

Off-line Cursive Handwriting Recognition using Hidden Markov Models

H. Bunke¹, M. Roth¹, E.G. Schukat-Talamazzini²

¹Institut für Informatik und angewandte Mathematik
Universität Bern, Länggassstrasse 51, CH-3012 Bern, Switzerland
E-mail: bunke@iam.unibe.ch

²Lehrstuhl für Informatik 5 (Mustererkennung)
Friedrich-Alexander-Universität Erlangen-Nürnberg
Martensstrasse 3, D-91056 Erlangen, Germany
E-mail: schukat@informatik.uni-erlangen.de

Abstract

A method for the off-line recognition of cursive handwriting based on Hidden Markov Models (HMMs) is described. The features used in the HMMs are based on the arcs of skeleton graphs of the words to be recognized. An average correct recognition rate of over 98% on the word level has been achieved in experiments with cooperative writers using two dictionaries of 150 words each.

CR categories and Subject Description: I.5: Pattern Recognition; I.4: Image Processing

General Terms: Algorithms

Additional key words: Optical character recognition, cursive script recognition, off-line recognition, hidden Markov model, skeleton graphs

1 Introduction

Optical character recognition (OCR) has been receiving much attention in the past few years although it is one of the oldest and most intensively studied subfields of pattern recognition [1]. The growing interest in this field has been driven by the steadily increasing power of computing machinery and an expanding commercial market. In OCR, we usually distinguish between the recognition of machine and hand printed text. The latter category can be further decomposed into the recognition of isolated characters and cursive script. For both isolated characters and cursive script, on-line and off-line recognition methods have been proposed. In on-line recognition, an electronic pen or a mouse is used as writing device, and the movement of this device as a function of time is recorded and taken as input data. In off-line recognition, by contrast, only the spatial distribution of the pixels belonging to script patterns, without any temporal information, is available.

The recognition of machine printed characters is considered a mature area and commercial products for a broad spectrum of applications are available today. Reviews of the state of the art can be found in [2],[3]. The field of handwriting recognition is less developed. During the past few years, most of the attention in handwriting recognition was directed to the on-line mode. This interest in on-line recognition has been stimulated by new “pen-type” computer interfaces that are currently under design to replace the traditional keyboard in a variety of applications. The two most recent surveys on on-line handwriting recognition are provided in [4],[5], where ref. [5] particularly focuses on oriental languages. More recently, neural networks [6]–[9], hidden Markov models [10],[11] and dynamic programming matching of character hypotheses against a dictionary [12] have been proposed for the on-line recognition of cursive handwriting.

Off-line recognition of both isolated characters and cursive script has applications in areas like postal address reading, or form and check processing at banks and insurance companies. Progress in off-line recognition of isolated characters achieved in the past years is quite remarkable. In two recent surveys, recognition rates of up to 99.5% and reliability of up to 100% have been reported [13],[14]. By contrast, the problem of off-line cursive handwriting recognition is still widely unsolved. The recognition rates reported in the literature vary between 50% and 96% depending on the experimental conditions and task definition. In [15], Bozinovic and Srihari described a system for off-line cursive script recognition. They applied a heuristic segmentation algorithm to create letter hypotheses, and a dictionary search to find the most plausible interpretation of an input word. A syntactic approach based on error-correcting parsing was proposed in [16]. Edelman et al. presented a method that is based on matching character prototypes against parts of an input word that may undergo an affine transformation [17]. A lexical verification module was employed for making the final decision about the identity of an unknown input. Simon and Baret extracted so-called descriptive chains, which represent structural features of an input word, and matched them against similar chains obtained from dictionary words [18]. A neural network based approach to off-line cursive script recognition was proposed in [19]. The authors used the quantized directions of the strokes contained in a word as descriptive features.

In the present paper, we propose a method for off-line cursive script recognition based on hidden Markov models (HMMs). The importance of HMMs in the area of speech recognition has been observed several years ago [20]. In the meantime, HMMs have also

been successfully applied to image pattern recognition problems such as shape classification [21] and face recognition [22]. HMMs qualify as a suitable tool for cursive script recognition for a number of reasons. First, they are stochastic models that can cope with noise and pattern variations occurring in human handwriting. Next, the number of tokens representing an unknown input word may be of variable length. This is a fundamental requirement in cursive handwriting because the lengths of the individual input words may greatly vary. Moreover, using an HMM-based approach, the segmentation problem, which is extremely difficult and error prone, can be avoided. This means that the features extracted from an input word need not necessarily correspond with letters. Instead, they may be derived from part of one letter, or from several letters. Thus the operations carried out by an HMM are in some sense holistic, combining local feature interpretation with contextual knowledge. Finally, there are standard algorithms known from the literature for both training and recognition using HMMs. These algorithms are fast and can be implemented with reasonable effort.

HMMs have been applied to the problem of off-line cursive script recognition by a number of researchers in a variety of ways. Kundu and Bahl built an HMM for the English language [23]. However, they require the input word being perfectly segmented into single characters. In a later paper, Chen and Kundu showed several other possibilities to apply HMMs to cursive script recognition where the segmentation algorithm is allowed to split a character into up to four parts [24]. Under “real world” conditions they obtained recognition rates between 74.5% and 96.8% depending on the size of the dictionary and the HMM architecture actually used. In [25] a status report about a system that is under development by Kaltenmeier et al. was given. The authors applied detectors that look for strokes of special shape within a sliding window in order to extract features. At present, the recognition rate of the system is about 85% on data from real mail addresses. An approach that assumes segmentation of a word into parts of letters like loops or upper and lower extensions was described in [26]. The authors have studied two different HMM topologies, one for large and another for small size vocabularies. The recognition rate of this system lies between 79% and 91%. It is dependent on the vocabulary size and the quality of the input data.

The approach proposed in this paper differs from other HMM-based methods described in the literature mainly in the type of features extracted from the input. We are using shape descriptors of the skeleton graph edges of an input word. These features are somehow at an intermediate level of abstraction, providing a good compromise between discriminatory power and extraction reliability and reproducibility. For example, they probably provide more distinctive information than the low level features used in [25]. But on the other hand, they can certainly be more reliably extracted than high level features such as complete letters [23]. It must be also mentioned that our motivation is different from that behind the work reported in [24]–[26]. Other authors aimed at the recognition of difficult data collected from real mail or bank checks. By contrast, we assume a cooperative writer who is willing to adapt his or her personal writing style such that the recognition performance of the system is improved. Such an assumption is realistic in the context of several applications, for example, computer input of personal notes written on regular paper, or data acquisition for banks or insurance companies via personal customer workstations.

The remainder of this paper is organized as follows. In section 2, the image preprocessing and feature extraction procedures used in our system will be described. Then an



Figure 1: An example of binarization, horizontal projection profile computation, and reference line extraction.

introduction to HMMs will be given in section 3. In section 4, more details about the application of HMMs to the problem of cursive script recognition will be provided. Experimental results will be presented in section 5. Finally, a discussion and conclusions will be given in section 6.

2 Image Preprocessing and Feature Extraction

Our input images are of format A4 and include typically 20 lines with four words in each line. The images are scanned at a resolution of 100dpi and binarized using Otsu's method [27]. Then the text lines are extracted using horizontal projection profiles. For each line of text, we extract four reference lines: the lower line, lower baseline, upper baseline, and upper line. These reference lines define three vertical zones, namely the lower, middle, and upper zone. Our procedure for reference lines extraction is similar to the one described in [15]. That is, the lower and upper lines are simply obtained from the minimum and maximum non-zero position in the horizontal projection profile, respectively, while the lower and upper baselines correspond to the local extrema of the first derivative. For a graphical illustration see Fig. 1. In order to become invariant to noise in the horizontal projection profile, a smoothing operation is combined with the computation of the first derivative. Moreover, a few heuristic rules are included in the reference line extraction procedure to handle cases where a complete line of text may not include any strokes in the upper or lower zone.

The process of word and reference line extraction is simplified by the fact that in our experiments we require each writer to use ruled paper with all four reference lines on it. However, the reference lines are not captured by our scanner. More sophisticated methods for word and reference lines extraction are described in [28]. It should be mentioned that our recognition approach does not require any slant correction. Therefore, our preprocessing procedures don't include any slant normalization routines.

After the reference lines have been found, the individual words are extracted using the vertical projection profile. Then some morphological smoothing on the binary image is performed, followed by the computation of the skeleton of each word by means of Tamura's algorithm [29]. A graphical illustration is given in Fig. 2. Note that our skeletons are based on 4-neighborhood.

The skeleton of a word can be interpreted as a graph where all pixels with exactly two neighbors are part of an edge, and pixels with either one, three, or four neighbors are nodes. The features used in the HMM for recognition are based on the edges of the

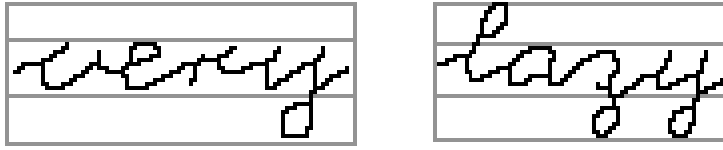


Figure 2: The two words shown in Fig. 1 after skeletonization.

skeleton graph of a word. Therefore, after the skeleton of a word has been obtained, we extract the edges from the skeleton graph.

As any HMM needs its input data in strict sequential order, the most important requirement for our edge extraction procedure is to ensure the edges being extracted in some canonical order. Procedures have been reported in the literature that attempt to recover the temporal stroke sequence from off-line cursive words [30]. However, as the original temporal stroke sequence of the input is not essentially required for recognition, we adopted another approach. The edge extraction procedure used in our implementation is guided by two main principles:

procedure edge_extraction

input: the skeleton graph of a word

output: the sequence $EDGES = (e_1, \dots, e_n)$ containing all edges of the skeleton graph in canonical order /* EDGES is initially empty */

method:

find initial node and put it on a list OPEN; /* OPEN is initially empty */

while OPEN is not empty **do** {

 remove first node i from OPEN and put it on a list CLOSED;

 /* CLOSED is initially empty */

 retrieve all edges e_{1_i}, \dots, e_{n_i} incident to i and remove them from the skeleton graph;

 add all nodes of degree 3 or 4 to OPEN that are incident to e_{1_i}, \dots, e_{n_i} , and are neither on OPEN nor on CLOSED;

 order e_{1_i}, \dots, e_{n_i} (and the corresponding nodes on OPEN) and add them to EDGES }

end edge_extraction

Figure 3: Algorithm for the extraction of the edge sequence from the skeleton graph.

- (i) The edges of each letter of the alphabet should always be extracted in the same order, independent of the context of the letter and of its variations in shape.
- (ii) If letter x has been written before letter y (or spatially speaking, if x is to the left of y), then all edges of x should be extracted before any of the edges of y .

A pseudo code description of our edge extraction procedure is given in Fig. 3. In this procedure, the left-most node of the skeleton graph is usually taken as initial node. However, there are a few exceptions to this rule in order to ensure the validity of principle (i) under the possible variations produced by different writers. Examples of possible

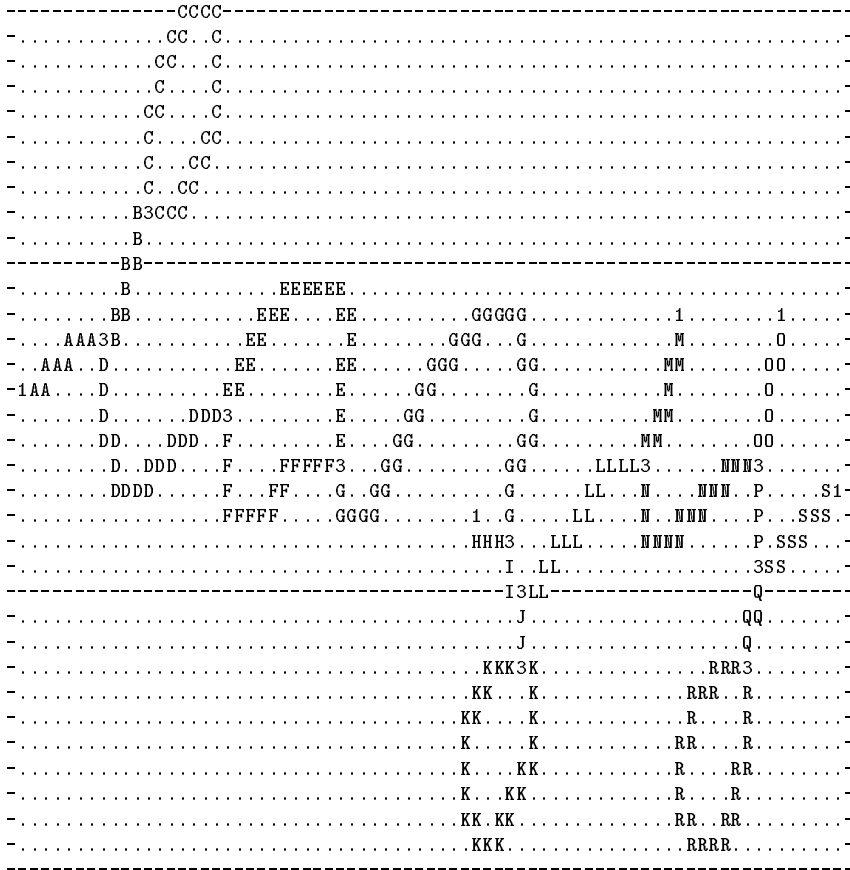


Figure 4: The result of extracting the edges from the second word in Fig. 2.

exceptions are letters like *B* or *I* (see Fig. 8) where the left-most node of the skeleton graph may randomly be either the upper or lower node of degree 1 at the left end of the word. Here, the procedure always picks the upper node regardless if it is in the left-most or the second-left position.

The edge ordering step in the algorithm of Fig. 3 is based on the directions in which the edges e_1, \dots, e_{n_i} leave the actual node i . The order of directions is west, north, south, east. For example, if node i is reached from west — in this case, the western edge has already been deleted from the skeleton graph and added to EDGES —, then an edge leaving i to the north will be included in EDGES before a southern edge, and a southern edge will be added before an eastern edge. For determining the direction of an edge, the first 50% of its pixels are examined.

The result of extracting the edges from the skeleton of the second word in Fig. 2 is shown in Fig. 4. Numbers 1 and 3 represent nodes of the graph and letters A, B, C, ... are used to label the edges. The letters also indicate the order of the edges in EDGES, i.e., A is the first element of EDGES, B the second one, and so on.

There are a few exceptions to the (west, north, south, east)-ordering rule. One affects loops that occur, for example, in letters like ℓ , z , or y (see Fig. 4). If such a loop is detected,

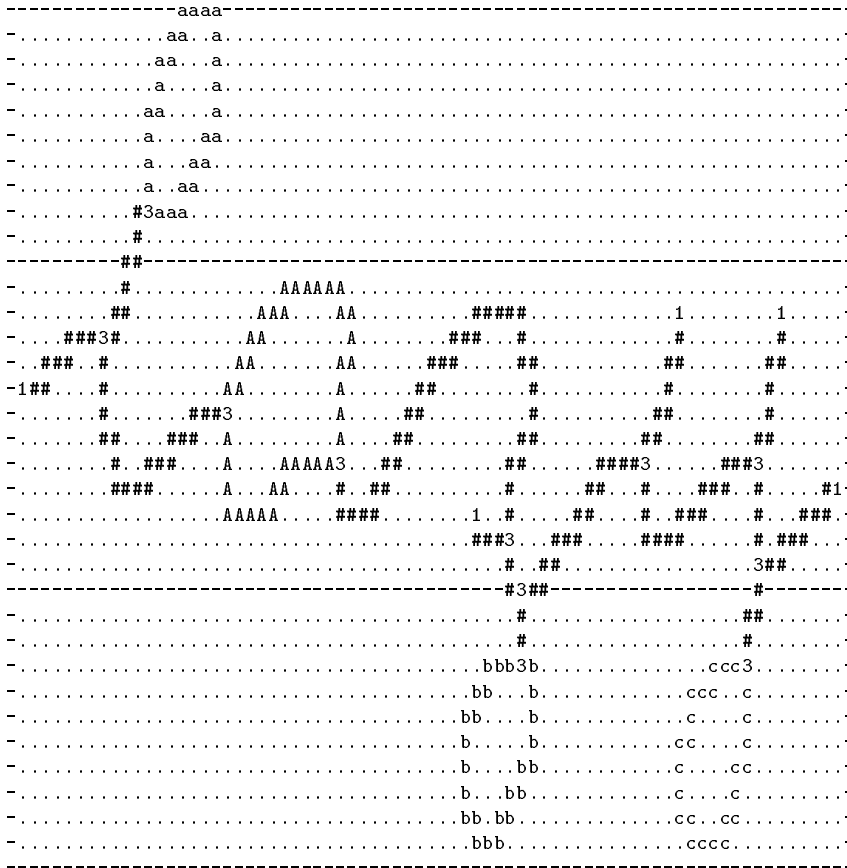


Figure 5: The result of loop extraction.

it will be added to EDGES not as last, but as second-last element. This exception is needed in order to guarantee the validity of the main principle (ii). As an example, the exception changes the order of loop C and edge D in Fig. 4. Similarly, also the order of K and L, and of R and S are changed.

During edge extraction, we check the skeleton graph for loops. This is not only needed for edge ordering. But it is also useful because loops carry important discriminatory information for recognition. For example, valley-shaped edges on the base line occur in many letters, e.g. in *v*, *e*, *y*, *l* or *a* in Fig. 2. But the range of possible interpretations, i.e. character hypotheses, can be reduced if we know whether the edge is part of a loop or not. Therefore, we include this information in the feature vector of an edge. A distinction is made between simple loops, which consist of just one edge, and complex loops including several edges. The result of loop extraction from Fig. 4 is shown in Fig. 5 where lower case and capital letters are used to label simple and complex loops, respectively, and the symbol # denotes non-loop pixels.

Having extracted the edges and loops from the skeleton graph of a word, we transform each edge into a feature vector of fixed length. A total of ten features f_1, \dots, f_{10} are used in our present implementation to describe an edge. Their meaning is described in more

detail below.

Features f_1, \dots, f_4 : These features describe the spatial location of an edge. Let R_1, R_2, R_3 and R_4 denote, from top to bottom, the reference lines shown in Fig. 1. As most of the edges are between R_2 and R_3 , we define a fifth reference line R'_2 exactly in the middle of R_2 and R_3 . Then f_1, \dots, f_4 are defined as the percentage of pixels of an edge lying between R_1 and R_2 , R_2 and R'_2 , R'_2 and R_3 , and R_3 and R_4 , respectively.

Feature f_5 : This is a binary feature indicating whether an edge is incident to a node of degree one or not.

Feature f_6 : This feature is a measure of curvature. It is defined as the ratio of the Euclidean distance of the two endpoints of an edge and its length. For straight lines, f_6 is equal to one while for simple loops it is equal to zero. We set f_6 equal to zero, by definition, if an edge belongs to a complex loop. In this way, some invariance is achieved with respect to breaking a simple loop into several pieces.

Features f_7, \dots, f_{10} : They contain more details about curved edges and are defined as the percentage of pixels lying above the top endpoint, below the bottom endpoint, left of the left-most endpoint, or right of the right-most endpoint of an edge, respectively. For edge **E** in Fig. 4, for example, these features take the following values: $f_7 = 21/23$, $f_8 = 0$, $f_9 = 0$, $f_{10} = 3/23$. Similarly for loop **K** we have $f_7 = 0$, $f_8 = 21/25$, $f_9 = 18/25$, $f_{10} = 5/25$.

It is easy to see that all features f_1 to f_{10} are invariant with respect to translation and scaling. Features f_5 and f_6 are also rotation invariant. Theoretically, the other features are not invariant under rotation. But in practice they are fairly robust under the degree of rotation that occurs in our experimental data. Also, the vector quantization process that maps feature vectors to symbols eliminates smaller variations as similar vectors will be mapped to the same symbol. This is a justification why no slant correction is applied in our approach.

3 Basic Theory and Implementation Tools for HMMs

For the purpose of self-containedness, we provide a brief introduction to the theory of HMMs in this section. Our notation follows the standard literature to which the reader is referred to for further details [20],[31].

Hidden Markov Models (HMM). The Hidden Markov Model is a straightforward generalization of ordinary probability distributions to the case of randomly generated sequences of discrete- or continuous-valued events. A discrete density HMM produces strings $O = O_1 \dots O_T$ of symbols from a finite alphabet $\{v_1, \dots, v_K\}$, while the continuous density version creates sequences of real-valued feature vectors $\mathbf{x} \in \mathbb{R}^d$.

The generation of the observable outputs O_t of the model is controlled by a doubly stochastic process. At each time instant $t = 1, \dots, T$, the model is in one out of N possible states $\{s_1, \dots, s_N\}$. The state q_t taken by the model at time t is a random

variable which depends only on the identity of its immediate predecessor state. According to this assumption the state distribution is completely determined by the parameters

$$\pi_i = p(q_1 = s_i) \quad \text{and} \quad a_{i,j} = p(q_t = s_j \mid q_{t-1} = s_i) \quad ;$$

in other words, the vector $\boldsymbol{\pi} = (\pi_1, \dots, \pi_N)^\top$ of initial probabilities together with the $(N \times N)$ -matrix $\mathbf{A} = [a_{i,j}]$ of transition probabilities form a first-order Markov chain.

The actual state sequence taken by the model serves as a probabilistic trigger for the production of the output sequence. The q_t themselves, however, remain hidden to an observer of the random process. According to a second model assumption, the probability distribution of an output symbol O_t (or an output vector, respectively) depends solely on the identity of the present state q_t ; thus, the distribution parameters

$$b_{j,k} = b_j(v_k) = p(O_t = v_k \mid q_t = s_j)$$

of a *discrete* density HMM constitute a $(N \times K)$ probability matrix $\mathbf{B} = [b_{j,k}]$. Consequently, the behaviour of an HMM with discrete output is entirely specified by the cardinality N of the state space, the alphabet size K , and a parameter array

$$\boldsymbol{\lambda} = (\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$$

of non-negative probabilities, obeying the $(1 + 2N)$ normalization conditions $\sum_i \pi_i = \sum_j a_{i,j} = \sum_k b_{j,k} = 1$.

Any of the state-dependent probability density functions (PDF)

$$b_j(\mathbf{x}) = p(O_t = \mathbf{x} \mid q_t = s_j)$$

of an HMM with *continuous* output can be reasonably well approximated by a convex combination

$$b_j(\mathbf{x}) = \sum_{k=1}^K b_{j,k} \cdot g_{j,k}(\mathbf{x}) = \sum_{k=1}^K b_{j,k} \cdot \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_{j,k}, \boldsymbol{\Sigma}_{j,k})$$

of multivariate Gaussian PDFs. The huge amount of statistical parameters found in a continuous mixture HMM as defined above — the model includes estimates of a distribution mean $\boldsymbol{\mu}_{j,k} \in \mathbb{R}^d$ and a covariance matrix $\boldsymbol{\Sigma}_{j,k} \in \mathbb{R}^{d \times d}$ for each of $(N \cdot K)$ index pairs — can be drastically reduced if all state-dependent sets $\{g_{j,k} \mid k = 1, \dots, K\}$ of mixture components are pooled into one global collection of PDFs. The resulting type of model is termed *semi-continuous* HMM; its output distributions

$$b_j(\mathbf{x}) = \sum_{k=1}^K b_{j,k} \cdot g_k(\mathbf{x}) \quad \text{with} \quad g_k(\mathbf{x}) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

all share the same global set $\mathbf{g} = (g_1, \dots, g_K)$ of Gaussians regardless of the state index j . The semi-continuous model is therefore characterized by the statistics $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \mathbf{A}, \mathbf{B}, \mathbf{g})$, where density function g_k is represented parametrically by its mean vector $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$. Evidently, our notation suggests that $\boldsymbol{\pi}$, \mathbf{A} , and \mathbf{B} can be interpreted as an ordinary discrete HMM and \mathbf{g} as the codebook of a K -class probabilistic (soft) vector quantizer, transforming continuous feature vectors \mathbf{x} into a likelihood array $(g_1(\mathbf{x}), \dots, g_K(\mathbf{x}))^\top$.

Recognition and training. When applied to pattern recognition problems, HMMs serve as generative models for the feature representations of the objects under analysis — spoken words or written words, for instance. In order to estimate the probability $p(O|\boldsymbol{\lambda})$ that a particular model $\boldsymbol{\lambda}$ generated a certain observation O , the expression

$$p(O|\boldsymbol{\lambda}) = \sum_{q_1} \dots \sum_{q_T} \left(\pi_{q_1} b_{q_1}(O_1) \cdot \prod_{t=2}^T a_{q_{t-1}, q_t} b_{q_t}(O_t) \right)$$

with the summations ranging over all possible state sequences $\mathbf{q} = (q_1, \dots, q_T)$ has to be evaluated. Fortunately, the above expression can be reformulated as a set of $N \cdot T$ recursive relations, allowing an efficient computation of $p(O|\boldsymbol{\lambda})$; this procedure is known as the *forward algorithm* (see, for instance, [20]). Provided that appropriate reference models $\boldsymbol{\lambda}_l$ for the competing pattern classes are available, we can decode an observed pattern O by selecting that class label l with maximum a posteriori probability

$$p(\boldsymbol{\lambda}_l | O) = \frac{p_l \cdot p(O | \boldsymbol{\lambda}_l)}{\sum_k p_k \cdot p(O | \boldsymbol{\lambda}_k)} \quad .$$

In many applications, the conditional probabilities on the right hand side of this decision rule are replaced by $p^*(O|\boldsymbol{\lambda}_l) = p(O, \mathbf{q}^*|\boldsymbol{\lambda}_l)$, where \mathbf{q}^* is a state sequence maximizing $p(O, \mathbf{q}|\boldsymbol{\lambda}_l)$. The optimal state sequence \mathbf{q}^* together with the quantity $p^*(O|\boldsymbol{\lambda}_l)$ is computed by the well-known *Viterbi algorithm* which operates usually much faster than the forward algorithm.

Maximum a posteriori classification is known to be the optimal decision rule provided that the involved models are correct with respect to the actual probability distribution of the underlying data. In order to satisfy this requirement, the structure as well as the parameters of the model have to be fitted to an appropriately designed training sample. Firstly, the sizes N and K of the model and the codebook (or, equivalently, the output alphabet) are chosen. In the subsequent training phase, the statistical parameters of the HMM are adapted in order to match the distribution of the training data. We obtain a reasonable estimate $\hat{\boldsymbol{\lambda}}$ of the model parameters by maximizing the total likelihood function

$$\mathcal{L}(\boldsymbol{\lambda}) = \prod_{m=1}^M p(O^{(m)} | \boldsymbol{\lambda})$$

of the particular observation sequences $O^{(1)}, \dots, O^{(M)}$ encountered in the training data. This Maximum-Likelihood estimate can be approximated by an iterative procedure — the *Baum-Welch algorithm* — which converges to a parameter point $\hat{\boldsymbol{\lambda}}$ that is at least locally optimal with respect to the criterion $\mathcal{L}(\boldsymbol{\lambda})$. A comprehensive treatment of Baum-Welch training algorithms for HMMs with different types of discrete, continuous, and semi-continuous output distributions can be found in [32]. It is worth mentioning that the vector quantization codebook itself is part of the semi-continuous model; as such it is subject to the reestimation process, too.

The ISADORA system. The classification stage of the cursive script recognizer described in section 4 is based on the ISADORA system [33]. This tool provides highly flexible HMM-based pattern analysis and the possibility to build hierarchically structured models from simple constituents. Following the theory of Recursive Markov Models [34],

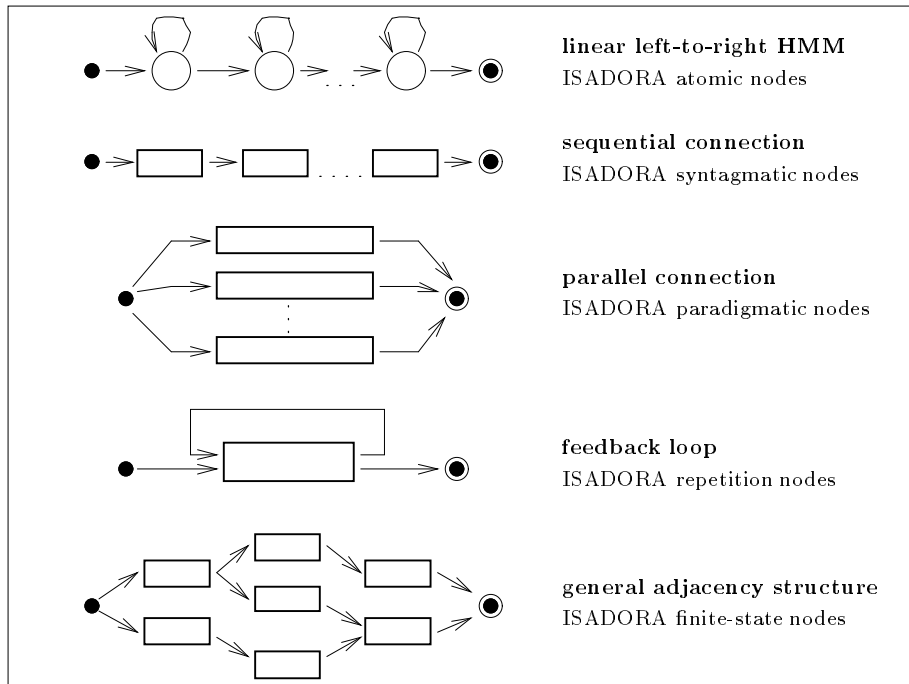


Figure 6: Elementary operations for HMM construction.

a rule mechanism is employed to represent complex objects of the application domain in an acyclic graph structure of nested Markov Models. Each node of this network defines a particular HMM related to the models of its successor nodes as specified by one of the following combination mechanisms (see Fig. 6):

- *Atomic* nodes form the elementary units of the network and, as such, consist of linear left-to-right HMMs, i. e., all transition probabilities $a_{i,j}$ vanish unless $i = j$ or $i = j - 1$.
- *Syntagmatic* nodes denote a sequential concatenation of the successor nodes' models in order to form a larger HMM.
- *Paradigmatic* nodes serve to represent the choice between alternative pattern classes.
- *Repetition* nodes denote an arbitrary reiteration of a single successor object implemented by self-looping the corresponding model.
- The most general means of hierarchical model construction is provided by the type of *Finite State Network* nodes. Its model is obtained by interconnecting all successor models according to a particular adjacency matrix.

Presently, ISADORA provides models with semi-continuous output distributions. A modified version of the Baum-Welch training algorithm supports context-dependent modelling as well as parameter estimation techniques like tying, Bayesian adaptation, and deleted interpolation.

At the decoding stage, ISADORA performs a beam-search driven Viterbi algorithm on the analysis model, trying to find a best-fitting alignment between its states and the input data. The result of this procedure is a nested hypothesis structure providing object identities along with their probabilistic scores as well as their temporal or spatial (depending on the application) boundaries in the input data.

Besides the application to Cursive Script Recognition (see below), ISADORA has been applied in the fields of Automatic Speech Recognition [35, 36] and the diagnosis of sensor data from moving vehicles [37].

4 Application to Cursive Script Recognition

word models:

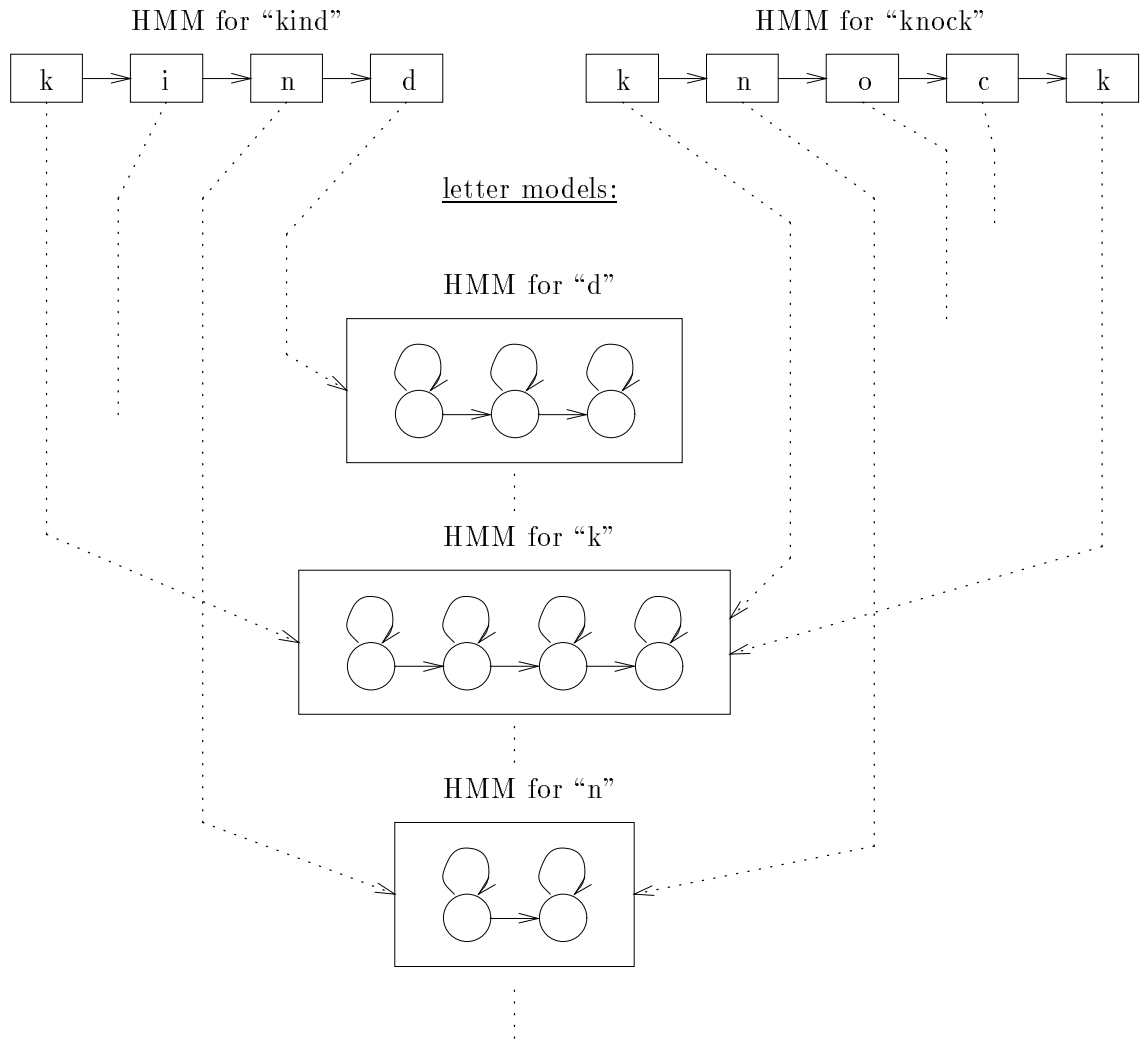


Figure 7: Graphical illustration of the model building process. The dotted lines are references from the word HMMs to the letter HMMs.

The preprocessing and feature extraction procedures generate a sequence $\mathbf{F} = \mathbf{f}_1 \dots \mathbf{f}_T$ of feature vectors for each word, where T is equal to the number of edges in the skeleton graph. Each \mathbf{f}_t consists of ten components $f_{t,1}, \dots, f_{t,10}$ (see section 2). After \mathbf{F} has been generated, each feature vector \mathbf{f}_t has to be transformed into a set of probabilistically weighted labels by soft vector quantization. In order to create an initial quantizer codebook, first the total number K of codebook symbols, or classes, has to be defined. Each codebook symbol should represent one class of similar feature vectors, i. e., edges of the skeleton graph. By manual inspection of the data set and experimentation, a suitable value $K = 28$ has been determined. Given the total number of codebook symbols, the estimate-maximize (EM) algorithm [38] is applied to identify the Gaussian K -mixture distribution of the training sample.

The linear model was adopted as basic topology of our HMMs. Examples of linear models consisting of three, four and two states are included in Fig. 7 (letter models for “d”, “k” and “n”). In the experiments described in more detail in section 5, we aimed at recognizing words from a dictionary of fixed size. In order to model a dictionary word, we first define a linear HMM for each letter of the alphabet. Then an HMM for each dictionary word is built by concatenating the appropriate letter models. Notice that there exists only one HMM per letter regardless of the number of occurrences of the letter in a word or in the dictionary. That is, the same letter model is shared by all instances of that letter occurring within the same word, and within all other dictionary words. A graphical illustration of the way the word models are built from the letter models is shown in Fig. 7. By means of this two-level approach, the number of training samples available for each letter is much larger than by defining a disjoint model for each individual word.

For the definition of the letter models it is important to observe that even when using a reference alphabet, the shape of the letters in handwriting samples delivered by different writers is not always the same. Particularly, the skeleton graphs of different instances of the same letter do not necessarily consist of the same number of edges. Therefore, the number of states in a letter model is not necessarily equal to the number of edges in the skeleton graph of that letter because this number is not uniquely defined. When defining the letter models, we set the number of states equal to the minimal number of edges that can be expected for a letter. Taking the minimal number of edges of the skeleton graph is an immediate consequence of using exclusively linear models. Otherwise, it could happen that a sequence of observation symbols is rejected by the correct HMM because the number of symbols is less than the number of states and, therefore, the HMM cannot reach its final state. In Table 1 the number of states for each letter model is given.

To illustrate the letter model definition process in more detail, consider the skeleton graph in Fig. 4. The minimal number of strokes for an “ l ” is the loop labelled with C and the stroke labelled with D. Clearly, there are instances of an “ l ” where either stroke A or B, or both of them are missing. The minimal number of strokes for an “ a ” is just as represented by the instance in Fig. 4 (edges E and F). Apart from this, there exist also instances of a ’s that have a short vertical stroke attached to the upper right part of edge E, for example. Such an instance of an “ a ” consists of four strokes which is certainly not minimal. Similar arguments apply to the other letters of the alphabet.

For linear HMMs, the initial state distribution vector is always $\pi = (1, 0, \dots, 0)$. In our application the initial probabilities of the output symbols are defined as uniform distributions. The state transition probabilities $a_{i,i+1}$ and $a_{i,i}$ are set to 0.2 and 0.8,

letter model	number of states	letter model	number of states	letter model	number of states	letter model	number of states
A	1	N	3	a	2	n	2
B	6	O	3	b	2	o	2
C	1	P	3	c	1	p	3
D	4	Q	3	d	3	q	3
E	2	R	4	e	2	r	3
F	5	S	3	f	4	s	1
G	3	T	3	g	3	t	3
H	4	U	2	h	3	u	3
I	2	V	4	i	2	v	2
J	3	W	6	j	3	w	3
K	4	X	4	k	4	x	3
L	4	Y	4	l	2	y	3
M	5	Z	3	m	3	z	3

Table 1: The number of states for each letter model.



Figure 8: The reference alphabet for our experiments.

respectively. For training, the Baum-Welch algorithm was used.

Finally, in the recognition phase we input an observation sequence \mathbf{F} that represents an unknown word into each HMM. Let λ_l be the HMM corresponding to the l -th word of the dictionary, $l = 1, \dots, W$. Then \mathbf{F} is classified as the dictionary word l with the maximum a posteriori probability $p(\lambda_l | \mathbf{F})$; a uniform a priori distribution $p_l = 1/W$ of dictionary words is assumed. As already mentioned in the preceding section, approximations of these quantities are computed by the Viterbi algorithm.

5 Experimental Results

In contrast with other approaches, where it is tried to recognize cursive script on mail or bank checks, we experimented with data produced by cooperative writers. That is, we asked our writers to adhere as closely as possible to the writing style taught to first grade pupils at Swiss elementary schools. The alphabet given to our writers as reference is shown in Fig. 8. Furthermore, all writers had to use ruled paper where for each line of text the four reference lines shown in Fig. 1 were given. (However, the reference lines were not captured by the scanner.)

Despite these standardization attempts, the differences in writing style were quite significant. A total of five writers participated in our experiments. In Fig. 9, a complete page written by writer 3 in the second experiment is shown.

Two experiments were carried out in order to assess the performance of the proposed

letter	a	b	c	d	e	f	g	h	i	j	k	l	m
no. of occurrences	72	18	34	31	131	13	27	35	65	6	11	59	24
letter	n	o	p	q	r	s	t	u	v	w	x	y	z
no. of occurrences	60	56	30	7	60	35	54	39	12	13	5	27	5

Table 2: Letter frequencies in experiment 1.

letter	A	B	C	D	E	F	G	H	I	J	K	L	M
no. of occurrences	6	6	6	6	6	6	6	6	6	6	6	6	9
letter	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
no. of occurrences	6	6	6	4	6	7	6	6	6	6	0	4	6
letter	a	b	c	d	e	f	g	h	i	j	k	l	m
no. of occurrences	75	15	18	14	113	22	25	31	54	0	7	53	5
letter	n	o	p	q	r	s	t	u	v	w	x	y	z
no. of occurrences	102	57	7	0	76	47	45	42	7	9	1	8	15

Table 3: Letter frequencies in experiment 2.

approach. In the first experiment, a dictionary of 150 randomly selected English words was used and the HMM built for each word. The complete dictionary is listed in Appendix A. We asked each writer to write each word four times. Thus we obtained four data sets, A, B, C and D, each consisting of 750 words.

We first trained the HMMs with data sets A, B, C and used data set D as test set. Then data sets A, B, D were used for training and C for testing, and so on. Eventually, a test set of 3000 words and a training set of 9000 words were obtained thus with the test data disjoint from the training data. For each word there were 60 training and 20 test instances. Table 2 shows the number of occurrences of the individual letters in the dictionary. As there is only one model for each letter that is shared by all dictionary words, there were $72 \times 60 = 4320$ training and $72 \times 20 = 1440$ test instances of letter *a*, $18 \times 60 = 1080$ training and $18 \times 20 = 360$ test instances of letter *b*, and so on. In this experiment, the substitution error rate on the word level was 1.6%. A more detailed representation of the results, showing the correct recognition rates for the individual writers is given in Table 4. One substitution error corresponds to a recognition rate decrease of 0.66%.

In the second experiment, we used a dictionary consisting of Swiss city and village names. In contrast with the first experiment where only lower case letters occurred, both capital and lower case letters were involved in the second experiment. Similar to the first experiment, the dictionary size was 150 words. For a complete listing of all dictionary words, see Appendix B. The letter frequencies are given in Table 3. All other conditions, like number of writers, size of test and training sets a.s.o., were exactly the same as in the first experiment. In the second experiment, the substitution error rate on the word level was 1.73%, see Table 5.

In Tables 4 and 5, only the top candidates, i.e. the models with maximum a posteriori probability, are taken into account. If we allow the correct word being among the top two (three) candidates, then the substitution error rate can be further reduced to 0.7% (0.4%) in experiment 1, and 0.97% (0.77%) in experiment 2; see Tables 6 to 9.

Our programs run on SUN Sparcstations under UNIX. All preprocessing steps together, starting with binarization and ending with inputting the sequence of symbols to the HMMs, take 0.3s for one word on the average. (Our programs were written with emphasis on flexibility and transparency rather than speed, and there is ample room for

Arosa Burgdorf Cham Dallenwil
Effretikon Frutigen Gaudria Horn
Stach Jona Klosters Langenthal
Murten Naefels Orbe Pratteln
Quinten Rueti Solothurn Thalwil
Urnaesch Vitznau Wohlen Yens
Zofingen Airolo Baden Carouge
Dielsdorf Eglisau Fiesch Gossau
Storgen Stanz Jansbach Koblenz
Locarno Murtten Morges Nidau
Oey Passugg Rapperswil Staefa
Schaffhausen Teufen Tgnach Visp
Mül Zermeg Arbon Bellinzona
Chillon Dornach Estavayer
Flawil Grenchen Hochdorf Isone
Juckern Kreuzlingen Siestal
Meilen Monthey Martigny
Nebikon Oberhofen Port Rotenschach
Stans Trivann Unterseen Valbella
Wetzikon Zurzach

Figure 9: Example of a complete page, written by writer 3.

making them faster.) Matching one input word against one HMM takes less than 5 ms on the average.

writer	test set A	test set B	test set C	test set D	average
1	99.33	96.67	98.67	98.00	98.17
2	97.33	96.67	95.33	97.33	96.67
3	100	99.33	100	100	99.83
4	96.67	96.00	99.33	98.00	97.50
5	100	100	100	99.33	99.83
average	98.67	97.73	98.67	98.53	98.40

Table 4: Results of experiment 1.

writer	test set A	test set B	test set C	test set D	average
1	98.67	94.67	96.00	98.00	96.83
2	98.67	99.33	98.00	99.33	98.83
3	96.67	96.67	96.67	95.33	96.33
4	100	100	99.33	100	99.83
5	99.33	100	98.67	100	99.50
average	98.67	98.13	97.73	98.53	98.27

Table 5: Results of experiment 2.

writer	test set A	test set B	test set C	test set D	average
1	99.33	99.33	99.33	99.33	99.33
2	98.00	98.67	97.33	100	98.50
3	100	100	100	100	100
4	98.00	98.00	100	99.33	98.83
5	100	100	100	99.33	99.83
average	99.07	99.20	99.33	99.60	99.30

Table 6: Results of experiment 1, allowing correct word among the top two candidates.

writer	test set A	test set B	test set C	test set D	average
1	99.33	100	99.33	100	99.67
2	98.67	98.67	98.00	100	98.83
3	100	100	100	100	100
4	99.33	99.33	100	99.33	99.50
5	100	100	100	100	100
average	99.47	99.60	99.47	99.87	99.60

Table 7: Results of experiment 1, allowing correct word among the top three candidates.

writer	test set A	test set B	test set C	test set D	average
1	99.33	96.00	98.00	98.00	97.83
2	99.33	99.33	98.67	100	99.33
3	97.33	98.00	98.00	99.33	98.17
4	100	100	100	100	100
5	100	100	99.33	100	99.83
average	99.20	98.67	98.80	99.47	99.03

Table 8: Results of experiment 2, allowing correct word among the top two candidates.

writer	test set A	test set B	test set C	test set D	average
1	99.33	96.67	98.00	98.00	98.00
2	99.33	99.33	99.33	100	99.50
3	99.33	98.00	98.67	99.33	98.83
4	100	100	100	100	100
5	100	100	99.33	100	99.83
average	99.60	98.80	99.07	99.47	99.23

Table 9: Results of experiment 2, allowing correct word among the top three candidates.

6 Conclusions

An approach to off-line recognition of cursive script based on HMMs was proposed in this paper. There are significant differences in the size of the vocabulary, quality of the data, and task definition, in general, between the handwriting recognition experiments reported in the literature. Therefore, it is perhaps not meaningful to compare the recognition performance of our approach directly to other methods, like those described in [15]–[19] and [23]–[26]. Nevertheless, a correct recognition rate of over 98% can be considered quite satisfactory in the context of any complex OCR problem, even if the data is of “good” quality. One of the conclusions that can be drawn from our experiments is that under the scenario of cooperative writers and a limited dictionary, it is possible to reach a recognition rate in off-line cursive handwriting close to that achieved in isolated handwritten character recognition [13],[14].

So far, we have not yet fully exhausted the potential of HMM methodology as implemented in the ISADORA package. In particular, our letter models are presently insensitive to their surrounding context, and no attempts have been made to build individual models

for significantly different realizations of the same letter.

In the future, we plan to use larger vocabularies and extend our experimental data set by including more writers. The fact that the substitution error rate didn't significantly increase when an alphabet consisting of capital and lower case letters was used instead of lower case letters only is an indication that the limits of the proposed method have not yet been reached.

Acknowledgments: We want to thank Ms. S. Jaskulke for making us available her first grade classnotes. Furthermore, we would like to thank Dr. T. Ha Minh, B. Achermann, D. Niggeler, R. Wittwer, T. Hänni and S. Weiss for participating in the experiments and giving valuable hints.

References

- [1] T. Pavlidis, S. Mori (editors): *Optical Character Recognition*, Special issue of Proc. of the IEEE, vol. 80, no. 7, 1992.
- [2] S. Impedovo, L. Ottaviano, S. Occhinegro: *Optical Character Recognition — A Survey*, Int. Journal of Pattern Recognition and Artificial Intelligence, vol. 5, nos 1 & 2, 1991, pp. 1–24.
- [3] S. Rice, J. Kanai, T.A. Nartker: *A Report on the Accuracy of OCR Devices*, TR-92-02, Information Science Research Institute, University of Nevada, Las Vegas, 1992.
- [4] C.C. Tappert, C.Y. Suen, T. Wakahara: *The State of the Art in On-Line Handwriting Recognition*, IEEE Transactions on PAMI, vol. 12, no. 8, 1990, pp. 787–808.
- [5] T. Wakahara, H. Murase, K. Odaka: *On-line Handwriting Recognition*, in [1], pp. 1181–1194.
- [6] G. Seni, S. Nij: *Towards an On-line Cursive Word Recognizer*, Pre-Proceedings 3rd Int. Workshop on Frontiers in Handwriting Recognition, 1993, pp. 122–131.
- [7] H. Weissman, M. Schenkel, I. Guyon, C. Nehl, D. Henderson: *Recognition-based Segmentation of On-line Run-on Handprinted Words: Input vs. Output Segmentation*, to appear in Pattern Recognition, 1994.
- [8] L. Schomaker: *Using Stroke- or Character-based Self-organizing Maps in the Recognition of On-line Connected Cursive Script*, Pattern Recognition, 26(3), 1993, pp. 443–450.
- [9] P. Morasso, L. Barberis, S. Pogliano, D. Vergano: *Recognition Experiments of Cursive Dynamic Handwriting with Self-Organizing Networks*, Pattern Recognition, 26(3), 1993, pp. 451–460.
- [10] S. Bercu, G. Lorette: *On-line Handwritten Word Recognition: An Approach Based on Hidden Markov Models*, Pre-Proceedings 3rd Int. Workshop on Frontiers in Handwriting Recognition, 1993, pp. 385–390.

- [11] J.-Y. Ha, S.-C. Oh, J.-H. Kim, Y.-B. Kwon: *Unconstrained Handwritten Word Recognition with Interconnected Hidden Markov Models*, Pre-Proceedings 3rd Int. Workshop on Frontiers in Handwriting Recognition, 1993, pp. 455–460.
- [12] I. Karls, G. Maderlechner, V. Pflug, S. Baumann, A. Weigel, A. Dengel: *Segmentation and Recognition of Cursive Handwriting with Improved Structured Lexica*, Pre-Proceedings 3rd Int. Workshop on Frontiers in Handwriting Recognition, 1993, pp. 437–442.
- [13] C.Y. Suen, C. Nadal, R. Legault, T.A. Mai, L. Lam: *Computer Recognition of Unconstrained Handwritten Numerals*, in [1], pp. 1162–1180.
- [14] C.Y. Suen, R. Legault, C. Nadal, M. Cheriet, L. Lam: *Building a New Generation of Handwriting Recognition Systems*, Pattern Recognition Letters, vol. 14, no. 4, 1993, pp. 303–315.
- [15] R.M. Bozinovic, S.N. Srihari: *Off-Line Cursive Script Word Recognition*, IEEE Trans. on PAMI, vol. 11, 1989, pp. 68–83.
- [16] K. Aoki, K. Yoshino: *Recognizer for Handwritten Script Words Using Syntactic Method*, Computer Recognition and Human Production of Handwriting, World Scientific, 1989.
- [17] S. Edelman, T. Flash, S. Ullman: *Reading Cursive Handwriting by Alignment of Letter Prototypes*, International Journal of Computer Vision, vol. 5, no. 3, 1990, pp. 303–331.
- [18] J.-C. Simon, O. Baret: *Regularities and Singularities in Line Pictures*, in H.S. Baird, H. Bunke, K. Yamamoto (editors): Structured Document Analysis, Springer Verlag, 1992, pp. 261–281.
- [19] A.W. Senior, F. Fallside: *An Off-line Cursive Script Recognition System Using Recurrent Error Propagation Networks*, Pre-Proceedings 3rd Int. Workshop on Frontiers in Handwriting Recognition, 1993, pp. 132–141.
- [20] L.R. Rabiner: *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE, vol. 77, no. 2, 1989, pp. 257–286.
- [21] Y. He, A. Kundu: *2-D Shape Classification Using Hidden Markov Model*, IEEE Trans. on PAMI, vol. 13, 1991, pp. 1172–1184.
- [22] F. Samaria, F. Fallside: *Face Identification and Feature Extraction Using Hidden Markov Models*, in G. Vernazza, A.N. Venetsanopoulos, C. Braccini (editors): Image Processing: Theory and Applications, Elsevier Science Publishers B.V., 1993, pp. 295–302.
- [23] A. Kundu, Y. He, P. Bahl: *Recognition of Handwritten Word: First and Second Order Hidden Markov Model Based Approach*, Pattern Recognition, 22(3), 1989, pp. 283–297.
- [24] M.Y. Chen, A. Kundu: *An Alternative to Variable Duration HMM in Handwritten Word Recognition*, Pre-Proceedings 3rd Int. Workshop on Frontiers in Handwriting Recognition, 1993, pp. 82–91.

- [25] A. Kaltenmeier, T. Caesar, J.M. Gloger, E. Mandler: *Sophisticated Topology of Hidden Markov Models for Cursive Script Recognition*, Proc. of the 2nd ICDAR, 1993, pp. 139–142.
- [26] M. Gilloux, M. Leroux, J.-M. Bertille: *Strategies for Handwritten Words Recognition Using Hidden Markov Models*, Proc. of the 2nd ICDAR, 1993, pp. 299–304.
- [27] N. Otsu: *A Threshold Selection Method from Gray-level Histogram*, IEEE Trans. on SMC, vol. 9, 1979, pp. 62–66.
- [28] T. Caesar, J.M. Gloger, E. Mandler: *Preprocessing and Feature Extraction for a Handwriting Recognition System*, Proc. of the 2nd ICDAR, 1993, pp. 408–411.
- [29] H. Tamura: *Further Considerations on Line Thinning Schemes*, Paper of IECEJ Tech. Group, PRL 75-66, 1975.
- [30] G. Boccignone, A. Chianese, L.P. Cordella, A. Marcelli: *Recovering Dynamic Information from Static Handwriting*, Pattern Recognition, 26(3), 1993, pp. 409–418.
- [31] L.R. Rabiner, B.H. Juang: *An Introduction to Hidden Markov Models*, IEEE ASSP Magazine, 1986, pp. 4–16.
- [32] X.D. Huang, Y. Ariki, M.A. Jack: *Hidden Markov Models for Speech Recognition*. Number 7 in Information Technology Series. Edinburgh University Press, Edinburgh, 1990.
- [33] E.G. Schukat-Talamazzini, H. Niemann: *Das ISADORA-System — ein akustisch-phonetisches Netzwerk zur automatischen Spracherkennung*. Mustererkennung 1991 (13. DAGM Symposium), pp. 251–258. Springer, Berlin, 1991.
- [34] J.J. Nijtmans: *Speech Recognition by Recursive Stochastic Modelling*. Technical report, Technische Universiteit Eindhoven, Den Haag, 1992.
- [35] E.G. Schukat-Talamazzini, H. Niemann, W. Eckert, T. Kuhn, S. Rieck: *Acoustic Modelling of Subword Units in the ISADORA Speech Recognizer*. Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, volume 1, pp. 577–580, San Francisco, 1992.
- [36] E.G. Schukat-Talamazzini, H. Niemann, W. Eckert, T. Kuhn, S. Rieck: *Automatic Speech Recognition Without Phonemes*. Proc. European Conf. on Speech Technology, pp. 129–132, Berlin, 1993.
- [37] G. Schmid, E.G. Schukat-Talamazzini, H. Niemann: *Analyse mehrkanaliger Meßreihen im Fahrzeugbau mit Hidden Markovmodellen*. in: S. Pöppel, editor, Mustererkennung 1993 (15. DAGM Symposium), Informatik aktuell, pp. 391–398. Springer, 1993.
- [38] A.P. Dempster, N.M. Laird, D.B. Rubin: *Maximum Likelihood from Incomplete Data via the EM Algorithm*. J. Royal Statist. Soc. Ser. B, vol. 39, no. 1, pp. 1–22, 1977.

Appendix A: Complete Dictionary Used in Experiment 1

able	decide	gesture	joke	moderate	pretend	telephone	wear
accomplish	deep	ghost	journey	morning	quantum	this	which
afternoon	disappoint	give	judge	name	quarrel	thread	wide
agree	discover	grill	ketchup	narrow	queen	tough	window
although	dream	harbour	kettle	naughty	question	transfer	woman
avenue	earth	help	key	neglect	quick	typical	xylonite
balance	easy	high	kidnap	night	quiver	umbrella	xylophone
beneficial	edit	highway	kind	nobody	ready	uncertain	yacht
blood	empty	honey	knock	obsolete	recommend	unique	yawn
breath	evening	hurry	lazy	obtain	remember	unlock	yes
broom	expensive	identical	legal	office	return	upward	yesterday
busy	familiar	ignore	lesson	omit	riddle	usually	yield
car	fetch	illegible	listen	opposite	rise	vegetables	young
casual	fiction	illuminate	lunch	oscillate	sample	very	zebra
celebrate	follow	incomplete	luxury	paper	school	visible	zero
chance	free	index	maintain	parallel	script	vision	zipper
cheap	furniture	jacket	man	persuade	shuffle	vote	zoom
cigarette	garage	jealous	merchandise	photograph	silent	water	
day	general	jewel	mighty	please	splendid		

Appendix B: Complete Dictionary Used in Experiment 2

Aarau	Davos	Glarus	Jona	Montreux	Payerne	Staefa	Vitznau
Airolo	Delsberg	Gossau	Juckern	Morges	Pfaeffikon	Stans	Wassen
Altdorf	Dielsdorf	Grenchen	Juf	Moutier	Port	Teufen	Wengen
Appenzell	Dietikon	Gstaad	Kandersteg	Murten	Pratteln	Thalwil	Wetzikon
Arbon	Dornach	Henniez	Klosters	Muttenz	Quarten	Thorberg	Wil
Arosa	Effretikon	Herisau	Kloten	Naefels	Quartino	Thun	Winterthur
Baden	Eglisau	Hochdorf	Koblentz	Naters	Quinten	Trub	Wohlen
Basel	Einsiedeln	Horgen	Kreuzlingen	Nebikon	Quinto	Twann	Yens
Bellinzona	Elm	Horw	Kriens	Neuenburg	Rapperswil	Unterseen	Yverdon
Bern	Engelberg	Huttwil	Langenthal	Nidau	Raron	Urdorf	Yvonand
Biel	Estavayer	Ibach	Lausanne	Nyon	Reinach	Urnaesch	Yvorne
Burgdorf	Fiesch	Ilanz	Liestal	Oberhofen	Riehen	Uster	Zermatt
Carouge	Flawil	Ins	Locarno	Oey	Rorschach	Uznach	Zernez
Cham	Flims	Interlaken	Lugano	Ofringen	Rueti	Uzwil	Zofingen
Chiasso	Frauenfeld	Ipsach	Luzern	Olivone	Sarnen	Valbella	Zuerich
Chillon	Fribourg	Isonne	Martigny	Olten	Schaffhausen	Vellerat	Zug
Chur	Frutigen	Jassbach	Meilen	Orbe	Schwyz	Vevey	Zurzach
Colombier	Gandria	Jaun	Meiringen	Parpan	Sion	Villars	
Dallenwil	Genf	Jegenstorf	Monthey	Passugg	Solothurn	Visp	

About the Author — HORST BUNKE received his M.S. and Ph.D. degree, both in computer science, from the University of Erlangen, Germany, in 1974 and 1979, respectively. In 1984 he joined the University of Bern where he is a full professor of computer science. Professor Bunke has published more than 150 papers and eight books, mainly in the fields of pattern recognition, computer vision, and artificial intelligence. He is one of the two editors-in-charge of the *International Journal of Pattern Recognition and Artificial Intelligence*.

About the Author — MARKUS ROTH was with the University of Bern as a student of computer science until January 1994 when he received his M.S. degree. During the specialization phase of his study, his interests were mainly in 2D object recognition and script recognition. Presently, he is with the international Zurich Insurance Group.

About the Author — ERNST GUENTER SCHUKAT-TALAMAZZINI received his diploma in Mathematics in 1982 from the University of Hannover, Germany. In 1986, he received his Ph.D. in Computer Science from the University of Erlangen-Nürnberg, Germany. Since 1982 he is with the Pattern Recognition Division of the Department for Computer Science of the latter institution. His research interest is the field of Statistical Pattern Recognition. Within the framework of national as well as European speech projects, he worked upon acoustic and linguistic modeling for Spoken Language Systems; current emphasis is on spontaneous speech as well as probabilistic approaches to conceptual decoding. Present activities concentrate, too, on the application of Hidden Markov Models and probabilistic grammars to the interpretation of sensory data in the areas of handwriting, car quality control, and granulometry. Dr. Schukat-Talamazzini is author, or coauthor, of about forty publications treating the above topics, including two monographs.