCR, where degradation models are used to generate realistic document image populations from ideal input images. Finally, the automatic generation of training data is much faster and cheaper than

collecting human written samples.

An overview of the thesis is given below:

Chapter 1: The present work is put into context, by providing a brief introduction to the field of off-line handwriting recognition and synthetic text generation. Besides, a clear objective of the research is defined, and the motivations behind it are explained.

Chapter 2: A concise description of the handwriting recognition system used throughout the thesis is given, including the basics of Hidden Markov Models.

Chapter 3: The first synthetic text line generation method, based on the geometrical perturbation of existing human written text line images, is presented and evaluated under several experimental conditions. For the evaluation, various configurations of the recognizer and the synthetic handwriting generation process are considered.

Chapter 4: The second synthetic text line generation method is described, where the handwritten text lines are synthesized from ASCII transcriptions using templates of characters and the Delta LogNormal model of handwriting generation. Comparison of synthetic and natural training data is also performed.

Chapter 5: A novel approach for text line segmentation into individual words is introduced, in order to perform segmentation-based text line recognition, which gives new insights into the effects of synthetically expanded training sets.

Chapter 6: The main conclusions of the work are summarized and suggestions for possible future work are provided.

Appendix A: The segmentation technique of Chapter 5 is generalized for arbitrary sets of objects having distances among them. The result is a generic divisive hierarchical clustering algorithm, where the set of objects to cluster is represented by a graph structure.

Chapter 2

Handwriting Recognition System

The application considered in the thesis is the off-line recognition of cursively handwritten text lines. For this purpose, a state-of-the-art Hidden Markov Model (HMM) based recognizer developed previously at our institute is used, which also incorporates high-level linguistic knowledge into the recognition process [72]. Considering its operation from the topmost level, it takes a grayscale image of a whole line of text as input, and provides the ASCII transcription of that line, as shown in Fig. 2.1. Of course, this is the ideal case, because it can also happen that the recognizer makes errors, i.e. the resulting transcription is different from the correct one.

A more detailed description of the recognition system can be seen in Fig. 2.2. According to this scheme, the text line image to be recognized is first normalized, and then feature vectors are extracted from the normalized image. These feature vectors, together with the trained HMMs and a language model, are used by the recognizer to produce the ASCII transcription of the text line. The implementation of the system is based on the Hidden Markov Model Toolkit (HTK) [120]. In the following sections of this chapter, the constituents of the recognition system depicted in Fig. 2.2 will be explained. For additional details the reader is referred to [72].

2.1 The IAM-Database

A large database of handwritten text images is a prerequisite for the development and evaluation of handwriting recognition systems. In this work, the IAM-Database is considered for this purpose [73].[1] For the acquisition of the database, sentences from the LOB-corpus [43] were printed on forms, and subjects were asked to write down those sentences on the empty area of the form, below the machine printed part. An example of

[1] See also: http://www.iam.unibe.ch/~fki/iamDB.

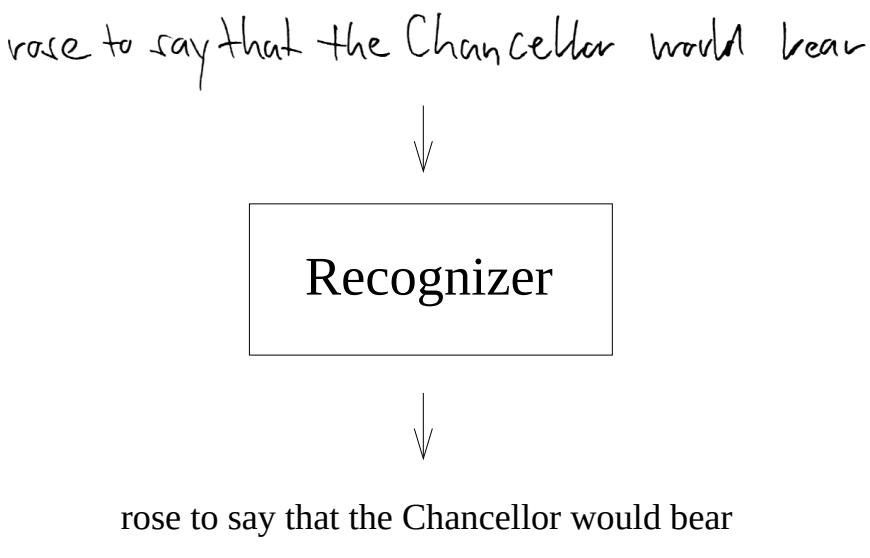


Figure 2.1: The recognizer provides the ASCII transcription of a handwritten text line image.

a completed form of the IAM-Database can be seen in Fig. 2.3. There was no constraint on the style and the writing instrument used.

The current version of the database contains 1,539 pages of scanned text[2] produced by 657 writers, including 115,320 instances of handwritten words distributed over 13,353 lines of text. The underlying lexicon includes more than 12,000 different words. The individual handwritten text lines of the scanned forms have been extracted and thus are also available separately, allowing to perform off-line handwritten text line recognition experiments directly, without any further segmentation steps. The extracted text lines are also contrast-enhanced to improve the separation between the handwriting ink and the background (i.e. to make handwriting ink darker, possibly with a white background). Otherwise, no further preprocessing to enhance image quality (e.g. by noise filtering) is done.

[2]8-bit grayscale images, scanned at 300dpi resolution.

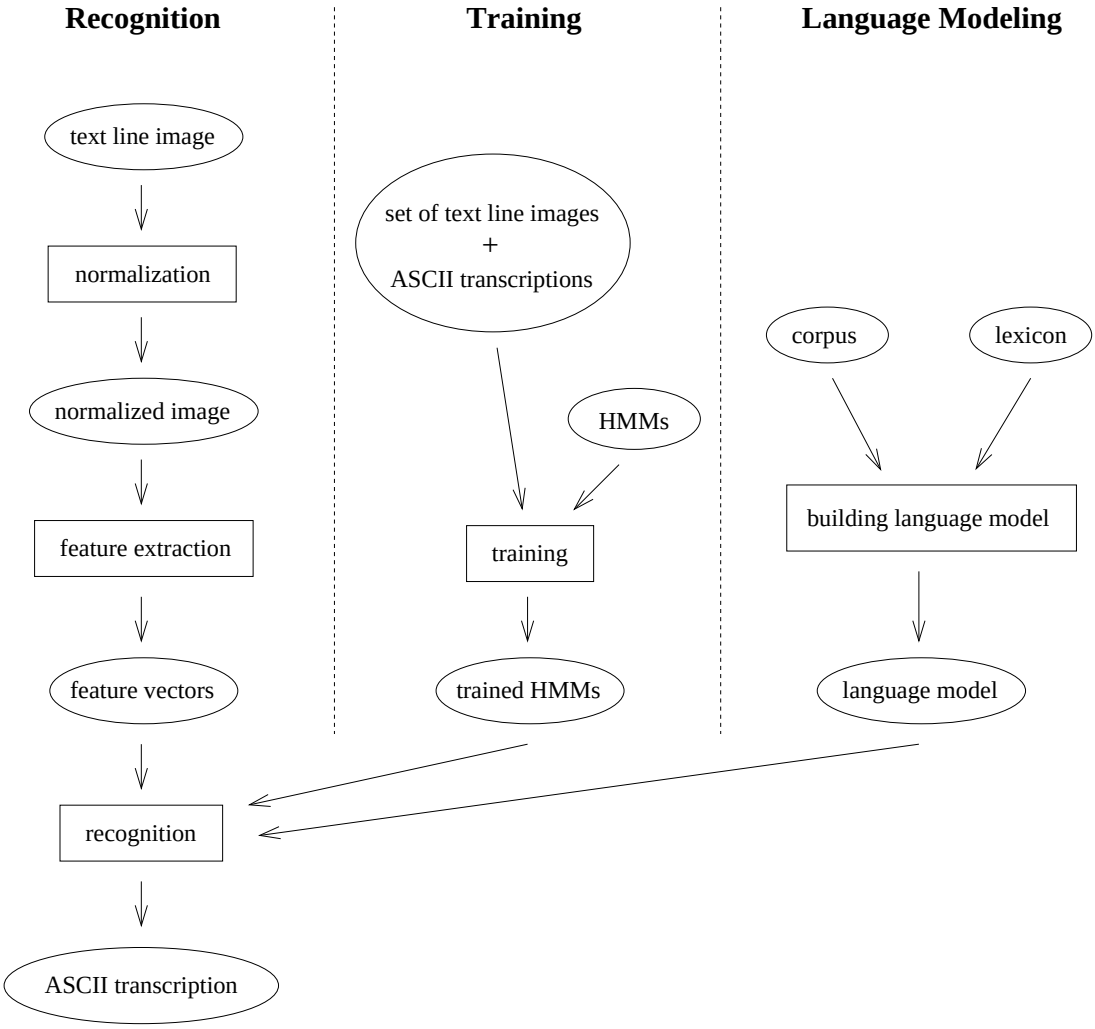
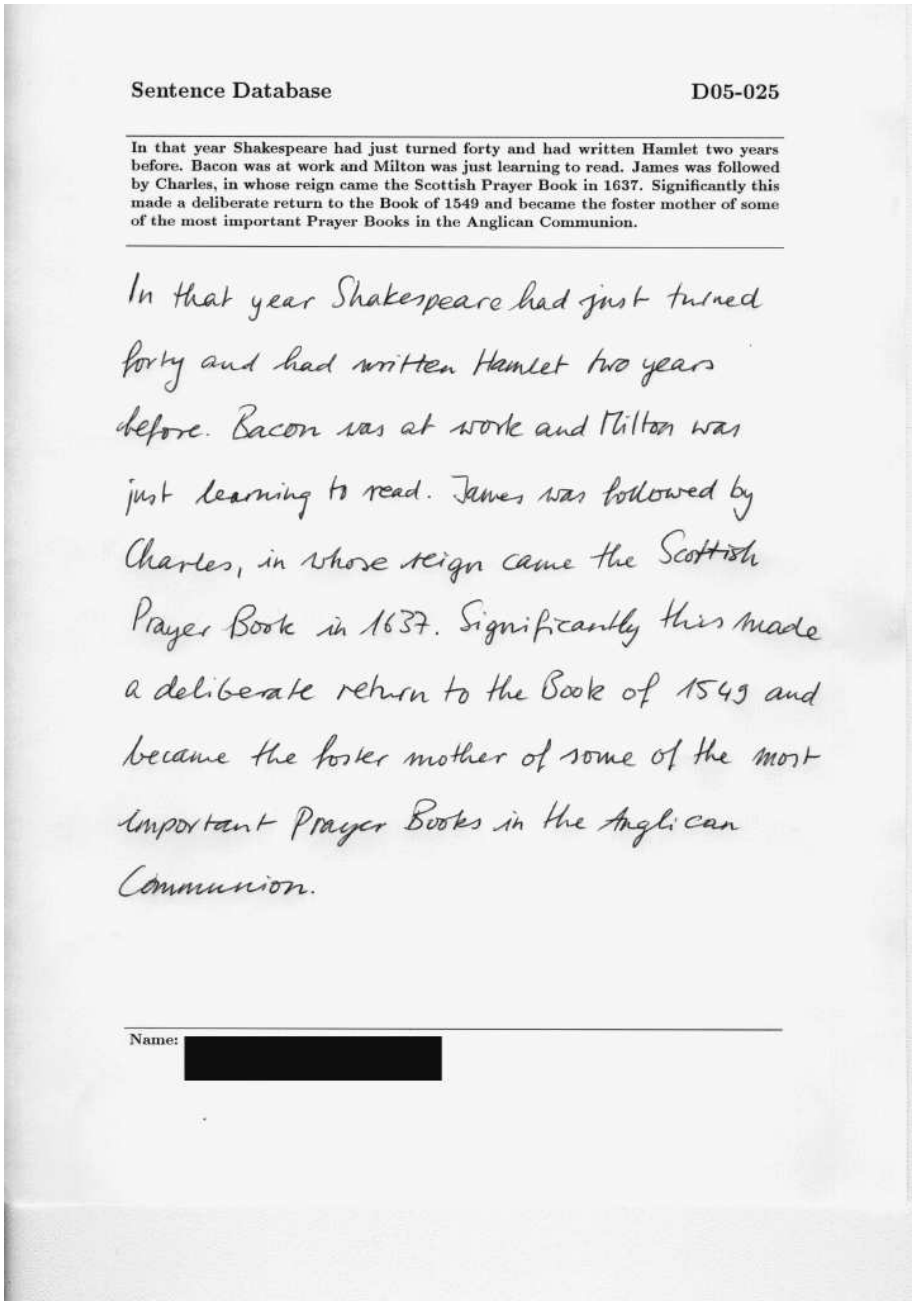


Figure 2.2: Detailed architecture of the recognition system.



Sentence Database D05-025

In that year Shakespeare had just turned forty and had written Hamlet two years before. Bacon was at work and Milton was just learning to read. James was followed by Charles, in whose reign came the Scottish Prayer Book in 1637. Significantly this made a deliberate return to the Book of 1549 and became the foster mother of some of the most important Prayer Books in the Anglican Communion.

In that year Shakespeare had just turned forty and had written Hamlet two years before. Bacon was at work and Milton was just learning to read. James was followed by Charles, in whose reign came the Scottish Prayer Book in 1637. Significantly this made a deliberate return to the Book of 1549 and became the foster mother of some of the most important Prayer Books in the Anglican Communion.

Name: [redacted]

Figure 2.3: Completed form of the IAM-Database: the writer was asked to write down the machine printed part using his/her own writing style.

2.2 Normalization

The purpose of normalization is to reduce the variability present in handwriting, in order to facilitate machine recognition. The normalization of a text line consists of several steps, as illustrated in Fig. 2.4:

- **Skew correction:** the text line is rotated such that the imaginary line on which the words were written, i.e. the lower baseline, is aligned horizontally.
- **Slant correction:** removes the slant of the letters using a shearing operation, so that the writing becomes upright.
- **Positioning:** the three main areas of the text line, namely the ascender part, the descender part, and the body of the text (middle part), are vertically scaled to a predefined height. The horizontal line that separates the ascender (descender) part from the body is called the upper (lower) baseline. In other words, the body of the text is located between the upper and lower baselines.
- **Width normalization:** the average letter width is estimated, and then adjusted to a predefined value by applying a horizontal scaling operation.

Note that each of the normalization steps is performed by applying a linear geometrical transformation on the text line image. Since the applied transformation is of global nature, local deviations of the property to be normalized are not removed. As an example, different letters might have different slants, so it is impossible to normalize them by only one global shearing transformation. Therefore the result of the normalization is practically never perfect. However, the same is true for those more sophisticated normalization algorithms that take local characteristics into account, too, although they may produce better results for certain text images. This is because the quality of the normalized image can only be judged subjectively, using the knowledge of what is actually written in the text line. That's why perfect normalization can be neither exactly defined nor required.

2.3 Feature Extraction

For the extraction of features from the normalized text line, a sliding window of one pixel width is moved from left to right over the image, and nine geometrical features are calculated at each window position, that is, for each pixel column. See illustration in Fig. 2.5. Thus an input text line is converted into a sequence of feature vectors in a 9-dimensional feature space. The aim of this conversion is to have a compact and tractable

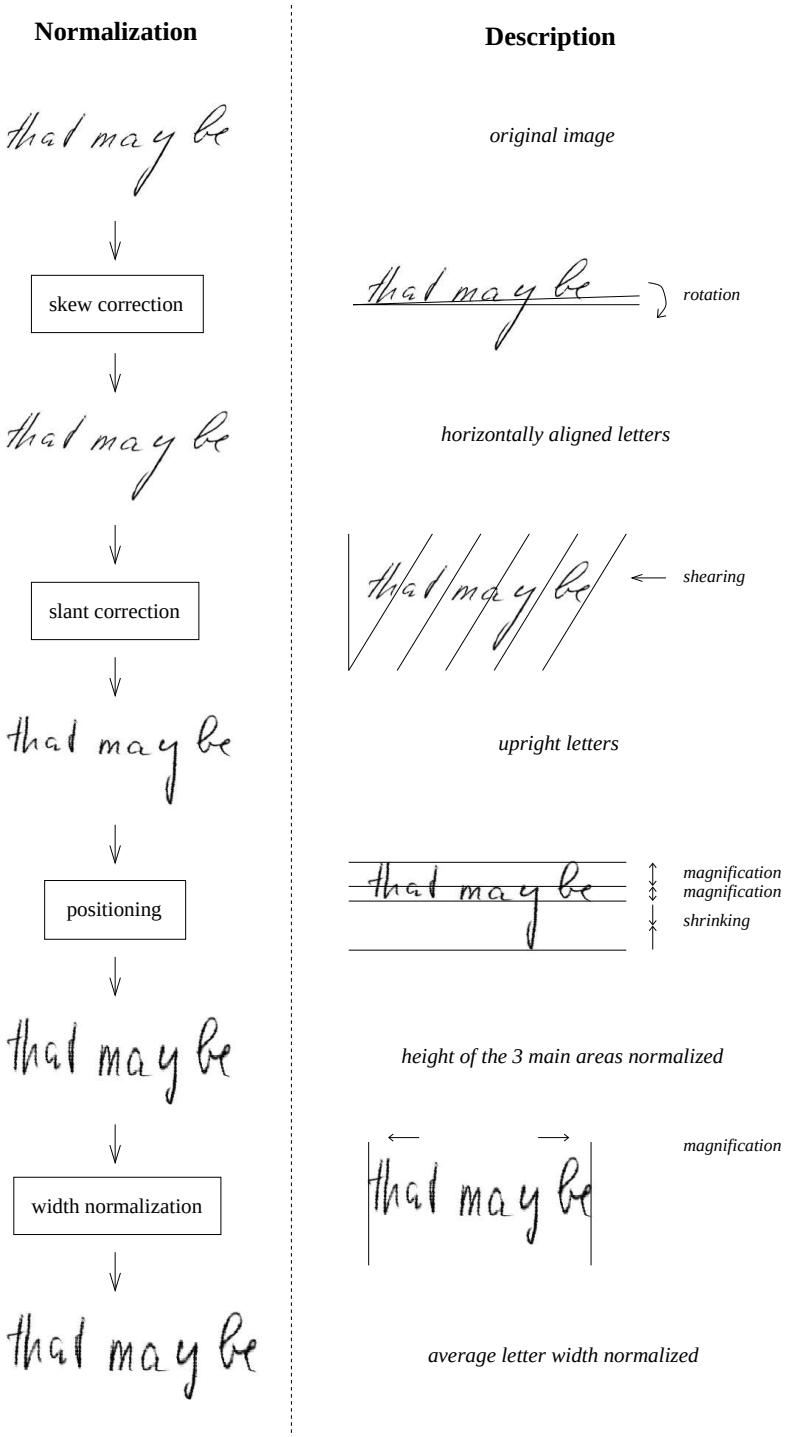


Figure 2.4: Illustration of the normalization steps: skew correction, slant correction, positioning, and width normalization.

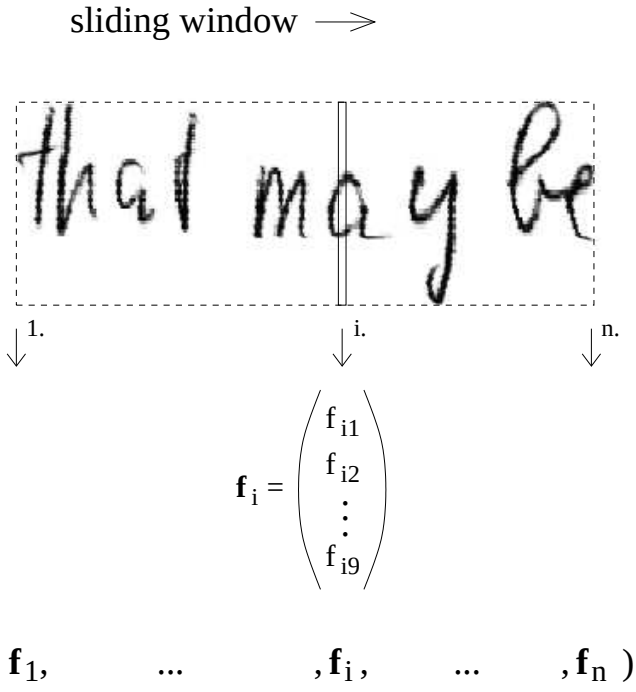


Figure 2.5: Feature extraction: sliding window of one pixel width moves from left to right, and nine geometrical features are calculated at each position (n is the width of the image in pixels).

representation of the text line, which contains relevant information needed for recognition, and which is appropriate for the further processing with HMMs (see Section 2.4).

The nine features used in the recognition system are the following:

1) Average gray value of the window.

2) Center of gravity of the window.

3) Second order moment of the window.

4-5) Position of the upper and lower contour in the window, respectively.

6-7) Gradient of the upper and lower contour at the window's position, respectively.

8) Number of black-white transitions in vertical direction.

9) Average gray value between the upper and lower contour.

To decide the contour of the writing for features 4 to 8, a threshold is used that separates foreground pixels (dark pixels that belong to the handwriting ink) from background pixels (light pixels that do not belong to the handwriting ink). Furthermore, to calculate features 6 and 7, the succeeding pixel column is also taken into consideration.

2.4 Hidden Markov Modeling

The core of the handwriting recognition system, that is, the underlying pattern recognition engine, is based on Hidden Markov Models, or shortly HMMs. This section presents the basics of HMMs as well as the way they are applied to the problem of handwritten text line recognition.

2.4.1 Basics of Hidden Markov Models

Hidden Markov Models are finite-state stochastic systems that generate output sequences during their operation. They have proved to be a useful abstraction of various real-world processes that produce observable output, due to their success in characterizing the statistical properties of the observations [88].

The operation of an HMM is a stepwise process, where the steps are performed at regularly spaced discrete points of time, $t = 1, 2, \ldots, T$, where T denotes the time of the last step. At each step, the HMM undergoes a transition of its state, and based on the new state, an output value is generated. The generated output values are observable, but the underlying sequence of states is not, i.e. the states are hidden for the observer.

The components of an HMM are the following:

- A finite set, $S = \{s_1, s_2, \ldots, s_N\}$, of possible states. The state at time t is denoted by q_t.
- The state transition probability matrix, $A = \{a_{ij}\}$, of size $N \times (N + 1)$. For $1 \leq i, j \leq N$, a_{ij} is the probability that the HMM gets into state s_j in the next step, provided that currently it is in state s_i. Formally, $a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i)$. The assumption is made that q_{t+1} depends exclusively on q_t.[3] For $j = N + 1$, a_{ij} gives the probability that the HMM terminates from state s_i, i.e. it will not perform any more steps.[4] For each state s_i, $\sum_{j=1}^{N+1} a_{ij} = 1$ must hold.

[3]Such HMMs are called first order HMMs. There also exist higher order types, where q_{t+1} depends on more than one of the preceding states.

[4]Termination probabilities are usually not part of standard HMM descriptions in the literature. However, in the current application they are a convenient means to concatenate character level HMMs.

- A set, V, of possible output values, which can be either finite or infinite. Typically the elements of V are numerical vectors.
- State dependent probability distributions, $B = \{b_i(o)\}$, of the output values, for all the N possible states of the HMM. Depending on the discrete or continuous nature of the distributions in B, the HMM is called discrete or continuous, respectively. Let $v \in V$, and let o_t denote the output value generated at time t. Then in the discrete case, $b_i(v) = P(o_t = v \mid q_t = s_i)$, which means that the output value v is generated with probability $b_i(v)$, provided that the HMM is in state s_i. In the continuous case the meaning is similar, but $b_i(v)$ is a likelihood rather than a probability. For each state s_i, $\sum_{v \in V} b_i(v) = 1$ must hold for discrete HMMs, and $\int_V b_i(v)dv = 1$ for continuous ones.
- The initial state distribution, $\pi = \{\pi_i\}$, where $\pi_i = P(q_1 = s_i)$ denotes the probability that the HMM is in state s_i at time $t = 1$. $\sum_{i=1}^{N} \pi_i = 1$ must hold.

None of the components above are time dependent. Thus the HMM is a stationary model, assuming that the properties of the observation sequence do not vary with time. For the ease of notation, an HMM is usually represented in the compact form $\lambda = (A, B, \pi)$, where λ stands for the name of the HMM.

When designing an HMM, one has to make two important decisions: one about the type of the output value distributions $b_i(\cdot)$, and the other about the number and topology of the HMM states. The topology reflects what kind of state transitions are allowed ($a_{ij} > 0$), and what are forbidden ($a_{ij} = 0$). An illustration of one of the simplest but rather common topology, namely the linear topology, is given in Fig. 2.6. Linear topology means that $\pi_1 = 1$, and $a_{ij} > 0$ implies $j = i$ or $j = i + 1$. Note that due to topological constraints, the 5-state HMM in Fig. 2.6 performs at least 5 steps before it terminates (i.e. $T \geq 5$), and thus it can only generate output sequences of length 5 or more.

After the design of an HMM, it has still many free parameters to be adjusted, so that the model $\lambda = (A, B, \pi)$ "best describes" how a given observed output sequence $O = o_1 o_2 \ldots o_T$ emerged from the underlying stochastic process. This task is called training and it is accomplished by the Baum-Welch algorithm [88], which tries to maximize $P(O \mid \lambda)$ iteratively.[5] Should there be multiple output sequences $O_1, O_2, \ldots, O_k$ to describe, the value $\prod_{l=1}^{k} P(O_l \mid \lambda)$ is maximized, which gives the overall probability (or likelihood) of the training samples.

Finally, let $O = o_1 o_2 \ldots o_T$ denote an observed output sequence produced by model λ. As it was mentioned earlier, the underlying state sequence $Q = q_1 q_2 \ldots q_T$ is hidden, so usually it is not possible to figure it out. But the most likely state sequence, for which $P(Q \mid O, \lambda)$ or equivalently $P(Q, O \mid \lambda)$ is maximal, can be calculated. This is called the

[5]For simplicity, from now on $P(\ldots)$ may denote either probability or likelihood.

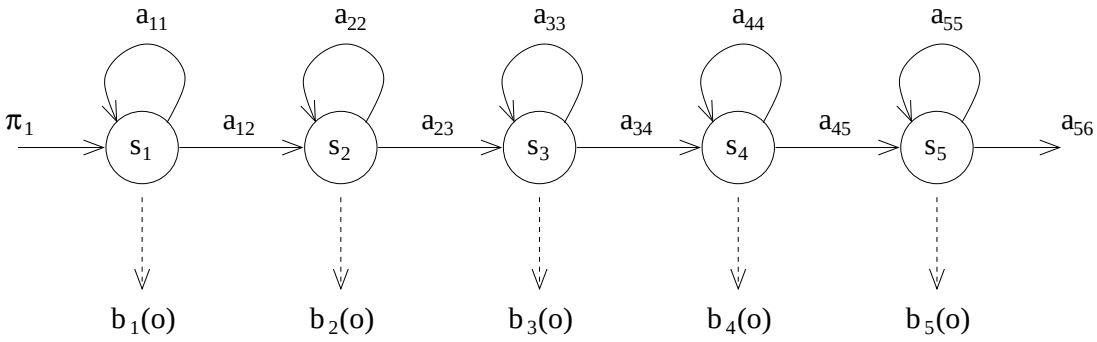


Figure 2.6: HMM with linear topology: all states are visited in a linear order before termination is reached.

decoding task, and its aim is to find the optimal state sequence Q_{opt} which "best explains" the observations. For a given state sequence Q,

$$P(Q, O \mid \lambda) = \pi_{q_1} \cdot b_{q_1}(o_1) \cdot a_{q_1 q_2} \cdot b_{q_2}(o_2) \cdot \ldots \cdot a_{q_{T-1} q_T} \cdot b_{q_T}(o_T) \cdot a_{q_T, N+1} \tag{2.1}$$

where q_t $(t = 1, 2, \ldots, T)$ acts for the corresponding index i for which $q_t = s_i$. Unfortunately, the number of possible state sequences is typically exponential in terms of T. Despite this fact, there exists an efficient way to solve the decoding task, the Viterbi algorithm which is based on dynamic programming techniques [88]. In the following, the value $P(Q, O \mid \lambda)$ will be referred to as the score of state sequence Q. Consequently, the optimal state sequence Q_{opt} has maximal score.

For more detailed introduction to HMMs see [88, 89].

2.4.2 Application to Text Line Recognition

To be able to apply HMMs in the recognition system, each input text line image has to be converted into a sequence O of observations. This is done by the feature extraction procedure in a left to right manner, as depicted in Fig. 2.5. The i-th observation value o_i corresponds to the i-th 9-dimensional feature vector $\boldsymbol{f}_i$, thus the feature vector sequence F is taken as the sequence of observations, i.e. $O = F$. As the set of possible observation values, the whole 9-dimensional Euclidean space $\mathbb{R}^9$ is considered.

For each character, a 14-state HMM is built.[6] In all HMMs the linear topology illustrated in Fig. 2.6 is used, which is intended to reflect the left to right direction of English handwriting. In each state s_i, the observation probability distribution is modeled by a

[6]The number of states was found empirically, see [72] for details.

continuous probability density function of Gaussian mixture type, i.e. by a weighted sum of Gaussian PDFs.[7] Formally,

$$b_i(\boldsymbol{o}) = \sum_{m=1}^{M} c_m \cdot \mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) \tag{2.2}$$

where $\boldsymbol{o} \in \mathcal{R}^9$ is the observation vector, M is the number of mixture components, $c_m \geq 0$ is the weight of the m-th component, $c_1 + c_2 + \ldots + c_m = 1$, and $\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a 9-dimensional Gaussian PDF with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, that is,

$$\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^9 |\boldsymbol{\Sigma}|}} \cdot e^{-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}-\boldsymbol{\mu})} \tag{2.3}$$

To reduce the number of free parameters, the covariance matrix $\boldsymbol{\Sigma}$ is set to diagonal, which is equivalent to the assumption that the features are statistically independent. In practice, this is a reasonable compromise between system complexity and performance, because usually there is insufficient training data and time to give reliable estimations of the non-diagonal elements. The restriction of using mixtures of Gaussians is not severe, since they are flexible enough to approximate other types of PDFs, too. The number of mixture components, M, is the same for all states in all of the HMMs. This parameter controls how detailed estimations can be stored in the HMM states about the feature value distributions.

The character models are concatenated to represent words and sequences of words. Such a composite HMM of the phrase "it is" is shown in Fig. 2.7. The transition probability between two consecutive characters is equal to the termination probability at the last state of the first character. The spaces between words are considered as instances of a special character named *sp*, with a corresponding HMM built for it. The dashed rectangles indicate the boundaries of the word level HMMs for "it" and 'is''. The space at the end of each word is included automatically.[8]

For training, the parameters of the HMMs are adjusted so that they fit to a set of handwritten text lines called the training set (see middle part of Fig. 2.2). This means that the overall likelihood of the training set given by $\prod_{k=1}^{K} P(F_k \mid \lambda_k)$ is maximized iteratively, where K denotes the size of the training set, F_k is the feature vector sequence of the k-th normalized text line image, and λ_k is the composite HMM built from the characters of the transcription provided for that line. The maximization is performed using the embedded Baum-Welch algorithm [120], which does not require information about the exact character positions within a text line image.

[7] PDF = probability density function

[8] In order that the last word is modeled properly, a small margin is added to the right side of every normalized text line image.

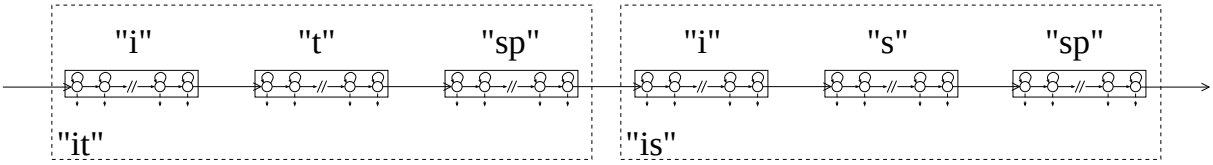


Figure 2.7: Concatenation of character HMMs to represent the phrase "it is".

In the recognition phase, word level HMMs are constructed for all the words in the lexicon by concatenating the corresponding character models. Then, through the interconnection of those HMMs, a so-called recognition network is compiled, a graph structure whose paths correspond to valid transcriptions of text lines and vice versa. On the lowest level, the nodes of the recognition network are HMM states, and the edges represent either initial state or transition probabilities. The state level paths whose first and last states are also the first and last state of a word level HMM, respectively, are called valid paths, since only they represent valid transcriptions. To recognize a text line of feature vector sequence F, the valid state level path having the highest score with respect to F needs to be found. The search is accomplished by the Token Passing algorithm [120], a variant of the Viterbi algorithm. The output of the recognizer is the word level transcription corresponding to the optimal state sequence. As a consequence, the difficult task of explicitly segmenting a line of text into isolated words is avoided, and the segmentation is obtained as a byproduct of the Viterbi decoding.

2.5 Bigram Language Model

Besides modeling the handwriting ink, the recognizer also makes use of high level linguistic knowledge, by incorporating a statistical language model. Statistical language models try to capture regularities of natural language, in order to improve the performance of related applications [92]. So far, the most successful models are the n-gram language models. Their assumption is that the probability distribution of the next word in a sequence depends only on the last $n-1$ words. In practice, the most common choices are $n = 2$ and $n = 3$, called bigram and trigram language model, respectively.

The recognizer considered in this thesis uses a bigram language model. It is based on the estimation of the following probabilities, where w_i and w_j denote words from the underlying lexicon:

- $P(w_i)$, the probability of the occurrence of word w_i, and
- $P(w_j \mid w_i)$, the conditional probability that word w_j follows immediately word w_i.

The estimation of these probabilities is based on the LOB-corpus [43]. According to the bigram language model, the a-priori probability of a word sequence $W = w_1 w_2 \dots w_l$, also called the language model score of W, is calculated by the following formula:

$$P(W) = P(w_1) \cdot P(w_2 \mid w_1) \cdot \ldots \cdot P(w_l \mid w_{l-1}) \tag{2.4}$$

The integration of the language model into the recognition process takes place in the recognition network described at the end of Subsection 2.4.2. The idea is that the original initial state and transition probabilities belonging to edges that lead to the first state of a word level HMM are multiplied by the corresponding language model probabilities $P(w_i)$ and $P(w_j \mid w_i)$, respectively [120].

This means that the new score of a state sequence Q will be the original score of Q multiplied by the corresponding language model score. In other words, the HMM score will be multiplied by the language model score. Formally, the new expression maximized at the recognition stage is $P(Q, F \mid \lambda) \cdot P(W)$, where Q is a valid state path, F is the feature vector sequence, λ is the composite HMM corresponding to Q, and W is the word sequence corresponding to Q. Thus the recognizer favors the more probable word sequences.

2.6 Evaluation of Performance

Considering one text line only, the output of the recognizer is a sequence of words, which has to be compared with the correct transcription of the text line image to assess the quality of the recognition result, that is, to assess how well the recognizer performed on that particular text line. In the literature, there are several approaches based on string matching to measure the performance of text line recognition systems [45, 120]. Unfortunately, due to the lack of boundary information, it is often not possible to provide an interpretation of a given performance measure that is precise and vividly descriptive at the same time, explaining also the origin of the errors the recognizer makes.

The first step of the evaluation is the optimal alignment of the two word sequences using a string matching algorithm based on dynamic programming [120]. An illustration of this alignment as well as the steps following it can be seen in Fig. 2.8. Punctuations are also treated as words. Then, certain statistics related to the found alignment are calculated, such as number of words of both the correct transcription and the recognition result, correctly aligned words, insertions,[9] deletions,[10] and substitutions. From these statistics, different measures of performance can be derived, among them are (see Fig. 2.8 for the

[9] Insertions usually occur when a word in the text line image is recognized as two or more words.
[10] Deletions typically arise when two or more neighboring word images are recognized as one word.

corresponding formulas):

- **Recognition rate:** intuitively, this is the percentage of the correctly recognized words of the text line.
- **Accuracy:** this measure is related to the total number of edit operations needed to transform the recognition output into the correct transcription.
- **Precision:** the percentage of the output words that are correctly assigned by the alignment method.

The recognition rate and the precision are always between 0% and 100%, while the accuracy can be negative, too. 100% accuracy means that the recognizer gave perfect transcription of the input text line. Concerning the recognition rate, even at 100% some extra insertions may occur.[11] Together with one or both of the other measures, precision can provide supplementary characterization of the system performance.

It is also noted that the recognition rate is analogous to the performance measure used at segmentation-based recognition systems, provided that the text line is correctly segmented into individual words. Accordingly, if the recognition of a word goes wrong, it is not interesting in terms of the recognition rate what exactly the erroneous result is. For example, if the word "sounding" in Fig. 2.8 was recognized as "saw a long" instead of "saw many", it would make a difference only in the accuracy and precision values. In summary, the usefulness of the different performance measures is task dependent. For example, the recognition rate is a suitable measure of performance if the task is the automatic indexing of handwritten documents, while the accuracy is more appropriate for the task of providing the correct transcription of a handwritten text, possibly followed by manual correction.

In the present work, the performance of the handwriting recognition system is measured in terms of the recognition rate. For the evaluation, a set of text lines, i.e. a test set, is considered. To compute the recognition rate for a whole set of lines rather than for a single line, the statistics used in the corresponding formula (i.e. H and N, see Fig. 2.8) are aggregated over the entire test set.

[11] Such extra insertions are quite rare, and generally mean that a punctuation symbol (e.g. dot, comma) is attached to the end of a correctly identified word.

Alignment

Correct transcription:	sounding	name	of	Lansdowne	Road	.	There	are
Recognition result:	saw	many	hours	of	Lansdowne	Road	.	Therefore

Statistics

Number of words: N = 8

Num. of result words: M = 8

Correct: H = 4 (of, Lansdowne, Road, .)

Deleted: D = 1 (are)

Substituted: S = 3 (sounding, name, There)

Inserted: I = 1 (many)

Performance measures

$$\text{Rec. rate} = \frac{H}{N} \times 100\% = 50\%$$

$$\text{Accuracy} = \frac{N - D - S - I}{N} \times 100\% = \frac{H - I}{N} \times 100\% = 37.5\%$$

$$\text{Precision} = \frac{H}{M} \times 100\% = 50\%$$

Figure 2.8: Alignment of the recognition result with the correct transcription, and measures for performance evaluation.

Chapter 3

Synthetic Handwriting by Perturbation Method

In this chapter, a perturbation model is presented to generate synthetic text lines from existing cursively handwritten lines of text produced by human writers. The motivation is to add synthetic data to the natural training data, rendered by human writers, so as to enlarge the training set. The basic idea of the approach is to use continuous nonlinear functions that control a class of geometrical transformations applied on the existing handwritten texts. The functions ensure that the distortions performed cannot be reversed by standard preprocessing operations of the handwriting recognition system. Besides the geometrical distortions, thinning and thickening operations are also part of the model.

A closer examination reveals, however, that the use of synthetic training data does not necessarily lead to an improvement of the recognition rate, because of two adversarial effects. First, it can be expected that the variability of the training set improves, which potentially leads to a higher recognition rate. On the other hand, synthetic training data may bias a recognizer towards unnatural handwriting styles, which can lead to a deterioration of the recognition rate, particularly if natural handwriting is used for testing.

The aim in this chapter is to find configurations of the recognizer presented in Chapter 2 and the synthetic handwriting generation process, by which the recognition performance can be significantly improved. The parameters examined include the number of Gaussian mixture components in the recognizer used for distribution estimation, distortion strength, training set size, number of writers in the training set, and the proportion of natural and synthetic training data. It is shown that significant improvement of the recognition performance is possible even when the original training set is large and the text lines are provided by many different writers. But to really achieve an improvement in this case, one has also to consider the capacity of the recognition system, which needs to be appropriately adjusted when expanding the training set with synthetic text lines.

3.1 Perturbation Model

Variation in human handwriting is due to many sources, including letter shape variation, variety of writing instruments, and others. In this section, a perturbation model for the distortion of cursive handwritten text lines is presented, where these sources of variation are modeled by geometrical transformations as well as thinning and thickening operations.

The design of the perturbation model was motivated by two important aspects: simplicity (including transparency) and nonlinearity. Simplicity is achieved by applying the same concept (underlying function, see Subsection 3.1.1) to each type of geometrical transformation, and considering only some basic types of distortions (shearing, scaling and shifting along one of the main axes). Nonlinearity is needed so that the distortions applied on the handwriting cannot be reversed by standard linear preprocessing operations of the handwriting recognition system.

The perturbation model incorporates some parameters with a range of possible values, from which a random value is picked each time before distorting a text line. There is a constraint on the text lines to be distorted: they have to be skew and slant corrected, because of the nature of the applied geometrical transformations. This constraint is not severe, because skew and slant correction are very common preprocessing steps found in almost any handwriting recognition system. In the following subsections the perturbation model is described in greater detail.

3.1.1 Underlying Functions

Each geometrical transformation in the model is controlled by a continuous nonlinear function, which determines the strength of the considered transformation. These functions will be called *underlying functions*.

The underlying functions are synthesized from a simple function, called *CosineWave*. A CosineWave is the concatenation of n functions, $f_1, f_2, \ldots, f_n$, where $f_i : [0, l_i] \rightarrow \mathbb{R}$, $f_i(x) = (-1)^i \cdot a \cdot \cos(\frac{\pi}{l_i} \cdot x),\ l_i > 0$.

An example is shown in Fig. 3.1. The functions f_i (separated by vertical line segments in Fig. 3.1) are called *components*. The *length* of component f_i is l_i and its *amplitude* is $|a|$. The amplitude does not depend on i, i.e. it is the same for all components.

To randomly generate a CosineWave instance, three ranges of parameter values need to be defined:

- $[a_{min}, a_{max}]$ for the amplitude $|a|$,
- $[l_{min}, l_{max}]$ for the component length,

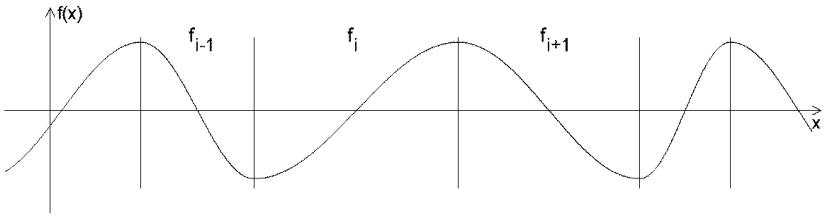


Figure 3.1: Example of a CosineWave function.

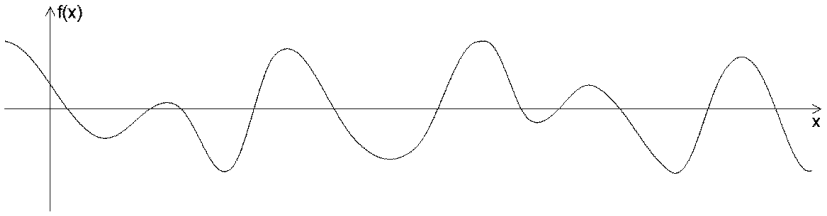


Figure 3.2: Example of a sum of two CosineWave functions.

- $[x_{min}, x_{max}]$ for the interval to be covered by the concatenation of all components.

The generation of a CosineWave is based on the following steps. First the amplitude is selected by picking a value $\alpha \in [a_{min}, a_{max}]$ randomly and letting $a = \alpha$ or $a = -\alpha$ with a 50% probability each. Then l_1 is decided by randomly picking a value from $[l_{min}, l_{max}]$. Finally the beginning of the first component (i.e. f_1) is chosen randomly from the $[x_{min} - l_1, x_{min}]$ interval. From this point on we only have to add additional components, one after the other, with randomly chosen lengths, until x_{max} is reached. For randomly picking a value from an interval, always the uniform distribution over that interval is used.

An underlying function is obtained by summing up a number, m, of such CosineWave functions. Fig. 3.2 depicts an example of such an underlying function with $m = 2$.

3.1.2 Geometrical Transformations

The underlying functions control several geometrical transformations, which are divided into two groups: the *line level* transformations applied on whole lines of text, and the *connected component level* transformations applied on the individual connected components of the considered line of text. The underlying function of each transformation is randomly generated, as described in Subsection 3.1.1. The parameters x_{min} and x_{max} are always

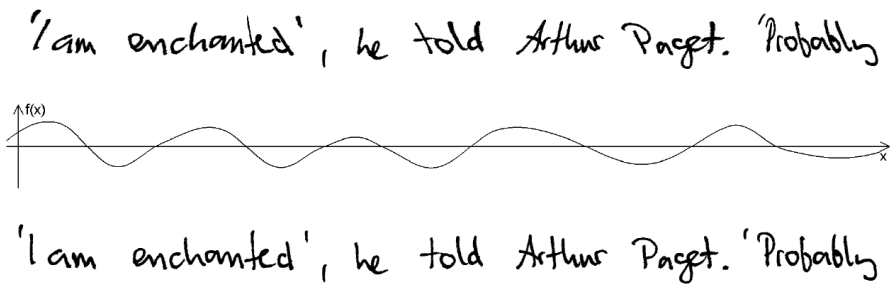


Figure 3.3: Illustration of shearing. The original text line is at the bottom, the underlying function is in the middle, and result of the distortion is on top.

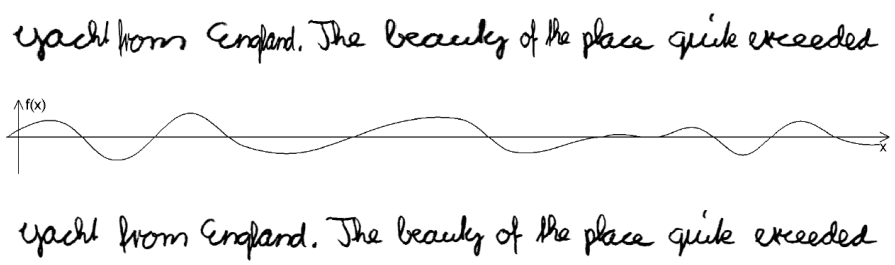


Figure 3.4: Illustration of horizontal scaling.

defined by the actual size of the image to be distorted. In the following the geometrical transformations will be defined and illustrated by figures. Note that the figures are only for illustration purposes, and weaker instances of the distortions are actually used in the experiments described later on.

There are four classes of geometrical transformations on the line level. Their purpose is to change properties, such as slant, horizontal and vertical size, and the position of characters with respect to the baseline. The line level transformations are these:

- **Shearing:** The underlying function, denoted by $f(x)$, of this transformation defines the tangent of the shearing angle for each x coordinate. Shearing is performed with respect to the lower baseline. An example is shown in Fig. 3.3. In this example and the following ones, the original text line is shown at the bottom, the underlying function in the middle, and the result of the distortion on top.

- **Horizontal scaling:** Here the underlying function determines the horizontal scaling factor, $1+f(x)$, for each x coordinate. This transformation is performed through

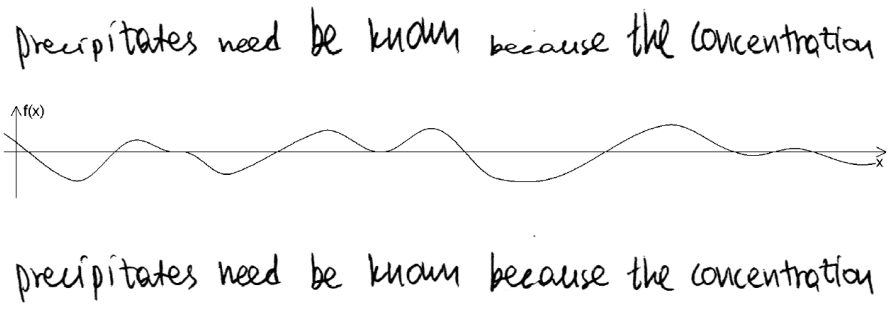


Figure 3.5: Illustration of vertical scaling.

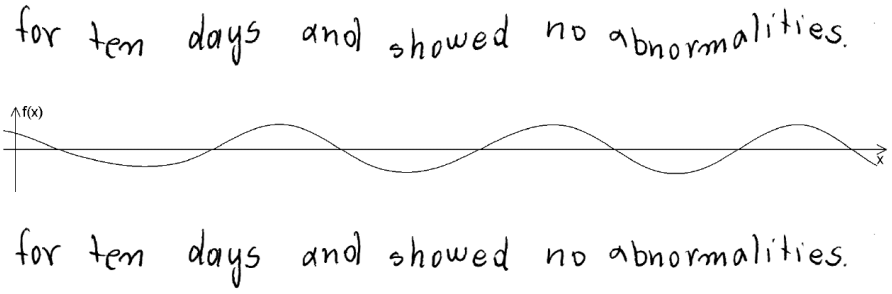


Figure 3.6: Illustration of baseline bending.

horizontal shifting of the pixel columns.[1] An example of this operation is shown in Fig. 3.4.

- **Vertical scaling:** The underlying function determines the vertical scaling factor, $1 + f(x)$, for each x coordinate. Scaling is performed with respect to the lower baseline. An example can be seen in Fig. 3.5.

- **Baseline bending:** This operation shifts the pixel columns in vertical direction, by the amount of $h \cdot f(x)$ for each x coordinate, where h is the height of the body of the text (i.e. the distance between the upper and lower baselines). An example is given in Fig. 3.6.[2]

The perturbation model also includes transformations, similar to the ones described above, on the level of connected components. These transformations change the structure of the

[1] The appropriate shifting value at x is given by $\int_0^x (1 + f(x))dx = x + \int_0^x f(x)dx$.
[2] It can be observed that the baseline is usually not a straight line, but rather of a wavy shape.

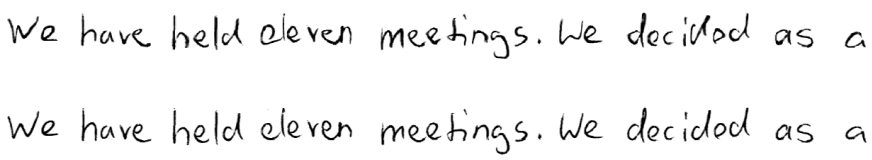


Figure 3.7: Illustration of connected component level distortions. The original text line is below, and the result of the distortions is above.

writing in a local context, i.e. within connected components. After the application of these transformations, the resulting connected components are scaled in both horizontal and vertical direction so that their bounding boxes regain their original sizes, and then they are placed in the image exactly at their original locations. For each connected component, individual underlying functions are generated. There are three classes of such transformations:

- **Horizontal scaling:** This transformation is identical to the line level horizontal scaling as described before, but it is applied to individual connected components rather than whole lines of text.
- **Vertical scaling 1:** This is the counterpart of horizontal scaling in the vertical direction.
- **Vertical scaling 2:** This transformation is identical to the line level vertical scaling, except that scaling is performed with respect to the horizontal middle-line of the bounding box.

The effect of all three transformations applied one after the other is shown in Fig. 3.7. In this figure, the lower text line is the original one, and above its distorted version is displayed. One can observe that in spite of the distortions the connected components underwent, their bounding boxes have remained the same.

3.1.3 Thinning and Thickening Operations

The appearance of a text line can also be changed by varying the thickness of its strokes. In the present perturbation model this is done by applying thinning or thickening steps iteratively. The method is based on a grayscale variant of the MB2 thinning algorithm [68]. (A general way to get the grayscale version of a specific type of thinning algorithm operating on binary images can be found in [103]). Thinning and thickening could also be performed using the morphological erosion and dilation operators, respectively, but this

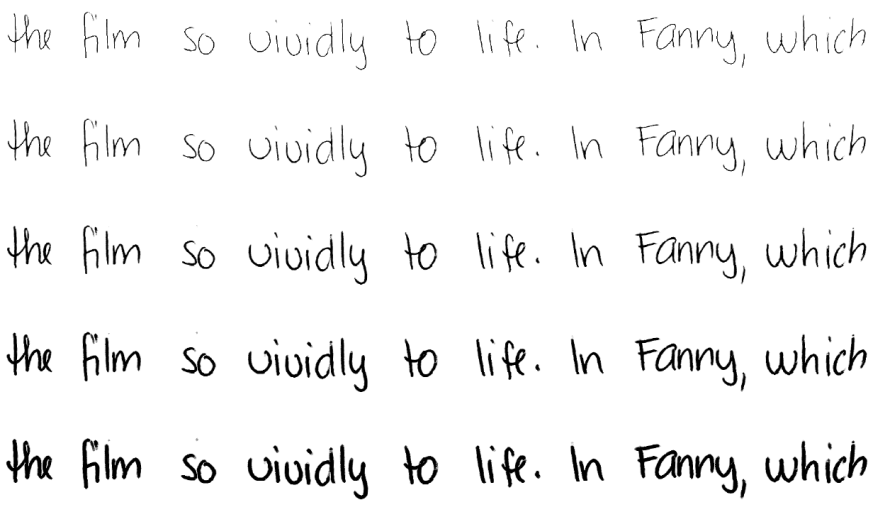


Figure 3.8: Illustration of thinning (above) and thickening (below) operations. The original text line is in the middle.

would not be safe when applied iteratively, because part of the original writing might be lost after too many steps of erosion. An illustration is given in Fig. 3.8, where the original text line is located in the middle, and above (below) it the results of two successive thinning (thickening) steps can be seen. The choice whether thinning or thickening is applied, as well as the number of steps (including zero) is randomly made.

3.1.4 Distorted Text Line Generation

Now that the main constituents of the perturbation model have been introduced, a simple scheme for the distortion of whole text lines can be designed. The steps of the perturbation method for distorting a given skew and slant corrected text line are the following:

1. Apply each of the line level transformations to the text line, one after the other, in the order given in Subsection 3.1.2.
2. For each individual connected component, apply the connected component level transformations, and make sure that the bounding boxes remain the same with respect to both size and location.
3. Apply thinning or thickening operations.

Of course, these steps are not required to be always rigorously followed. In particular, one can omit one or several of the transformations. The method is demonstrated in Fig. 3.9.

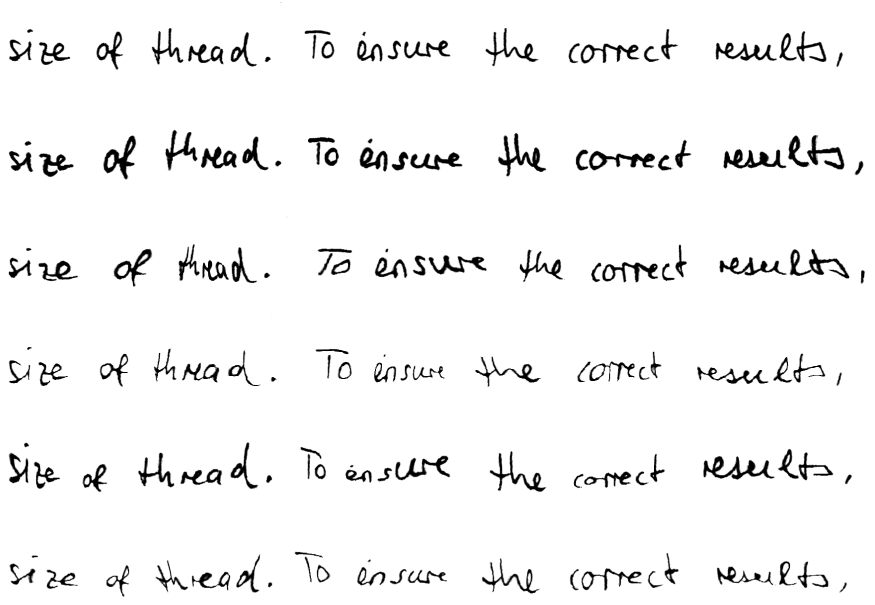


Figure 3.9: Demonstration of the perturbation method. The original human written text line is on top, and below it five distorted versions can be seen.

The original human written text line is on top, and below there are five synthetically generated versions of that line. It can be seen that all of the characters have somewhat changed in each generated line. Note that due to the random nature of the perturbation method, virtually all generated text lines are different. Other examples are given in Section 3.2.

3.1.5 Why these transformations?

In the field of handwritten character recognition, numerous different methods are reported to perturb character images. As for geometrical transformations, translation, scaling, rotation, shearing, shrinking, interpolation between character samples, and also nonlinear deformations were tried [16, 23, 69, 74], without claim to completeness. Other types of perturbations include erosion and dilation [16], and pixel inversion noise [69].

Although they seem to be very different approaches, surprisingly almost all of them have been applied successfully to generate additional training samples for character recognition systems, yielding improvements in the recognition performance.[3] Thus the character recognition experiments suggest that most of the reasonable (i.e. simple and smooth)

[3]The only exception is shrinking, which deteriorated the system performance in [16].

perturbations might improve the recognition rate. Furthermore, there is no comparative study showing that one or more of these approaches are superior to the others.

With this background, and considering that there are no similar results for the problem of handwritten text line recognition, the main goal was to design a perturbation model for handwritten text lines which is conceptually simple, transparent, and involves smooth transformations. Additional requirement was the nonlinearity of the applied transformations.

That's why only a few standard transformations were considered: shifting, scaling, and shearing, which were to be governed locally by the same type of nonlinear underlying function, to facilitate transparency. As underlying function, the *CosineWave* function meets the design goals: it is derived from one of the simplest possible nonlinear functions (i.e. the *cosine* function), by simple scaling and concatenation steps. Its derivative and integral are easy to calculate, and those functions are of similar type (phase-shifted *CosineWave*, or in other words *SineWave*). Its average value (integral divided by length) over the whole text line is expected to be zero, which means that the transformations are not expected to be partially reversed by normalization operations that detect and correct the average of some property, e.g. the average slant. Similar properties hold when the sum of *CosineWave* functions is considered. Additionally, thinning and thickening operations are also standard means of handwritten image manipulation.

3.2 Experimental Evaluation

The purpose of the experiments is to investigate whether the performance of the off-line handwritten text recognizer described in Chapter 2 can be improved by adding synthetically generated text lines to the training set. Three different configurations with respect to training set size and number of writers are examined: small training set with only a few writers, small training set with many writers, and large training set with many writers. Additionally, synthetic training set expansion will be compared with natural one, and the proportion of natural and synthetic training data is also addressed.

3.2.1 Background Information on the Experiments

Writer Independent Task

All the experiments are writer-independent, i.e. the population of writers who contributed to the training set is disjoint from those who produced the test set. This makes the task of the recognizer very hard, because the writing styles found in the training set can be

	number of text lines	number of writers	lexicon size	extended
Pool1	541	6	412	no
Pool2	1,993	400	6,012	yes
Pool3	200	200	11,897	yes

Table 3.1: Subsets of the IAM-Database used in the handwriting recognition experiments. The lexicon includes all the words that occur in the corresponding set of text lines, but it may also be extended by additional words.

totally different from those in the test set, especially if the training set was provided by only a few writers.

Nevertheless, the writer-independent scenario is assumed to be more relevant to the application of synthetic training data than that of the writer-dependent scenario, i.e. when the training and test sets contain the same writers. This is because a given training set is supposed to be less representative of the test set in the writer-independent case, so greater benefit can be expected from the additional synthetic training data in that case. Furthermore, commercial systems with a huge number of possible clients, i.e. those reading bank checks, postal addresses and filled forms, require writer-independent handwriting recognition engines.

Subsets of the IAM-Database

Three subsets of the extracted handwritten text lines of the IAM-Database are considered for the handwriting recognition experiments: a set of 541 lines produced by 6 writers only, a set of 1,993 lines from 400 different writers, and a set of 200 lines written by 200 persons. These sets are denoted by *Pool1*, *Pool2*, and *Pool3* in Table 3.1, respectively. The first two sets have 30 text lines in common (5 from each of the 6 writers of *Pool1*), while *Pool3* is disjoint from the other two with respect to both text lines and writers.

Each set has a corresponding lexicon, which is considered in the experiments related to that set, and contains all possible words that are allowed to occur in the text lines to recognize. All the words in each of the three sets of text lines are included in the corresponding lexicon. For *Pool2* and *Pool3*, the lexicon also contains additional words that are not present in the text lines of those sets, respectively. Furthermore, the lexicon of *Pool3* is a superset of the other two.[4] Obviously, the recognizer's task is easier if there is a smaller lexicon of allowed words.

[4]The lexicon of *Pool2* consists of all the words of those forms that have at least one line in *Pool2*, while the lexicon of *Pool3* includes all the words of the IAM-Database that were available at the beginning of the present research.

Synthetic Text Line Generation

Finally, some general notes about the synthetic text line generation are given. If not mentioned otherwise, all the three steps described in Subsection 3.1.4 are applied to distort a natural text line. Underlying functions are obtained by summing up two randomly generated *CosineWave* functions (two is the minimum number to achieve peaks with different amplitudes, see Figs. 3.1 and 3.2). Concerning thinning and thickening operations, there are only three possible events allowed: one step of thinning, one step of thickening, or zero steps (i.e. nothing happens), with zero steps having the maximal probability of the three, while the two other events are equally probable.

As it was mentioned in Subsection 2.4.2, the text lines in the training set are also normalized, including skew and slant correction, positioning, and width normalization (see Fig. 2.4). Since the text lines to be distorted have to be skew and slant corrected, synthetic training text line generation takes place right after the skew and the slant of the text line have been normalized. This means that only the two remaining normalization steps are performed on the distorted text lines, i.e. positioning and width normalization.

3.2.2 Small Training Set with a Small Number of Writers

The purpose of the experiments described in this subsection is to test the potential of the proposed method in relatively simple scenarios, i.e. the case of a small training set and only of few writers.[5] For the experiments, the subset *Pool1* of Table 3.1, containing 541 text lines from 6 different writers, was considered.[6] The underlying lexicon consisted of 412 different words. The six writers who produced the data used in the experiments will be denoted by *a*, *b*, *c*, *d*, *e* and *f* in the following. Subsets of writers will be represented by sequences of these letters. For example, *abc* stands for writers *a*, *b*, and *c*.

Three groups of experiments were conducted, in which the text lines of the training sets were distorted by applying three different subsets of the distortions described in Section 3.1. The three subsets were the set of *all distortions*, the set of geometrical transformations on the *line level*, and the set of *connected component level* geometrical transformations. In each case, five distorted text lines per given training text line were generated and added to the training set. So the extended training set was six times larger than the original one.

Fig. 3.10 shows examples of natural and synthetically generated pairs of text lines used in the experiments where all the distortions were applied. For each pair of text lines the

[5]The bigram language model was also simplified to the special case of $P(w_i) = P(w_i, w_j) = \frac{1}{N}$, where w_i and w_j denote arbitrary words of the underlying lexicon, and N is the size of the lexicon. Thus each word had the same probability to occur at any position in a text line. See also Section 2.5.

[6]Each writer produced approximately 90 text lines.

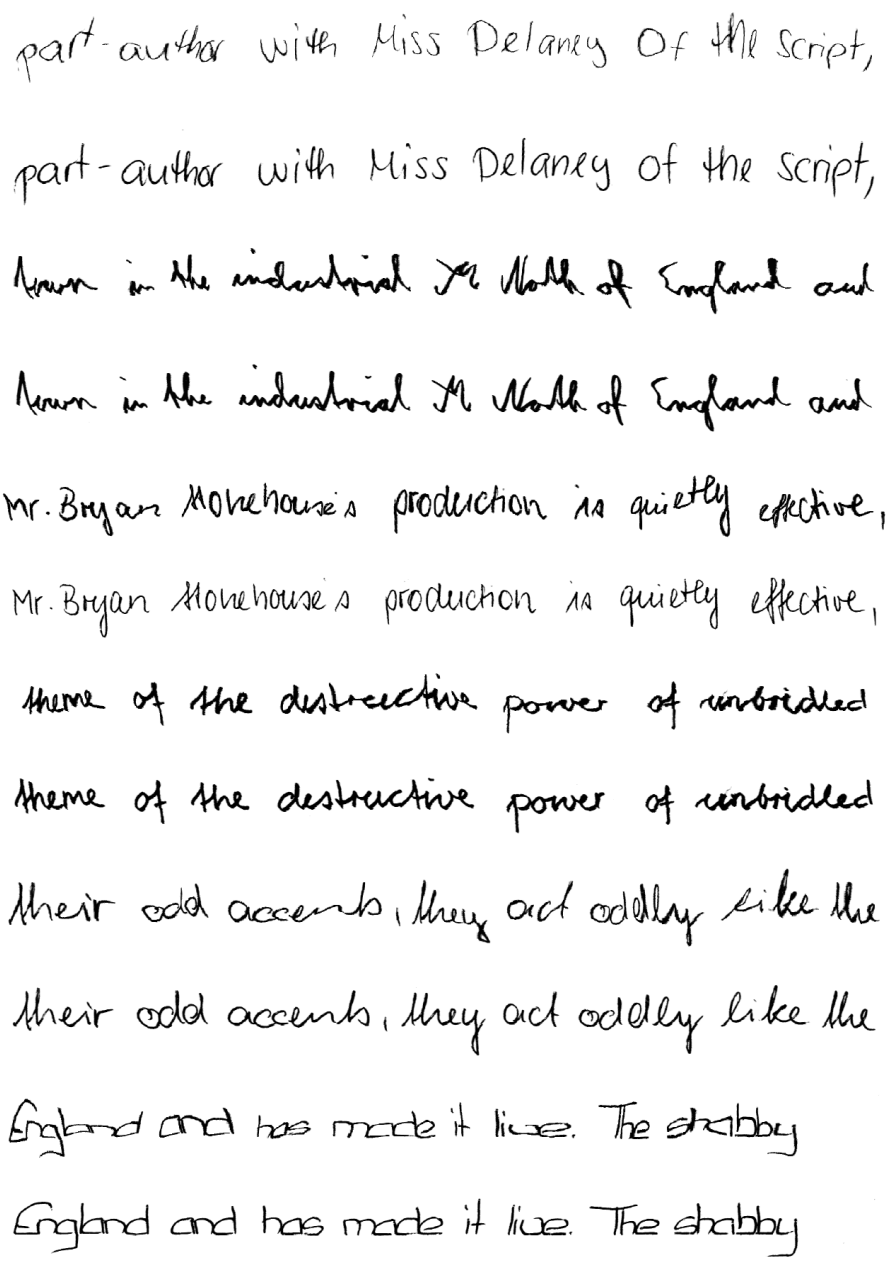


Figure 3.10: Natural (below) and synthetic (above) text lines for writers *a-f*.

natural one is shown below, while the synthetic one is above it. The first pair belongs to writer *a*, the second to writer *b*, and so on.

The recognition results of the three experiments are shown in Table 3.2, where the rows

	original	all dist.	line level	cc. level
a	33.14	48.98	47.06	38.69
b	38.68	43.07	40.41	42.61
c	39.16	49.31	46.80	44.41
d	30.56	53.14	48.62	43.02
e	54.40	59.61	58.88	54.24
f	18.83	31.98	26.90	27.76
ab	60.69	73.46	75.79	54.92
cd	56.84	61.30	62.44	59.66
ef	63.84	68.46	67.54	67.51
abc	75.19	74.11	75.78	74.83
def	65.35	68.87	67.04	68.74

Table 3.2: Results of the experiments described in Subsection 3.2.2 (in %).

correspond to the different training modalities. The test set is always the complement of the training set, and consists of natural text only. For example, the test set corresponding to the first row consists of all natural text lines written by writers *bcdef*, while the training set is given by all natural text lines produced by writer *a* plus five distorted instances of each natural text line. In the first column, the results achieved by the original system that uses only natural training data are given for the purpose of reference. The other columns contain the results of the three groups of experiments using expanded training sets, i.e. the results for all, line level, and connected component level distortions, respectively. In those three columns each number corresponds to the median recognition rate of three independent experimental runs. In each run a different recognition rate is usually obtained because of the random nature of the distortion procedure.

In Table 3.2 it can be observed that adding synthetic training data leads to an improvement of the recognition rate in 29 out of 33 cases. Some of the improvements are quite substantial, for example, the improvement from 33.14% to 48.98% in row *a*.

Augmenting the training set of a handwriting recognition system by synthetic data as proposed in this paper may have two adversarial effects on the recognition rate. First, adding synthetic data increases the variability of the training set, which may be beneficial when the original training set has a low variability, i.e. when it was produced by only one or a few writers. On the other hand, the distortions may produce unnatural looking words and characters, which may bias the recognizer in an undesired way, because the test set includes only natural handwriting.

The greatest increase in recognition performance can be observed in Table 3.2 for those cases when there is only one writer in the training set. Then the variability of the training

set is low and the addition of synthetic data leads to a better modeling of the test set. In this case, the application of all distortions outperforms the use of only line level or connected component level distortions. Where multiple writers are used for training, the variability of the training set is larger and the increase in recognition performance becomes smaller when synthetic training data is added. Also, in this case using all distortions does not always result in higher recognition rate than applying just line level or connected component level distortions.

Since in the majority of the experimental runs, an improvement of the recognition rate was observed, it can be concluded that the use of synthetic training data can potentially lead to improved handwriting recognition systems, in case of only a few writers in the training set.

In all experiments described in this subsection, single Gaussians were used in the HMMs' states to estimate observation probability distributions. As we will see in the following, the number of Gaussians should be increased if the training set contains handwriting samples from many writers.

3.2.3 Small Training Set with a Large Number of Writers

In these experiments, the case where many writers are represented in a small training set was considered. The subset *Pool2* of Table 3.1 was used for the experiments, so the size of the underlying lexicon was 6,012 words, which is considerably larger than that of the previous experiments. The effects of adding synthetic data to the training set were investigated in terms of three parameters.

The first parameter was the *number of Gaussians* to model the observation probability distributions in an HMM. It was shown in [31] that using six Gaussian mixture components for distribution estimation results in much better recognition rates than using only a single Gaussian. Hence the number of Gaussians was considered as an important parameter of the recognition system, and its influence on the effectiveness of synthetic training data was studied.

The second parameter to examine was the *distortion strength*, which can be controlled by changing the interval of the possible amplitude values for the underlying functions described in Section 3.1. Four levels of strength were defined based on a subjective assessment: *very weak*, *weak*, *middle* and *strong*. Note that these terms indicate only the relative order of the four levels, rather than absolute categories.[7] In Fig. 3.11, two examples are shown, where the text lines on top were distorted using all four different distortion strengths. For the distorted text line generation, all of the distortions were applied, in

[7] The strength was increased by *jointly* increasing the amplitude parameters for all the transformations. For thinning/thickening, the probability of zero steps was decreased.

	original	very weak	weak	middle	strong
Ga=1, Size=81	36.05	53.06	58.73	46.49	53.97
Ga=6, Size=81	**54.20**	62.81	**65.76**	63.72	62.81
Ga=1, Size=162	60.54	62.81	63.95	63.27	59.26
Ga=6, Size=162	**63.04**	70.29	**72.79**	70.29	67.80
Ga=1, Size=243	67.80	53.97	65.31	62.81	42.63
Ga=6, Size=243	**70.07**	71.20	**72.79**	71.88	*67.80*
Ga=1, Size=324	50.11	68.25	65.99	62.13	58.28
Ga=6, Size=324	**71.20**	73.47	**74.60**	*70.52*	*70.07*

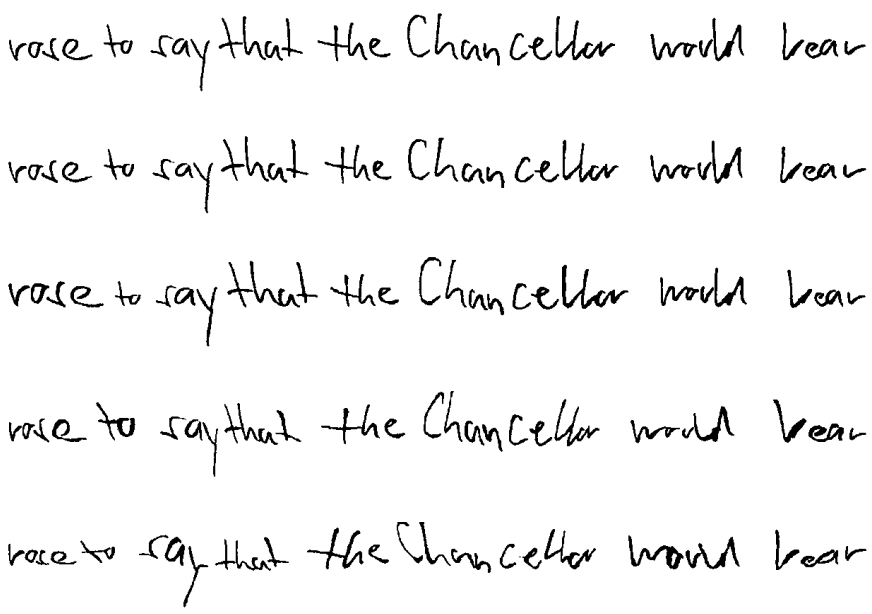
Table 3.3: Results of the experiments of Subsection 3.2.3 (in %). Various number of Gaussians, distortion strengths and training set sizes were used. The *weak* distortion strength at $Ga = 6$ performed the best, highlighted using boldface. Those results where distorted text lines deteriorated the recognition performance are printed in italic.

the way described in Subsection 3.1.4. A trade-off between quality and variability of the generated text lines can be observed, which is governed by the distortion strength. That is, stronger distortions usually introduce more variability, but on the other hand, the generated text lines tend to look less natural. Thus tuning the distortion strength is expected to be beneficial.

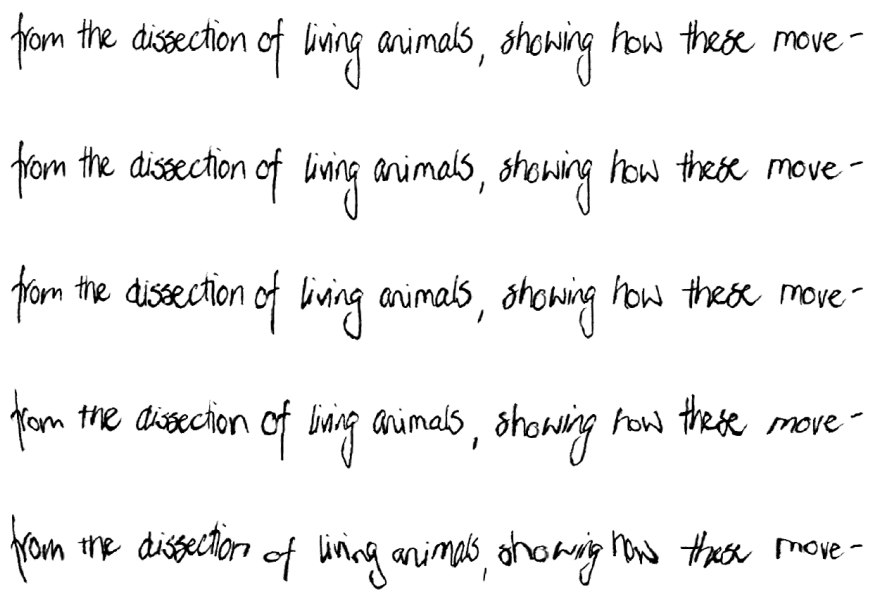
The third parameter to be investigated in the experiments was the *number of natural text lines* in the training set. Generally, the larger this subset is, the greater is its variability and the better is the recognition rate anticipated. Thus smaller improvements in the recognition rate are expected when adding synthetic data to a larger set of natural text lines.

To examine the effects of the three parameters mentioned above, 324 text lines from 219 writers, and 47 text lines from 27 writers were considered for training and testing, respectively. The writers contributing to the training set were different from those who provided the test set. Hence the experiment was again writer-independent. In order to study the influence of the third parameter mentioned in the last paragraph, four different training set sizes were examined: 81, 162, 243 and 324 text lines, where each set was a subset of its successor. The test set consisted always of the same 47 lines. In each case, five distorted text lines per given training text line were generated and added to the training set.[8] The recognition results are shown in Table 3.3. In column *original*, the results without addition of synthetic text lines can be seen, column *very weak* contains the results achieved by adding synthetic text lines of *very weak* strength, and so on. The results using a single Gaussian for distribution estimation are shown in rows with *Ga=1*,

[8] A discussion on the number of synthetically generated text lines follows in Subsection 3.2.6.



a)



b)

Figure 3.11: Illustration of levels of distortion strength used in the experiments of Subsection 3.2.3. From top to bottom, for both a) and b) parts: original, very weak, weak, middle and strong.

while *Ga=6* means that mixtures of six Gaussians were used.

As it can be seen, using mixtures of six Gaussians instead of single Gaussians always yielded (much) better recognition rates. Furthermore, using single Gaussians the recognition rate under the different distortion strategies exhibits a great degree of variation in both the positive and negative direction. By contrast, for six Gaussians, great changes occur only in the positive direction, especially for smaller training set sizes. A possible explanation of this phenomenon is that since only natural text lines are used for testing, unnatural looking synthetic text lines in the training set may cause serious damage in parameter estimation when single Gaussians are used. In the case of six Gaussians, outliers in the training set cause less damage, i.e. the recognizer is less biased towards the unnatural variability of the synthetic training data.

In the following the case of six Gaussians is discussed in greater detail. First of all, the results in Table 3.3 show that *weak* distortions perform best (those results together with their original counterparts are highlighted using boldface). It can also be observed that for training set sizes 81 and 162, all four distortion strengths yield improvements in the recognition rate, while at size 243 the strong, and at size 324 both the strong and the middle distortions produce negative results (printed in italic). This shows that, when adding synthetic data to larger training sets, the positive effect of increased variability of the training set becomes smaller, and the negative effect of training the recognizer on unnatural looking text lines can become dominant.

Since using six Gaussians and weak distortions has proved to be the most promising scenario, the experiments of the *weak* column in Table 3.3 were repeated three more times, using other mutually disjoint subsets of *Pool2*.[9] The sets were similar to those used in Table 3.3 with respect to size and number of writers. All three test sets contained 46 text lines from at least 28 writers. Table 3.4 shows the results. It can be seen that there were always improvements in the recognition rates. This observation confirms the results in Table 2. When using six Gaussians and weak distortions, an improvement in the recognition rate can be expected for small training set sizes. The larger the training set is, the smaller are the anticipated improvements.

3.2.4 Large Training Set with Many Writers

In the following, the case where there are many writers and a large training set is considered. For the experiments, the 1,993 text lines of *Pool2* of Table 3.1 were used. These text lines were produced by 400 different writers, and the underlying lexicon contained

[9]Disjoint in terms of text lines. However, the sets of writers in the four different experimental setups of this subsection (the one before and the three repetitions) may overlap, since their sets of text lines were sampled randomly from *Pool2*, in a way that only the writer independence of the corresponding training set-test set pairs was assured.

	Exp1		Exp2		Exp3	
	original	weak	original	weak	original	weak
Ga=6, Size=80	59.38	69.42	51.42	60.78	53.63	59.62
Ga=6, Size=160	67.86	74.55	59.26	64.49	62.61	64.96
Ga=6, Size=240	71.65	77.23	60.57	64.92	61.97	64.74
Ga=6, Size=320	75.45	78.57	61.66	63.62	62.82	65.60

Table 3.4: Results of the experiments of Subsection 3.2.3 (in %). Confirmation of the results reported in Table 3.3 on three different data sets.

6,012 different words. This set of text lines was randomly divided into *training*, *validation* and *test set*, such that their sets of writers were pairwise disjoint. The training and validation set contained 1,433 text lines from 288 writers, and 160 text lines from 32 writers, respectively. The test set contained 400 text lines from 80 writers.

First, the training and the validation set were used to find the optimal parameters for the system that uses natural training data only, and for the system that uses a mixture of natural and synthetic training data. In the following, these two optimized systems will be referred to as *Original System* and *Expanded System*, respectively.

The optimization was performed in terms of *capacity* and *distortion strength*. The capacity of the recognition system is defined as the number of free parameters to be estimated from the training set. It determines how much information the recognizer can store to express its knowledge about the handwriting represented by the training set.[10,11] A capacity too high may cause overfitting on the training data. On the other hand, a capacity too low may lead to a poor handwriting model. Since the synthetically expanded training set contains increased variability (both natural and unnatural), its optimal capacity is expected to be higher than the recognizer's optimal capacity for the original training set. That is, if the capacity of the system is not increased after the expansion of the training set, there is the danger that the capacity may be too low, such that the system is biased towards the unnatural variability introduced by the additional synthetic text lines, to such an extent which may cause the recognition performance to drop. In the experiments, the capacity was varied through changing the number of Gaussian mixture components used for estimating the feature value distributions in the states of the Hidden Markov Models. The number of Gaussian mixtures, Ga, is the same in all HMMs. If this parameter, Ga, is increased, then it enables the system to model the distributions of the features extracted from the handwriting more accurately. Thus the capacity of the system is increased.

[10] Apart from the already existing a-priori knowledge which is not considered in the present discussion, since it is left untouched by the training process.

[11] In the literature, there are other approaches that capture the concept of capacity by emphasizing the representational power of the system rather than the information storage aspect. See for example [99].

	original	very weak	weak	middle	strong
Ga=6	67.04	65.45	66.12	65.52	62.81
Ga=12	69.95	69.69	71.41	69.76	70.09
Ga=15	70.48	70.88	**72.27**	**71.54**	70.48
Ga=18	70.15	**72.20**	**72.47**	**72.40**	71.01
Ga=21	69.62	71.61	**72.40**	**72.01**	71.54
Ga=24	70.48	71.34	**73.00**	**73.33**	71.21
Ga=27	70.22	71.48	**72.87**	**73.86**	**71.67**
Ga=30	69.49	**71.67**	**72.14**	**73.20**	**71.74**

Table 3.5: Results of the optimization stage of the experiments of Subsection 3.2.4 (in %). Statistically significant improvements are highlighted using boldface.

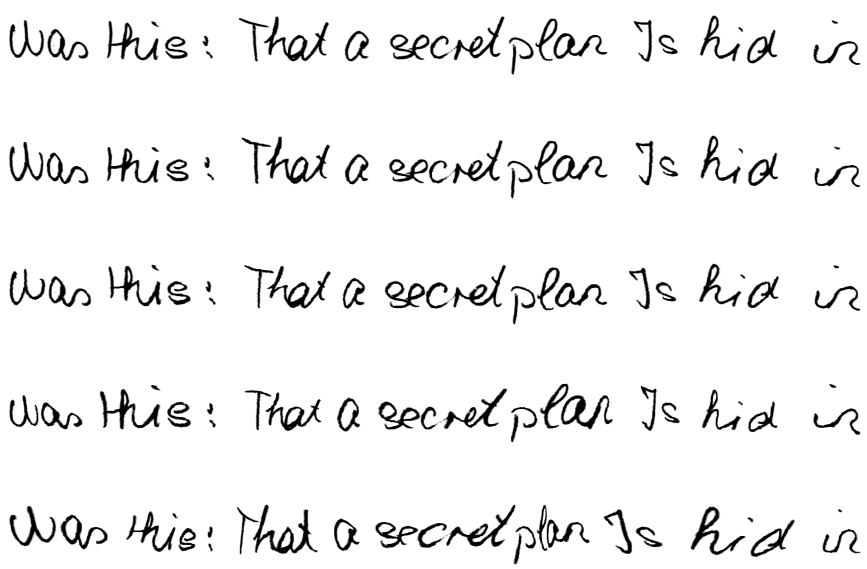
The generation of synthetic training data was similar to that of Subsection 3.2.3, but the distortion strengths denoted by *middle* and *strong* were slightly weaker.[12] Obviously this is a rather sparse sampling of the whole spectrum of possible distortion strengths. The reason is that the training of HMMs using multiple Gaussian mixture components and an expanded training set is an extremely time consuming process. So the available computational resources strictly limited the number of different distortion strengths that could have been considered. The risk of missing the optimal range of distortion strength was intended to be reduced by explicitly defining a few strength values that seemed reasonable for sampling based on subjective assessment.

Detailed results of the optimization stage are reported in Table 3.5. In the HMM training procedure, the training set, consisting of natural and synthetic training data, was used, while the recognition rates were measured on the validation set, which consisted of natural text lines only. Column *original* corresponds to the system using exclusively natural training data. According to the best result, the system with $Ga = 15$ is chosen as the *Original System*, which achieved a recognition rate of 70.48%. The other four columns, namely *very weak*, *weak*, *middle* and *strong*, show the recognition rates of the system using a mixture of natural and synthetic training data. For each text line in the training set, always five distorted text lines were generated, thus the expanded training set was always six times larger than the original one. Those results which correspond to statistically significant improvements with respect to the *Original System* (with a significance level higher than 90%), are highlighted using boldface.[13]

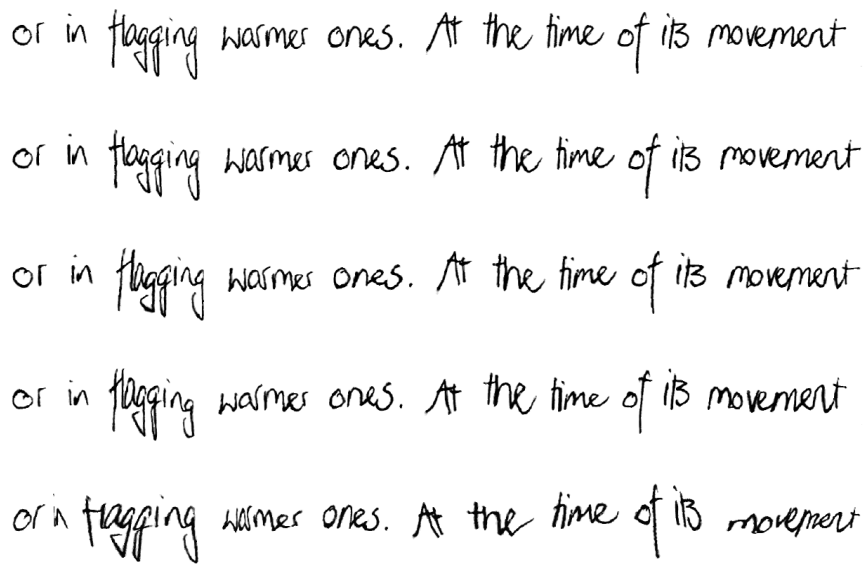
It can be seen that increasing the capacity is beneficial for expanded training sets. Rows

[12]This way the distortion strength was sampled in equal steps in terms of the parameter values of the perturbation model.

[13]The significance level of an improvement was calculated from the writer level recognition rates, by applying a statistical z-test for matched samples.



a)



b)

Figure 3.12: Illustration of levels of distortion strength used in the experiments of Subsection 3.2.4. From top to bottom, for both a) and b) parts: original, very weak, weak, middle and strong.

	Ga	strength	recognition rate
Original System	15	–	76.85%
Expanded System	27	middle	79.54%

Table 3.6: Results on the test set of the experiments of Subsection 3.2.4.

$Ga = 6$ and $Ga = 12$ show the effects of low capacity after training set expansion with synthetic data, resulting in lower recognition rates in the majority of the cases. With an increasing strength of the distortions, the optimal capacities become higher: from column *original* to column *strong* the optimal Ga's were 15, 18, 24, 27 and 30, respectively. This can be explained by the increasing variability of the training set. (Note that for strength *strong*, the optimal capacity is possibly above $Ga = 30$.) The most significant improvements came at strengths *weak* and *middle*. All significant improvements in these columns have a significance level greater than 95%. The most significant area is at strength *middle*, from $Ga = 24$ to $Ga = 30$. Here the significance level is greater than 99%. Thus the *Expanded System* was chosen among these, namely the one with $Ga = 27$, where the recognition rate was 73.86%.

After the optimization stage, the *Original System* was trained on the union of the training and validation set, and the *Expanded System* on the union of the expanded training and expanded validation set. For each natural text line in the validation set, five synthetic text lines were generated at strength *middle* to get the expanded validation set. Then, using the test set for testing on previously unseen examples, the recognition results of the *Original System* and the *Expanded System* were 76.85% and 79.54%, respectively, as shown in Table 3.6. This shows that using synthetic text lines, the recognition performance could be improved by more than 2.5%. The significance level of this improvement is greater than 99%. (The recognition rates on the test set differ a lot from those measured on the validation set. This can be explained by the relatively small size of the validation set. The magnitude of the validation set is limited by the amount of text lines in the training set, so that the training set has approximately the same optimal capacity as its union with the validation set. This way the negative effects of too low capacity can be avoided at the testing phase. But, as it was mentioned before, the choice of the training set size is also constrained by the computational complexity of the training process.)

However, one might note that the samples of the test set were not unseen in the literal sense, since *Pool2* was also used in the experiments of Subsection 3.2.3. But on the other hand, the methodology introduced in Subsection 3.2.3, that is, exploring different configurations of distortion strength and number of Gaussians, has not changed at all. So there was no change made based on such experiments in which the test set was anyhow involved, i.e. there was no optimization on the test set.[14]

[14]Concerning the experiments of Subsection 3.2.2, the test set used here is disjoint from *Pool1*.

	Ga	strength	recognition rate
Original System	15	–	75.05%
Expanded System	27	middle	76.41%

Table 3.7: Recognition results on the 200 text lines of *Pool3*, using the *Original System* and *Expanded System* of Subsection 3.2.4.

Moreover, the methodology can be considered *robust*, since it consists of such optimizations that must always be done, independently of the underlying datasets:

- The most appropriate distortion strength between zero and extremely strong can only be found empirically, because it may depend on the details of the recognizer under consideration, as well as on the concrete dataset.[15]
- Finding the optimal the number of Gaussians (or more generally, the optimal capacity) is a must in a multi-Gaussian system, because it is dependent on the characteristics of the training set. The same optimization is needed for the synthetically expanded training set, in order to have a fair comparison with the original system. (One purpose of this subsection was to illustrate why this latter optimization should not be overlooked.)

Nevertheless, the testing of the *Original System* and the *Expanded System* was repeated using the text lines of *Pool3* as test set. None of those lines were considered in the previous experiments. Also the larger lexicon of *Pool3* was used. Such change of the lexicon is straightforward, due to the use of character level HMMs. The results can be seen in Table 3.7. The HMMs trained on synthetically expanded training set performed better, with a significance level greater than 95%.

Thus, the experiments show that expansion of the available set of text lines by synthetically generated instances makes it possible to significantly improve the recognition performance of a handwritten text line recognizer, even when the original training set is large and contains handwriting from many writers.

3.2.5 Comparing Natural and Synthetic Expansion

So far it has been shown that it is possible to improve the recognition performance by using synthetically expanded training sets. In this subsection, the improvements achieved

[15]It would be more robust to optimize on the space of all possible parameterizations of the perturbation method, but it is not feasible because the dimensionality of the search space would be intractably large. Instead, the strength-related parameters are increased/decreased jointly.

with additional synthetic training data are compared to those achieved by expanding the training set using natural, i.e. human written, text lines.

To examine the system performance as a function of the training set size, four different training set sizes were considered: 160, 320, 638 and 1,275 text lines from *Pool2*, where each smaller set was a subset of all larger sets. This means that at each training set expansion, i.e. when going from one set to the next larger one, the size was approximately doubled. Furthermore, at each expansion, the additional text lines came from writers who had not yet been represented in the training set. The numbers of writers in the four training sets were 32, 64, 128 and 256, respectively. (So the number of writers was also doubled at each training set expansion.) To evaluate the system performance, a test set of 398 text lines from *Pool2*, written by 80 persons was used. The creation of the training and test sets was made by randomly sampling the corresponding number of writers from *Pool2*. All the experiments were writer-independent, i.e. the population of writers who contributed to the training sets were disjoint from those who produced the test set.

For distorted text line generation, the distortion strength *middle* of Subsection 3.2.4 was used. To expand a training set by synthetically generated text lines, five distorted text lines per given natural training text line were generated and added to the training set. The synthetic text lines were not generated separately for each training set size, but only once for the largest training set of 1,275 text lines, because this way the same subset relation holds for the synthetically expanded training sets as for their natural counterparts.

In the experiments, the highest possible recognition performance achieved with natural data was compared to that obtained with a mixture of natural and synthetic data. For this purpose, optimization in terms of capacity was performed. This means that for each training set, the number of Gaussian mixture components, Ga, for which the recognition rate was maximal, was selected. In other words, the question was how efficiently the recognizer can use the different training sets to model a given test set.

In Table 3.8, the recognition results on the test set for the four different training set sizes as well as their synthetically expanded counterparts can be seen. In each row, results for a specific training set size are shown. For example, in row *Size=160* it can be seen that the optimal recognition rate using the training set of 160 text lines was 62.86%, at $Ga = 6$. Furthermore, when this training set of 160 natural text lines was expanded by synthetically generated text lines (which means in this case a total of $160 \cdot 6 = 960$ text lines), the optimal recognition rate was 70.58%, at $Ga = 24$.

This recognition rate of 70.58% is comparable to that 70.44% we could achieve using the training set of 638 natural text lines (see row *Size=638*). So the synthetic expansion of 160 training text lines had a similar effect as if the number of natural text lines in the training set had been increased by a factor of four. The improvement from 62.86% to 70.58%, achieved by adding synthetic text lines to the training set is quite substantial. For training set sizes of 320 and 638 natural text lines, synthetic expansions also yielded substantial

	Natural text only		Synth. expanded set	
	Rec. rate	Opt. *Ga*	Rec. rate	Opt. *Ga*
Size=160	62.86	6	70.58	24
Size=320	68.59	9	73.05	18
Size=638	70.44	9	74.66	24
Size=1275	73.96	18	75.98	27

Table 3.8: Comparison of the best recognition rates (in %) on the test set achieved by natural and synthetically expanded training sets of different sizes.

improvements, from 68.59% to 73.05% and from 70.44% to 74.66%, respectively. Observe that these improvements are higher than those achieved by doubling the size of the training set using additional natural text lines. For size 1,275, synthetic expansion also improved the recognition rate, but there was no larger natural training set in our experiments to which this improvement could be compared.

Since the optimal number of Gaussians was always greater for the synthetically expanded training set than for its natural counterpart, the results also confirmed that increasing capacity has a beneficial effect when augmenting the training set by synthetic data. Furthermore, the recognition rates in Table 3.8 suggest that in terms of recognition performance, the acquisition of a remarkable amount of new natural text lines can be substituted by generating synthetic text lines from the available ones.

3.2.6 Proportion of Natural and Synthetic Training Data

In the previous experiments, the number of distorted text lines to be generated for each natural text line of the training set was set to five. This number was chosen due to the following reasons:

- In general, it was expected that increasing the number of synthetic text lines yields improved recognition rates.[16]
- Because of time complexity issues of the training process, it was not feasible to generate arbitrarily many synthetic text lines.
- 5 generated text lines per each natural training text line was thought to be a reasonable compromise, with still acceptable additional cost of the training time.

In the following, the behavior of the recognizer is studied in the case when the proportion of the synthetic text lines increases in the training set. A small subset of *Pool2* was con-

[16] On the other hand, if the distortions applied are too strong, inverse effects can be expected.

	orig.	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10
Ga=6	**70.81**	71.82	70.54	69.31	71.08	70.40	70.40	71.76	71.15	72.37	71.49
Ga=9	70.74	**73.79**	73.18	72.30	74.81	74.00	74.61	74.13	74.95	74.20	73.93
Ga=12	70.06	72.84	**74.13**	74.13	74.75	73.66	74.61	75.63	**76.44**	74.81	75.56
Ga=15	68.70	71.49	73.73	73.93	75.22	74.47	75.76	75.83	75.83	76.04	75.90
Ga=18	-	72.84	73.39	**74.88**	75.08	75.15	75.42	76.44	75.76	**76.37**	75.97
Ga=21	-	-	72.91	74.75	**75.42**	**75.36**	**76.58**	76.17	75.56	75.97	76.58
Ga=24	-	-	-	73.79	75.08	74.13	76.17	76.31	75.22	76.31	**77.12**
Ga=27	-	-	-	-	74.61	73.86	76.31	**76.92**	75.49	76.24	76.99

Table 3.9: Comparing recognition rates (in %) on the test set using different proportions of natural and synthetic training data. The number of synthetic text lines per each natural one varies from 0 to 10. In each column, the best recognition result is indicated by boldface.

sidered for the experiments. 320 text lines from 64 writers were used for training, and 10 distorted text lines were generated for each of them at *middle* strength of Subsection 3.2.4. The test set consisted of 160 text lines from 32 writers (disjoint from those of the training set). Optimization in terms of the number of Gaussians was performed.

The results are reported in Table 3.9. In the first column, the results of the original, i.e. natural, training set are shown. In the next column denoted by *+1*, the first synthetically generated text lines were added to the training set, resulting in a total size of $320 + 320 = 640$ lines. In column *+2*, the first two synthetic lines were added, and so on. The maximal recognition rates are highlighted in boldface.[17]

It can be observed that both the maximal recognition rate as well as the corresponding optimal capacity tend to increase with the number of additional synthetic text lines. There is a sudden drop in the optimal capacity at columns *+8* and *+9*. A possible explanation is that from 7 or 8 additional synthetic text lines the optimal capacity may be above 27 Gaussians. The highest recognition rate, 77.12% was achieved when all the 10 synthetic lines were added to the natural training set. For comparison, the natural training set was expanded using human written data, such that it included 1,435 text lines from 288 writers, and it was found that its optimal recognition rate on the test set was 77.19%, at $Ga = 18$.

Although the results seem to confirm the expectation that increasing the number of synthetic text lines yields improved recognition rate, two additional notes need to be made about the optimal number of synthetic text lines:

- The optimal proportion of natural and synthetic training data is very likely to be dependent also on the distortion strength applied. For example, if extremely strong

[17]For 3 Gaussians, i.e. for $Ga = 3$, the original training set achieves only 63.34%.

distortions are applied, the recognition rate is expected to drop after adding even a very little amount of such unnatural data to the natural training set.

- It may also not be useful to generate a huge amount of synthetic data, because each synthetic text line is strongly correlated with the corresponding human written one. For example, the topology of the characters, and respectively the intrinsic font of the writing, cannot be changed by the perturbation method.

3.3 Discussion: Capacity and Saturation

The main goal of synthetic training set expansion was to improve the recognition performance, by adding synthetic text lines to the original, i.e. human written, training set. With respect to this goal, the most important observation of the experiments presented in Section 3.2 can be summarized as follows:

Observation: the larger the original training set is, the more Gaussians are needed so that the synthetic training set expansion can improve the recognition rate.

To further examine this phenomenon, an experiment was conducted using gradually increasing training sets with increasing number of writers, while keeping the test set as well as the number of Gaussian components (i.e. the capacity) fixed. The natural training and validation set defined in Subsection 3.2.4 was used for training and testing, respectively. The numbers of Gaussians considered were 1 and 6. The two corresponding curves of recognition rates are shown in Fig. 3.13, where different proportions of the training set were used for training, while the test set was always the same. The percentages on the horizontal axis are to be understood with respect to the union of the training set and the validation set (the union consists of $1433 + 160 = 1,593$ text lines).

Based on these curves, two statements can be made:

- For 1 Gaussian, we cannot expect further improvements above approximately 20% of $1,593 \approx 320$ training text lines. Nevertheless, there can be some insignificant random fluctuation of the recognition rate for larger training set sizes.

- For 6 Gaussians, we cannot expect further improvements above approximately 50% of $1,593 \approx 800$ training text lines, except for the small fluctuation just mentioned.

This leads to the intuitive notion of *saturation point*, which is not an exact definition, but rather a thinking model, or practical experience:

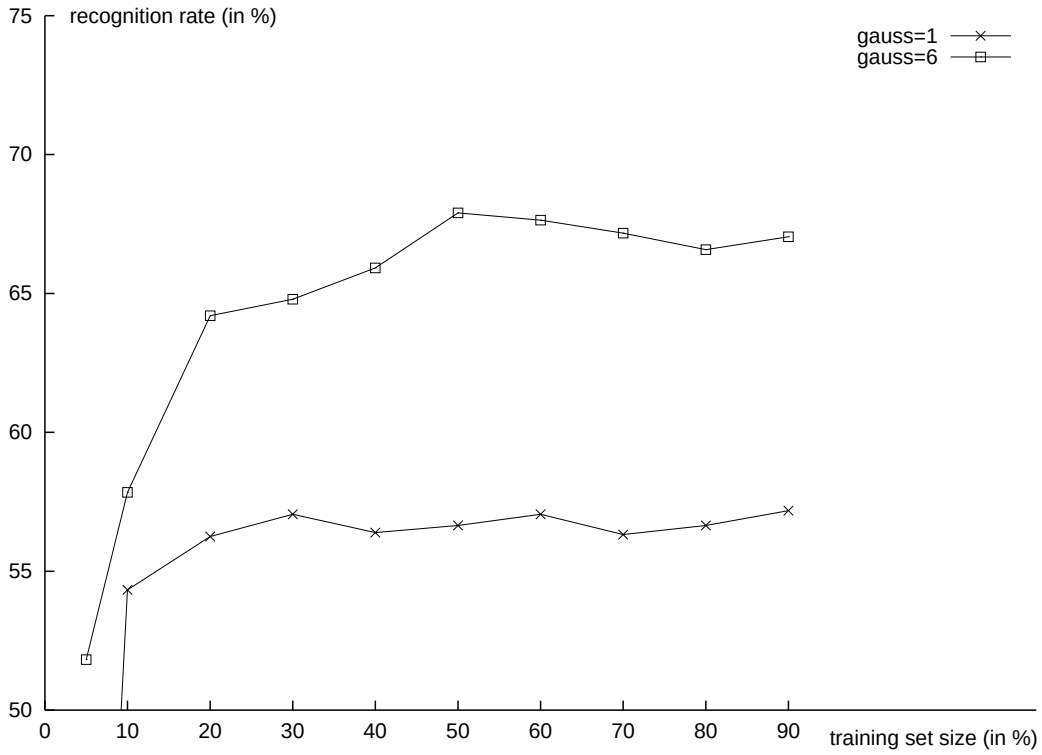


Figure 3.13: Recognition rates on the test set using increasing training set sizes and fixed capacity of the recognizer.

Saturation point: the minimal amount of natural training data above which further improvements in the recognition rate cannot be expected, given some fixed properties of the recognition system.

Comparing 1 Gaussian to 6 Gaussians, in the latter case the recognizer can store more detailed knowledge about the handwriting ink, and thus higher recognition performance is anticipated.[18] However, for a reliable estimation of finer details more training data is needed. Until there is insufficient amount of training data, the estimations become better and better as the size of the training set increases, resulting in an increasing recognition rate, too. Once the estimations of the free parameters become stable, i.e. at the saturation point, no further improvements of the recognition rate can be expected.

A possible explanation of the behavior beyond the saturation point can be, for example, that for those limited training set sizes practically available, the values of the free

[18]This is meant asymptotically, with an increasing amount of training data. For a fixed training set size, it is possible that more Gaussians perform worse, as a result of overfitting.

parameters exhibit signs of fluctuation rather than convergence.[19] Or alternatively, one can assume converging behavior, but in a way that the asymptotical recognition rate is approached from both sides, below *and* above. Nevertheless, both explanations imply that in practice it cannot be predicted beyond the saturation point whether increasing the training set size yields improved or deteriorated recognition performance.

Apparently, if the amount of natural training data is near or above the saturation point, we cannot expect any positive change in the recognition rate through the expansion with synthetic data either, since even additional natural data does not help.[20] To the contrary, the negative effect of unnaturality inherent in the synthetic data can become dominant, causing the recognition rate to drop. So the main conclusion of the experiment, concerning synthetic training set expansion, can be expressed in the following way:

Conclusion: near or above the saturation point, additional synthetic training data does not help, but rather it can deteriorate the recognition performance.

As an illustration for 1 Gaussian, it can be observed in Table 3.3 that among all synthetic expansions at *Size=243* and *Size=324*, only the very weak strength at *Size=324* performed slightly better (68.25%) than the natural training set of *Size=243* (67.80%).[21] For 6 Gaussians, in Table 3.5 the recognition rate dropped because the system was already saturated (note that the same data was used here to illustrate saturation).

Based on the above discussion, one might think it is easy to overcome the problem of saturation: if the recognizer is saturated given a natural training set, all we have to do is to increase the number of Gaussians, since this way the saturation point is shifted, thus we open up room for further improvement.

According to the author's opinion, this might not be the case. To illustrate the idea of the argument, in Fig. 3.14 some of the most important assumptions about handwriting made by the recognizer are shown inside the ellipses, organized in a hierarchical structure. For example, such an assumption is that the pixel column based feature vector sequence representation of a text line is sufficient for recognition, or that HMMs are appropriate models of the feature vector sequences. The arrows indicate that an assumption is further specified by another one, e.g. the arrow between *HMMs* and *linear topology* means that the possible HMMs used are restricted to that of linear type.

In the current system, the assumptions enclosed by dashed ellipses are adjustable, usually

[19] The training procedure may itself be sensitive to the initial conditions, i.e. a small change in the training set may cause considerable changes in the free parameter estimations.

[20] Assuming that natural data is more appropriate than synthetic data for the estimation of details of natural handwriting.

[21] The unusually low recognition rate of 50.11% at *Size=324* is due to fact that a single Gaussian can be the most easily biased towards such variabilities in the training set that are less representative of the test set, especially when the training set and the test set are *both* small.

Figure 3.14: Hierarchical representation of some important assumptions made by the recognition system used in the thesis. The arrows indicate specializations, and the dashed ellipses denote such assumptions that are adjustable, e.g. by training.

results of an optimization process like training or validation, while the other assumptions are fixed (i.e. they represent a-priori assumptions/knowledge). Obviously, the fixed assumptions set an upper bound on the recognition performance (in the ideal case, it is the perfect recognition[22]), which is independent of the other, lower-level adjustable assumptions of the hierarchy.

Since there are several other properties/assumptions of the system besides the *number of components* (see Fig. 3.14), it is conceivable that fixing some of them (particularly, those inside the solid ellipses in Fig. 3.14) might also set an upper bound on the recognition performance lower than the perfect recognition, even if the number of Gaussians is allowed

[22]For simplicity, the intrinsic error of the recognition problem is neglected, i.e. it is assumed that perfect recognition is possible.

to be optimized. And when the system performance gets close to that upper bound, saturation is experienced, due to the reasons mentioned before.[23]

More generally, it can occur that when the available natural training set is large enough, the saturation point can only be shifted (and thus room for further improvement opened) if some of the higher-level fixed assumptions are made adjustable, or changed either thoroughly (e.g. using Time Delay Neural Networks instead of HMMs) or only in their fixed parameterizations (e.g. using 20 states instead of 14 states). Such modification affects the hierarchy under the corresponding assumptions, too. As an example, if in spite of the enlarged training set, increasing the number of Gaussian components does not help in achieving better recognition performance, one can try to change the *linear topology* assumption to that of *left-to-right topology*[24] assumption, by which the number of free parameters to estimate from the training data increases, so it might help in utilizing the available amount of training data better. In other words, when changing the number of free parameters of the system, not only the quantitative but also the qualitative aspects of the new free parameters must be taken into account, so that the desirable effect can be achieved.

Finally, to summarize the discussion, a three-step strategy for handling saturation, i.e. for shifting the saturation point, is given:

> **Step 1:** Increase the capacity related to the adjustable assumptions.
>
> **Step 2:** If Step 1 does not help, or cannot be applied, make a fixed assumption (or more) adjustable, or change its fixed parameterization such that the capacity of the hierarchy under it increases.
>
> **Step 3:** If Step 2 fails, try to make thorough changes in some of the fixed assumptions.

As it can be seen from the experiments described in Section 3.2, first Step 2 was applied to resolve the single-Gaussian assumption of Subsection 3.2.2,[25] and then Step 1 proved to be sufficient in Subsection 3.2.4, using the multi-Gaussian assumption, to overcome the saturation problem. Steps 1 and 2 are very important, because not considering them, i.e. leaving the recognition system unchanged after the synthetic training set expansion,

[23] Unfortunately, at present time the available resources are not sufficient to experimentally establish such an upper bound of recognition performance for the current system.

[24] The left-to-right topology is an extension of the linear topology, because the transition from a given state s_i is allowed not only to itself and the next state s_{i+1}, but to any other state s_j where $j \geq i$. See also Subsection 2.4.1.

[25] Actually, Step 2 was applied twice: first to change to a fixed number of 6 Gaussians, and then to make the number of Gaussians adjustable. Also note that Step 1 could not be applied when the number of Gaussians was fixed.

may lead to the incorrect conclusion that synthetic data is not useful for the recognizer, given the available amount of natural training data.

However, it is still an open problem whether Step 3 will ever have to be applied, or from what amount of natural training data, if any, will all the three steps fail, i.e. from what amount of natural training data will the synthetic expansion become useless, *independently* of the types of improvements made (in this case it is also interesting to examine whether additional *natural* training data helps).

3.4 Summary and Conclusions

A method for training set expansion by generating randomly perturbed versions of natural text lines rendered by human writers was presented and evaluated under several experimental conditions in writer-independent experiments. It was demonstrated that using such expanded training sets, improvements in the recognition rate can be achieved rather easily when the original training set is small and contains handwriting from only a limited number of writers. In the second experiment, where there was a large number of writers and a small training set, the applied distortion strength and the number of Gaussian mixture components needed to be adjusted to get an improvement over the original system. It turned out that if the number of Gaussians is increased, the unnatural looking synthetic text lines present in the expanded training set cause less damage in the parameter estimation during the training phase, and the positive effect of increasing the variability of the training set becomes dominant. Then it was shown that significant improvement in the recognition rate is possible to achieve even in the case of a large training set provided by many writers. In this case, the applied distortion strength needs to be adjusted, and the capacity of the recognizer (i.e. the number of Gaussians used for distribution estimations) plays an important role. The capacity has to be optimized after training set expansion, because the optimal capacity of the recognition system trained on the expanded training set is expected to be higher than the optimal capacity of the system trained on the original training set. If the capacity is not properly adjusted when using the synthetically expanded training set, there is the danger that the capacity may become too low, such that the system is biased towards unnatural handwriting styles in an undesired way, causing the recognition performance to drop.

Synthetic and natural training set expansions were also compared, and the results of the experiments suggest that in terms of recognition performance, the acquisition of a remarkable amount of new natural text lines can be substituted by generating synthetic text lines from the available natural ones.

The proportion of natural and synthetic training data was also addressed, and it was found that increasing the number of synthetically generated text lines yielded improvements in

the recognition rate. On the other hand, it is reasonable to assume that if extremely strong distortions are used, the opposite outcome can be expected.

Finally, the empirical observations of the experiments were discussed, based on the intuitive concept of saturation. The most important point is that besides capacity, also the saturation has to be taken into account, because neither synthetic nor natural training set expansion can improve the recognition rate when the recognition system is already saturated by the available amount of natural training data. However, when the system is far from being saturated by the available natural training data, substantial improvements can be achieved by synthetic training set expansion.

3.5 Future Work

Since the problem of synthetic training data was addressed from a rather general point of view in the experiments, many questions mostly related to the enhancement of the baseline perturbation method are still open:

- Extending the perturbation method to explicitly model spacing variations (although it is partially done implicitly by shearing and horizontal scaling).
- Considering other types of distortions as well as underlying functions.
- Examining the suitability of the individual distortions. Such a test would be extremely time consuming, keeping in mind that a distortion might be more useful in an ensemble rather than alone.
- Applying not only one but several distortion strengths when expanding the training set.
- Performing the capacity adjustment on a much finer level, i.e. for each character HMM separately. This can become important when the capacity has to be seriously increased, particularly in the case of synthetic training set expansion. The reason is that, for example, 27 Gaussians might be optimal for those characters that occur frequently in the training set, but there is a danger of overfitting e.g. for most of the capital letters.
- Not adding all the generated texts to the natural training set, but detecting and excluding "badly distorted" ones, by an appropriate rejection mechanism. For example, if a word recognition task is considered, those distorted word instances that are not correctly recognized by the *original* recognizer would be regarded as too unnatural and thus would be rejected. But also the HMM score of the original and

the distorted word, respectively, could be compared to determine the quality of the distorted one.

- Using style dependent distortions as well as distortion strengths to facilitate the creation of expanded training sets of better quality.
- Applying the distortions in the testing phase, to make the recognizer insensitive to small distortions of the text to be recognized. The idea is that several slightly distorted images are generated, and their recognition results are combined, e.g. by using standard combination schemes from the field of multiple classifier systems.
- Comparing the improvements achieved in the writer-independent case with those of the writer-dependent case (i.e. when the training set and the test set contain the same writers). It can be expected that there are also substantial improvements for the writer-dependent scenario if the original training set is small. Furthermore, synthetic texts might facilitate the adaptation of the recognizer to a specific writer or group of writers, while considerably reducing the amount of natural training texts that need to be collected for the adaptation.
- Investigating the merits of the proposed perturbation method using other types of recognizers, for example, nearest neighbor classifiers or neural networks, considering the task of character or word recognition.
- Extensively testing the recognizer's performance on differently distorted texts that are still readable by humans.
- Building a handwriting-based CAPTCHA based on the perturbation model.

Chapter 4

Template-based Synthetic Handwriting Generation

Generating synthetic text lines from existing human written text lines has several limitations. First of all, the generated lines will be strongly correlated with the corresponding original lines. Furthermore, we don't have the possibility of creating new styles of characters, only to distort existing ones, which is likely to introduce unnaturality since the same kinds of perturbations are applied at any part of the text line, regardless of its actual content. And finally, natural handwriting samples are needed to be at disposal to generate synthetic handwriting.

To address these limitations, especially the last one, a method for synthesizing English handwritten text lines from ASCII transcriptions is presented in this chapter. The method includes perturbations, too, but it also has a finer control over the generation process, all down to the stroke level.

The method is based on templates (or prototypes) of handwritten characters built manually using Bézier splines [37]. As a first step to generate a text line image corresponding to a given ASCII transcription, the appropriate series of character templates are perturbed and concatenated. The resulting static image is then decomposed into strokes of straight segments and circular arcs [58]. These strokes are then randomly expanded and overlapped in time. For each stroke, a delta-lognormal curvilinear velocity profile is generated, and the skeleton image of the text line is drawn, followed by grayscale thickening operations to make the generated script look more natural.

The delta-lognormal velocity profiles as well as the decomposition into straight and circular strokes are taken from the Delta LogNormal theory of handwriting generation [85]. The advantage of using a handwriting generation model is that the variations resulting from perturbing its parameters may better reflect psychophysical phenomena of human handwriting than applying geometrical ad-hoc distortions on a static image. Additionally,

this way we can get on-line data, not only static images of texts.

The text lines generated by the proposed method are used to train the Hidden Markov Model (HMM) based off-line handwritten text line recognizer described in Chapter 2. The aim is to examine how the synthesized training sets compare to the natural ones, in terms of the recognition rate.

The approach followed here is similar to that of [57], where Korean characters are synthesized using templates of characters, and a variant of the Delta LogNormal model adapted to the characteristics of Korean handwriting. However, there are two major differences: in the present work, cursive English text lines are synthesized,[1] and the synthesized texts are used as training data for a handwriting recognition system.

The work presented in the following was a joint work with Daniel Kilchhofer, who has contributed the majority of the implementation of the method, as well as the application of Bézier splines for character template creation and concatenation. In addition to this chapter, a description of the method can be found in [49].

4.1 Character Templates and their Concatenation

The basis of the proposed handwriting generation method are the character templates that represent the ideal shapes of the different letters. Such templates were built for the upper and lower case characters of the English alphabet, for the ten digits, and for many special symbols including punctuations. Each prototype is a Bézier spline [37], which was put together manually from a series of Bézier arcs, one after the other, in a predefined writing order. An example is shown in Fig. 4.1, where a template for the lower case letter 'a' can be seen. The template consists of 5 Bézier arcs. In general, the start and end point of the letter, the points where the tangent is horizontal or vertical, and the corner points where the tangent is ambiguous, separate the Bézier arc segments of the template from each other. That is, those points correspond to the start and end points of the individual Bézier arcs.

Two versions of the English alphabet prototypes have been used: block and cursive style. They are shown on the top and in the middle part of Fig. 4.2, respectively, while the prototypes for the special symbols can be seen at the bottom in the same figure. Cursive writing style means that a character is linked to the preceding one. The linking of two consecutive characters is usually done by removing the last and the first segment of the first and the second character, respectively, and connecting the new end point of the first character to the new start point of the second one, using a Bézier arc. An example of

[1] Oriental and Western scripts have several different characteristics, e.g. Oriental characters consists of many disconnected line segments, and thus the pen must be lifted more often during the course of writing.

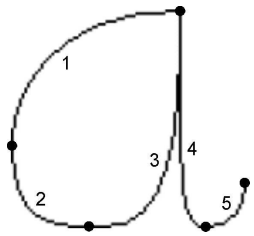


Figure 4.1: Template of letter 'a', consisting of 5 consecutive Bézier arcs.

Figure 4.2: Prototypes for the block style and cursive style English alphabet (above and in the middle, respectively), and for the special symbols including punctuations (below).

linking letter 'c' to 'e' is shown in Fig. 4.3.

Now suppose that there is an ASCII transcription of a text line given. The first step towards the generation of a corresponding image of that text line is that the ASCII character sequence (including spaces) is transformed into a sequence of template characters using the available prototypes. If a character class has more than one prototype available (e.g. block and cursive style), one of them is chosen according to a random parameter.

Next, each template in the series undergoes some geometrical perturbations, including scaling, shifting, and changing the slant. Since the character templates are defined by the control points of the Bézier spline, it is sufficient to perform the perturbations only on the control points. The perturbation method also includes shifting all control points using randomly chosen displacement vectors.

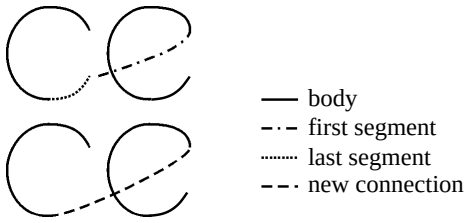


Figure 4.3: Linking letter 'c' to letter 'e'.

Figure 4.4: Static image after the perturbation and concatenation of character templates.

The static image of the text line is produced by concatenating the perturbed templates along a horizontal baseline. This means that the perturbed templates are put one after the other and cursive style letters are linked. An example can be seen in Fig. 4.4. Many of the curves in this figure seem to contain too much noise. However, reducing the level of noise seriously reduces the diversity of character shapes.

To alleviate this problem, in the next section elements of a handwriting generation model will be incorporated into the synthesizing process. This enables to generate naturally looking handwriting with high character shape variability while keeping the noise level low.

4.2 The Delta LogNormal Handwriting Generation Model

The problem of machine generation of handwriting is approximately of the same age as the problem of handwriting recognition [27]. The better understanding of human handwriting generation can be of great benefit for many different disciplines related to handwriting. Besides those already mentioned in Section 1.2, the possible applications include on-line handwriting recognition [86], on-line signature verification [77], and educational tools for teaching children handwriting [20].

Numerous handwriting generation models have been proposed in the literature to account for the variability present in handwriting [87]. Today there are two competing approaches: oscillatory [41] and discrete [85] models. The former views a continuous piece of handwriting as the result of constrained modulation of an underlying basic oscillation movement,

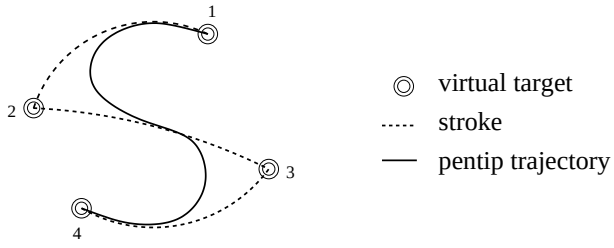


Figure 4.5: Action plan for letter 's' – virtual targets are linked by circular strokes.

while the latter views it as the result of temporal superimposition of simple, discontinuous strokes.

Usually the models aim at explaining various phenomena of *existing* on-line handwriting samples, rather than generating new samples from scratch. That is, given an existing sample, the model parameters are adjusted so that the sample can be *reproduced* with minimal error, in both of the spatial and temporal domains. Generating *new* samples from scratch (e.g. from ASCII transcription) would require extensive knowledge about the correlations of the model parameters in human handwriting, which is thought to be time- and context-dependent, to create new realistic styles. This latter problem is currently not in the focus of handwriting generation research.

The Delta LogNormal model of handwriting generation, introduced during the past decade in [80, 81, 82, 85], is a relatively new and successful model of the discrete type, based on simple and clear concepts, with a solid mathematical foundation [84]. It describes fluent handwriting as the vectorial superimposition of consecutive strokes in time [85]. The strokes of a letter or a word can either be straight line segments or circular arcs, and they are part of a so-called action plan connecting a sequence of virtual targets. Such an action plan regarding letter 's', for example, is depicted in Fig. 4.5.

Each stroke results from an action of rapid writing, produced by the synergy of two parallel neuromuscular systems, one being the agonist and the other the antagonist to the movement, resulting in a delta-lognormal velocity profile of the pentip [80]:

$$v(t) = D_1 \cdot \Lambda_1(t, t_0, \mu_1, \sigma_1^2) - D_2 \cdot \Lambda_2(t, t_0, \mu_2, \sigma_2^2) \tag{4.1}$$

where $\Lambda_i(t, t_0, \mu_i, \sigma_i^2)$ $(i = 1, 2)$ is a lognormal function:

$$\Lambda_i(t, t_0, \mu_i, \sigma_i^2) = \frac{1}{\sigma_i \sqrt{2\pi} \cdot (t - t_0)} \cdot e^{-\frac{\left(\ln(t-t_0)-\mu_i\right)^2}{2\sigma_i^2}} \tag{4.2}$$

D_i denotes the amplitude of the input command to the neuromuscular system, occurring at

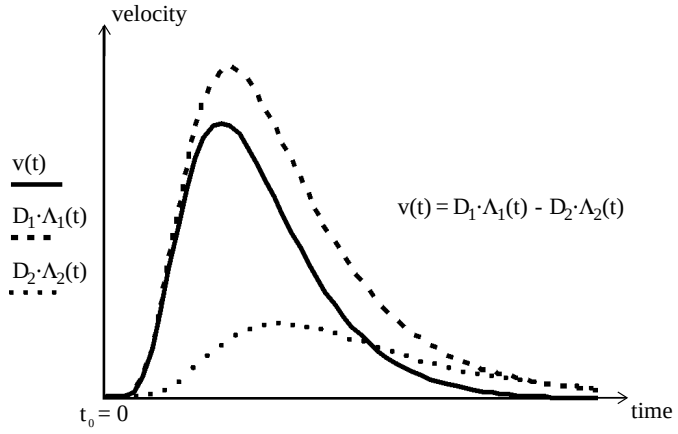


Figure 4.6: Calculation of curvilinear velocity for a single stroke.

time t_0, while μ_i and σ_i are called the logtime delay and the logresponse time, respectively (see [84] for a detailed discussion on the meaning of these parameters).

An illustration of the curvilinear velocity calculation is shown in Fig. 4.6. Since the total integral of the lognormal function is equal to 1, the stroke length is given by $D = D_1 - D_2$. However, since the velocity curve is unbounded with respect to time ($t \to \infty$), in practice the stroke is truncated at some time, t', where $v(t')$ is close to zero.

The execution of two or more consecutive strokes can partially overlap in time, which means that a stroke can be initiated before the preceding one reaches its target [80]. If, say n, strokes overlap at time t, the resulting velocity vector of the pentip is obtained as the vectorial summation of the n individual velocity vectors. This temporal superimposition of discontinuous strokes results in a continuous trajectory of the pentip, as for the example depicted in Fig. 4.5. It also can be seen that the virtual targets are usually never reached (except for the very first and the very last one), so the actual trace of the pentip differs from that of the action plan.

As a first step towards the incorporation of the Delta LogNormal model in the method, the static image described in Section 4.1 is approximated by segments of straight lines and circular arcs, without overlapping [58]. The following points are considered as segmentation points: start or end point of a connected component, local maxima of curvature being greater than a predefined threshold, inflection points, and corner points. These points can be easily extracted from the Bézier spline. In Fig. 4.7, such an approximation of letter 'a' of Fig. 4.1 can be seen.

This kind of approximation can already be considered as a simple action plan, where the segmentation points correspond to the virtual targets, and the segments to the strokes of the action plan. However, in real handwriting the action plan is usually of greater extent

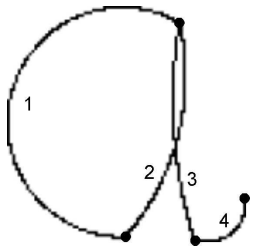


Figure 4.7: Approximation of letter 'a' by circular arc segments.

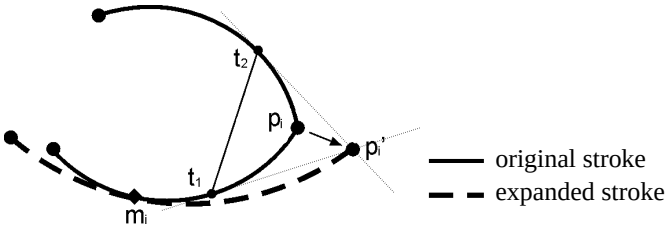


Figure 4.8: Illustration of the stroke expansion method.

than the actual trace of the writing, due to the stroke overlapping effect (see the example in Fig. 4.5). That's why a stroke expansion method is applied on the simple action plan. The idea of the method is illustrated in Fig. 4.8. Here the i-th and the $(i+1)$-th strokes join together at virtual target p_i. We choose randomly two points, t_1 and t_2, on either side of p_i. The new virtual target, p_i', is the intersection of the tangents at t_1 and t_2. Finally, the expanded i-th stroke (dashed line in Fig. 4.8) is a circular arc intersecting the original i-th stroke at some point, m_i (this condition ensures that the expansion will not be too large).

After the action plan has been expanded, delta-lognormal velocity profiles are generated for each stroke, and the degree of overlapping is decided for each consecutive pair of strokes, using randomly perturbed parameters of the Delta LogNormal model (see [30] for an analysis on the effect of various parameter values). Based on this on-line information, the text line can be generated. In Fig. 4.9, a generated version of letter 'a' of Fig. 4.1 can be seen. Here it is noted that a complete text line is drawn component by component, i.e. pen lifting is not modeled in this work.

To make the text line look more realistic, thickening operations are also applied, using gradually decreasing gray level values. Some examples of the final result of the text line generation method are shown in Fig. 4.10. As it can be seen, the curves are smoother compared to Fig. 4.4, due to the overlapping effect.

Figure 4.9: Letter 'a' generated using delta-lognormal velocity profiles and stroke overlapping.

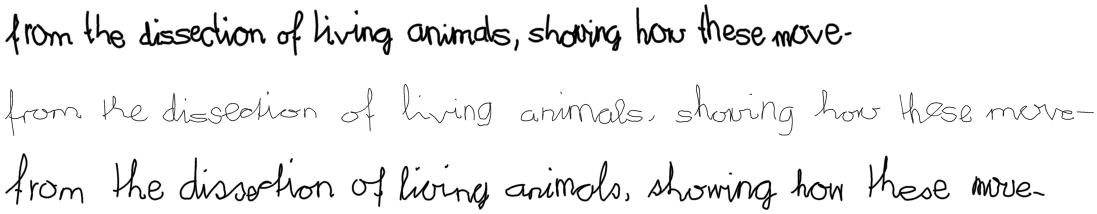


Figure 4.10: Examples of complete text lines generated using delta-lognormal velocity profiles and stroke overlapping.

4.3 Experimental Results

In this section, the results of the experiments are reported, which were conducted to explore the potential advantages and limitations of the proposed method, in terms of some simple scenarios. The application considered was the off-line recognition of handwritten text lines, using the Hidden Markov Model (HMM) based handwritten text line recognizer described in Chapter 2.

For the experiments, the same subsets of *Pool2* (see Table 3.1 in Subsection 3.2.1) were used as in Subsection 3.2.6. That is, the training set consisted of 320 text lines from 64 writers, and the test set of 160 text lines from 32 writers. The writers of the training and the test set were disjoint, i.e. the experiments were writer-independent. The underlying lexicon consisted of 6,012 words. For each of the 320 natural text lines in the training set, 5 synthetic ones were generated, using the ASCII transcription of the corresponding natural text line. This means that altogether 320 natural and $320 \cdot 5 = 1{,}600$ synthetic text lines were available. To measure the recognition performance, always the natural text lines of the test set were used.

Training set	Recognition rate (in %)	Capacity (Ga)
natural only (Tr_1)	70.81	6
synthetic only (Tr_2)	61.78	21
natural + 1 synthetic (Tr_3)	72.30	9
natural + 5 synthetic (Tr_4)	70.06	27

Table 4.1: Recognition rates on the test set, using different types of training sets.

The effects of two factors were investigated. The first was the proportion of natural and synthetic text lines in the training data. For this purpose, four different training sets were produced from the pool of available training text lines: $Tr_1 = \{$all 320 natural text lines$\}$, $Tr_2 = \{$all 1,600 synthetic text lines$\}$, $Tr_3 = \{$320 natural + 320 synthetic text lines$\}$, $Tr_4 = \{$320 natural + 1,600 synthetic text lines$\}$. The second factor was the capacity of the recognition system, i.e. the number of free parameters to be estimated, which had turned out to be important when using synthetic training data, see Chapter 3. The capacity was controlled by varying the number of Gaussian mixture components, Ga, used in the states of the HMMs to estimate observation probability distributions (see also Subsection 2.4.2).

In Table 4.1, the best results using Tr_1, Tr_2, Tr_3, and Tr_4 for training are shown, together with the corresponding capacity values. Although these are the first results on limited data sets, some conclusions can already be drawn. First of all, the natural training set performed much better than the training set consisting exclusively of synthesized text lines. Nevertheless, the experiments also suggest that augmenting the natural training set by synthetic text lines can improve the recognition rate, but the proportion of the natural and synthetic data is an important factor. Only with set Tr_3, where the amount of natural and synthetic training data are the same, an improvement was observed. In case of Tr_2 and Tr_4, a recognition rate lower than that for Tr_1 is achieved. The experiments reported in Table 4.1 confirm the earlier results of Chapter 3, which show that increasing the capacity of the system is beneficial when the training set is expanded by synthetic text lines. This is because in case of too low capacity the system is biased towards the unnatural variability introduced by the synthetic training data.

To compare the results with those of the perturbation method presented in Chapter 3, the same natural training and test sets were considered as in Subsection 3.2.6. According to Table 3.9, 5 and 10 additional synthetic text lines yielded the recognition rates of 75.36% and 77.12%, respectively. Thus at present time the synthetic text lines generated by the perturbation method are more appropriate for synthetic training set expansion. On the other hand, the proposed template-based synthetic handwriting generation method can be used even when there are no natural training text lines available, to create a baseline recognizer.

4.4 Summary and Conclusions

A method for synthesizing handwritten text lines from ASCII transcriptions was presented. The method is based on character templates and the Delta LogNormal handwriting generation model. The aim was to use the synthetic text lines for the training of a handwritten text line recognizer. The experimental results showed that the natural training set performed much better than the one that contained exclusively synthesized text lines generated by the proposed method. On the other hand, the addition of synthetic text lines to the natural training set may improve the recognition rate, but the proportion of natural and synthetic text lines plays an important role.

4.5 Future Work

It is clear that additional work needs to be conducted in the future to make the synthetic text lines represent more diverse writing styles. Since the Delta LogNormal theory is still an active topic of investigation [21, 83, 84], following the developments can also help a lot in making progress.

The possible future improvements of the method include the following:

- Increasing the number of templates per character would be a straightforward step towards the generation of more diverse writing styles.
- Improving the grayscale thickening operations to better reflect the characteristics of natural text line images can also be beneficial, since the recognizer uses grayscale information of the images.
- Making the static image resulting from the perturbation of character templates smoother, by preserving the unambiguous tangent at the control points where two consecutive Bézier arcs meet, as well as the sign of the curvature between inflection points.
- Modeling of the pen-lifting movements, in order to generate on-line data not only on the connected component level but also on the text line level, to be able to work with on-line text line recognizers.
- Refinement of the stroke expansion scheme and its relation with the degree of overlapping. Moreover, following the new results related to the Delta LogNormal model, to find out more about the correlation of its individual parameters in human handwriting.[2]

[2]The problem of correlation has been addressed recently.

Chapter 5

Segmentation-based Text Line Recognition

In this chapter, a novel method for the extraction of words from handwritten text lines is proposed, that uses a tree structure built for the text line to perform the segmentation into individual words. The aim is to implement a segmentation-based methodology for handwritten text line recognition, based on the recognizer of Chapter 2. Although segmentation errors influence adversely the recognition (i.e. there is practically no chance to recognize an erroneously extracted word), on the other hand the knowledge of the exact word boundaries may be advantageous for the recognition of correctly extracted words. For comparing the two approaches, experiments using HMMs trained on both natural and synthetically expanded training data are presented.

5.1 Word Extraction from Handwritten Text Lines

Word extraction from handwritten text lines usually involves the calculation of a line specific threshold which separates the gaps between words from the gaps inside the words in that line. In this section, a novel method is presented that improves this approach, by making the decision about a gap not only in terms of a threshold, but also depending on the context of that gap, i.e. the relative sizes of the surrounding gaps are taken into account. For this purpose, building a structure tree of the text line is proposed, whose nodes represent possible word candidates. Such a tree is traversed in a top-down manner to find the nodes that correspond to words of the text line. Experiments with different gap metrics as well as threshold types show that the proposed method can yield significant improvements over conventional word extraction methods.

5.1.1 Introduction

The segmentation of either machine printed or handwritten text lines is typically based on the assumption that the gaps between words (*inter-word gaps*) are larger than those inside the words (*intra-word gaps*) [48, 51, 52, 65, 66, 71, 96]. As a consequence, such methods often work as follows: first the text line is decomposed into a series of components. A component can either be a single connected component [71, 96], or a set of horizontally overlapping connected components [52]. The next step is the calculation of the distances between adjacent components, using some heuristic called *gap metric* [65]. For example, one of the simplest gap metrics is to consider the horizontal distance between the bounding boxes of the two consecutive components. Finally, a gap is classified as being an inter-word gap if the size of the gap (i.e. the corresponding distance) is above a threshold value. Otherwise, a gap is classified as being an intra-word gap. The threshold can be determined by clustering the distances between the adjacent components [51, 52, 66], or it can be extracted using some specific features of the text line image [48, 71]. After this classification step, the words can be extracted from the text line and further processed. The extracted words may contain punctuation marks (e.g. dot, comma, etc.) attached.[1]

There are other recognition-free approaches (i.e. there is no attempt made to find out the lexical content of the text line) to word segmentation, including neural networks to determine segmentation points [50], scale space techniques [67], and the utilization of semantic knowledge [26].

In this section, a method is proposed that is intended to make the threshold-based approach described above more flexible, by allowing the inter-word gaps to be smaller than the threshold, based on the context of the gaps. A so-called *structure tree* is built for each considered text line. The nodes of the tree are groups of consecutive components, and they represent the possible word candidates. For each node, the distances inside the corresponding group must be smaller than the distances to the left and to the right side of the group, up to a predefined factor $\alpha \geq 1$. The tree is traversed in a top-down manner to find the nodes that correspond to words of the text line. The rule is that a word is found if all the gaps within the node are smaller than a predefined threshold. The threshold value is extracted directly from the image of the text line, and does not need to be defined by the user beforehand.

The method can be considered as an extension of traditional threshold-based methods. Experiments with different gap metrics as well as threshold types show that the new method can yield significant improvements above threshold-based word extraction methods. It is also illustrated through a simple scheme that punctuation detection can further improve both types of methods.

[1] Punctuations are usually not considered as words in word extraction tasks, but they are rather to be detected in a subsequent recognition phase.

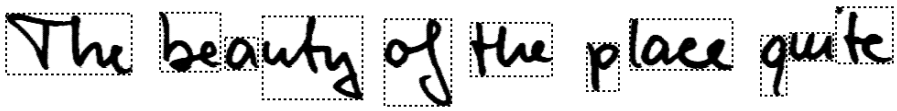


Figure 5.1: Components of a handwritten text line and their bounding boxes.

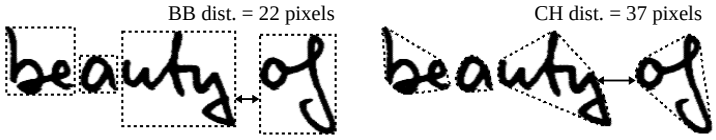


Figure 5.2: Illustration of the bounding box (left) and the convex hull (right) gap metrics.

5.1.2 Components and Distances

Before applying the word extraction method, the text line images are normalized, which consists of those steps described in Section 2.2. Then the lines are decomposed into components separated by white spaces in the vertical projection profile of the line.[2] An example together with the bounding boxes of the components can be seen in Fig. 5.1. Note that one component can consist of several horizontally overlapping connected components.

In this work, the following two gap metrics are used (for an illustration see Fig. 5.2):

- *Bounding box* (*BB*) *metric*: the distance between two adjacent components is the horizontal distance between the corresponding bounding boxes.
- *Convex hull* (*CH*) *metric*: the distance is the minimal white run-length[3] between the convex hulls of the two adjacent components. If the components do not overlap vertically (i.e. there is no horizontal white run connecting them), the bounding box distance is taken.

5.1.3 Threshold Computation

For the calculation of the threshold, an approach similar to the one described in [71] was taken. The threshold is a value of a feature of the text line image, multiplied by some constant factor $\gamma > 0$.

[2] In the grayscale text line images, the Otsu threshold [79] is used to determine whether a pixel belongs to the foreground (black pixel) or the background (white pixel). A column of a text line image is considered white space if it contains only white pixels.

[3] A *white run-length* is the length of a series of white pixels connecting two black ones in the same row.

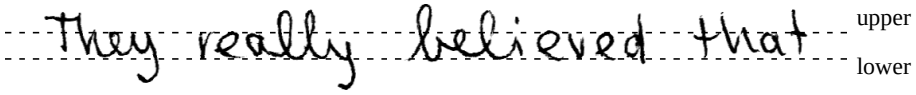


Figure 5.3: Upper and lower baselines of a text line.

Two kinds of features were considered, both of which are based on the white run-lengths between the upper and lower baselines[4]:

- *Median white run-length* (*MWR*): the median of the set of white run-lengths between the upper and lower baselines.
- *Average white run-length* (*AWR*): the median number of white pixels in a row divided by the median number of black-white transitions in a row, considering the rows between the upper and lower baselines.

So in the baseline method for word extraction, first the components are extracted (see Subsection 5.1.2), followed by the calculation of the distances between adjacent components using one of the gap metrics (see Subsection 5.1.3). Then, those gaps smaller than the threshold $\gamma \cdot f$, where $\gamma > 0$ is a predefined parameter and f is either the *MWR* or the *AWR* feature value of the text line, are considered to be intra-word gaps, while the others are the inter-word gaps. Finally, the words can be extracted using this information.

Depending on the choice of the gap metric as well as the threshold type, there are four possible variants of the baseline method. We denote them by *BB–MWR*, *BB–AWR*, *CH–MWR*, and *CH–AWR*.

5.1.4 Structure Tree

Consider a gap metric and a series S of adjacent components. Let *maxgap* be the maximal distance within S, while *leftgap* and *rightgap* the distances on the left and right hand side of S, respectively. If S consists of only one component, then consider *maxgap* as zero. Furthermore, if there is no component preceding (succeeding) S, consider *leftgap* (*rightgap*) as positive infinite.

We say that S is a group iff

$$\alpha \cdot maxgap < \min\{leftgap, rightgap\} \tag{5.1}$$

[4]The *upper* and *lower baselines* separate the ascenders and descenders from the body of the text, respectively. See Fig. 5.3.

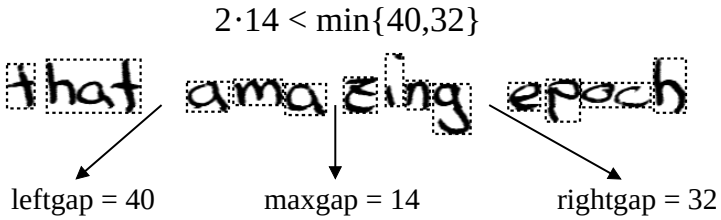


Figure 5.4: Illustration of group concept: the series of components corresponding to the word "amazing"is a group for $\alpha = 2$, using the *BB* gap metric.

where $\alpha \geq 1$ is a predefined, fixed parameter.

The concept of group is illustrated in Fig. 5.4, using the *BB* gap metric. Here the series of components corresponding to "amazing" is a group for $\alpha = 2$, because $2{\cdot}14 < \min\{40, 32\}$.

Some basic properties of the group definition, which can be easily checked by the reader, are listed below:

- Each single component is a group for any $\alpha \geq 1$ (*singleton group*).
- The series of all the components of the text line is a group for any $\alpha \geq 1$ (*trivial group*).
- Any two groups are either disjoint or one of them completely includes the other. In other words, there is no overlapping between groups, except for total inclusion.
- With an increasing value of α, the number of groups in a text line decreases.

A *division* of a group means that its components are divided into at least two sub-groups (i.e. each component belongs to exactly one sub-group). The division is called *minimal division* if the number of sub-groups is minimal. For example, a minimal division of the line in Fig. 5.4 contains three sub-groups corresponding to "that", "amazing", and "epoch", given the fixed value of $\alpha = 2$. (Note that none of them can be further expanded to remain a group different from the whole line.)

Two important properties related to the concept of division, which follow from the properties of groups mentioned earlier, are these:

- A singleton group has no division.
- The minimal division of a non-singleton group is unique.

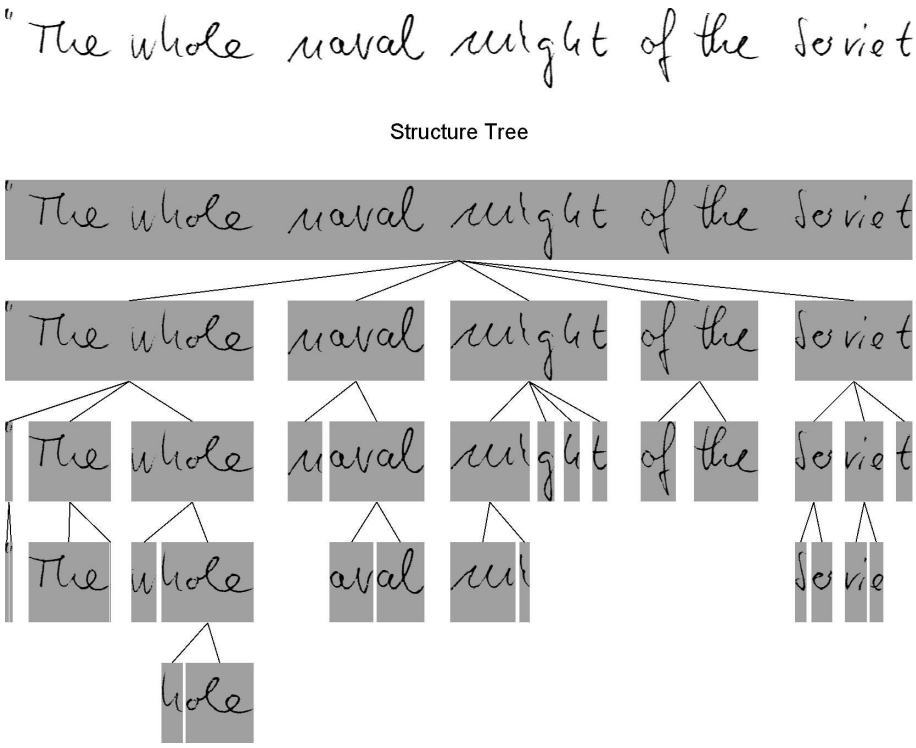


Figure 5.5: Structure tree of a handwritten text line with *BB* gap metric and $\alpha = 1.5$.

Now the *structure tree* of a text line can be defined in the following recursive way:

Structure tree: Its root is the trivial group. The children of a node are the sub-groups of its minimal division, if a minimal division exists.

The concept of structure tree is illustrated in Fig. 5.5, using the *BB* gap metric and $\alpha = 1.5$. Note the following three important properties of the tree:

- The leaves of the tree are the singleton groups, i.e. the individual components.
- Each group of the text line is a node somewhere in the tree.
- The structure tree is unique, because of the uniqueness of the minimal division.

5.1.5 Novel Segmentation Method

The proposed method to improve the baseline method (see Subsection 5.1.3) is the following: starting from the trivial group, the structure tree is built gradually in a top-down

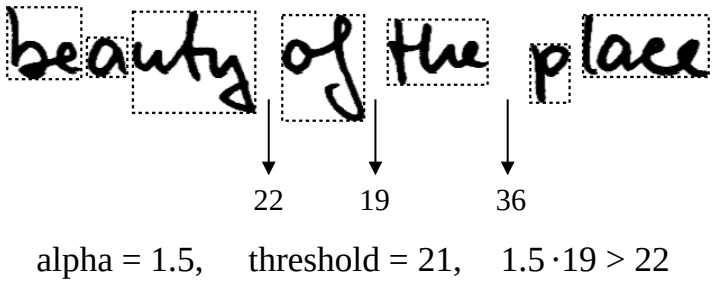


Figure 5.6: A case where the novel approach performs better than the threshold-based one: $19 < threshold$, but $\alpha \cdot 19 < \min\{22, 36\}$ does not hold.

manner by always dividing its actual leaves using minimal division. When an actual leaf has a *maxgap* value less than the threshold (which equals $\gamma \cdot f$, see Subsection 5.1.3), it is not divided further, because then it is considered as being a word.

So the result of the segmentation method is a partial structure tree (equivalently it can be considered as the result of a top-down search in the *full* structure tree), whose leaf nodes correspond to the words of the text line. Once all leaf nodes of the tree have been constructed, the corresponding words can be extracted from the text line.

The method requires parameters, α and γ. The first of these parameters, α, controls the tree structure, while the second parameter is identical to that of the baseline method (see Subsection 5.1.3). Note that for $\alpha = 1$, the new method gives exactly the same result as the baseline method. That's why it can be considered as an extension of the simple threshold-based method.

The expected advantage of the new method is shown in Fig. 5.6. Here, in contrast to the baseline method, the word pair "of the" is not accepted as being one word, because, although the gap between them being smaller than the threshold, they do not satisfy the group condition for $\alpha = 1.5$. So they will be separated and extracted correctly. This makes it clear that in the novel approach an inter-word gap is allowed to be smaller than the threshold, resulting in a more flexible way of how the threshold value influences the segmentation result.

A possible drawback of the new method is that introducing the group condition might result in missing a word. This happens if the word does not satisfy the group condition for the given α parameter value. In this case the word is not represented as a node in the structure tree, and consequently cannot be extracted correctly. Fortunately, there are not many of such words: for $\alpha = 1$, $\alpha = 1.5$, and $\alpha = 2$, approximately 99%, 98%, and 95% of the words (optionally with some of the neighboring punctuations attached) in the datasets used later in the experiments satisfy the group condition, respectively, for either gap metrics. See Subsection 5.1.7 for details.

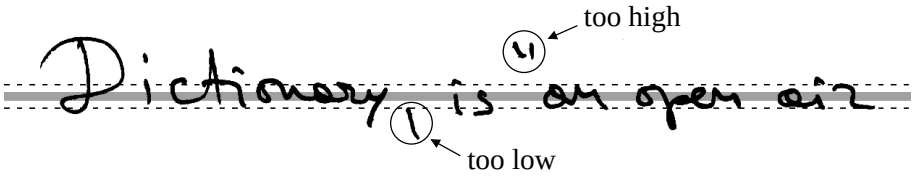


Figure 5.7: Punctuation detection: components located too low or too high are not considered for segmentation.

5.1.6 Punctuation Detection

Punctuations may cause many errors during the segmentation of a text line, because they divide the inter-word gaps into two smaller ones, which can be mistaken with intra-word gaps by the segmentation method [96]. That's why it was tested how a simple punctuation detection scheme affects the accuracy of the best performing variant of the segmentation methods.

The detection scheme simply finds those components that do not intersect the middle-third stripe between the upper and lower baselines. These components are not taken into consideration when segmenting the line. The idea is illustrated in Fig. 5.7. All the components that do not intersect the gray area are eliminated. However, after the segmentation of the "punctuation-free" line has finished, each eliminated component is attached to the closest word found, so as not to lose e.g. the last letter of a word.

5.1.7 Experimental Results

Four variants of the baseline and the novel method have been introduced, respectively. The four variants are denoted by *BB–MWR*, *BB–AWR*, *CH–MWR*, and *CH–AWR* (see Subsection 5.1.3). The purpose of the experiments described in this subsection was, for all four variants, to compare the baseline and the novel method against each other, and see whether the novel method really yields improvements above the baseline method. The accuracy of word extraction was measured as the percentage of correctly extracted words in relation to the total number of words.

For the experiments, two subsets, H_{dev} and H_{test}, of the IAM-Database (see Section 2.1) were considered, by random sampling. The first set, H_{dev}, was used for the development of the various segmentation methods, including the optimization of their free parameters (γ for the variants of the baseline method and (α, γ) for the variants of the novel method). This set contained altogether 7,880 word instances in 1,044 handwritten text lines written by 120 writers. Since the variant *CH–AWR* performed best on H_{dev}, the punctuation detection scheme described in Subsection 5.1.6 was tried with that. This

	baseline method		novel method		
	Acc.	γ	Acc.	α	γ
BB-MWR	91.48	1.30	92.90	1.80	1.40
BB-AWR	92.78	0.85	93.25	1.55	0.85
CH-MWR	93.51	1.35	94.23	1.60	1.70
CH-AWR	94.75	0.95	95.17	1.50	1.0
CH-AWR-PD	95.12	0.95	95.67	1.50	1.0

Table 5.1: Word extraction accuracies of the different methods on the test set H_{test} (in %), together with their optimal parameters.

resulted in another, fifth variant for both the baseline and the novel approach, denoted by *CH−AWR–PD*. Optimization of the corresponding free parameter(s) was performed on H_{dev} for this new variant, too.

The other set, H_{test}, was used to compare the performances of the optimized baseline and novel methods. This set contained 7868 word instances in 1,025 handwritten text lines produced by 120 writers. The writers of H_{dev} and H_{test} were disjoint.

The accuracies of the various methods on H_{test} are shown in Table 5.1. As it can be seen, the novel method yielded improved accuracy for all of the five considered variants. All the improvements are statistically significant at a level greater than 90%.[5] For *CH−AWR–PD* the significance level is greater than 98%.

From Table 5.1 it is concluded that the convex hull based gap metric (*CH*) performed much better than the bounding box based one (*BB*). This is in accordance with other results reported in the literature [52, 65, 66, 96]. For the threshold types, *AWR* seems to be superior to *MWR*. The fact that the optimal value of γ is about 1 when *AWR* is used together with *CH* might indicate the relevance of the *AWR* feature to the problem of word segmentation. It can also be observed that the novel method brought the *MWR* and *AWR* results closer to each other, for either of the gap metrics. This suggests that the novel method is less sensitive to the type of threshold used.

Furthermore, the novel method usually had a higher optimal γ value, due to its more flexible use of the threshold (see Subsection 5.1.5). Finally it is concluded that punctuation detection, despite of being realized in a rather simple way, improved the accuracy significantly. However, the *CH−AWR* variant of the novel method still performed slightly better than the baseline method with punctuation detection.

The results reported in Table 5.1 are comparable to those of [67] and [71], where the per-

[5]The significance level of an improvement was calculated from the writer level accuracies, by applying a statistical z-test for matched samples.

formance was evaluated in a similar way as here, although different datasets for evaluation were used. In the experiments it was also noticed that for the punctuation detection case at $\alpha = 1.5$ the structure tree contained 98.45% of the words in set H_{test}, although the word extraction accuracy reported in Table 5.1 is only 95.67%. Hence, there is still room for further improvement.

5.1.8 Summary and Future Work

A novel approach for the extraction of words from handwritten text lines was proposed. The method makes use of a structure tree built for each text line, enabling more flexibility than those methods making decisions by simply comparing the gaps with a threshold. This is reflected by the fact that according to the new method, inter-word gaps are allowed to be smaller than the threshold, based on their context. The nodes of the structure tree represent possible word candidates, and the words are extracted by applying a top-down search procedure. When a certain condition is true at a node during the search, the node is considered as being a word, and no further search is performed on its sub-tree.

Experiments with different gap metrics as well as threshold types showed the effectiveness of the approach. The performance was further improved by incorporating a simple punctuation detection scheme.

In the future, the performance can be further improved by using more sophisticated heuristics for the detection and interpretation of punctuation marks, as well as different rules for accepting a node in the structure tree as a word. Clustering-based thresholds are also to be examined. Furthermore, several variations and generalizations of the group rule and the corresponding structure tree can be elaborated, e.g. the one described in Appendix A. Finally, the structure tree may also be useful for the generation of multiple word segmentation hypotheses, instead of producing only one, fixed segmentation output.

5.2 Recognition Methodology

The methodology used in this chapter for segmentation-based text line recognition is based on the word extraction method described in Section 5.1 and on the original segmentation-free recognizer presented in Chapter 2.

An illustration of the methodology can be seen in Fig. 5.8. First the normalized text line image is segmented into individual words (denoted by the vertical lines), and then for each extracted word image the recognizer of Chapter 2 provides an N-best list of candidate transcriptions based on the HMM score (see Section 2.4). It is noted that the extracted words may contain punctuation symbols attached, and the recognizer is aware of this fact, so it also calculates the HMM score for such possible phrases like “, that”,

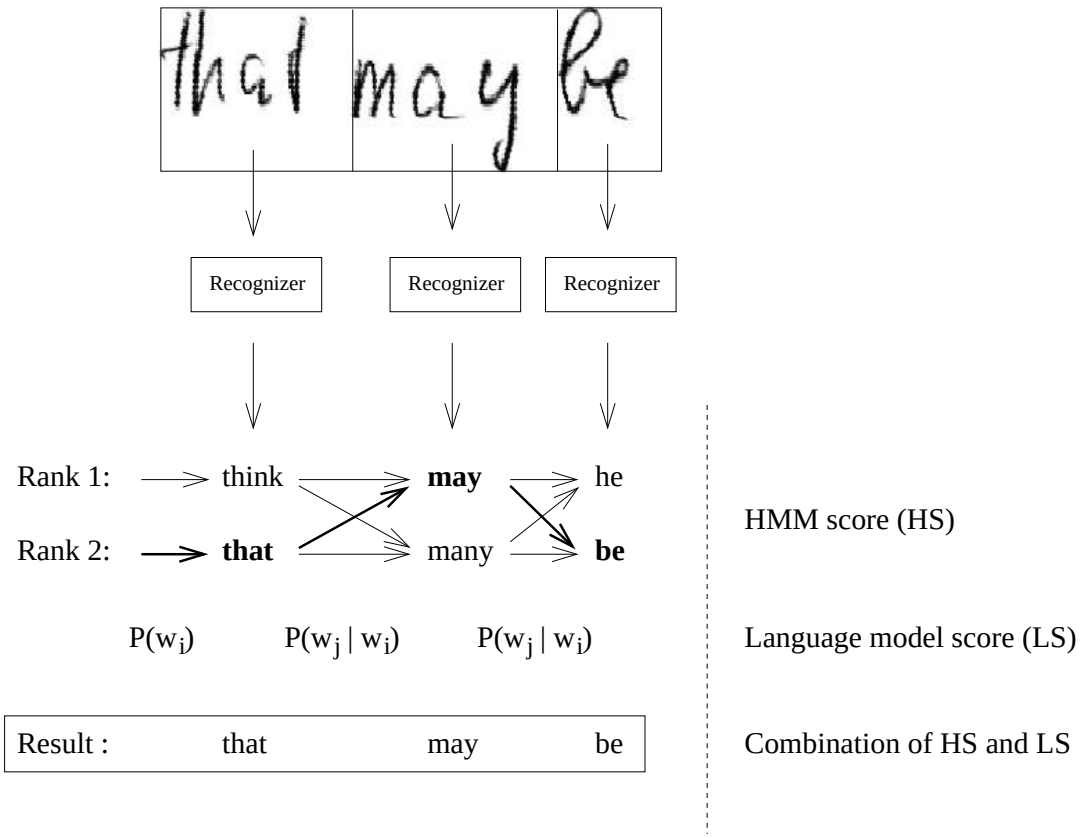


Figure 5.8: Illustration of the methodology for segmentation-based text line recognition: the normalized text line image is first segmented into individual words, then for each extracted word image the recognizer of Chapter 2 provides an N-best list of candidate transcriptions based on the HMM score (in the example, $N = 2$), and then the optimal path with respect to the combination of the overall HMM score and language model score is taken as recognition result.

". That,", or even ",". The rule is that there may be arbitrarily many punctuations, but at most one non-punctuation (i.e. word) is allowed in the recognition result. Using the words of the N-best lists as nodes, a directed graph is built: each word of an N-best list is connected to all words in the succeeding N-best list, as shown in the example of Fig. 5.8. The weight of an edge connecting two words is equal to the corresponding word transition probability of the underlying bigram language model (see also Section 2.5). There are also edges leading to the words of the very first N-best list (technically, a dummy node can be added for this purpose, from which these edges originate): the weight of such an edge is the probability of occurrence of the corresponding word, according to the language

model. Finally, the optimal path (highlighted in boldface in Fig. 5.8) along which the product of all the HMM scores (corresponding to the nodes) and all the language model scores (corresponding to the edges) is maximal, is taken as recognition result.

As a special case, $N = 1$, i.e. the 1-best list when only the top choice is taken for each extracted part of the text line image, practically means that no language model is considered, because the recognition result is based exclusively on the HMM score.

Since the result of the recognition process is a sequence of words (including punctuation symbols, too), the evaluation is the same as for the original, segmentation-free system, described in Section 2.6. That is, at the evaluation the recognized punctuation symbols also count as words.

5.3 Experimental Results

The purpose of the experiments was to compare the performances of the segmentation-free and the segmentation-based approaches, particularly when using HMMs trained on both natural and synthetically expanded training sets.

For word extraction, the variant of the proposed novel method that performed best in the experiments of Subsection 5.1.7, namely *CH−AWR−PD*, was applied.

As segmentation-free recognizer, the *Original System* and the *Expanded System* of Subsection 3.2.4 were considered. The former was trained on 1,593 natural text lines, while the latter was trained on the same natural training set but expanded with 5·1,593=7,965 synthetically generated text lines at *middle* strength, see details in Subsection 3.2.4. For testing, the same natural test set of 400 text lines from 80 writers used in Subsection 3.2.4 was considered, but 50 lines were removed because their writers were also involved in the set denoted by H_{dev}, which was used for developing the segmentation method (see Subsection 5.1.7). Thus the test set used consisted of 350 text lines produced by 70 writers. The underlying lexicon included 6,012 words.

The results using different N values for the size of the N-best lists are shown in Table 5.2. As it can be seen, for $N = 1$, i.e. when no language model score was involved, the *Original System* performed better than the *Expanded System*, which is quite surprising if we consider that in the experiments of Subsection 3.2.4 the *Expanded System* performed significantly better.

To find out the reason for this phenomenon, and to compare the segmentation-free approach with the segmentation-based one, the recognition rates on the test set were also measured for the original, segmentation-free recognizer, first without language model (i.e. only HMM score is taken into account, see Section 2.4), and then with bigram lan-

N	Original System	Expanded System
1	63.45	63.04
5	70.96	70.39
10	70.86	71.31
20	71.18	71.88
30	**71.94**	71.91
40	71.43	72.16
50	71.56	72.47
60	71.88	72.54
70	71.62	72.76
80	71.62	72.79
90	71.50	**72.98**
100	71.59	**72.98**

Table 5.2: Recognition rates on the test set (in %), for the segmentation-based handwritten text line recognition systems trained on natural (*Original System*) and synthetically expanded (*Expanded System*) training sets, using different N-best list sizes.

	Original System	Expanded System
no language model	63.38	62.75
bigram language model	75.95	78.66

Table 5.3: Recognition rates on the test set (in %), for the segmentation-free recognition systems trained on natural (*Original System*) and synthetically expanded (*Expanded System*) training sets, with and without language model.

guage model (see Section 2.5).[6] The results are shown in Table 5.3.

As expected according to the previous results, it can be observed that when bigram language model was used, the system trained on the expanded training set performed significantly better than the system trained on natural training data only. On the other hand, when no language model was used, the *Original System* performed better. A possible explanation is that in Subsection 3.2.4 the distortion strength was optimized for the bigram language model, not for the case when no language model is used. That is, it cannot be stated that the HMMS trained on the synthetically expanded training set are "better", because this might only be true for that particular language model (or type of

[6]The experiments with bigram language model are exactly the same as those reported in Table 3.6, except for that 50 text lines that were removed from the original test set of 400 text lines.

language model) for which the distortion strength was optimized.[7]

For further explanation, consider again Table 5.2. The greater the value of N is, the more possibilities the bigram language model has to affect the recognition result. For the *Original System*, the recognition rate stops improving from $N = 30$, while for the *Expanded System* it increases monotonically up to $N = 100$. The explanation is that although the synthetically expanded training set did not help in moving the correct word up into the top rank position (that's why the worse recognition rate when $N = 1$), it did help in moving the correct word into a *better* position, closer to the top ranked (but incorrect) words, and thus it has become easier for the correct word to jump on the top with the help of the language model score. In other words, the synthetically expanded training set made the bigram language model (for which it was optimized) more effective. The fact that the improvement in the recognition rate is considerably lower than that for the segmentation-free approach is due to the incorrectly segmented words, which caused the bigram language model to be less effective, and consequently less profitable for the *Expanded System*.

The results also show that when no language model was used, the segmentation-free and the segmentation-based approaches achieved approximately the same recognition performance. This means that the negative effect of incorrectly segmented words was compensated by the knowledge of the exact word boundaries of the correctly extracted words. For the case when bigram language model was integrated into the recognition process, the segmentation errors deteriorated its effectiveness in a way that it could not have been compensated.

5.4 Summary and Conclusions

In this chapter, a novel method for the extraction of words from handwritten text lines was presented, which traverses a structure tree built for the text line to find the possible word candidates. Based on this method and the original segmentation-free recognizer described in Chapter 2, a segmentation-based methodology for handwritten text line recognition was devised.

In the experiments, it was observed that when no language model was used, the segmentation-based recognizer achieved approximately the same results as the original segmentation-free recognizer, i.e. the segmentation errors were compensated by the advantage that the recognizer knew the exact boundaries of the correctly segmented words. However, when bigram language model was incorporated, the segmentation-free recognizer performed considerably better, i.e. the segmentation errors deteriorated the applicability of the bigram

[7] A few recently conducted experiments suggest that for no language model the *weak* and the *very weak* distortion strengths of Subsection 3.2.4 perform better than the *middle* strength.

language model to such a degree that it could not have been compensated.

The results of the experiments also suggest that the underlying language model plays an important role when synthetic training data is used. That is, if the parameters of the synthetic text line generation method are optimized using a given language model, changing the language model afterwards may deteriorate the recognition performance compared to that achieved by the natural training data.

5.5 Future Work

The experimental results indicate that there are two different directions to continue the research:

- Performing segmentation and recognition simultaneously, by traversing the structure tree in a top-down manner, and using the recognition results of the nodes to decide whether a certain node corresponds to a word or not. This way the segmentation may not cause serious damages when a bigram language model is also incorporated, since the vast majority of the words are included among the hypotheses provided by the structure tree.
- Further investigating the relationship between the properties of the language model (e.g. perplexity, see [72]) and the effectiveness of different types of synthetically expanded training sets (e.g. variable distortion strengths), in terms of either segmentation-free or segmentation-based recognition tasks.

Chapter 6

Summary and Conclusions

In this thesis, the generation and use of synthetic training data was investigated, in terms of the problem of off-line cursive handwritten text line recognition. For this purpose, an already existing Hidden Markov Model (HMM) based handwritten text line recognizer described in Chapter 2 was considered. The main motivation was to examine whether the recognition performance can be improved by expanding the natural training set using synthetically generated text lines, because the automatic generation of training data is much faster and cheaper than collecting additional human written samples.

First, in Chapter 3 a perturbation model was introduced, which uses geometrical transformations as well as thinning and thickening operations to distort existing human written lines of text. After examining several configurations of the recognizer and the perturbation method, it was shown that synthetically expanded training sets can improve the recognition performance significantly, even when the original training set is large and the text lines are produced by many different writers. It was found that the two most important factors to optimize are the strength of the distortions and the capacity of the recognizer. The latter was defined as the number of free parameters of the system to be estimated from the available training data, and was controlled by the number of Gaussians used for distribution estimations inside the HMM states. It was shown that increasing the capacity after training set expansion is beneficial, because the optimal capacity of the recognition system trained on the expanded training set is expected to be higher than that of the system trained on the original training set. Furthermore, if the capacity is not properly adjusted when using the synthetically expanded training set, there is the danger that the capacity may become too low, and the system is biased in such a way towards unnatural handwriting styles present in the synthetic texts that may cause the recognition performance to drop. Finally, it was argued that saturation must also be taken into account, because neither synthetic nor natural training set expansion can improve the recognition rate when the recognition system is already saturated by the available amount of natural

training data. On the other hand, when the system is far from being saturated, synthetically expanding the natural training set can yield substantial improvements in the recognition performance.

Another method for synthetic text line generation was presented in Chapter 4, where handwritten text lines were synthesized directly from the ASCII transcription. The method is based on templates of characters as well as on the Delta LogNormal model of human handwriting generation. One advantage over the perturbation method is that no human written text lines are needed to generate the synthetic texts. Second, by incorporating elements of a handwriting generation model, the results may better reflect psychophysical phenomena of human handwriting than applying only geometrical ad-hoc distortions on a static image. The experimental results showed that the addition of such synthetic text lines to the natural training set may improve the recognition rate, but the proportion of natural and synthetic text lines is important to be tuned properly. At present time, the synthetic text lines generated by the perturbation method of Chapter 3 are more appropriate for synthetic training set expansion, but the template-based method can be used even when there are no human written training text lines available.

In Chapter 5, a novel method for the extraction of words from handwritten text lines was proposed, which traverses a structure tree built for the text line to find the possible word candidates. Based on this method and the recognizer used in the previous experiments, a segmentation-based methodology for handwritten text line recognition was devised. Regarding the use of synthetic training data, the experimental results suggest that the underlying language model plays an important role. That is, if the parameters of the synthetic text line generation method are optimized using a given language model, changing the language model afterwards may deteriorate the recognition performance compared to that achieved by the natural training data.

Although there are still many open problems to investigate in the future (e.g those mentioned at the end of the individual chapters), based on the work presented in this thesis it can be concluded that the use of synthetic training data can lead to improved recognition performance of handwriting recognition systems.

Bibliography

[1] L. Ahn, M. Blum, N.J. Hopper, and J. Langford. The CAPTCHA Web Page. `http://www.captcha.net`, 2000.

[2] L. Ahn, M. Blum, N.J. Hopper, and J. Langford. CAPTCHA: Telling Humans and Computers Apart. In E. Biham, editor, *Advances in Cryptology*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2003.

[3] L. Ahn, M. Blum, and J. Langford. Telling Humans and Computers Apart Automatically – How Lazy Cryptographers Do AI. *Communications of the ACM*, 47(2):57–60, 2004.

[4] N. Arica and F.T. Yarman-Vural. An Overview of Character Recognition Focused on Off-line Handwriting. *IEEE Trans. on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, 31(2):216–233, 2001.

[5] H.S. Baird. Document Image Defect Models. In H.S Baird, H. Bunke, and K. Yamamoto, editors, *Structured Document Image Analysis*, pages 546–556. Springer, 1992.

[6] H.S. Baird. Document Image Defect Models and their Uses. In *Proc. 2nd Int. Conf. on Document Analysis and Recognition*, pages 62–67, Tsukuba Science City, Japan, 1993.

[7] H.S. Baird. The State of the Art of Document Image Degradation Modeling. In *Proc. 4th IAPR Workshop on Document Analysis Systems*, pages 1–13, Rio de Janeiro, Brasil, 2000.

[8] H.S. Baird, A.L. Coates, and R.J. Fateman. PessimalPrint: a Reverse Turing Test. *Int. Journal on Document Analysis and Recognition*, 5(2-3):158–163, 2003.

[9] H.S. Baird and R. Fossey. A 100-Font Classifier. In *Proc. 1st Int. Conf. on Document Analysis and Recognition*, pages 332–340, St.-Malo, France, 1991.

[10] H.S. Baird, V. Govindaraju, and D.P. Lopresti. Document Analysis Systems for Digital Libraries: Challenges and Opportunities. In S. Marinai and A. Dengel, editors, *Document Analysis Systems VI*, volume 3163 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2004.

[11] H.S. Baird and M. Luk. Protecting Websites with Reading-based CAPTCHAs. In *Proc. 2nd Int. Web Document Analysis Workshop*, pages 53–56, Edinburgh, Scotland, 2003.

[12] H.S. Baird and G. Nagy. A Self-Correcting 100-Font Classifier. In *Proc. IS&T/SPIE Symposium on Electronic Imaging: Science and Technology*, volume 2181, pages 106–115, San Jose, California, USA, 1994.

[13] H.S. Baird and T. Riopka. ScatterType: a Reading CAPTCHA Resistant to Segmentation Attack. In *Proc. 12th SPIE/IS&T Document Recognition and Retrieval Conference*, volume 5676, San Jose, California, USA, 2005.

[14] C. Barrière and R. Plamondon. Human Identification of Letters in Mixed-Script Handwriting: An Upper Bound on Recognition Rates. *IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics*, 28(1):78–82, 1998.

[15] H. Bunke. Recognition of Cursive Roman Handwriting – Past, Present and Future. In *Proc. 7th Int. Conf. on Document Analysis and Recognition*, pages 448–459, Edinburgh, Scotland, 2003.

[16] J. Cano, J.C. Pérez-Cortes, J. Arlandis, and R. Llobet. Training Set Expansion in Handwritten Character Recognition. In *Proc. 9th SSPR / 4th SPR*, pages 548–556, Windsor, Ontario, Canada, 2002.

[17] M. Chew and H.S. Baird. BaffleText: a Human Interactive Proof. In *Proc. 10th SPIE/IS&T Document Recognition and Retrieval Conference*, volume 5010, pages 305–316, Santa Clara, California, USA, 2003.

[18] H. Choi, S.J. Cho, and J.H. Kim. Generation of Handwritten Characters with Bayesian Network based On-line Handwriting Recognizers. In *Proc. 7th Int. Conf. on Document Analysis and Recognition*, pages 995–1001, Edinburgh, Scotland, 2003.

[19] H. Choi, S.J. Cho, and J.H. Kim. Writer-Dependent Online Handwriting Generation with Bayesian Networks. In *Proc. 9th Int. Workshop on Frontiers in Handwriting Recognition*, pages 130–135, Kokubunji, Tokyo, Japan, 2004.

[20] S. Djeziri, W. Guerfali, R. Plamondon, and J.M. Robert. Learning Handwriting with Pen-based Systems: Computational Issues. *Pattern Recognition*, 35(5):1049–1057, 2002.

[21] M. Djioua, R. Plamondon, A.D. Cioppa, and A. Marcelli. Delta-lognormal Parameter Estimation by Non-linear Regression and Evolutionary Algorithm: A Comparative Study. In *Proc. 12th Conf. of the International Graphonomics Society*, pages 44–48, Salerno, Italy, 2005.

[22] D. Doermann and S. Yao. Generating Synthetic Data for Text Analysis Systems. In *Proc. 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 449–467, Las Vegas, Nevada, USA, 1995.

[23] H. Drucker, R. Schapire, and P. Simard. Improving Performance in Neural Networks Using a Boosting Algorithm. In S. Hanson et. al., editor, *Advances in Neural Information Processing Systems 5*, pages 42–49. Morgan Kaufmann, 1993.

[24] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

[25] D. Elliman and N. Sherkat. A Truthing Tool for Generating a Database of Cursive Words. In *Proc. 6th Int. Conf. on Document Analysis and Recognition*, pages 1255–1262, Seattle, WA, USA, 2001.

[26] M. Feldbach and K.D. Tönnies. Word Segmentation of Handwritten Dates in Historical Documents by Combining Semantic A-Priori-Knowledge with Local Features. In *Proc. 7th Int. Conf. on Document Analysis and Recognition*, pages 333–337, Edinburgh, Scotland, 2003.

[27] J.J.D. Gon, J.Ph. Thuring, and J. Strackee. A Handwriting Simulator. *Physics in Medicine and Biology*, 6(3):407–414, 1962.

[28] S. Gopisetty, R. Lorie, J. Mao, M. Mohiuddin, A. Sorin, and E. Yair. Automated Forms-processing Software and Services. *IBM Journal of Research and Development*, 40(2):211–230, 1996.

[29] V.K. Govindan and A.P. Shivaprasad. Artificial Database for Character Recognition Research. *Pattern Recognition Letters*, 12(10):645–648, 1991.

[30] W. Guerfali and R. Plamondon. The Delta LogNormal Theory for the Generation and Modeling of Cursive Characters. In *Proc. 3rd Int. Conf. on Document Analysis and Recognition*, pages 495–498, Montreal, Canada, 1995.

[31] S. Günter and H. Bunke. Optimizing the Number of States, Training Iterations, Gaussians in an HMM-based Handwritten Word Recognizer. In *7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland*, pages 472–476, 2003.

[32] S. Günter and H. Bunke. Multiple Classifier Systems in Offline Handwritten Word Recognition – On the Influence of Training Set and Vocabulary Size. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 18(7):1302–1320, 2004.

[33] I. Guyon. Handwriting Synthesis from Handwritten Glyphs. In *Proc. 5th Int. Workshop Frontiers in Handwriting Recognition*, pages 309–312, Essex, United Kingdom, 1996.

[34] I. Guyon, R.M. Haralick, J.J. Hull, and I.T. Phillips. Data Sets for OCR and Document Image Understanding Research. In H. Bunke and P.S.P. Wang, editors, *Handbook of Character Recognition and Document Image Analysis*, pages 779–799. World Scientific, 1997.

[35] T.M. Ha and H. Bunke. Off-line Handwritten Numeral Recognition by Perturbation Method. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(5):535–539, 1997.

[36] D. Harel. *Computers Ltd: What They Really Can't Do.* Oxford University Press, 2003.

[37] D. Hearn and M.P. Baker. *Computer Graphics.* Prentice-Hall, 1986.

[38] M. Helmers and H. Bunke. Generation and Use of Synthetic Training Data in Cursive Handwriting Recognition. In *Proc. 1st Iberian Conf. on Pattern Recognition and Image Analysis*, pages 336–345, Puerto de Andratx, Mallorca, Spain, 2003.

[39] T.K. Ho and H.S. Baird. Evaluation of OCR Accuracy Using Synthetic Data. In *Proc. 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 413–422, Las Vegas, Nevada, USA, 1995.

[40] T.K. Ho and H.S. Baird. Large-Scale Simulation Studies in Image Pattern Recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(10):1067–1079, 1997.

[41] J.M. Hollerbach. An Oscillation Theory of Handwriting. *Biological Cybernetics*, 39:139–156, 1981.

[42] S. Impedovo, P.S.P. Wang, and H. Bunke, editors. *Automatic Bankcheck Processing.* World Scientific, 1997.

[43] S. Johansson, G.N. Leech, and H. Goodluck. *Manual of Information to Accompany the Lancaster-Oslo/Bergen Corpus of British English, for Use with Digital Computers.* Department of English, University of Oslo, Oslo, Norway, 1978.

[44] T. Kanungo, R.M. Haralick, and I. Phillips. Global and Local Document Degradation Models. In *Proc. 2nd Int. Conf. on Document Analysis and Recognition*, pages 730–734, Tsukuba Science City, Japan, 1993.

[45] T. Kanungo, G.A. Marton, and O. Bulbul. Paired Model Evaluation of OCR Algorithms. Technical Report LAMP-TR-030/CAR-TR-903/CS-TR-3972, University of Maryland, College Park, Maryland, USA, 1998.

[46] R. Kasturi, L. O'Gorman, and V. Govindaraju. Document Image Analysis: A Primer. *Sādhanā*, 27(1):3–22, 2002.

[47] G. Kaufmann, H. Bunke, and T.M. Ha. Recognition of Cursively Handwritten Words Using a Combined Normalization/Perturbation Approach. In A.C. Downton and S. Impedovo, editors, *Progress in Handwriting Recognition*, pages 21–28. World Scientific, 1997.

[48] E. Kavallieratou, N. Fakotakis, and G.Kokkinakis. An Unconstrained Handwriting Recognition System. *Int. Journal on Document Analysis and Recognition*, 4(4):226–242, 2002.

[49] D. Kilchhofer. Synthetisches Generieren von Handschriftlichen Textzeilen. Master's thesis, IAM, Universität Bern, Bern, 2004.

[50] G. Kim and V. Govindaraju. Handwritten Phrase Recognition as Applied to Street Name Images. *Pattern Recognition*, 31(1):41–51, 1998.

[51] S.H.K. Kim, C.B. Jeong, H.K. Kwag, and C.Y. Suen. Word Segmentation of Printed Text Lines based on Gap Clustering and Special Symbol Detection. In *Proc. 16th Int. Conf. on Pattern Recognition*, pages 320–323, Quebec City, Canada, 2002.

[52] S.H.K. Kim, G.-S. Lee, and C.Y. Suen. Word Segmentation in Handwritten Korean Text Lines based on Gap Clustering Techniques. In *Proc. 6th Int. Conf. on Document Analysis and Recognition*, pages 189–193, Seattle, WA, USA, 2001.

[53] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. On Combining Classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[54] A.L. Koerich, R. Sabourin, and C.Y. Suen. Large Vocabulary Off-line Handwriting Recognition: A Survey. *Pattern Analysis & Applications*, 6(2):97–121, 2003.

[55] L.I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[56] A. Kundu. Handwritten Word Recognition Using Hidden Markov Model. In H. Bunke and P.S.P. Wang, editors, *Handbook of Character Recognition and Document Image Analysis*, pages 157–182. World Scientific, 1997.

[57] D.-H. Lee and H.-G. Cho. A New Synthesizing Method for Handwriting Korean Scripts. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 12(1):46–61, 1998.

[58] X. Li, M. Parizeau, and R. Plamondon. Segmentation and Reconstruction of On-line Handwritten Scripts. *Pattern Recognition*, 31(6):675–684, 1998.

[59] M.D. Lillibridge, M. Abadi, K. Bharat, and A.Z. Broder. Method for Selectively Restricting Access to Computer Systems. U.S. Patent No. 6,195,698, February 2000.

[60] N. Lindgren. Machine Recognition of Human Language, Part III – Cursive Script Recognition. *IEEE Spectrum*, 2:104–116, 1965.

[61] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten Digit Recognition: Benchmarking of State-of-the-Art Techniques. *Pattern Recognition*, 36(10):2271–2285, 2003.

[62] G. Lorette. Handwriting Recognition or Reading? – What is the Situation at the Dawn of the 3rd Millenium? *Int. Journal on Document Analysis and Recognition*, 2(1):2–12, 1999.

[63] Y. Lu and M. Sridar. Character Segmentation in Handwritten Words – An Overview. *Pattern Recognition*, 29(1):77–96, 1996.

[64] S. Madhvanath and V. Govindaraju. The Role of Holistic Paradigms in Handwritten Word Recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(2):149–164, 2001.

[65] U. Mahadevan and R.C. Nagabushnam. Gap Metrics for Word Separation in Handwritten Lines. In *Proc. 3rd Int. Conf. on Document Analysis and Recognition*, pages 124–127, Montreal, Canada, 1995.

[66] U. Mahadevan and S.N. Srihari. Hypotheses Generation for Word Separation in Handwritten Lines. In *Proc. 5th Int. Workshop on Frontiers in Handwriting Recognition*, pages 453–456, Essex, United Kingdom, 1996.

[67] R. Manmatha and N. Srimal. Scale Space Technique for Word Segmentation in Handwritten Documents. In M. Nielsen, P. Johansen, O.F. Olsen, and J. Weickert, editors, *Scale-Space Theories in Computer Vision*, volume 1682 of *Lecture Notes in Computer Science*, pages 22–33. Springer, 1999.

[68] A. Manzanera and T.M. Bernard. Improved Low Complexity Fully Parallel Thinning Algorithm. In *Proc. 10th Int. Conf. on Image Analysis and Processing*, pages 215–220, Venice, Italy, 1999.

[69] J. Mao and K.M. Mohiuddin. Improving OCR Performance Using Character Degradation Models and Boosting Algorithm. *Pattern Recognition Letters*, 18:1415–1419, 1997.

[70] V. Märgner and M. Pechwitz. Synthetic Data for Arabic OCR System Development. In *Proc. 6th Int. Conf. on Document Analysis and Recognition*, pages 1159–1163, Seattle, WA, USA, 2001.

[71] U.-V. Marti and H. Bunke. Text Line Segmentation and Word Recognition in a System for General Writer Independent Handwriting Recognition. In *Proc. 6th Int. Conf. on Document Analysis and Recognition*, pages 159–163, Seattle, WA, USA, 2001.

[72] U.-V. Marti and H. Bunke. Using a Statistical Language Model to Improve the Performance of an HMM-based Cursive Handwriting Recognition System. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 15(1):65–90, 2001.

[73] U.-V. Marti and H. Bunke. The IAM-Database: an English Sentence Database for Off-line Handwriting Recognition. *Int. Journal on Document Analysis and Recognition*, 5(1):39–46, 2002.

[74] M. Mori, A. Suzuki, A. Shio, and S. Ohtsuka. Generating New Samples from Handwritten Numerals based on Point Correspondence. In *Proc. 7th Int. Workshop on Frontiers in Handwriting Recognition*, pages 281–290, Amsterdam, The Netherlands, 2000.

[75] S. Mori, C.Y. Suen, and K. Yamamoto. Historical Review of OCR Research and Development. In L. O'Gorman and R. Kasturi, editors, *Document Image Analysis*, pages 244–273. IEEE Computer Society Press, 1995.

[76] G. Nagy, T.A. Nartker, and S.V. Rice. Optical Character Recognition: An Illustrated Guide to the Frontier. In *Proc. IS&T/SPIE Symposium on Electronic Imaging: Science and Technology*, volume 3967, pages 58–69, San Jose, CA, USA, 2000.

[77] V.S. Nalwa. Automatic On-Line Signature Verification. *Proceedings of the IEEE*, 85(2):215–239, 1997.

[78] U. Neisser and P. Weene. A Note on Human Recognition of Handprinted Characters. *Information and Control*, 3:191–196, 1960.

[79] N. Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans. on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[80] R. Plamondon. A Kinematic Theory of Rapid Human Movements. Part I: Movement Representation and Generation. *Biological Cybernetics*, 72:295–307, 1995.

[81] R. Plamondon. A Kinematic Theory of Rapid Human Movements. Part II: Movement Time and Control. *Biological Cybernetics*, 72:309–320, 1995.

[82] R. Plamondon. A Kinematic Theory of Rapid Human Movements. Part III: Kinetic Outcomes. *Biological Cybernetics*, 78:133–145, 1998.

[83] R. Plamondon and M. Djioua. Handwriting Stroke Trajectory Variability in the Context of the Kinematic Theory. In *Proc. 12th Conf. of the International Graphonomics Society*, pages 250–254, Salerno, Italy, 2005.

[84] R. Plamondon, C. Feng, and A. Woch. A Kinematic Theory of Rapid Human Movement. Part IV: A Formal Mathematical Proof and New Insights. *Biological Cybernetics*, 89:126–138, 2003.

[85] R. Plamondon and W. Guerfali. The Generation of Handwriting with Delta-lognormal Synergies. *Biological Cybernetics*, 78:119–132, 1998.

[86] R. Plamondon, D.P. Lopresti, L.R.B. Schomaker, and R. Srihari. On-Line Handwriting Recognition. In J.G. Webster, editor, *Encyclopedia of Electrical and Electronics Engineering*, volume 15, pages 123–146. Wiley-Interscience, 1999.

[87] R. Plamondon and S. Srihari. On-line and Off-line Handwriting Recognition: A Comprehensive Survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[88] L. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[89] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[90] S.V. Rice, F.R. Jenkins, and T.A. Nartker. The Fifth Annual Test of OCR Accuracy. Technical Report ISRI-TR-96-01, University of Nevada, Las Vegas, Nevada, USA, 1996.

[91] F. Roli, J. Kittler, and T. Windeatt, editors. *Proc. 5th Int. Workshop on Multiple Classifier Systems*, Cagliari, Italy, 2004. Springer.

[92] R. Rosenfeld. Two Decades of Statistical Language Modeling: Where do We Go from Here? *Proc. of the IEEE*, 88(8):1270–1278, 2000.

[93] H.A. Rowley, M. Goyal, and J. Bennett. The Effect of Large Training Set Sizes on Online Japanese Kanji and English Cursive Recognizers. In *Proc. 8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 36–40, Niagara-on-the-Lake, Ontario, Canada, 2002.

[94] A. Rusu and V. Govindaraju. Handwritten CAPTCHA: Using the Difference in the Abilities of Humans and Machines in Reading Handwritten Words. In *9th Int. Workshop on Frontiers in Handwriting Recognition*, pages 226–231, Kokubunji, Tokyo, Japan, 2004.

[95] K.M. Sayre. Machine Recognition of Handwritten Words: A Project Report. *Pattern Recognition*, 5(3):213–228, 1973.

[96] G. Seni and E. Cohen. External Word Segmentation of Off-line Handwritten Text Lines. *Pattern Recognition*, 27(1):41–52, 1994.

[97] A.W. Senior and A.J. Robinson. An Off-line Cursive Handwriting Recognition System. *IEEE Trans. on Patten Analysis and Machine Intelligence*, 20(3):309–321, 1998.

[98] S. Setlur and V. Govindaraju. Generating Manifold Samples from a Handwritten Word. *Pattern Recognition Letters*, 15:901–905, 1994.

[99] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[100] P. Simard, Y.L. Cun, and J. Denker. Efficient Pattern Recognition Using a New Transformation Distance. In S. Hanson et. al., editor, *Advances in Neural Information Processing Systems 5*, pages 50–58. Morgan Kaufmann, 1993.

[101] P. Simard, R. Szeliski, J. Couvreur, and I. Calinov. Using Character Recognition and Segmentation to Tell Computer from Humans. In *Proc. 7th Int. Conf. on Document Analysis and Recognition*, pages 418–423, Edinburgh, Scotland, 2003.

[102] J.-C. Simon. Off-line Cursive Word Recognition. *Proceedings of the IEEE*, 80(7):1150–1161, 1992.

[103] P. Soille. *Morphological Image Analysis*. Springer, 1999.

[104] M. Sridar and F. Kimura. Segmentation-based Cursive Handwriting Recognition. In H. Bunke and P.S.P. Wang, editors, *Handbook of Character Recognition and Document Image Analysis*, pages 123–156. World Scientific, 1997.

[105] S.N. Srihari. Handwritten Address Interpretation: a Task of Many Pattern Recognition Problems. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 14(5):663–674, 2000.

[106] T. Steinherz, E. Rivlin, and N. Intrator. Offline Cursive Script Word Recognition – a Survey. *Int. Journal on Document Analysis and Recognition*, 2(2):90–110, 1999.

[107] D.G. Stork. Toward a Computational Theory of Data Acquisition and Truthing. In D. Helmbold and B. Williamson, editors, *Computational Learning Theory*, volume 2111 of *Lecture Notes in Computer Science*, pages 194–207. Springer, 2001.

[108] C. Y. Suen, M. Berthod, and S. Mori. Automatic Recognition of Hand Printed Characters – The State of the Art. *Proceedings of the IEEE*, 68(4):469–487, 1980.

[109] C.Y. Suen and R.J. Shillman. Low Error Rate Optical Character Recognition of Unconstrained Handprinted Letters based on a Model of Human Perception. *IEEE Trans. on Systems, Man, and Cybernetics*, 7:491–495, 1977.

[110] D. Trier, A.K. Jain, and T. Taxt. Feature Extraction Methods for Character Recognition – a Survey. *Pattern Recognition*, 29(4):641–662, 1996.

[111] C.H. Tung and H.J. Lee. Performance Analysis of an OCR System via a Handwritten Character Generator. *Pattern Recognition*, 27(2):221–232, 1994.

[112] O. Velek, Ch.-L. Liu, and M. Nakagawa. Generating Realistic Kanji Character Images from On-line Patterns. In *Proc. 6th Int. Conf. on Document Analysis and Recognition*, pages 556–560, Seattle, WA, USA, 2001.

[113] O. Velek and M. Nakagawa. The Impact of Large Training Sets on the Recognition Rate of Off-line Japanese Kanji Character Classifiers. In *Proc. 5th IAPR Workshop on Document Analysis Systems*, pages 106–109, Princeton, New Jersey, USA, 2002.

[114] B. Verma, M. Blumenstein, and S. Kulkarni. Recent Achievements in Off-line Handwriting Recognition Systems. In *Proc. 6th Int. Conf. on Computational Intelligence and Multimedia Applications*, pages 27–33, Melbourne, Australia, 1998.

[115] A. Vinciarelli. A Survey on Off-line Cursive Word Recognition. *Pattern Recognition*, 35(7):1433–1446, 2002.

[116] A. Vinciarelli, S. Bengio, and H. Bunke. Offline Recognition of Unconstrained Handwritten Texts Using HMMs and Statistical Language Models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(6):709–720, 2004.

[117] L. Vuurpijl, R. Niels, M. Erp, L. Schomaker, and E. Ratzlaff. Verifying the UNIPEN devset. In *Proc. 9th Int. Workshop on Frontiers in Handwriting Recognition*, pages 586–591, Kokubunji, Tokyo, Japan, 2004.

[118] J. Wang, C. Wu, Y.-Q. Xu, H.-Y. Shum, and L. Ji. Learning based Cursive Handwriting Synthesis. In *Proc. 8th Int. Workshop on Frontiers in Handwriting Recognition*, pages 157–162, Niagara-on-the-Lake, Ontario, Canada, 2002.

[119] X. Ye, M. Cheriet, and C.Y. Suen. A Generic Method of Cleaning and Enhancing Handwritten Data from Business Forms. *Int. Journal on Document Analysis and Recognition*, 4(2):84–96, 2001.

[120] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book*. Entropic, 2002.

[121] M. Zimmermann and H. Bunke. N-Gram Language Models for Offline Handwritten Text Recognition. In *Proc. 9th Int. Workshop on Frontiers in Handwriting Recognition*, pages 203–208, Kokubunji, Tokyo, Japan, 2004.

Appendix A

Minimum Spanning Tree Clustering

In this appendix, the segmentation technique described in Section 5.1 is generalized for arbitrary sets of objects having distances among them. That is, the objects do not need to be linearly aligned like in the case of text line segmentation. The result is a generic divisive hierarchical clustering algorithm, where the set of objects to cluster as well as the distances among them are represented by a weighted graph G.

Based on a straightforward generalization of the group concept of Section 5.1, it is proved that the minimum spanning tree (MST) of G contains all the information needed to perform the clustering, i.e. to build the unique hierarchy of clusters. Thus the use of the MST is justified mathematically.

A.1 Basics and Notations

Suppose we have a connected, non-empty, weighted undirected graph G, whose vertices represent objects and the cost (or weight) of an edge between two vertices represents the (finite) distance between the corresponding two objects.[1]

The following notations are used in the appendix:

- $\boldsymbol{MST}$**:** a minimum spanning tree of a graph.
- $\boldsymbol{S}$**:** a sub-graph of some graph.
- $\boldsymbol{V(g)}$**:** the set of vertices of graph g.
- $\boldsymbol{c(e)}$**:** the cost of edge e.

[1] In practice, G is a complete graph, but the clustering method will be defined for the more general case of connected graphs.

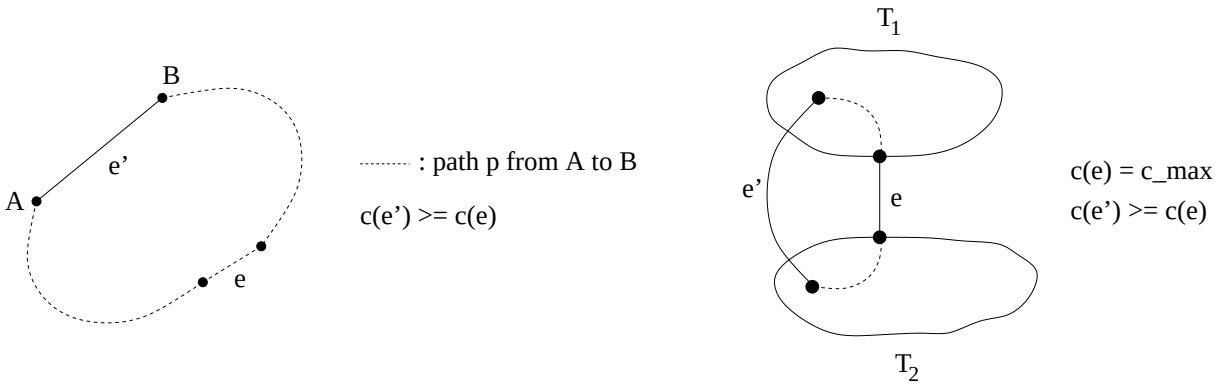


Figure A.1: Illustration of a) Theorem 1, and b) Theorem 2.

A.2 Properties of the Minimum Spanning Tree

Theorem 1 *Let p be the (unique) path from vertex A to vertex B in an MST of graph g. If there is an edge e' of g between A and B, then $c(e') \geq c(e)$, where e is an arbitrary edge on p.*

Proof: If $c(e') < c(e)$ was true for any e edge on p, then substituting e by e' would result in a spanning tree with a smaller cost, which is a contradiction. (See also Fig. A.1/a.) □

Theorem 2 *Let the maximal cost in an MST of graph g be c_{max}, and let S be an arbitrary connected subgraph of g containing all the vertices of g. Then the maximal cost in S is greater or equal than c_{max}.*[2]

Proof: Let e be an edge of the *MST* with $c(e) = c_{max}$. This edge divides the *MST* into two parts, T_1 and T_2, which parts are connected only by edge e. Since S is connected, there is an edge e' of S that connects T_1 with T_2. Because of Theorem 1, $c(e') \geq c(e)$ must hold. (See also Fig. A.1/b.) □

A.3 Group Concept and its Relation to the *MST*

Definition 1 (group condition) *Let S be a non-empty, connected sub-graph of graph G. We say that S satisfies the group condition iff $\forall e, e' \colon \alpha \cdot c(e) < c(e')$, where e is an*

[2]In other words, in an *MST* not only the sum of the costs but also the maximal cost is minimal.

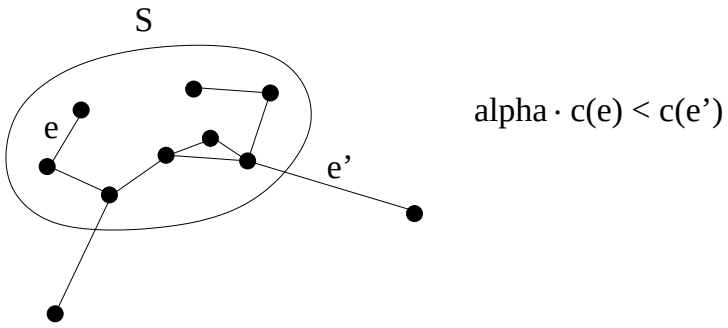


Figure A.2: Illustration of the group condition.

edge of S, e' is an edge of G going out of S (i.e. e' is an edge that goes between a vertex of S and a vertex that does not belong to S), and $\alpha \geq 1$ is a predefined parameter. (See also Fig. A.2.)

Definition 2 (group concept) *Let H be a non-empty subset of the vertices of graph G. H is a group (in G) iff there exists an S connected subgraph of G, such that $V(S) = H$, and S satisfies the group condition.*

Corollary 1 *For any $\alpha \geq 1$, a single vertex $v \in V(G)$ is a group (*singleton group*), and the same is true for $V(G)$ (*trivial group*).*

Lemma 1 *If $H \subseteq V(G)$ is a group, then it induces a sub-tree in any MST of G.*

Proof: Since H is a group, it has a corresponding $S \subseteq G$, where $V(S) = H$ (see Def. 2). Now suppose that H does not induce a sub-tree in an MST, but an F forest of n sub-trees $T_1, \ldots, T_n$, where $n \geq 2$ (see Fig. A.3). To make F become a T' tree (i.e. connected), add $n-1$ edges of S to F (and respectively to the MST, resulting in an MST' graph). This can be done, since S is connected. But now we have cycle(s) in MST'. Each cycle has to go through an edge that goes out of T', because T' does not contain any cycles, since it is a tree. So next, to eliminate the cycles, remove $n-1$ edges from MST', in an iterative manner, that go out of T' and cause a cycle. Since the edges of S that were added to F have smaller costs than those going out of T' (see Def. 1, and observe that $V(S) = V(F) = V(T') = H$), we get a spanning tree with a smaller cost than that of the MST after the last, cycle eliminating step, which is a contradiction. □

Theorem 3 *If H is a group in G, then H is also a group in an arbitrary MST of G.*

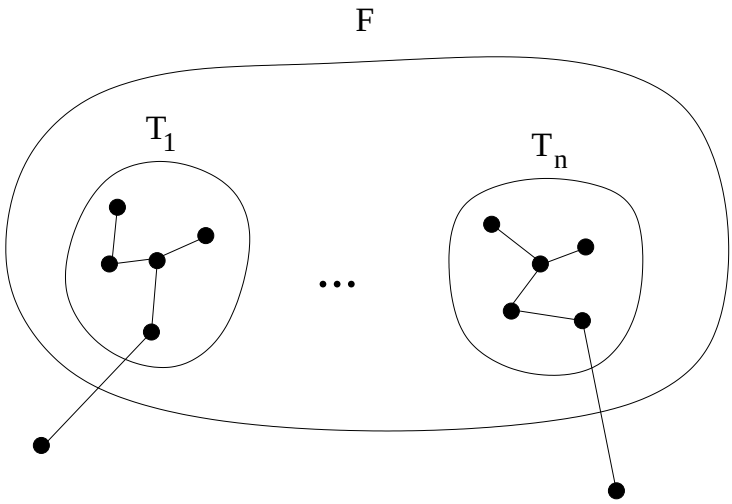


Figure A.3: Illustration to the proof of Lemma 1.

Proof: Since H is a group, it has a corresponding $S \subseteq G$, where $V(S) = H$ (see Def. 2). Let T denote the corresponding sub-tree of an arbitrary MST of G, where $H = V(T)$. Such T exists because of Lemma 1. It can be easily seen that T — being part of a minimum spanning tree — has to be a minimum spanning tree itself, of the sub-graph of G induced by the vertex set H. That's why its maximal cost has to be less or equal than the maximal cost in S, according to Theorem 2. Consequently, T satisfies the group condition. This means that H remains a group also if we restrict G to the MST. □

Theorem 4 *If H is a group in an MST of G, then H is also a group in G.*

Proof: We prove that the sub-tree T of the MST induced by H satisfies the group condition also in graph G. Let e' be an edge of the MST that goes out of T, having a minimal cost among such edges. Since T satisfies the group condition in the MST, its enough to show that for any e'' edge in G that goes out of T, $c(e'') \geq c(e')$ holds. This is true, because if there was such an edge e'' with $c(e'') < c(e')$, then adding e'' to the MST would result in a cycle that would have two edges (including e'') going out of T. Then substituting the edge different from e'' by e'' would yield a spanning tree with a smaller cost than that of the MST, which is a contradiction. □

Corollary 2 *It is enough to consider an arbitrary MST of G to extract the groups.*

Theorem 5 *Two groups, H_1 and H_2, are either disjoint or one of them completely includes the other.*

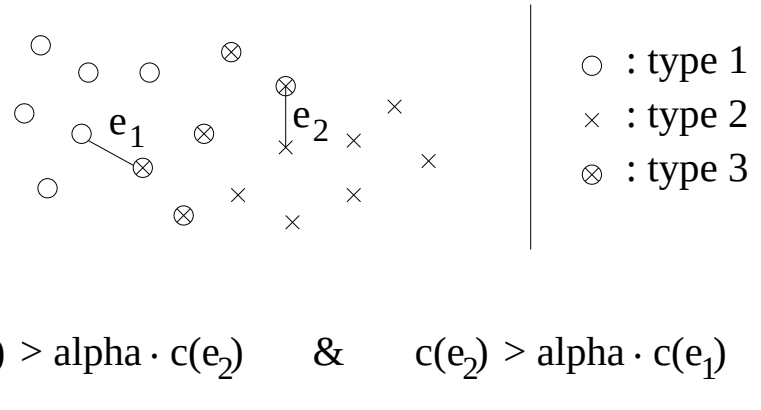


Figure A.4: Illustration to the proof of Theorem 5.

Proof: Let S_1 and S_2 denote the corresponding sub-graphs of G, for H_1 and H_2, respectively, according to Def. 2. Suppose that H_1 and H_2 are neither disjoint nor one of them is a subset of the other. Then, there are three kinds of vertices in $H_1 \cup H_2$: the ones belonging only to H_1 (type 1), the ones belonging only to H_2 (type 2), and the ones belonging to both H_1 and H_2 (type 3). Since S_1 is connected, there has to be at least one edge e_1 between a type 1 vertex and a type 3 vertex. Similarly, there has to be an edge between a type 2 vertex and a type 3 vertex (see Fig. A.4). Since e_1 is inside S_1 and e_2 goes out of S_1, the group condition implies that $c(e_2) > \alpha \cdot c(e_1)$ must hold. Due to similar reasons, $c(e_1) > \alpha \cdot c(e_2)$ must also hold, which is a contradiction since $\alpha \geq 1$. □

Corollary 3 *The division of a non-singleton group $H \subseteq V(G)$ into a minimal number, $m \geq 2$, of disjoint sub-groups is unique (i.e. the* minimal division *of a non-singleton group is unique).*

Corollary 4 *The* structure tree *of $V(G)$ can be built, and it is unique (the definition as well as the other properties are the same as written in Subsection 5.1.4).*

Corollary 5 *The clustering task of Section 5.1 is the special case where the minimum spanning tree of G consists of one single path only.*

Furthermore, if $\alpha = 1$ and all the costs in G are different, the structure tree will be the same as the one we would get by the single-linkage hierarchical clustering algorithm [24]. Several different variants of the group condition can be elaborated, but those are beyond the scope of the present thesis.

Curriculum Vitae

Personal details

First Name:	Tamás
Last Name:	Varga
Date of Birth:	March 19^{th}, 1978
Place of Birth:	Szigetvár, Hungary
Address:	Mühledorfstrasse 28/601 CH-3018 Bern, Switzerland
Nationality:	Hungarian

Education

2001 - 2006	Ph.D. in Science (expected by January 2006), at University of Bern, Switzerland
1999 - 2001	Master of Science, Study of Economics, Computer Science, and Mathematics at University of Szeged, Hungary
1996 - 1999	Bachelor of Science, Study of Computer Science and Mathematics at József Attila University, Szeged, Hungary
1992 - 1996	Matura, Study at Zrínyi Miklós High School, Szigetvár, Hungary
1984 - 1992	Study at Tinódi Lantos Sebestyén Primary School, Szigetvár, Hungary

Publications

Journal Articles and Book Chapters

T. Varga and H. Bunke, "Offline Handwriting Recognition Using Synthetic Training Data Produced by Means of a Geometrical Distortion Model", *Int. Journal of Pattern Recognition and Artificial Intelligence*, Vol. 18, No. 7, pages 1285–1302, 2004

H. Bunke and T. Varga, "Off-line Roman Cursive Handwriting Recognition", *to appear*

Conference Papers

T. Varga and H. Bunke, "Generation of Synthetic Training Data for an HMM-based Handwriting Recognition System", *Proc. 7th Int. Conf. on Document Analysis and Recognition*, Vol. 1, pages 618–622, Edinburgh, Scotland, 2003

T. Varga and H. Bunke, "Effects of Training Set Expansion in Handwriting Recognition Using Synthetic Data", *Proc. 11th Conf. of the Int. Graphonomics Society*, pages 200–203, Scottsdale, Arizona, USA, 2003

T. Varga and H. Bunke, "Off-line Handwritten Textline Recognition Using a Mixture of Natural and Synthetic Training Data", *Proc. 17th Int. Conf. on Pattern Recognition*, Vol. 2, pages 545–549, Cambridge, United Kingdom, 2004

T. Varga and H. Bunke, "Comparing Natural and Synthetic Training Data for Off-line Cursive Handwriting Recognition", *Proc. 9th Int. Workshop on Frontiers in Handwriting Recognition*, pages 221–225, Kokubunji, Tokyo, Japan, 2004

T. Varga, D. Kilchhofer, and H. Bunke, "Template-based Synthetic Handwriting Generation for the Training of Recognition Systems", *Proc. 12th Conf. of the Int. Graphonomics Society*, pages 206–211, Salerno, Italy, 2005

T. Varga and H. Bunke, "Tree Structure for Word Extraction from Handwritten Text Lines", *Proc. 8th Int. Conf. on Document Analysis and Recognition*, Vol. 1, pages 352–356, Seoul Olympic Parktel, Seoul, Korea, 2005