Short Papers

An HMM-Based Approach for Off-Line Unconstrained Handwritten Word Modeling and Recognition

A. El-Yacoubi, M. Gilloux,
R. Sabourin, *Member*, *IEEE*, and
C.Y. Suen, *Fellow*, *IEEE*

Abstract—This paper describes a hidden Markov model-based approach designed to recognize off-line unconstrained handwritten words for large vocabularies. After preprocessing, a word image is segmented into letters or pseudoletters and represented by two feature sequences of equal length, each consisting of an alternating sequence of shape-symbols and segmentation-symbols, which are both explicitly modeled. The word model is made up of the concatenation of appropriate letter models consisting of elementary HMMs and an HMM-based interpolation technique is used to optimally combine the two feature sets. Two rejection mechanisms are considered depending on whether or not the word image is guaranteed to belong to the lexicon. Experiments carried out on real-life data show that the proposed approach can be successfully used for handwritten word recognition.

Index Terms—Handwriting modeling, preprocessing, segmentation, feature extraction, hidden Markov models, word recognition, rejection.

1 Introduction

Handwriting is one of the easiest and most natural ways of communication between humans and computers. However, early investigations in automatic handwriting recognition were limited by the memory and power of the computers available at that time which did not permit the design of real-time systems. Thanks to the recent progress in electronics and to the latest generation of computers, these problems have been overcome; therefore, since the beginning of the 1980s, there has been a dramatic increase of research in this field. According to the way handwriting data are generated, two classes are distinguished: If the data provided to the system correspond to the pixels of a static image obtained with a scanner or a CCD camera after the writing is completed, then we are in the off-line recognition case. If the data correspond to the sequence of pixels (defined by their coordinates) drawn by the user on a digitized tablet and transmitted to the system during the writing, then we are in the on-line recognition case. Off-line and on-line systems are also distinguished by the applications they are devoted to. The former are dedicated to bank check processing, mail sorting, commercial forms-reading, etc., while the latter are mainly dedicated to pen computing industry and security domains such as signature verification and author authentication. Off-line handwriting recognition is a more difficult task, because the temporal information, such as the number and the order of the strokes and the pressure, is not available as in the on-line case. On the other hand, off-line systems can achieve huge economic benefits even with low recognition rates, while on-line systems must achieve high recognition rates to be used in a commercial system. In the remainder of this paper, we shall talk about off-line handwriting recognition.

Despite the impressive progress achieved in handwriting recognition, the results are still far from human performance. This is a reason why researchers have limited their studies to particular problems and applications. In this context, isolated character recognition can be seen as a less complicated task where satisfactory solutions are already available. In word recognition tasks, the application specifies the lexicon of possible words. For small lexicons, as in bank check processing, most approaches are global, where a word is considered as an indivisible entity [1], [2], [3]. For large lexicons, as in postal applications [4], [5], [6], the segmentation of words into basic units such as letters is required. Owing to the difficulty of this operation, most successful approaches are segmentation-recognition methods in which words are first loosely segmented into letters or pieces of letters, and a *dynamic programming* technique is used in recognition to choose the definitive segmentation [7], [8]. Although these methods are less robust when the segmentation process fails to split a pair of letters (or more), they have many advantages over global ones. Indeed, for a given learning database, it is more reliable to train a small set of letters than whole words. Furthermore, unlike analytic approaches, global approaches are possible only for lexicon-driven problems and do not satisfy the portability criterion since, for each new application, the set of the lexicon words must be trained.

During the last decade, *hidden Markov models* (*HMM*s), which can be thought of as a generalization of dynamic programming techniques [9], have become the predominant approach to automatic speech recognition [9], [10], [11]. These stochastic models have been shown to be well-adapted to summarize variability phenomena involved in time-varying signals. The success of HMMs in speech recognition has recently led many researchers to apply them to handwriting recognition by representing each word image as a sequence of observations. According to the way this representation is carried out, two approaches can be distinguished: implicit segmentation [4], [12], which leads to a speech-like representation of the handwritten word image, and explicit segmentation [5], [6], which requires a segmentation algorithm to split words into letters or pseudoletters.

In this paper, we propose an explicit segmentation-based HMM approach to recognize unconstrained handwritten words (uppercase, cursive and mixed). This system uses *three sets of features:* The first two are related to the shape of the segmented units, while the features of the third set describe segmentation points between these units. The first set is based on global features, such as loops, ascenders, and descenders, and the second set is based on features obtained by the analysis of the bidimensional contour transition histogram of each segment. Finally, segmentation features correspond to either spaces, possibly occurring between letters or words, or the vertical position of segmentation points that split connected letters. Given that the two sets of shape-features are separately extracted from the image, we represent each word by two feature sequences of equal length, each consisting of an alternating sequence of shape-symbols and segmentation-symbols. In the problem we are dealing with, we consider a vocabulary

- *A. El-Yacoubi, R. Sabourin, and C.Y. Suen are with the Centre for Pattern Recognition and Machine Intelligence, Department of Computer Science, Concordia University, 1455 de Maisonneuve Boulevard West, Suite GM-606, Montréal, Canada H3G 1M8. A. El-Yacoubi is also with the Departamento de Informatica, Pontificia Universidade Catolica do Parana, Av. Imaculada Conceicao, 1155-Prado Velho, 80.215-901 Curitiba-PR-Brazil. R. Sabourin is also with Ecole de Technologie Supérieure, Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle (LIVIA), 1100 Notre-Dame Ouest, Montréal, Canada H3C 1K3.*
E-mail: yacoubi@ppgia.pucpr.br.
- *M. Gilloux is with Service de Recherche Technique de La Poste, Département Reconnaissance, Modélisation et Optimasation (RMO), 10, rue de l'île Mâbon, 44063 Nantes Cedex 02, France.*

Manuscript received 26 May 1998; revised 10 Mar. 1999.
Recommended for acceptance by J. Hull.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 107682.

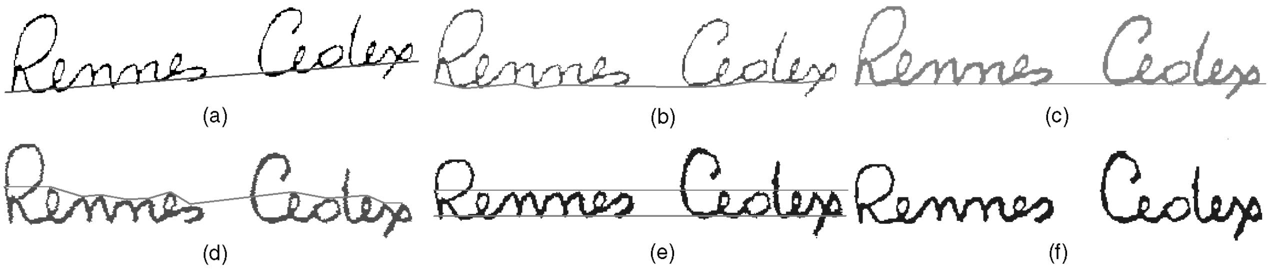


Fig. 1. Preprocessing steps: (a) original image, (b) and (c) baseline slant normalization, (d) character slant normalization, (e) lower-case letter area normalization, (f) definitive image after smoothing.

which is large but dynamically limited. For example, in city name recognition, the contextual knowledge brought by the postal code identity can be used to reduce the lexicon of possible city names to a small size. Since the entire vocabulary of words is large, it is more realistic to model basic units, such as letters, rather than whole words. Indeed, this modeling needs only a reasonable number of models to train (and to store). Then, each word (or word sequence) model can be dynamically built by concatenating letter models. This modeling is also more appropriate for available learning databases, which often do not contain all the possible words to be recognized. Our system also contains a mechanism to reject unreliable decisions.

This paper is organized as follows. Section 2 describes the fundamentals of hidden Markov models. Section 3 details the steps of preprocessing, segmentation, and feature extraction. Section 4 deals with the application of HMMs to handwritten word recognition in a dynamic vocabulary. Section 5 presents the experiments performed to validate the approach. Section 6 concerns the rejection mechanism considered by our system. Finally, Section 7 gives some concluding remarks and perspectives.

2 Hidden Markov Models

Hidden Markov models have been applied in several areas during the last 15 years, including speech recognition [9], [10], [11], language modeling [13], handwriting recognition [4], [5], [6], on-line signature verification [14], etc. A hidden Markov model is a doubly stochastic process, with an underlying stochastic process that is not observable (hence the word hidden), but can be observed through another stochastic process that produces the sequence of observations [11]. The hidden process consists of a set of states connected to each other by transitions with probabilities, while the observed process consists of a set of outputs or observations, each of which may be emitted by each state according to some probability density function (pdf). Depending on the nature of this pdf, several HMM classes can be distinguished. If the observations are naturally discrete or quantized using vector quantization [15], and drawn from an alphabet or a codebook, the HMM is said to be discrete [10], [11]. If these observations are continuous, we are dealing with a continuous HMM [11], [16], with a continuous pdf usually approximated by a mixture of normal distributions. Another family of HMMs, a compromise between discrete and continuous HMMs, are semi-continuous HMMs [17] that mutually optimize the vector quantized codebook and HMM parameters under a unified probabilistic framework. Although HMMs have some limitations such as the assumption of conditional independence of observations given the state sequence, these limitations are behind the well-defined theoretical foundations of HMMs and the existence of powerful algorithms for decoding and training. Particularly, a procedure called the Baum-Welch algorithm [11] can iteratively and automatically adjust HMM parameters given a training set of observation sequences. This algorithm, which is an implementation of the EM (expectation-maximization) algorithm [18] in the HMM case, guarantees that the model converges to a local maximum of the probability of observation of the training set according to the maximum likelihood estimation (MLE) criterion. The local maximum depends on the initial HMM parameters.

In some applications, it is useful to allow transitions with no output in order to model for instance a missing event in a given stochastic process, e.g., the absence of an expected character in a word due to undersegmentation or misspelling. It has been shown, in this case, it is more convenient to produce observations by transitions rather than by states [10]. To accommodate these changes, we have to define an additional HMM parameter a'_{ij} which stands for the probability of null transition between states i and j, i.e., that produces no output, a_{ij} being the conventional nonnull transition between these two states. We also define, for discrete HMMs, for instance, $b_{ij}(k)$ as the probability of observing the symbol k given the transition between states i and j. In this case, the stochastic constraints, for an N-state discrete HMM with an alphabet of size M, become:

$$\sum_{j=1}^{N}(a_{ij} + a'_{ij}) = 1 \quad \text{and} \quad \sum_{k=1}^{M} b_{ij}(k) = 1. \tag{1}$$

Taking this into account, slight changes occur in the classical Baum-Welch and Viterbi [19] algorithms for which the various forward and backward recursions still hold.

3 Representation of Word Images

Markovian modeling assumes that a word image is represented by a sequence of observations. These observations should be statistically independent once the underlying hidden state sequence is known. Therefore, we first preprocess each input image to get rid of information that is not meaningful to recognition and that may lead to dependence between observations (character slant, etc.). Then, segmentation and feature extraction processes are carried out to transform the image into an ordered sequence of symbols.

3.1 Preprocessing

The goal of preprocessing is to reduce irrelevant information such as noise and intraclass variability (e.g., character slant) that causes high writer-sensitivity in classification, therefore increasing the task complexity in a writer-independent recognizer. In our system, the preprocessing stage consists of four steps [20]: *baseline slant normalization*, *lower case letter area (upper-baseline) normalization* when dealing with cursive words, *character skew correction,* and, finally, *smoothing* (Fig. 1). The first two attempt to ensure a robust extraction of our first feature set, mainly ascenders and descenders, while the third step is required since the second feature set shows a significant sensitivity to character slant.

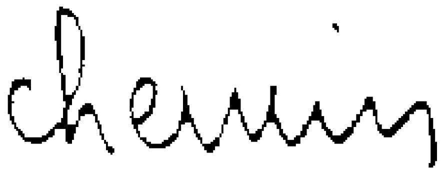


Fig. 2. Ambiguity in handwritten words, here the French word `Chemin`.

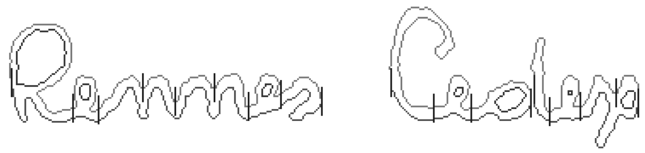


Fig. 3. Segmentation of words into letters or pseudoletters.

Baseline slant normalization is performed by aligning the minima of the lower contour after having filtered those corresponding to descenders. Upper-baseline normalization is similar and consists of aligning the maxima of the upper contour after having filtered those corresponding to ascenders or upper-case letters. However, the transformation here is nonlinear since it must keep the normalized lower-baseline horizontal. The ratio of the number of filtered maxima over the total number of maxima is used as an a priori selector of the writing style: cursive or upper-case (in which case no normalization is done). Character skew is estimated as the average slant of elementary segments obtained by sampling the word image contour, without taking into account horizontal and pseudohorizontal segments. Finally, we apply smoothing to eliminate the noise appearing at the borders of the word image due to the normalizations mentioned above.

3.2 Segmentation of Words into Characters

In speech recognition, the basic units correspond to phonetic events, for instance, *phonemes*. As it is hard to achieve an a priori explicit segmentation of words into those units, the techniques employed consist of sampling the speech signal into successive frames with a sufficiently high frequency. This representation is suitable because such a frequency allows a slow description of the speech signal in such a way that the different phonetic events can more or less be separately detected using minimal supervised learning techniques [9], [10]. *When dealing* with handwritten words, the basic units are naturally the alphabet letters. The employed segmentation techniques are numerous, but can be categorized into either implicit or explicit methods. Implicit methods are inspired by those considered in speech recognition and can either work at the pixel column level [4], [12] or realize an a priori scanning of the image with sliding windows [21]. Explicit methods, by contrast, use some characteristic points, such as upper (or lower) contour minima, intersection points, or spaces, to propose possible segmentation points (SPs). Due to the bidimensional character of off-line handwritten word images and to the overlap between letters, implicit methods are less efficient here than in speech recognition or on-line handwriting recognition. Indeed, vertical sampling loses the sequential aspect of the strokes, which is better represented by explicit methods. Moreover, in implicit methods, SPs have to be learned also. Nevertheless, implicit methods complement explicit ones and are particularly efficient in dealing with discrete touching characters. On the other hand, because of the ambiguity encountered in handwritten words, it is impossible to correctly segment a word into characters without resorting to the recognition phase. Indeed, the same pixel representation may have several interpretations, according to context. In Fig. 2, for instance, the group of letters at the right of `e` could be interpreted—in the absence of context—by many combinations of letters `i`, `m`, `n`, `r`, `u`, `v`, or `w`.

We choose an explicit segmentation algorithm that deliberately proposes a high number of SPs, offering in this way several segmentation options, the best one to be validated during recognition. This algorithm is based on the following two hypotheses: 1) There exists natural SPs corresponding to disconnected letters; 2) the physical SPs between connected letters are located at the neighborhood of the image upper contour minima. To segment a word, we make use of the upper and lower contours, loops, and upper contour minima. Then, each minimum satisfying some empirical rules gives rise to an SP. Mainly, we look in the neighborhood of this minimum for the upper contour point that permits a vertical transition from the upper contour to the lower one without crossing any loop, while minimizing the vertical transition histogram of the word image. If the crossing of a loop is unavoidable, no SP is produced. This strategy may produce correctly segmented, undersegmented, or oversegmented letters, as shown in Fig. 3, for example.

3.3 Feature Extraction

The goal of the feature extraction phase is to extract, in an ordered way (suitable to Markovian modeling), a set of relevant features that reduce redundancy in the word image while preserving the discriminative information for recognition. Our main philosophy in this step is that, unlike isolated character recognition, lexicon-driven word recognition approaches do not require features to be very discriminative at the character or pseudo character level because other information, such as context (particular letter ordering in lexicon words), word length, etc., are available and permit high discrimination of words. Thus, we consider features at the segment level with the aim of clustering letters into classes. In our system, the sequence of segments obtained by the segmentation process is transformed into a sequence of symbols by considering two sets of features.

The first feature set is based on global features, namely loops, ascenders, and descenders. Ascenders (descenders) are encoded in two ways according to their relative size compared to the height of the upper (lower) writing zone. Loops are encoded in various ways according to their membership in each of the three writing zones and their relative size compared to the sizes of these zones. The horizontal order of the median loop and the ascender (or descender) within a segment are also taken into account to ensure a better discrimination between letters such as `b` and `d` or `p` and `q`.[1] Each combination of these features within a segment is encoded by a distinct symbol, leading in this way to an alphabet of 27 symbols. For example, in Fig. 3, the first segment is encoded by the symbol *L*, reflecting the existence of a large ascender and a loop located above the core region. The second segment is encoded by the symbol *o*, indicating the presence of a small loop within the body of the writing. The third segment is represented by the symbol *-*, which encodes shapes without any interesting feature, etc.

The second feature set is based on the analysis of the bidimensional contour transition histogram of each segment in the horizontal and vertical directions. After a filtering phase consisting of averaging each column (row) histogram value over a five pixels-wide window centered in this column (row) and rounding the result, the histogram values may be equal to 2, 4, or 6. In each histogram, we focus only on the median part, representing the stable area of the segment, and we determine the *dominant transition number* defined as the value k (2, 4, or 6) for which the number of columns (rows) with a histogram value equal to k is

1. Henceforth, alphanumeric characters will be designated by `courier` format and feature symbols with *italic* style.

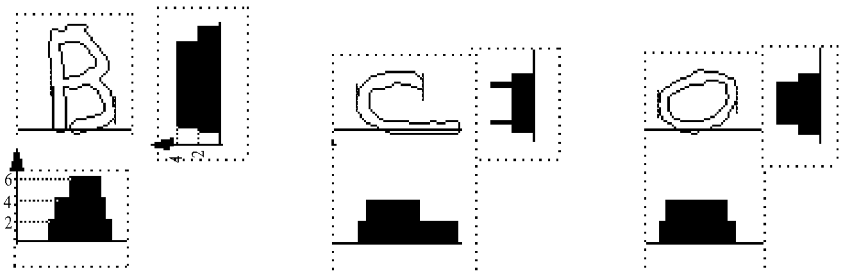


Fig. 4. Transition histograms of segmented shapes.

maximum. Each different pair of dominant transition numbers is then encoded by a different symbol or class. After having created some further subclasses by a finer analysis of the segments, this coding leads to a set of 14 symbols. For instance, in Fig. 4, Letters B, C, and O, for which the pairs of dominant transition numbers are (6, 2), (4, 2), and (4, 4), are encoded by symbols called *B*, *C*, and *O*, respectively, but this, of course, is not always the case.

We also use five segmentation features that try to reflect the way segments are linked together. For connected segments, two configurations are distinguished: If the space width is less than a third of the average segment width (ASW), we consider that there is no space, and encode this configuration by the symbol n. Otherwise, we validate the space and encode it in two ways, depending on whether the space width is smaller (symbol @) or larger (symbol #) than ASW. If the two segments are connected, the considered feature is the segmentation point vertical position which is encoded in two ways (symbols s or u) depending on whether the segmentation point is close to or far from the writing baseline. Finally, given an input word image, the output of the feature extraction process is a pair of symbolic descriptions of equal length, each consisting of an alternating sequence of segment shape symbols and associated segmentation point symbols (Fig. 5).

4 Markovian Modeling of Handwritten Words

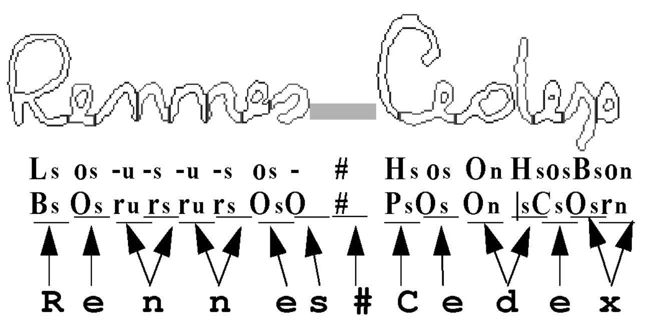
This section presents the application of HMMs in handwritten word recognition. After briefly describing some related works in this field, we give the justifications behind the design of the model we propose and we detail the steps of learning and recognition as used in our system.

4.1 Use of HMMs in Handwritten Word Recognition

Recently, HMMs have been applied to several areas in handwriting recognition, including character recognition [22], on-line word recognition [23], [24] and off-line word recognition. In the latter application, Gillies [4] was one of the first to use an implicit segmentation-based HMM for cursive word recognition. First, a label is given to each pixel in the image according to its membership in strokes, holes, and concavities. Then, the image is transformed into a sequence of symbols which result from the vector quantization of each pixel column. Each letter is characterized using a different discrete HMM, the parameters of which are estimated on hand-segmented data. The Viterbi algorithm is used in recognition and allows an implicit segmentation of words into letters as a by-product of the word matching process. Magdi and Gader [12] use a similar technique in which the observations are based on the location of black-white and white-black transitions on each image column and a 12-state left-to-right HMM is designed for each character. Cho et al. [21] also use an implicit segmentation technique in which a cursive word image is first split into a sequence of overlapping vertical gray-scale bitmap frames, which are then encoded into discrete symbols using principal component analysis and vector quantization. A word is modeled as an interconnection network of character and ligature HMMs. To improve the recognition strategy, several combinations of Forward and Backward Viterbi procedures were investigated. Chen et al. [6] use an explicit segmentation-based continuous density variable duration HMM where the observations are based on geometrical and topological features, pixel distributions, etc. Each letter is identified with a state which can account for up to four segments per letter. The parameters of the HMM are estimated using the lexicon and the manually labeled training data. A modified Viterbi algorithm is applied to provide several outputs, which are postprocessed using a general string editing method. Finally, Bunke et al. [25] propose an HMM approach to recognize cursive words produced by cooperative writers. The features used in their scheme are based on the edges of the skeleton graph of a word. A semicontinuous HMM (the Isadora system [26]) is considered for each character and the number of Gaussians was defined by manual inspection of the data set. Recognition is performed using a beam search-driven Viterbi algorithm.

4.2 The Proposed Model

As shown above, several HMM architectures can be considered for handwritten word recognition. This stems from the fact that handwriting is certainly not a Markovian process and, even if it were so, the correct HMM architecture is actually not known. The usual solution to overcome this problem is to first make structural assumptions and then use parameter estimation to improve the probability of generating the training data by the models. In our case, the assumptions to be made are related to the behavior of the segmentation process. As our segmentation process may produce either a correct segmentation of a letter, a letter omission, or an oversegmentation of a letter into two or three segments, we built an eight-state HMM having three paths to take into account these configurations (Fig. 6). In this model, observations are emitted along transitions. Transition t_{07}, emitting the null symbol Φ,



Ls Os -u -s -u -s Os - # Hs Os On HsOsBsOn
Bs Os ru rs ru rs OsO # PsOs On |sCsOsrn
R e n n e s # C e d e x

Fig. 5. Pair of feature sequences representing a word (word sequence) image.

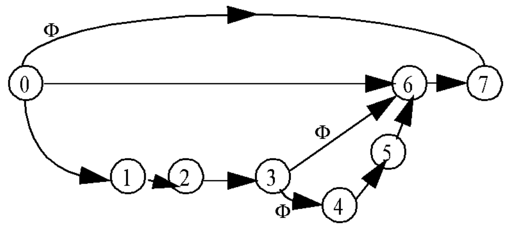


Fig. 6. The character model.

models the letter omission case. Transition t_{06} emits a symbol encoding a correctly segmented letter shape, while transition t_{67} emits a symbol encoding the nature of the segmentation point associated with this shape. Null transition t_{36} models the case of oversegmentation into only two segments. Transitions t_{01}, t_{23}, and t_{56} are associated with the shapes of the first, second, and third parts of an oversegmented letter, while t_{12} and t_{45} model the nature of the segmentation points that gave rise to this oversegmentation. Note that the rare occurrence of splitting a letter into three pieces makes the associated parameters likely not to be reliably estimated. The solution to this problem is to share the transitions involved in this phenomenon (t_{34}, t_{36}, t_{45}, t_{56}) over all character models, by calling for the tied states principle. Nevertheless, this procedure is not carried out for letters M, W, m, or w for which the probability of segmentation into three pieces is high and, therefore, there are enough examples to separately train the third segment parameters for each of these letters. Finally, a refinement of the character model consisted of considering context-dependent models for upper-case letters depending on their position in the word: first position, whether in an upper-case or cursive word, or any different position in an upper-case word. The motivation behind this is that features extracted from these two categories of letters can be very different since they are based on global features, such as ascenders, which strongly depend on the writing style.

Our model architecture is somewhat similar to that of other approaches, such as [2], [27], but with some differences. Here, the first segment presented to a character model is produced by two different transitions depending on whether it corresponds to the entire shape of a correctly segmented character (t_{06}) or to the first part of an oversegmented character (t_{01}), while in [2], [27], for example, the same transition is shared between these two configurations. Our architecture allows the transitions in the model to be fed by homogeneous data sources, leading to less variability and higher accuracy (e.g., the first part of an over-segmented d and a correctly segmented d, which are very different, would be presented to different kinds of transitions, t_{06} and t_{01}, respectively). In other words, the variability coming from the inhomogeneity in the source data, since it is known a priori, is eliminated by separate modeling of the two data sources. In addition, we have a special model for interword space, in the case where the input image contains more than one word. This model simply consists of two states linked by two transitions, modeling a space (in which case only the symbols corresponding to spaces @ or # can be emitted) or no space between a pair of words (Fig. 7).

4.3 The Learning Phase

Since the exact orthographic transcription (labeling) of each training word image is available, the word model is made up of the concatenation of the appropriate letter models, the final state of an HMM becoming the initial state of the next one, and so on (Fig. 8).

Note that, here, we use an *embedded Baum-Welch training algorithm* for which the segments produced by the segmentation algorithm need not be manually labeled. This is an important consideration for two reasons: First, manually segmenting a database is a very expensive process and is therefore not desirable; second, assuming we have a sufficient learning database, embedded Baum-Welch training allows the recognizer to capture contextual effects and permits the segmentation of the feature sequence into letters and the reestimation of the associated transitions so as to optimize the likelihood of the training database. Thus, the recognizer decides for itself what the optimal segmentation might be, rather than being heavily constrained by a priori knowledge based on human intervention [9]. This is particularly true if we bear in mind the inherent incorrect assumptions made about the HMM structure. From an implementation point of view, given a word composed of L letters, a new parameter corresponding to the index of the currently processed letter is added to the probabilities involved in the Baum-Welch algorithm. Then, the results of the final forward (initial backward) probabilities at the last (initial) state of the elementary HMM associated with a letter are moved forward (backward) to become the initial forward (final backward) probabilities at the initial (last) state of the elementary HMM associated with the following (previous) letter. If $\alpha_t^l(i)$ ($\beta_t^l(i)$) denotes the forward (backward) probability associated with the letter of index l, then this process is carried out according to the following equations:

$$\alpha_t^{l+1}(0) = \alpha_t^l(N-1) \quad l = 0, \ldots, L-2 \quad t = 0, 1, \ldots, T-1 \tag{2}$$

$$\beta_t^{l+1}(0) = \beta_t^l(N-1) \quad l = 0, \ldots, L-2 \quad t = 0, 1, \ldots, T-1, \tag{3}$$

where 0 and $N-1$ are the initial and final states of elementary letter HMMs and t is time index. In addition to the learning set, we use a validation set on which the reestimated model is tested after each training iteration. *The* training stops when the likelihood of the training set becomes sufficiently small or, more formally, when the following inequality becomes true:

$$\xi_t = \frac{P_\tau^T - P_{\tau-1}^T}{P_\tau^T + P_{\tau-1}^T} < \varepsilon. \tag{4}$$

Here, P_τ^T is the likelihood of the training set at iteration τ, ξ_t is the normalized increase of P_τ^T and ε is *a* sufficiently small threshold, typically 10^{-3}, 10^{-4}, etc. Once the training phase is over, the stored optimal model parameters are those corresponding to the iteration maximizing the likelihood of the validation set (and not the last iteration). This strategy allows the model to acquire a better generalization over unknown samples.

4.4 The Recognition Phase

The recognition process consists of determining the word maximizing the *a posteriori probability* that a word w has generated an unknown observation sequence O,

$$\Pr(\hat{w}|O) = \max_w \Pr(w|O). \tag{5}$$

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

$$\Pr(w|O) = \frac{\Pr(O|w)\Pr(w)}{\Pr(O)}. \tag{6}$$

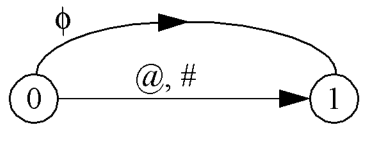


Fig. 7. The interword space model.

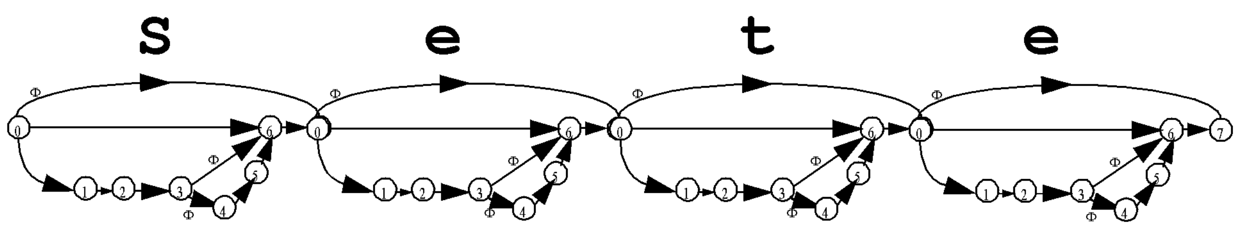


Fig. 8. Training model for the French word `Sete`.

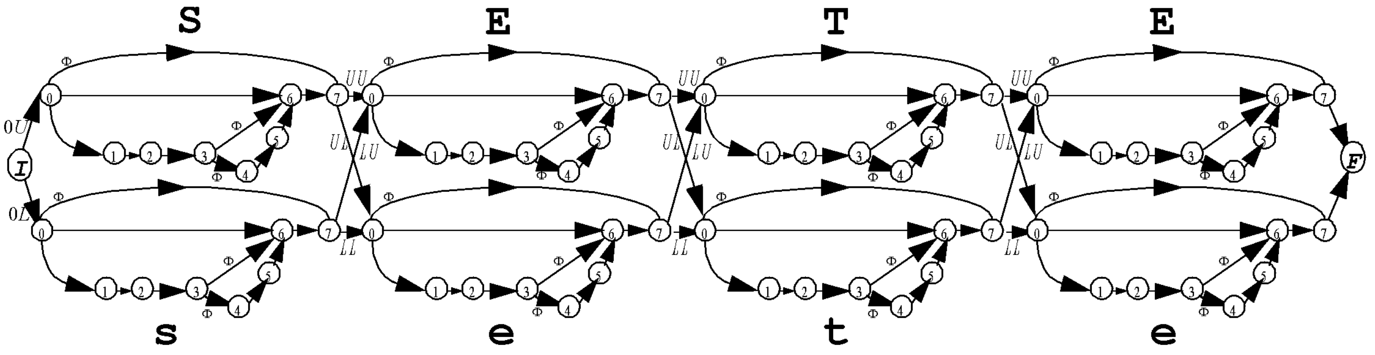


Fig. 9. Global recognition model for lexicon word `SETE`.

Since $\Pr(O)$ does not depend on w, recognition becomes equivalent to maximizing the joint probability,

$$\Pr(w, O) = \Pr(O|w)\Pr(w). \tag{7}$$

$\Pr(w)$ is the a priori probability of the word w and is related to the language of the considered task. The estimation of $\Pr(O|w)$ requires a probabilistic model that accounts for the shape variations O of the word w. We assume that such a model consists of a global Markov model created by concatenating letter HMMs. The model architecture remains basically the same as in training; however, as the writing style is unknown in recognition, a letter model here actually consists of two models in parallel, associated with the upper and lower case modes of writing a letter (Fig. 9). As a result, two consecutive letter models are now linked by four transitions associated with the various ways two consecutive letters may be written: uppercase-uppercase (*UU*), uppercase-lowercase (*UL*), lowercase-uppercase (*LU*), and lowercase-lowercase (*LL*). The probabilities of these transitions, as well as those of starting a word with an upper-case (*0U*) or lower-case (*0L*) letter, are estimated by their occurrence frequency in the learning database.

This architecture is more complete than usually adopted methods which generate *a priori two or three possible* ASCII configurations of words (fully upper-case, fully lower-case, or lower-case word beginning with an upper-case letter). Indeed, these methods quickly become tedious and time-consuming when dealing with a sequence of words rather than a single word, besides the fact that they cannot handle the problem of mixed handwritten words. The proposed model elegantly avoids these problems, allowing an implicit detection of the writing style during recognition owing to the Viterbi algorithm.

4.5 Combination of Feature Sets in a Unifying HMM Architecture

As mentioned earlier, the extraction of two feature types from segments led us to represent each word image by two feature sequences. An obvious solution to handle the two sequences is to use two independent word recognition engines and combine their outputs in a subsequent stage. This strategy, however, would be computationally heavy, and also less accurate, since the redundancy between the two feature sets is dropped in this case. Therefore, we should wonder how to optimally use these two sets while keeping in mind the number of parameters which must be trained. In this perspective, we were inspired by an HMM-based interpolation technique proposed in [10]. The idea is to build two HMMs, one in which no simplification assumptions are made and the other, analogous to the first (with the same structure) but with fewer parameters using the tied states principle [10]. The first model is more accurate but has a relatively large number of parameters, while the second is less accurate but has a relatively small number of parameters and, hence, is more reliably trained. The second step is to combine the two HMMs with a linear interpolation, the parameters of which are estimated by a third HMM. In our case, instead of using the tied states principle, the design of the two models is obtained by considering the way the two feature sequences are fed to the HMMs. Indeed, for each transition emitting shape symbols, two hypotheses can be made: First, the two symbols corresponding to the two feature sets, say o_k^1 and o_k^2, are jointly emitted by the transition and, in this case, we have to estimate probabilities of the form $p_1(o_k^1, o_k^2|t_{ij})$. This is the same thing as the consideration of a new alphabet, the elements of which are all the possible combinations of the symbols of the two alphabets; second, this time the two symbols are supposed to be independently emitted by the transition and, in this case, we have to estimate probabilities of the form $p_2(o_k^1|t_{ij})$ and $p_3(o_k^2|t_{ij})$. It is easy to see then, that the number of parameters is proportional to $N_1 \times N_2$ and $N_1 + N_2$, respectively, for the first and second assumptions, where N_1 and N_2 are the sizes, respectively, of the first and second alphabets. Hence, we obtain two HMMs, say M_1 and M_2, with complementary merits and drawbacks as mentioned above. The last step is to take, as final estimation of $p(o_k^1, o_k^2|t_{ij})$, a linear combination of p_1 and $p_2 \times p_3$ given by:

$$\Pr(O_k^1, O_k^2|t_{ij}) = \lambda_i \times p_1(O_k^1, O_k^2|t_{ij}) + (1-\lambda_i) \times \left[p_2(O_k^1|t_{ij}) \times p_3(O_k^2|t_{ij})\right] \tag{8}$$

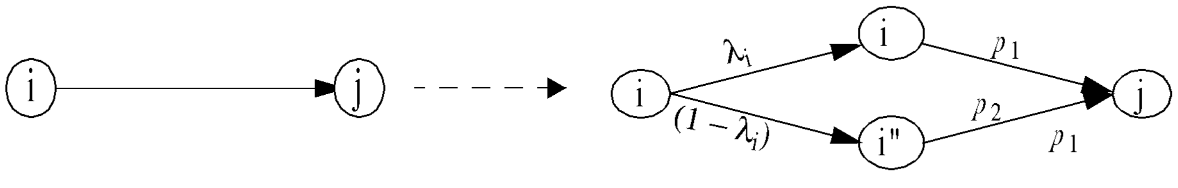


Fig. 10. The duplication process to generate the interpolated HMM.

with λ_i varying between 0 and 1. This equation can be interpreted as an associated Markov source in which each state is replaced by three states. However, since, in our model (Fig. 6), the problem of reliable parameter estimation solely deals with transitions emitting a shape observation, only transitions t_{01}, t_{06}, t_{23}, and t_{56} are involved in the duplication process (Fig. 10).

In the resulting HMM, say M_3, p_1 and p_2 are assumed to be known and are given by trained models M_1 and M_2. Therefore, only parameters λ_i must be trained. For more details on this section, see [28].

5 Experiments

Experiments were carried out on unconstrained handwritten French city name images manually localized on real mail envelopes which were scanned at a resolution of 200 dpi. The sizes of the learning, validation and test sets were 11,410, 3,724, and 4,313, respectively. To simulate the address recognition task, we assume that the city names in competition are independent for a given list of corresponding postal code hypotheses. This is only partly true, since confusable codes may belong to the same geographic area and, in some cases, the cities in a given area tend to share a common element (e.g., BOULOGNE SUR SEINE, postal code 92100, and NEUILLY SUR SEINE, postal code 92200). Nevertheless, under this assumption, for each image in the test set, we choose $N-1$ city names from a vocabulary of 9,313 city names. The prior probabilities were assumed to be equal so that the average perplexity was maximum and equal to N. Indeed, for large vocabularies, $\Pr(w)$ is very difficult to estimate due to the lack of sufficient training samples. Moreover, we cannot rely on bigram or trigram statistics in this case because there is no reason for the size of a city to be correlated to its ASCII name. For example, PARIS is a big city, but PALIS is not. The best way for a reliable estimation of $\Pr(w)$ is, actually, to use city information, such as the population size, the postal code (in France, a zip code ending with 000 corresponds to a big city), the economic power, etc. To simulate the case where one, two, or three digits in the postal code are ambiguous (one, two, or three ambiguous digits give rise to 10, 100, or 1,000 possible postal codes), the values chosen for N were 10, 100, and 1,000. Recognition was carried out using the logarithmic version of the Viterbi algorithm, while, for training, we used the Baum-Welch algorithm. Since segment data were not labeled, the model parameters were initialized with random values. Training experiments carried out with uniform and various random starts showed essentially the same performances. The recognition rates obtained using the models M_1, M_2, and M_3 (defined in Section 4.5) are reported in Table 1 and some well-recognized French city name examples are given in Fig. 11.

TABLE 1
Recognition Rates Obtained with Models M_1, M_2, and M_3

Lexicon	M_1	M_2	M_3
10	99.0%	99.0%	99.2%
100	96.0%	96.1%	96.3%
1,000	87.7%	87.9%	88.9%

From the above results, it is clear that HMMs enable the system to achieve good performance even though the features used are not very discriminative at the *grapheme* level (a grapheme consists of a letter, left part, or right part of a letter) for which the recognition rate is only around 23 percent. Actually, it is the redundancy brought by the ordered association of features to describe a word that is discriminative and this discrimination is as high as the word length. The use of the segmentation features and preprocessing has also significantly contributed to recognition accuracy [20]. The similarity of the results obtained with models M_1 and M_2 is not surprising since the advantages of M_1 are the drawbacks of M_2 and vice versa (Section 4.5). The interpolation technique yielded a significant improvement, especially if we keep in mind that only the models associated with nonfrequent characters in the training set took advantage from it. Confusions in our system come from many sources, mainly from words with several overlapping characters (Fig. 12a), poor images (Fig. 12b), images with underline or with noise or remnant strokes above them (Fig. 12c), or the lack of examples to reliably estimate some model parameters.

6 Rejection

So far, system performances were expressed in terms of *recognition rates*. In real applications, however, systems are required to have *confusion rates* (*CR*) lower than some threshold based on economical criteria. A typical value of accepted *CR* in mail sorting is 1 percent. Therefore, it is necessary to consider in our approach a *rejection* criterion. In this perspective, we must go back to the Bayes' formula in (6) to compute $\Pr(O)$. When O is known to belong to the lexicon, as in our previous experiments, $\Pr(O)$ is obtained by:

$$\Pr(O) = \sum_{w} \Pr(O|w) \times \Pr(w). \tag{9}$$

Then, rejection can be established by requiring $\Pr(w|O)$ to be greater than a given threshold. Fig. 13 shows the evolution of the recognition rate, the confusion rate, and reliability (defined as the proportion of correct answers among the accepted images) as a function of the rejection rate (by varying the threshold value) when the correct answer is guaranteed to belong to a lexicon of size $N = 100$ or $1,000$. The test for a lexicon of size 10 has been dropped since the error rate is already sufficiently small in this case.

In real applications, however, the processed word image is not guaranteed to belong to the lexicon since it can be the result of a city name mislocation or a wrong dynamic generation of the lexicon (due to an important error in postal code recognition). To simulate this problem, we have generated random lexicons, half of

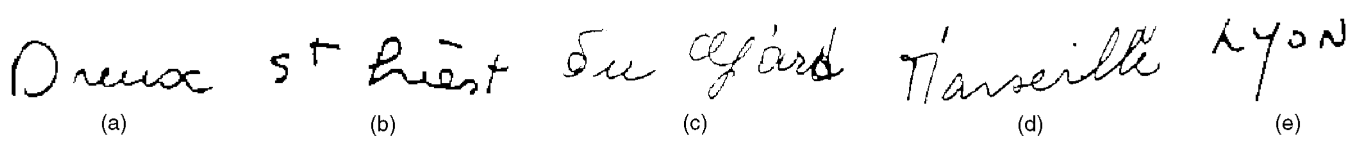


Fig. 11. Some examples of well-recognized images of French city names.

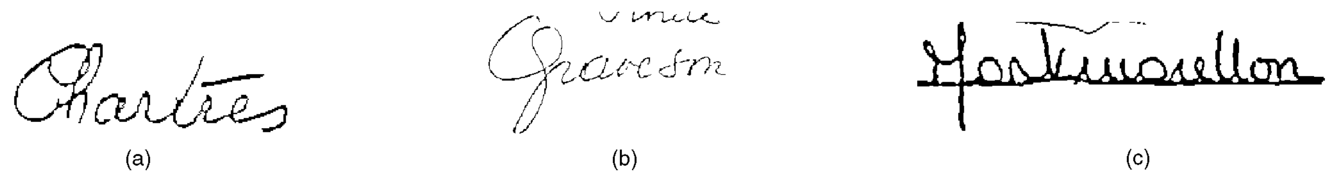


Fig. 12. Some examples of misrecognized images.

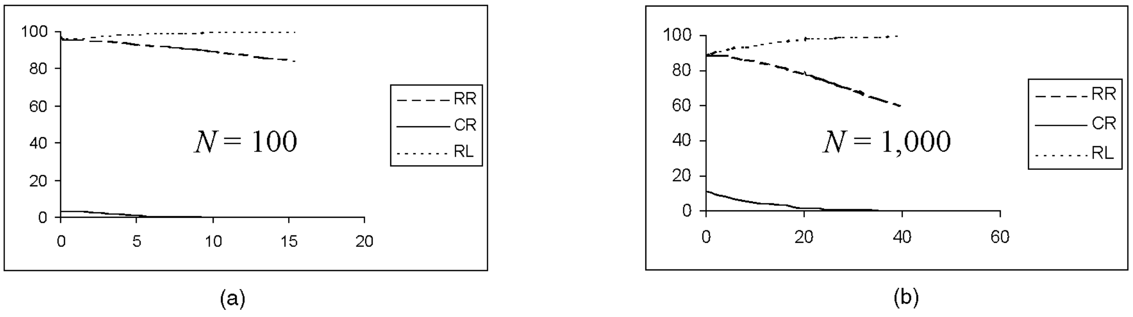


Fig. 13. Recognition rate (*RR*), confusion rate (*CR*) and reliability (*RL*) as a function of rejection rate when the correct answer is guaranteed to belong to the lexicon.

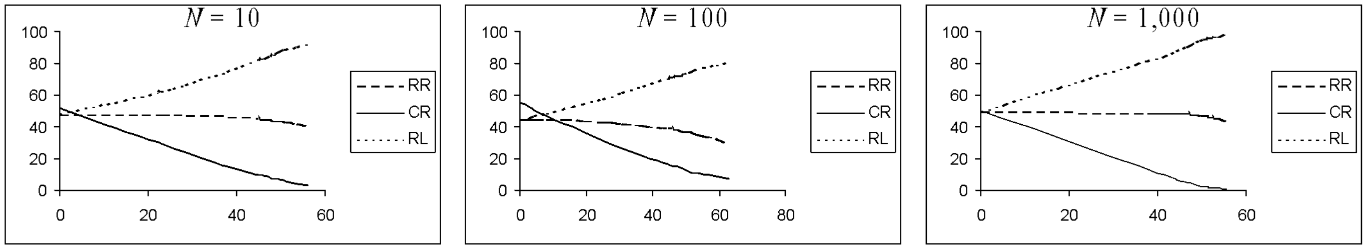


Fig. 14. Recognition rate (RR), confusion rate (CR) and reliability (RL) as a function of rejection rate when the probability that the word image belongs to the lexicon is $P_{\text{in}} = 0.5$.

which do not contain the correct answers. In this case, Bayes' probability becomes [5]:

$$\Pr(w|O) = \frac{P_{in} \times \Pr(O|w) \times \Pr(w)}{P_{in} \times \sum_w \Pr(O|w) \times \Pr(w) + p_{out} \times \Pr(O|out)}, \quad (10)$$

where $p_{\text{in}} = 0.5$ is the a priori probability that w belongs to the lexicon and p_{out} is its complement; $p_{\text{out}} = 1 - p_{\text{in}}$. Note that a more realistic value for p_{in} would have been chosen, in the case of city name recognition, by an estimation based on the knowledge of the accuracy of the recognition postal code and the automatic location of the city name. We approximated the term $\Pr(O|out))$ by the output of an ergodic HMM trained on the same set used to train the character models, although a more accurate set should also have included words that do not correspond to city names. The number of states of this HMM was set to 14 after several trials. Fig. 14 shows the evolution of the recognition rate, confusion rate, and reliability as a function of the rejection rate when the correct answer is not guaranteed to belong to a lexicon of size $N = 10, 100$, or $1,000$. Note that, in this experiment, the recognition rate cannot exceed 50 percent, given that p_{in} was set at 0.5.

7 Conclusion

In this paper, we presented a complete system for recognizing unconstrained handwritten words. We consider that our approach achieves good performance given that the data correspond to real word images and were not filtered; however, it is hard to compare with other approaches since we are not using the same databases. The main strength of the proposed system lies in its training phase, which does not require any manual segmentation of the data to train the character models. We used character HMMs to model explicitly both segmented shapes and associated segmentation points, leading to a better discrimination between characters. By building the word model as a sequence of character models, each consisting of a pair of associated upper-case and lower-case HMMs, the writing style is implicitly detected during recognition. We proved also that rejection can be efficiently implemented using HMMs. We believe that our system can still be improved, mainly in feature extraction and recognition phases. As mentioned in

Section 3.3, we use simple features to describe the segments and we rely on the context to discriminate the words. This may not be sufficient for small words which only differ in one or two letters (e.g., Reims versus Rennes, Nantes versus Mantes, etc.). In this case, low-level features describing the pixel distributions in the segments are necessary to eliminate the ambiguity between such words. An elegant method could be a hierarchical data representation which provides more details as the feature sequence length gets smaller and the ambiguity between the dynamic lexicon candidates (to be computed dynamically) gets higher. This could be an optimal compromise between performance and speed requirements in commercial systems. Markovian modeling could be improved by replacing discrete HMMs by semicontinuous HMMs in order to avoid the inherent loss of information when producing our feature sets, particularly histogram-based and segmentation-based features; or by considering hybrid Markovian/neural models to overcome the lack of discrimination power in conventional HMMs. Finally, our model architecture could be optimized by adding in the character model (Fig. 6) a null transition from state 6 to state 4 (t_{64}), generating in this way a loop that can model the oversegmentation of a letter into any number of graphemes. This would be particularly interesting for the design of generic models that can accommodate an important writing style change possibly occurring with the use of new sources of handwriting data.

ACKNOWLEDGMENTS

This work was supported by the Service de Recherche Technique de la Poste (SRTP) at Nantes, France, 1 École de Technologie Supérieure (ETS) at Montréal, Canada, and the Centre of Pattern Recognition and Machine Intelligence (CENPARMI) at Montréal, Canada. The authors would like to thank J.M. Bertille from the SRTP for discussions regarding this work.

REFERENCES

[1] M. Gilloux and M. Leroux, "Recognition of Cursive Script Amounts on Postal Cheques," *Proc. Fifth U.S. Postal Service Advanced Technology Conf.*, pp. 545-556, 1992.

[2] S. Knerr, O. Baret, D. Price, and J.C. Simon, "The A2iA Recognition System for Handwritten Checks," *Proc. Document Analysis Systems*, pp. 431-494, 1996.

[3] C.Y. Suen, L. Lam, D. Guillevic, N.W. Strathy, M. Cheriet, J.N. Said, and R. Fan, "Bank Check Processing System," *Int'l J. Imaging Systems and Technology*, vol. 7, pp. 392-403, 1996.

[4] A.M. Gillies, "Cursive Word Recognition Using Hidden Markov Models," *Proc. Fifth U.S. Postal Service Advanced Technology Conf.*, pp. 557-562, 1992.

[5] M. Gilloux, M. Leroux, and J.M. Bertille, "Strategies for Cursive Script Recognition Using Hidden Markov Models," *Machine Vision and Applications*, vol. 8, no. 4, pp. 197-205, 1995.

[6] M.Y. Chen, A. Kundu, and S.N. Srihari, "Variable Duration Hidden Markov Model and Morphological Segmentation for Handwritten Word Recognition," *IEEE Trans. Image Processing*, vol. 4, no. 12, pp. 1,675-1,687, Dec. 1995.

[7] R.M. Bozinovic and S.N. Srihari, "Off-Line Cursive Word Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no.1, pp. 68-83, Jan. 1989.

[8] F. Kimura, M. Shridhar, and Z. Chen, "Improvements of a Lexicon Directed Algorithm for Recognition of Unconstrained Handwritten Words," *Proc. Int'l Conf. Document Analysis and Recognition*, pp. 18-22, Oct. 1993.

[9] J. Picone, "Continuous Speech Recognition Using Hidden Markov Models," *IEEE Signal Processing Magazine*, pp. 26-41, July 1990.

[10] L. Bahl, F. Jelinek, and R. Mercer, "A Maximum Likelihood Approach to Speech Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, no. 2, pp. 179-190, Mar. 1983.

[11] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257-286, Feb. 1989.

[12] M. Magdi and P. Gader, "Handwritten Word Recognition Using Segmentation-Free Hidden Markov Modeling and Segmentation-Based Dynamic Programming Techniques," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 5, pp. 548-554, May 1996.

[13] F. Jelinek, R.L. Mercer, and S. Roukos, "Principles of Lexical Language Modeling for Speech Recognition," *Advances in Speech Signal Processing*, S. Furui and M.M. Sondhi, eds., pp. 651-699, 1992.

[14] L. Yang, B.K. Widjaja, and R. Prasad, "Application of Hidden Markov Models for Signature Verification," *Pattern Recognition*, vol. 28, no. 2, pp. 161-170, Feb. 1995.

[15] R.M. Gray, "Vector Quantization," *IEEE Signal Processing Magazine*, pp. 4-29, Apr. 1984.

[16] L.A. Liporace, "Maximum Likelihood Estimation for Multivariate Observation of Markov Sources," *IEEE Trans. Information Theory*, vol. 28, no. 5, pp. 729-734, Sept. 1982.

[17] X.D. Huang and M.A. Jack, "Semi-Continuous Hidden Markov Models for Speech Signals," *Computer Speech and Language*, vol. 3, pp. 239-251, 1989.

[18] T.K. Moon, "The Expectation-Maximization Algorithm," *IEEE Signal Processing Magazine*, pp. 47-60, Nov. 1996.

[19] G.D. Forney, "The Viterbi Algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268-278, Mar. 1973.

[20] A. El-Yacoubi, J.M. Bertille, and M. Gilloux, "Towards a More Effective Handwritten Word Recognition System," *Proc. Fourth Int'l Workshop Frontiers in Handwriting Recognition*, pp. 378-385, Dec. 1994.

[21] W. Cho, S.W. Lee, and J.H. Kim, "Modeling and Recognition of Cursive Words with Hidden Markov Models," *Pattern Recognition*, vol. 28, no. 12, pp. 1,941-1,953, Dec. 1995.

[22] R. Nag, K.H. Wong, and F. Fallside, "Script Recognition Using Hidden Markov Models," *IEEE Int'l Conf. Acoustics Signal and Speech Processing*, pp. 2,071-2,074, 1986.

[23] S. Bercu and G. Lorette, "On-Line Handwritten Word Recognition: An Approach based on Hidden Markov Models," *Proc. Third Int'l Workshop Frontiers in Handwriting Recognition*, pp. 385-390, May 1993.

[24] J.Y. Ha, S.C. Oh, and J.H. Kim, "Recognition of Unconstrained Handwritten English Words With Character and Ligature Modeling," *Int'l J. Pattern Recognition and Artificial Intelligence*, vol. 9, no. 3, pp. 535-556, June 1995.

[25] H. Bunke, M. Roth, and E.G. Schukat-Talamazzini, "Off-Line Cursive Handwriting Recognition Using Hidden Markov Models," *Pattern Recognition*, vol. 28, no. 9, pp. 1,399-1,413, Sept. 1995.

[26] E.G. Schukat-Talamazzini and H. Niemann, "Das ISADORA-System—Ein Akustischphonetisches Netzwerk zur Automatischen Spracherkennung," *Musterekennung, 13. DAGM Symp.*, pp. 251-258, 1991.

[27] M.Y. Chen, A. Kundu, and J. Zhou, "Off-Line Handwritten Word Recognition Using a Hidden Markov Model Type Stochastic Network," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 481-496, May 1994.

[28] A. El-Yacoubi, R. Sabourin, M. Gilloux, and C.Y. Suen, "Improved Model Architecture and Training Phase in an Off-Line HMM-Based Word Recognition System," *Int'l Conf. Pattern Recognition*, pp. 16-20, Aug. 1998.