HMM Based On-Line Handwriting Recognition

Jianying Hu, *Member, IEEE*, Michael K. Brown, *Senior Member, IEEE*, and William Turin, *Senior Member, IEEE*

Abstract—Hidden Markov Model (HMM) based recognition of handwriting is now quite common, but the incorporation of HMM's into a complex stochastic language model for handwriting recognition is still in its infancy. We have taken advantage of developments in the speech processing field to build a more sophisticated handwriting recognition system. The pattern elements of the handwriting model are subcharacter stroke types modeled by HMM's. These HMM's are concatenated to form letter models, which are further embedded in a stochastic language model. In addition to better language modeling, we introduce new handwriting recognition features of various kinds. Some of these features have invariance properties, and some are segmental, covering a larger region of the input pattern. We have achieved a writer independent recognition rate of 94.5% on 3,823 unconstrained handwritten word samples from 18 writers covering a 32 word vocabulary.

Index Terms—On-line handwriting recognition, hidden Markov models, subcharacter models, evolutional grammar, invariant features, segmental features.

1 Introduction

COMPUTER recognition of handwritten cursive script has received relatively little attention, until recently, when compared to Optical Character Recognition (OCR), speech recognition, and other image or scene analysis areas. Interest began to grow significantly in the 1980s and 1990s (for a good survey, see [1]) with the introduction of small, but sufficiently powerful, portable computers to serve as the support platforms. Hidden Markov Model (HMM) based handwriting recognition has now become quite common for both on-line and off-line systems (e.g., [2], [3], [4], [5], 6[], [7]). However, the incorporation of subcharacter HMMs into a complex stochastic language model is still in its infancy, and there has been relatively little effort devoted to feature development. In this paper, we describe the application of language and stochastic modeling methods developed originally for speech recognition to the problem of on-line handwriting recognition, as well as the development of new features. We begin with a Hidden Markov Model (HMM) based system with subcharacter model units using well known point oriented features like stroke tangents, and then introduce new features for handwriting recognition that are invariant with respect to the three common factors of geometric distortion—translation, rotation, and scaling. These purely stochastic methods are combined with interleaved segmental features that capture a larger region of segmented shape information. The stochastic processing is repeatedly interrupted to insert these features into partial hypotheses. This new method yields substantially improved recognition accuracy.

A partial diagram of the recognition system, called AEGIS (Automatic Evolutional Grammar Interpretation System), is shown in Fig. 1 (the application generic components of AEGIS were originally developed for speech applications [8]). The system may perform both recognition and training, depending on the mode of operation and the loaded grammar(s). There are three application generic components: an HMM scoring module, an Evolutional Grammar (EG) processing module, and a back-tracing module. In addition, there are application specific modules that perform feature extraction (FX), state scoring, model parameter learning, and input flow control.

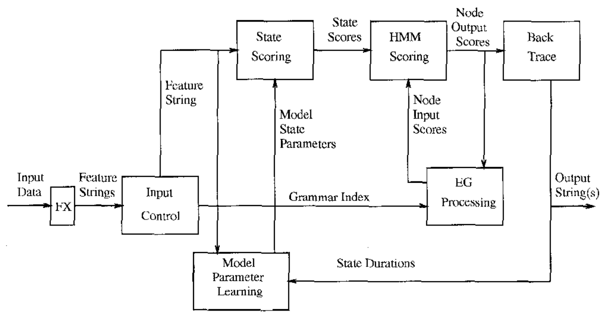


Fig. 1. Partial diagram of AEGIS.

The handwriting data was collected using a newly developed graphics input tablet [9]. The sampling rate is 200 samples per second; the tablet dimensions are 8.5 in × 11 in; and resolution is 0.1 mm. Writers were asked to write on a lined sheet of paper with no constraints on speed or style. The lines on the paper suggest an implied preferable orientation and size.

Preprocessing consists of two steps: noise reduction and normalization. Input device noise is reduced using a spline smoothing operator. A spline kernel is convolved with the x and y coordinates of the input sample points to generate the coordinates of sampled points of a local approximating cubic spline [10]. Cusps are detected beforehand and treated as boundary points during the smoothing operation to avoid loosing important shape features. The only normalization operation currently applied is deskewing of word samples. After preprocessing, an equal-arclength resampling procedure is applied to remove variations in writing speed. Each word sample is then represented as a time-ordered sequence of observations in the form of feature vectors.

In the next section, we give detailed descriptions of the methods used in our HMM-based recognition engine, including model topology, development of subcharacter model units, stochastic language modeling, decoding, and parameter learning. Section 3 introduces invariant features, describes their advantages, and develops two new invariant features. Segmental features are introduced in Section 4, where a feature representing holistic letter shape information is incorporated into the stochastic model decoding process. We then present experimental results and conclude in Sections 5 and 6.

2 Model Design and Training

Consider a hidden Markov chain with N states and transition probability matrix $\mathbf{A} = [a_{ij}]_{N,N}$. If s_t is the state index of the Markov chain at sample point t, then $a_{ij} = Pr(s_{t+1} = j \mid s_t = i)$. Suppose the process observations O_t are drawn from a finite observation set $V = [v_1, v_2, \ldots, v_M]$, the state conditional distributions are represented by the conditional probabilities: $b_j(k) = Pr(O_t = v_k \mid s_t = j)$. Detailed description of HMMs can be found in [11], [12].

Among the models describing speech and handwriting the

• *J. Hu and M.K. Brown are with Lucent Technologies Bell Laboratories, 700 Mountain Ave., Murray Hill, NJ 07974.*
E-mail: {jianhu, mkb}@rbell-labs.com.
• *W. Turin is with AT&T Bell Laboratories, 600 Mountain Ave., Murray Hill, NJ 07974. E-mail: wt@research.att.com.*

Manuscript received Feb. 10, 1995; revised May 15, 1996. Recommended for acceptance by J.J. Hull.
For information on obtaining reprints of this article, please send e-mail to: transpami@computer.org, and reference IEEECS Log Number P96068.

most popular are the so-called left-to-right HMM's in which $a_{ij} = 0$ for $j < i$. The subcharacter and character models we have adopted are even more restrictive disallowing in addition state skipping ($a_{ij} = 0$ for $j > i + 1$). We have selected this relatively simple topology because it has been shown to be successful in speech recognition, and there has not been sufficient proof that more complex topologies would necessarily lead to better recognition performance. Furthermore, in unconstrained handwriting, skipping of segments seem to happen most often for ligatures, which in our system is handled by treating ligatures as special "connecting" letters and allowing them to be bypassed in the language model.

2.1 Nebulous Stroke Models

Early HMM based on-line handwriting recognition systems used word models as basic units, where a separate model is designed for each word [2]. Systems using letter models as basic units have become popular recently (e.g., [5], [6]). By tying models to letters instead of words, the model set does not increase with the size of the vocabulary. Furthermore, modification of the vocabulary can be done by simply changing the grammatical constraints of the system while leaving the model set intact. However, letter models are still unduly inefficient. English alphabet, like most western alphabets, contains patterns that are common to more than one letter. When letter models are used, the constituent patterns of individual letters are modeled independently regardless of their similarities, causing redundancy in storing and comparing the models. One way to solve this problem and further improve model efficiency is to use subcharacter models as basic units [4]. This approach has another advantage: shared patterns can be better trained with the same amount of training data, because by sharing models they also share training samples.

Several difficulties are involved in developing subcharacter models. First, it is not clear how to break letters into subcharacter units. Various sets of subcharacter primitives have been proposed before for non-HMM based on-line systems (e.g., [13]). More recently, Bercu and Lorette proposed segmenting an on-line handwriting sample into a succession of primitives classified as loops, humps and cusps and using them as feature observations in an HMM based recognition system [4]. A common drawback of the above approaches is that the handwriting sample has to be segmented before recognition, which often prematurely limits the hypothesis space. Furthermore, training of subcharacter models can be difficult, because while training of letter models can be initiated using samples of isolated letters, it is hard to obtain reliable samples of isolated subcharacter patterns.

We introduce an approach using subcharacter models called *nebulous stroke models*. A letter model is a concatenation of several such stroke models. Sharing among letters is enforced by referring to the same stroke models. A stroke could be any segment of a handwritten script. This is the major novelty of this approach: we do not attempt to impose rigidly defined stroke models, instead, we make the system learn indefinite stroke models through training, hence the term *nebulous*. The only constraints given to the system are the number of strokes in each letter, and which of the strokes are shared. No manual segmentation is involved in training: the stroke models are trained first on isolated letters and later on whole word samples, but never on isolated or segmented strokes (which would be impossible to obtain since there is no specific definition of a stroke). Using this approach, the system is allowed the freedom to decide on the most "natural" way of segmenting letters into strokes. Furthermore, this decision can be adjusted automatically through training when more and more new features are taken into consideration. Fig. 2 shows the segmentation of several letter samples as a result of the training procedure when a one-state HMM is used to model each stroke, with a single tangent slope angle feature. In the samples shown in the figure, there are six, eight, and five strokes in "a," "g," and "j" (without the dot), respectively. "a" and "g" share the first 4 strokes s1, s4, s5, s6; "g" and "j" share the last four strokes **s18**, **s19**, **s20**, **s1**; stroke **s1** (corresponding to upward ligature) is shared by all three samples. The training procedure will be discussed in more detail later in Section 2.4.

Ligatures are attached to letters only during training. At recognition time they are treated as special, one stroke "connecting" letters inserted between "core" letters and can be skipped with no penalty (see Fig. 3). This treatment insures that our system can handle mixed style handwriting as opposed to pure cursive only.

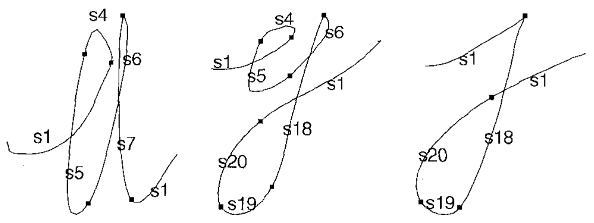


Fig. 2. Stroke segmentation of three letter samples.

In English cursive script, crosses (for "t"s and "x"s) and dots (for "i"s and "j"s) are referred to as delayed strokes. In previous methods delayed strokes are first detected in preprocessing and then either discarded or used in postprocessing. There are two drawbacks to these approaches. First, the information contained in delayed strokes is wasted in the first case and inadequately used in the second case because stroke segmentation cannot be influenced by this data. Second, it is often difficult to detect delayed strokes reliably during preprocessing. The approach we have taken is to treat delayed strokes as special letters in the alphabet. A word with delayed strokes is given alternative spellings to accommodate all possible sequences with delayed strokes in different positions. During recognition, delayed strokes are considered as inherent parts of a script just like normal letters and contribute directly to the scoring of the hypotheses. Although potentially this approach can lead to a large increase in the number of hypotheses, the actual increase of searching space can be controlled through pruning in a beam search mechanism [14]. Alternatively, for large vocabulary tasks one could first use a simpler network with inexact delayed stroke modeling (e.g., one optional delayed stroke module at the end of the network shared by all paths) to obtain the top few most likely candidates and then use exact delayed stroke modeling on the reduced candidate set to obtain the final recognition result.

2.2 Grammatical Constraints

The nebulous stroke models described above are concatenated according to a lexicon to form letter models, which are also left to right HMMs without state skipping. A letter could be written in different patterns in which case each pattern is modeled by a separate HMM. Currently the entire lexicon is specified manually. The letter models are further embedded in a grammar network, which can represent a variety of grammatical constraints, e.g., word dictionaries, statistical N-gram language models and context free grammars.

The three primary components of the evolutional grammar are a grammar arc table, a null arc (unlabeled grammar arc) table, and a Recursive Transition Network (RTN) table which records the grammar rules for expanding the nonterminals [8]. Labeled grammar arcs can have either a nonterminal label representing another RTN subnetwork or an HMM label representing an HMM defined in another table called the lexicon, which describes the structure of the HMM for each letter pattern. The EG initially contains a degenerate grammar consisting of only a start node, an end node and a single arc with a non-terminal label. During the evolution process, upon encountering a non-terminal arc, the arc is first replaced with the subnetwork it represents and is examined again. This process continues until all nonterminal references on the earliest arcs are eliminated. If a resulting label references an HMM, the appropriate model structure is built as indicated by the lexicon. Once all leading HMM references are built, HMM score integration proceeds. As a score emerges from an HMM and needs to be propagated further in the network, additional evolution of the EG may occur. In this way, only those regions of the grammar touched by the HMM search are expanded. Beam search methods [14] can be used to limit the amount of grammar expansion.

Fig. 3 shows part of the evolutional grammar for a dictionary containing the word "can" and how it evolves during decoding. Labels in brackets are nonterminal labels and the others are terminal labels. Labels composed of a single letter followed by different digit subscripts refer to the separate HMM's for different patterns of the same letter; "lg1" refers to the upward ligature model. Unlabeled arcs are the null-arcs. For any given dictionary, a grammar compiler [15] is used to convert a list of word specifications into an optimized network with shared prefixes and suffixes.

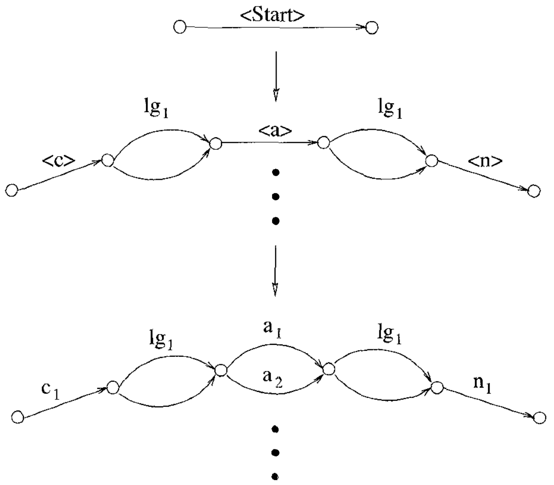


Fig. 3. Partial diagrams of a EG network.

2.3 Decoding

The Viterbi algorithm is used to search for the most likely state sequence corresponding to the given observation sequence and to give the accumulated likelihood score along this best path [11]. Suppose that for any state i, $q_i(t)$ denotes the selected state sequence (hypothesis) leading to i at sample point t, and $\delta_i(t)$ denotes the accumulated log-likelihood score of that hypothesis. O_t represents the observation at sample point t, and $\Delta_i(O_t)$ represents the log-likelihood score of O_t in state i. In our current model, for efficiency reasons, we assume that all the state-preserving probabilities a_{ii} are constant and therefore need not be included in the accumulated likelihood scores. (We have experimented with variable state preserving probabilities and they showed no significant improvement in recognition performance over the constant ones.) Since each letter model is a left-to-right HMM with no state skipping, within each letter model the hypothesis and its likelihood are updated as:

$$q_i(t) = \begin{cases} q_{i-1}(t-1), i & \text{if} \quad \delta_{i-1}(t-1) > \delta_i(t-1) \\ q_i(t-1), i & \text{otherwise} \end{cases} \qquad (1)$$

$$\delta_i(t) = \max \delta_{i-1}(t-1),\ \delta_i(t-1) + \Delta_i(O_t) \qquad (2)$$

We now explain how scores are propagated through grammar nodes. Suppose g is a grammar node and $p(g)$ and $s(g)$ denote the sets of preceding and succeeding letter pattern classes corresponding to the incoming and outgoing arcs respectively. For each letter pattern class l, $m(l)$ denotes the HMM used to model the pattern; $h(l)$ denotes the initial state of the model; and $f(l)$ denotes the final state of the model. At each sample point t during the Viterbi search, the maximum of all the accumulated scores at the final states of the preceding letter models, also called *incoming scores*, is found and propagated to the initial state of each of the succeeding models, along with the corresponding state sequence. The operation is carried out as follows:

$$k = \underset{l \in p(g)}{argmax}\, \delta_{f(l)}(t-1); \qquad (3)$$

and for each state $j = h(l);\ l \in s(g)$:

$$q_j(t) = \begin{cases} q_{f(k)}(t-1), j & \text{if } \delta_{f(k)}(t-1) > \delta_j(t-1) \\ q_j(t-1), j & \text{otherwise} \end{cases} \qquad (4)$$

$$\delta_j(t) = \begin{cases} \delta_{f(k)}(t-1) + \Delta_j(O_t) & \text{if } \delta_{f(k)}(t-1) > \delta_j(t-1) \\ \delta_j(t-1) + \Delta_j(O_t) & \text{otherwise} \end{cases} \qquad (5)$$

2.4 Model Training

Models are trained using the well known iterative segmental training method [11]. Given a set of training samples, the HMM for each sample is instantiated by concatenating HMMs for the appropriate letters, ligatures and delayed strokes, which are in turn instantiated by concatenating the composing stroke models. The training procedure then is carried out through iterations of segmentation of training samples by Viterbi algorithm using the current model parameters, followed by parameter reestimation using the means along the path. The iterative procedure stops when the difference between the likelihood scores of the current iteration and those of the previous one is smaller then a preset threshold.

We have developed a training process composed of three consecutive stages. The initial parameters for each stage are the output from the previous stage, while the initial parameters for the first stage are obtained through equal-length segmentation of the training samples. No manual segmentation is involved at any stage.

The first stage—*letter training*, is carried out on isolated letter samples including ligatures and delayed strokes. This stage essentially serves as a model initializer—it is left to the later training stages to fully capture the characteristics of cursive handwriting and variations among different writers. The model parameters obtained are then passed on as initial parameters for the second stage of training—*linear word training*, which is carried out on whole word samples. We call it linear because during this stage each word sample is bound to a single sequence of stroke models. In other words, each sample is labeled not only by the corresponding word, but also by the exact letter pattern sequence corresponding to the particular style of that sample, which is then converted to a unique stroke sequence according to the lexicon. Such highly constrained training is necessary to obtain reliable results when the models are not yet adequately trained. The disadvantage is that the letter pattern sequence corresponding to each sample has to be manually composed, which is a demanding and error prone process, especially when the training set is large.

This is the reason why we introduce the third training stage—*lattice word training*. As the name suggests, during this stage each word is represented by a lattice, or finite state network, that includes all possible stroke sequences that can be used to model the word. The finite state network corresponding to each word is described by a subgrammar. Each training sample is labeled only by the word (or the index to the subgrammar representing the word). The stroke sequence that best matches the sample is chosen by the decoding algorithm and the resulting segmentation is used for parameter re-estimation. This stage of training involves minimal amount of human supervision, and therefore can conveniently accommodate a large amount of training data. Fig. 4 shows the segmentation of a sample "rectangle" at the end of lattice word training. The solid squares show the boundaries between letters (including ligatures), and the stars indicate the boundaries between strokes.

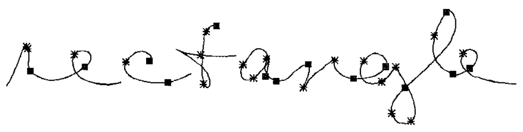


Fig. 4. Segmentation of a training word sample.

3 Invariant Features

In choosing handwriting features, we face the problem of variability in handwriting caused by the geometric distortion of letters and words by rotation, scaling and translation. In general, translation is not a serious problem because it is easy to chose features that are invariant with respect to translation. Examples include handwriting stroke tangents and curvature. Unfortunately, these features are not invariant with respect to the other two factors. For example, stroke tangents are invariant with respect to scale, but not rotation; curvature is invariant with respect to rotation, but not scale.

There are two principal methods for dealing with variability in pattern recognition. The patterns can be normalized before feature extraction by some set of preprocessing transformations, or features can be chosen to be insensitive to the undesirable variability. These two methods often need to be combined because neither one can solve the problem completely by itself. On one hand, excessive preprocessing is undesirable because it may result in premature, limiting decisions or loss of information. On the other hand, certain features that are not completely invariant (e.g., tangent slope angle) prove to be important in distinguishing different symbols. We have adapted one of the common features for handwriting recognition—tangent slope angle, which is invariant to translation and scaling, but not rotation. In this section, we introduce two new features for handwriting recognition that are invariant with respect to all three factors of geometric distortion.

We define a *similitude* transformation to be a combination of translation, rotation and scaling described by:

$$\mathbf{w} = c\,\mathbf{U}\,\mathbf{r} + \mathbf{v} = c\begin{bmatrix}\cos\omega & -\sin\omega\\ \sin\omega & \cos\omega\end{bmatrix}\mathbf{r} + \begin{bmatrix}v_x & v_y\end{bmatrix}^T,$$

where c is a positive scalar. Two curve segments are equivalent if they can be obtained from each other through a similitude transformation. Invariant features are features that have the same value at corresponding points on different equivalent curve segments.

Suppose that a smooth planar curve $\mathbf{P}(t) = (x(t), y(t))$ is mapped into $\tilde{\mathbf{P}}(\tilde{t}) = (\tilde{x}(\tilde{t}), \tilde{y}(\tilde{t}))$ by a reparameterization $t(\tilde{t})$ and a similitude transformation, i.e., $\tilde{\mathbf{P}}(\tilde{t}) = c\,\mathbf{U}\,\mathbf{P}(t(\tilde{t})) + \mathbf{v}$. Without loss of generality, assume that both curves are parameterized by arc length (natural parameter), i.e., $t = s$ and $\tilde{t} = \tilde{s}$. Obviously, $d\tilde{s} = c\,ds$, thus the corresponding points on the two curves are related by $\tilde{\mathbf{P}}(\tilde{s}) = c\,\mathbf{U}\,\mathbf{P}((\tilde{s} - \tilde{s}_0)/c) + \mathbf{v}$. It can then be shown [16] that curvature (the reciprocal of radius) at the corresponding points of the two curves is scaled by $1/c$, i.e., $\tilde{\kappa}(\tilde{s}) = \frac{1}{c}\kappa((\tilde{s} - \tilde{s}_0)/c)$. It follows that:

$$\tilde{\kappa}'(\tilde{s})\tilde{\kappa}^{-2}(\tilde{s}) = \kappa'\big((\tilde{s} - \tilde{s}_0 / c)\kappa^{-2}\big((\tilde{s} - \tilde{s}_0)/c\big)\big) \qquad (6)$$

where $\tilde{\kappa}' = d\tilde{\kappa}/d\tilde{s}$ and $\kappa' = d\kappa/ds$. Equation (6) defines an invariant feature which we shall refer to as *normalized curvature*.

An other set of invariants that require lower orders of derivatives can be obtained by using the invariance of distance ratios between corresponding points. Consider again the two equivalent curves $\mathbf{P}(t)$ and $\tilde{\mathbf{P}}(\tilde{t})$ defined above. Suppose P_1 and P_2 are two points on curve $\mathbf{P}(t)$ whose tangent slope angles differ by θ, P is the intersection of the two tangents on $\mathbf{P}(t)$. Similarly, $\tilde{P}_1$ and $\tilde{P}_2$ are two points on curve $\tilde{\mathbf{P}}(\tilde{t})$ whose tangent slope angles also differ by θ, and $\tilde{P}$ is the intersection of the two tangents on $\tilde{\mathbf{P}}(\tilde{t})$ (Fig. 5). Since angles and hence turns of the curve are invariant under the similitude transformation, it can be shown that if point $\tilde{P}_1$ corresponds to point P_1, then points $\tilde{P}_2$ and $\tilde{P}$ correspond to points P_2 and P, respectively [16], thus $|\tilde{P}_1\tilde{P}| = c|P_1P|$ and $|\tilde{P}\tilde{P}_2| = c|PP_2|$. Therefore we have:

$$\frac{|\tilde{P}\tilde{P}_2|}{|\tilde{P}_1\tilde{P}|} = \frac{|PP_2|}{|P_1P|}. \qquad (7)$$

If we fix the value of θ to a constant θ_0, then (7) defines another invariant feature which can be computed at each sample point. We call this feature *ratio of tangents*. In order to enhance the distinctive power of the feature, we augment it by the sign of the local curvature. The resulted feature is called *signed ratio of tangents*, and is used instead of ratio of tangents in the experiments described later.

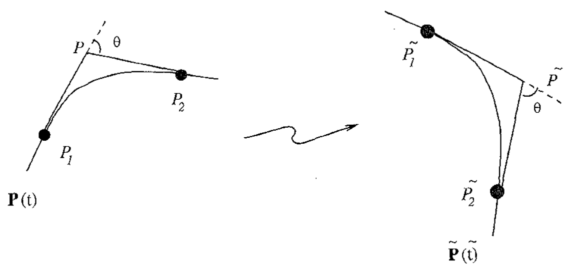


Fig. 5. Ratio of tangents.

To evaluate accurately the invariant features described above, high quality derivative estimates of up to the third order have to be obtained from the sample points. Obviously simple finite difference based methods for derivative estimation do not provide the needed insensitivity to spatial quantization error or noise. We have applied a set of smoothing spline operators of up to the fifth order for this purpose [10].

With the addition of the two invariant features the observation vector now contains three dimensions. Even though these vectors are continuous in nature, we chose to use discrete HMM's instead of continuous density HMM's to avoid making assumptions on the form of the underlying distribution. To simplify our models, we also chose to treat the features as being independent from each other. A separate distribution is estimated for each feature in each state during training. The joint probability of observing symbol vector $S_{k_1k_2k_3} = \{k_1, k_2, k_3\}$ in state j is:

$$b_j\left(S_{k_1k_2k_3}\right) = \prod_{i=1}^{3} b_{ji}(k_i),$$

where $b_{ji}(k_i)$ is the probability of observing symbol k_i in state j according to the probability distribution of the ith feature. In order to adjust the influence of different features according to their discriminative power, we compute the *weighted log-likelihood*:

$$\hat{L}_j\left(S_{k_1k_2k_3}\right) = \sum_{i=1}^{3} w_i \ln\left[b_{ji}(k_i)\right] - \ln\left(N_j\right),$$

derived form the *weighted probabilities*:

$$\hat{b}_j\left(S_{k_1k_2k_3}\right) = \prod_{i=1}^{3}\left[b_{ji}(k_i)^{w_i}/N_j\right],$$

where N_j is the state normalization factor such that the weighted probabilities satisfy the condition:

$$\sum_{k_1k_2k_3} \hat{b}_j\left(S_{k_1k_2k_3}\right) = 1\ ;\ \forall j,$$

thus are not biased towards any particular state. The weights w_i are positive scalars and specify the relative dominance of each feature.

4 Segmental Features

The HMM system described so far relies on localized features. During the decoding process, for each new sample data point taken in a time ordered sequence, the HMM hypotheses scores are discretely integrated and propagated through the HMM network. The incremental score at each step depends only on the local feature at the current point. We will refer to methods of this type as *point oriented*. The advantage of point oriented methods is that all knowledge sources are integrated into a single model. Because of this, all possible segmentations and identifications of the input pattern are considered in an efficient manner. On the other hand, point oriented methods have the disadvantage of using only localized observation measurements. Thus, shape information on larger scales is missing from the process. One way to remedy this problem is to extract features from a window around each sample point, where the window could be of fixed (e.g., [17]) or variable size (e.g., features in [6], the ratio of tangents feature described in the previous section, etc.). The weakness of this method is that it does not adapt to the varying characteristics of pattern shapes, sizes, or segmentation boundaries.

There are methods that trade the efficiency advantage of point oriented methods for the greater accuracy of measurements over larger regions. These methods fall into the class called *segment oriented* [4] where a script is first presegmented into letters or subcharacter primitives according to certain predefined boundary conditions (pen-ups, cusps, etc.) and one observation feature vector is computed for each segment. To avoid loosing potentially good hypotheses, segment oriented methods should first generate all possible segmentations because the scoring knowledge is not available at the time of segmentation and will be applied later as a post-process, but in practices no system actually does this because the number of possible segmentations makes the problem intractable.

We propose a new segment oriented method that ameliorates the usual tradeoff between efficiency and accuracy. We call this the *interleaved segmental* method. Point oriented methods are used to obtain partial segmentation hypotheses which are augmented with observation measurements made on the hypothesized segments. The resulted system is called an *augmented HMM system.*

In Section 2.3, we explained how hypotheses are propagated through a grammar node g in a commonly used implementation ((3), (4), (5)). In order to incorporate segmental shape information (in this case letter shape information) into the search, we augment the incoming scores with letter matching scores, computed using global letter shape models. To be more specific, let $\alpha_l(t_1, t_2)$ be the likelihood score of the segment from sample point t_1 to t_2 on the observation sequence being matched to letter pattern class l, the augmented incoming scores are defined as $\delta'_{f(l)}(t-1) = \delta_{f(l)}(t-1) + \alpha_l(t_l, t-1)$, where $t_l = t - d_{m(l)}(t-1)$ and $d_{m(l)}(t-1)$ is the number of sample points assigned to letter model $m(l)$ (letter duration) up to sample point t–1. Using these augmented scores, (3), (4), (5) are replaced by the following:

$$k = \underset{l\in p(g)}{argmax}\, \delta'_{f(l)}(t-1); \tag{8}$$

and for each state $j = h(l)$; $l \in s(g)$:

$$q_j(t) = \begin{cases} q_{f(k)}(t-1), j & \text{if } \delta'_{f(k)}(t-1) > \delta_j(t-1) \\ q_j(t-1), j & \text{otherwise} \end{cases} \tag{9}$$

$$\delta_j(t) = \begin{cases} \delta'_{f(k)}(t-1) + \Delta_j(O_t) & \text{if } \delta'_{f(k)}(t-1) > \delta_j(t-1) \\ \delta_j(t-1) + \Delta_j(O_t) & \text{otherwise} \end{cases} \tag{10}$$

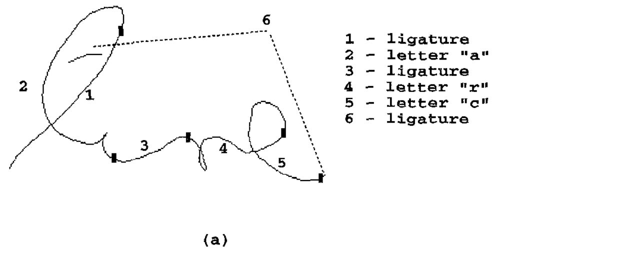
By augmenting the incoming scores at each grammar node with the letter matching scores, the overall shapes of letters are taken into consideration when the hypotheses leading to the grammar node are ranked and the one with the highest rank is chosen and propagated to the succeeding letter models. Through this mechanism, letter matching scores computed on dynamically allocated segments directly affect the decision making at each point during the Viterbi search, so that the system is biased towards sequences with better matches at the letter level.

It should be pointed out that in such an augmented HMM system, the state sequence resulted from Viterbi search is no longer guaranteed to be the optimal sequence, because now the accumulated score of a sequence leading to state i at sample point t not only depends on the previous state, but also on *how* the previous state was reached (history reflected in the letter duration). This dependence violates the basic condition for the Viterbi algorithm to yield optimum solution. However, as shall be shown later, our experimental results suggest that the gain obtained by incorporating segmental features by far outweighs the loss in the optimality of the algorithm.

The segmental matching score $\alpha(t_1, t_2)$ is computed using a correlation based metric inspired by the metrics developed by Sinden and Wilfong [18]. Given a segment a with coordinate sequence $\eta_a = \langle(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\rangle$, the *instance vector* of a is defined as: $v_a = (\bar{x}_1, \bar{y}_1, \bar{x}_2, \bar{y}_2, \ldots, \bar{x}_n, \bar{y}_n)$, where $\bar{x}_i = x_i - x_a$ and $\bar{y}_i = y_i - y_a$ for $1 \le i \le n$, and (x_a, y_a) is the centroid of a. The *normalized instance vector* of a, $u_a = v_a/|v_a|$, is a translation and scale independent representation of segment a in R^{2n}. Through a resampling procedure, a sample segment of arbitrary length can be mapped to a vector in R^{2N} where N is a predetermined number. The difference between any two sample segments a and b is defined as: $D(a, b) = 0.5(1 - u_a \cdot u_b)$, whose value ranges from 0 (when a and b are identical) to 1. The segmental matching score $\alpha_l(t_1, t_2)$, is then defined as: $\alpha_l(t_1, t_2) = -w_\alpha D(a_l, a_{t_1,t_2})$, where a_l is the *model segment* for letter pattern l, a_{t_1,t_2} is the segment from sample point t_1 to t_2 on the input sample sequence and w_a is a weight factor.

In order to compute the above segment matching score, a single model segment needs to be derived for each letter pattern class. Let $u_1, u_2, \ldots, u_M$ be the normalized instance vectors of a set of prototypes for the letter pattern class l (which can be easily obtained as side products of segmental training). A single model segment representing this class is represented by vector w which minimizes the sum of distances from individual prototypes. It can be easily shown that $w = \tilde{u}/|\tilde{u}|$, where $\tilde{u} = \sum_{i=1}^{M} u_i$.

Fig. 6 illustrates the effect of incorporating letter matching scores in the HMM system by comparing the different letter level segmentations obtained without and with letter matching scores. Fig. 6a shows the segmentation of a sample of word "line" when the basic HMM system was used and the sample was falsely recognized as "arc." Fig. 6b shows the segmentation of the same sample when letter matching scores are applied and the sample is correctly recognized. The hypothesis shown in Fig. 6a is not selected now because the segment corresponding to letter "a" does not match the corresponding model segment well and therefore yields a poor letter matching score.



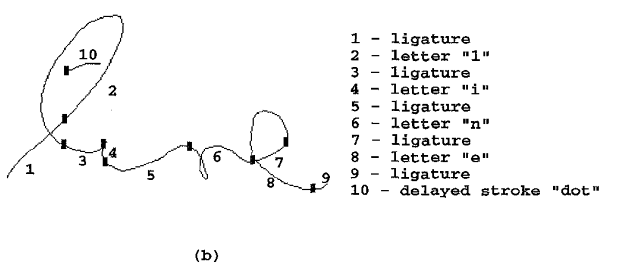


Fig. 6. Different letter segmentations of a sample of word "line".

5 Experimental Results

To test our system, we composed a vocabulary of 32 words, targeting an underlying application of a pen driven graphics editor. The vocabulary covers all 26 lower case letters in the English alphabet, contains many groups of easily confused words, such as "line," "lines" and "spline," "cut" and "out," and has both very short words such as "in" and relatively long ones such as "rectangle." Samples from 18 writers have been collected, the writer group containing men and women, left-handed and right-handed, of many different cultural origins: American, European, Asian, and South American. During sample collection, the writers were told to write in their most natural way with no explicit constraints. Each word was written 15 times by each writer. After removing invalid samples (samples with misspelling or parts missing due to hardware problems), the final data set consists of 8,595 word samples.

The isolated letter samples used for letter training were all written by one writer, imitating all writing styles known to the authors. 10–15 samples were written for each unique style of each letter and a particular stroke sequence is bound to those samples. There are all together 54 lower-case letter models (as sequences of stroke models), including delayed strokes and ligatures, composed of a total of 93 stroke models. Each stroke is currently modeled by a single state.

Ten writers were chosen (after data collection) to be the "training writers." The whole word training set is composed of about 10 samples of each word from each training writer, a total of 3,180 samples. About 600 of the 3,180 training samples are used for linear word training, and the whole training set is used for lattice word training. The test set is composed of all samples not used for training, divided into two groups. Group A is the multiple-writer test group which contains 1,592 samples from the 10 training writers, group B is the writer-independent test group composed of 3,823 samples from the eight other writers.

Table 1 summarizes the performance of our recognition system. Error rates from three experiments as well as the settings for each experiment are listed. The top line of Fig. 7 shows some of the samples from test set B that were not recognized correctly. In fact they are so sloppy that even human beings can hardly recognize them correctly. The bottom line of the same figure shows some of the more legible samples from the same writer that were recognized correctly.

TABLE 1
Error Rates from Three Experiments

	A (multiple-writer)	B (writer-independent)
1	10.3%	23.4%
2	5.6%	10.7%
3	3.1%	5.5%

1. *Basic HMM system with single tangent slope angle feature.*
2. *Basic HMM system with two additional, invariant features: normalized curvature and signed ratio of tangents.*
3. *Augmented HMM system.*

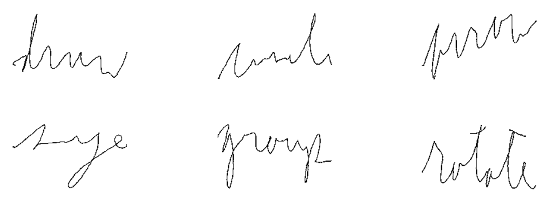


Fig. 7. Top: examples of scripts recognized incorrectly ("draw," "circle," and "arrow"); bottom: examples of scripts recognized correctly ("size," "group," and "rotate").

6 Concluding Remarks

We have described experiments in handwriting recognition using hidden Markov modeling and stochastic language modeling methods originally developed for speech applications. These methods are generalized in the AEGIS architecture. Subcharacter models called nebulous stroke models are used to model the basic units in handwriting. We introduce two new features for handwriting that are invariant under translation, rotation and scaling. Invariant features have been discussed extensively in computer vision literature. However, they have been rarely used previously in real applications due to the difficulty involved in the estimation of high order derivatives. We have demonstrated that these high order invariant features can indeed be made useful with careful implementation.

A method for combining segment oriented features in a stochastic pattern recognizer has been developed. In this method, called interleaved segmental method, partial segmentation hypotheses obtained using the point oriented features in a conventional dynamic programming search are combined with scores based on segmental shape measurements made on the hypothesized segments. Although certain optimality characteristics of the HMM system are sacrificed in the process, significant reduction in recognition error was achieved by this method, reducing writer-independent error rate by nearly 50%.

Finally, we would like to point out that although we report only recognition results on a relatively small vocabulary of 32 words in this paper, none of the techniques presented is inherently dependent on the size of the vocabulary. The system can be easily adjusted to handle a large or unlimited vocabulary by imposing different grammatical constraints. For example, one could use a statistical N-gram grammar instead of a dictionary to allow an unlimited vocabulary. Experiments are to be carried out in the future to test the system's performance on large and unlimited vocabularies.

REFERENCES

[1] C.C. Tappert, C.Y. Suen, and T. Wakahara, "The State of the Art in On-Line Handwriting Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 8, pp. 787-808, Aug. 1990.

[2] R. Nag, K.H. Wong, and F. Fallside, "Script Recognition Using Hidden Markov Models," *Proc. ICASSP '86*, vol. 3, pp. 2,071-2,074, Japan, Apr. 1986.

[3] A. Kundu and P. Bahl, "Recognition of Handwritten Script: A Hidden Markov Model Based Approach," *Proc. ICASSP '88*, vol. 2, pp. 928-931, New York, Apr. 1988.

[4] S. Bercu and G. Lorette, "On-Line Handwritten Word Recognition: An Approach Based on Hidden Markov Models," *Proc. Third IWFHR*, pp. 385-390, Buffalo, N.Y., May 1993.

[5] J. Makhoul, T. Starner, R. Schartz, and G. Lou, "On-Line Cursive Handwriting Recognition Using Speech Recognition Models," *Proc. ICASSP '94*, pp. v125-v128, Adelaide, Australia, Apr. 1994.

[6] K.S. Nathan, H.S. M. Beigi, J. Subrahmonia, G.J. Clary, and H. Maruyama, "Real-Time On-Line Unconstrained Handwriting Recognition Using Statistical Methods," *Proc. ICASSP '95*, pp. 2,619,-2,622, Detroit, Mich., June 1995.

[7] 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.

[8] M.K. Brown and S.C. Glinski, "Stochastic Context-Free Language Modeling with Evolutional Grammars," *Proc. ICSLP '94*, vol. 2, pp. 779-782, Yokohama, Japan, Sept. 1994.

[9] G.L. Miller and R. Boie, "Capacitive Proximity Sensors," U.S. Patent #5,337,353,9, Aug. 1994.

[10] J. Hu, M.K. Brown, and W. Turin, "Invariant Features for HMM Based Handwriting Recognition," *Proc. ICIAP '95*, pp. 588-593, Sanremo, Italy, Sept. 1995.

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

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

[13] S.A. Guberman and V.V. Rozentsveig, "Algorithm for the Recognition of Handwritten Text," *Automation and Remote Control*, vol. 37, pp. 751-757, May 1976. (Translated from *Automatika i Telemekhanika*, vol. 37, no. 5, pp. 122-129, May 1976.)

[14] B.T. Lowerre and D.R. Reddy, "The HARPY Speech Understanding System," *Trends in Speech Recognition*, W.A. Lean, ed., chapter 15, pp. 340-360. Prentice Hall, 1980.

[15] M.K. Brown and J.G. Wilpon, "A Grammar Compiler for Connected Speech Recognition," *IEEE Trans. Signal Processing*, vol. 39, no. 1, pp. 17-28, Jan. 1991.

[16] A.M. Bruckstein, R.J. Holt, A.N. Netravali, and T.J. Richardson, "Invariant Signatures for Planar Shape Recognition Under Partial Occlusion," *CVGIP: Image Understanding*, vol. 58, pp. 49-65, July 1993.

[17] M. Schenkel, I. Guyon, and D. Henderson, "On-Line Cursive Script Recognition Using Time Delay Neural Networks and Hidden markov Models," *Special Issue of Machine Vision and Application on Cursive Script Recognition*, R. Plamondon, ed. Springer Verlag, 1995.

[18] F. Sinden and G. Wilfong, "Method of Recognizing Handwritten Symbols," U.S. Patent #5,333,209, July 26, 1994.

A New Methodology for Gray-Scale Character Segmentation and Recognition

Seong-Whan Lee, *Member, IEEE Computer Society*,
Dong-June Lee, *Member, IEEE*, and
Hee-Seon Park, *Member, IEEE*

Abstract—Generally speaking, through the binarization of gray-scale images, useful information for the segmentation of touched or overlapped characters may be lost in many cases. If we analyze gray-scale images, however, specific topographic features and the variation of intensities can be observed in the character boundaries. We believe that such kinds of clues obtained from gray-scale images may work for efficient character segmentation and recognition. In this paper, we propose a new methodology for character segmentation and recognition which makes the best use of the characteristics of gray-scale images. In the proposed methodology, the character segmentation regions are determined by using projection profiles and topographic features extracted from the gray-scale images. Then a nonlinear character segmentation path in each character segmentation region is found by using multi-stage graph search algorithm. Finally, in order to confirm the nonlinear character segmentation paths and recognition results, recognition-based segmentation method is adopted. Through the experiments with various kinds of printed documents, it is convinced that the proposed methodology is very effective for the segmentation and recognition of touched and overlapped characters.

Index Terms—Character segmentation and recognition, topographic feature, gray-scale character recognition, multistage graph search, recognition-based segmentation.

1 INTRODUCTION

IT is a challengeable issue to develop a practical system which can maintain a high recognition accuracy, independent of the quality of the input documents and the character fonts. Very often even in printed text, adjacent characters tend to be touched or overlapped. Therefore, it is essential to segment a given string correctly into its character components. Any failure or error in this segmentation step produces a negative effect on character recognition [1].

The complexity of character segmentation stems from the wide variety of fonts, rapidly expanding text styles, and image characteristics such as poor-quality printing and poor binary images. Touched, overlapped, separated, and broken characters are major factors for causing segmentation errors. Moreover, when a document is composed of multiple languages, (e.g., Hangul with alphanumeric characters), it is more difficult to segment characters due to differences in character sizes and touching types of each language.

Previous methods for character segmentation can be roughly classified into three categories: straight segmentation method, recognition-based segmentation method, and cut classification method.

• *S.-W. Lee is with the Dept. of Computer Science and Engineering, Korea University, Anam-dong, Seongbuk-ku, Seoul 136-701, Korea.*
E-mail: swlee@human.korea.ac.kr.

• *D.-J. Lee is with the Telecommunication Network Research Lab., Korea Telecom, Woomyun-dong, Suhcho-ku, Seoul 137-792, Korea*
E-mail: djlee@tigermask.kotel.co.kr.

• *H.-S. Park is with the Multimedia Lab, Samsung Electronics Co. Ltd., Suwon P.O. Box 105, Kyungki-do 440-600, Korea.*
E-mail: hspark@coda.info.samsung.co.kr.

Manuscript received Mar. 13, 1995; revised Dec. 26, 1995. Recommended for acceptance by J.J. Hull.
For information on obtaining reprints of this article, please send e-mail to: transpami@computer.org, and reference IEEECS Log Number P96081.