

The parametric eigenspace representation enables us to determine an illumination that is optimal when the poses of the objects are unknown. The planning process is further simplified if the poses of the objects are in fact fixed. In this case, the eigenspace representation of each object is a point and only the distances between points need be computed to determine the optimal illumination.

In the experiments, a single parameter was used to represent object pose (rotation) for a given stable configuration. For certain applications, three degree of freedom (DOF) may be needed to describe object pose. In such cases, for any given illumination, object appearance is represented in eigenspace as a 3-DOF manifold. This, of course, involves the acquisition of a larger number of object images for each illumination. Further, illumination itself can be described using additional parameters, including source size, distance, color, and the number of sources. In [7], optimization of illumination color is described and demonstrated by experiments.

The planning approach can also be used to simultaneously optimize multiple parameters. The only requirement is that these parameters be varied during the acquisition of the planning image set. Clearly, for multiple parameters, acquiring image sets, computing parametric eigenspaces, and determining the optimal parameter values can be very time consuming. Therefore, our planning method may prove impractical when more than three illumination parameters must be jointly optimized. A lesser number of parameters, however, can be easily accommodated since illumination planning is typically done off-line and only once. As a result, it is generally not subject to severe time constraints.

REFERENCES

- [1] B. G. Batchelor, "A Prolog lighting advisor," *SPIE Proc. Intelligent Robots and Computer Vision VIII: Systems and Applications*, B. G. Batchelor, Ed., vol. 1193, 1993.
- [2] C. Cowan and A. Bergman, "Determining the camera and light source location for a visual task," in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 509-514, 1989.
- [3] C. Cowan, and D. Nitzan, "Automatic placement of vision sensors" in *Proc. 18th Annual NSF Conf. Design and Manufacturing Systems*, Georgia Tech University, Atlanta, GA, Jan., 1992.
- [4] H. Murase, F. Kimura, M. Yoshimura, and Y. Miyake, "An improvement of the auto-correlation matrix in pattern matching method and its application to handprinted 'HIRAGANA'," *Trans. IECE*, vol. J64-D, no. 3, 1981.
- [5] H. Murase and M. Lindenbaum, "Spatial temporal adaptive method for partial eigenstructure decomposition of large images," NTT Tech. Rep. No. 6527, Mar. 1992. Also in *IEEE Trans. Image Processing*, in press.
- [6] H. Murase and S. K. Nayar, "Visual learning and recognition of 3D objects from appearance," *Int. J. Comput. Vision*, vol. 14, 1995.
- [7] —, "Illumination planning for object recognition in structured environments," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 31-38, 1994.
- [8] E. Oja, *Subspace methods of Pattern Recognition*. Hertfordshire: Research Studies, 1983.
- [9] W. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C*. Cambridge: Cambridge University Press, 1988.
- [10] S. Sakane, M. Sato, and M. Kakikura, "Automatic planning of light source placement for an active photometric stereo system," in *Proc. IEEE Int. Workshop on Intelligent Robots and Systems*, pp. 559-566, 1990.
- [11] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 586-591, June 1991.
- [12] S. Yi, R. M. Haralick, and L. G. Shapiro, "Optimal sensor and light source positioning for machine vision," in *Proc. Int. Conf. Pattern Recognition*, 1990.
- [13] S. Yi, "Illumination control expert for machine vision: A goal driven approach," Ph.D. dissertation, Dept. of Computer Science and Engineering, Univ. of Washington, Seattle, Mar. 1990.

A Fast Statistical Mixture Algorithm for On-Line Handwriting Recognition

Eveline J. Bellegarda, Jerome R. Bellegarda,
David Nahamoo, and Krishna S. Nathan

Abstract—The automatic recognition of on-line handwriting is considered from an information theoretic viewpoint. Emphasis is placed on the recognition of unconstrained handwriting, a general combination of cursive written word fragments and discretely written characters. Existing recognition algorithms, such as elastic matching, are severely challenged by the variability inherent to unconstrained handwriting. This motivates the development of a probabilistic framework suitable to the derivation of a fast statistical mixture algorithm. This algorithm exhibits about the same degree of complexity as elastic matching, while being more flexible and potentially more robust. The approach relies on a novel front-end processor that, unlike conventional character or stroke-based processing, articulates around a small elementary unit of handwriting called a frame. The algorithm is based on 1) producing feature vectors representing each frame in one (or several) feature spaces, 2) Gaussian K -means clustering in these spaces, and 3) mixture modeling taking into account the contributions of all relevant clusters in each space. The approach is illustrated on a simple task involving a 81-character alphabet. Both writer-dependent and writer-independent recognition results are found to be competitive with their elastic matching counterparts.

Index Terms—On-line handwriting recognition, statistical modeling, frame-based processing, mixture output distributions.

I. INTRODUCTION

As they allow natural interactions between man and machine without the use of a keyboard, speech and handwriting are increasingly gaining acceptance as input to computer devices. Although speech is better suited for bulk data entry, handwriting is an attractive input medium in noisy acoustic environments, when a quiet operation is desired, or where privacy is a primary concern. Examples of applications where handwriting is the preferred input channel include form filling, personal note taking, document editing, record keeping, and meeting support. To address such applications, computer interfaces must include the capability of automatically recognizing natural handwriting.

Pen-based user interfaces most often capture handwriting through a special pen operating on an electronic tablet. The handwriting data recorded on the tablet is sent to a recognizer that outputs a string of typeset characters. Such operation requires the on-line recognition of the handwritten data. It is therefore to be distinguished from optical character recognition (OCR), that is always performed off-line. In OCR the basic input unit is a bitmap, whereas in on-line handwriting it is the trace the pen leaves on the tablet [1]. As a result, OCR devices operate on a static image of the handwriting, while on-line devices

Manuscript received August 19, 1993; revised December 15, 1993. Recommended for acceptance by Associate Editor R. Kasturi.

The authors are with IBM Research, T. J. Watson Research Center, Yorktown Heights, NY 10598 USA.

IEEE Log Number 9405790.

capture the dynamic (or temporal) characteristics of the writing, that can be exploited when performing recognition. In addition, in on-line operation the line thickness of the trace is essentially one pixel wide, which avoids the need for skeletonization or contour processing found in OCR. Finally in on-line mode there is immediate feedback from the recognizer to the user, which potentially enables the user to adjust for his or her writing idiosyncrasies.

One can distinguish four different types of handwriting, each of increasing difficulty for an automatic handwriting recognition system. *Discrete handwriting* (also referred to as handprint) is obtained when the user is constrained to write either within predefined boxes or with significant spacing between characters, so that the characters are naturally segmented. Slightly less constrained is discrete handwriting produced in *run-on mode*, when the characters are allowed to touch each other. A single stroke cannot include more than one character, so the transition from one letter to the other always corresponds to a pen lift. This makes segmentation relatively straightforward. In *pure cursive* handwriting, all the letters forming a word are connected, with the exception of a small number of well-defined cases: dots on *i* and *j*, crosses on *t* and *x*. Recognition can be performed either at the word level or at the character level. In the former case, each word must be segmented into a string of characters, a nontrivial and challenging problem [2]. Postprocessing, such as dictionary checking, is often necessary to attain acceptable recognition accuracy. Finally, the most common way to write is through a mixture of discrete and cursive modes; such *unconstrained handwriting* is the most likely to occur in a practical situation.

Variability within and between writers is more pronounced in cursive or unconstrained handwriting than in discrete or handprint. Between-writer variability comes from the fact that an idealized model of a character always undergoes writer-specific distortion. Within-writer variability arises from contextual distortions that depend on the position of each character within a given word. In both cases, characters are deformed into different topological variations called allographs [3]. Thus, correct modeling of each character (or, more generally, of each unit to be recognized) is a key aspect of the overall performance of a handwriting recognition system.

The aim of this paper is to develop an efficient modeling suitable for all types of handwriting. We propose a statistical framework inspired from one originally developed for speech recognition [4]. Moreover, to better capture some of the topological variations, we adopt a modeling philosophy based on a much finer unit of handwriting than considered so far.

The paper is organized as follows. In the next section we propose a general modeling formalism to provide a unifying framework for the rest of the paper. In Section III, we develop a novel feature representation, based on a two-stage process aiming at extracting the most informative events in handwriting. We refer to the resulting features as the *chirographic* representation of handwriting. Section IV discusses the basic model topology used to describe the observed feature vectors, as well as issues related to training and prototype establishment. In Section V, we explain how recognition is performed, and what advantages are brought about by the new framework. Finally, Section VI presents some experimental results illustrating the above approach on a simple task with an 81-character alphabet. Both writer-dependent and writer-independent experiments are conducted.

II. GENERAL FORMALISM

Let $\vec{C}$ be the sequence of characters that the user wants to write and $\vec{S}$ the corresponding scribble sequence recorded on the tablet. In this notation, handwriting production consists in generating the (recorded) scribble sequence $\vec{S}$ from the (intended) character sequence $\vec{C}$,

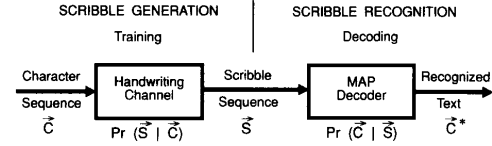


Fig. 1. Handwriting production and recognition: An information-theoretic perspective.

while handwriting recognition consists in retrieving the (unknown) character sequence $\vec{C}$ from the (observed) scribble sequence $\vec{S}$. These two complementary processes are illustrated in Fig. 1.

The goal in a recognition process is to retrieve the actual character sequence $\vec{C}$ that was written by the user. The problem is therefore to find a character sequence that is as close as possible to the written sequence. In a statistical sense, this is achieved by minimizing the probability of recognition error. During the scribble recognition process, we therefore compute the character sequence $\vec{C}^*$ that has the maximum *a posteriori* (MAP) probability given the information-bearing input sequence $\vec{S}$. In other words, $\vec{C}^*$ is obtained as:

$$\vec{C}^* = \arg \max_{\vec{C}} \Pr(\vec{C} | \vec{S}) \quad (1)$$

where the maximization is performed over all possible character sequences, $\vec{C}$, that the user might write. Using Bayes' rule, we can write: $\Pr(\vec{C} | \vec{S}) = \Pr(\vec{S} | \vec{C}) \Pr(\vec{C}) / \Pr(\vec{S})$. Since the denominator in this expression does not depend on $\vec{C}$, (1) can be rewritten as:

$$\vec{C}^* = \arg \max_{\vec{C}} \Pr(\vec{S} | \vec{C}) \Pr(\vec{C}). \quad (2)$$

In (2), the term $\Pr(\vec{C})$ is the source model, i.e., it characterizes the possible sequence of characters in the language considered. (This could include English words, digit strings, and/or sets of mathematical expressions, for example.) It can therefore be obtained through a *language model*, which may vary in complexity from a simple dictionary to more complex statistical language models based on (letter or word) *n*-grams.

As for the term $\Pr(\vec{S} | \vec{C})$, it characterizes all scribble sequences that can be generated for each intended sequence $\vec{C}$. Clearly, the scribble production process is a one-to-many mapping, since the particular sequence $\vec{S}$ is essentially unreproducible. Loosely speaking, $\Pr(\vec{S} | \vec{C})$ characterizes the distortion that occurs as the sequence of characters $\vec{C}$ gets produced, which depends on many external factors such as writer identity, tablet and pen used, noise environment, etc. Borrowing from information theory, we will call it the handwriting channel distortion. Therefore $\Pr(\vec{S} | \vec{C})$ can be calculated through a *handwriting channel model*, fully specified during training¹ (cf. Fig. 2).

While the above framework is valid for all choices of scribble representation, it is clearly desirable to define one that is well-suited for all types of handwriting, including unconstrained handwriting. Our basic premise is that for unconstrained handwriting, where there are numerous topological variations on each character, efficiency can only be attained by working with small, elementary handwriting units. This leads to the decomposition of the scribble sequence into frames, each corresponding to a small, uniquely informative, portion of handwriting. More specifically, let the scribble sequence $\vec{S}$ be obtained as: $\vec{S} = f_1 \cdots f_N$, where each f_i is a scribble unit (frame) to be described shortly. Similarly, let the character sequence

¹Note that the terminology "training" used here does not imply writer dependence. For a writer-independent task, $\Pr(\vec{S} | \vec{C})$ is typically obtained using training data from a pool of different writers, while for a writer-dependent task, it is obtained using training data from just one writer.

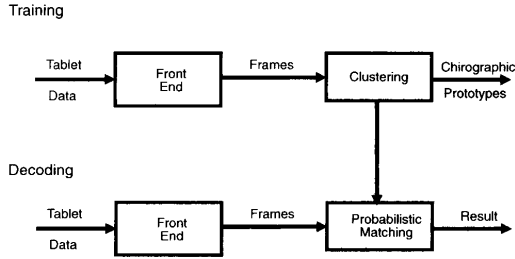
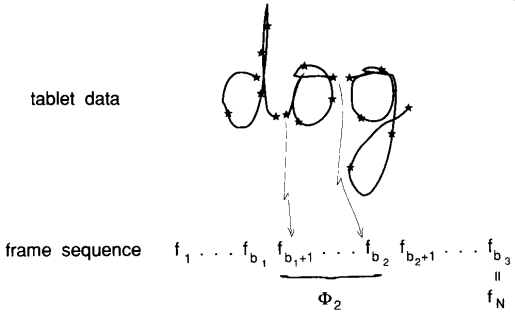


Fig. 2. System overview.

Fig. 3. Example of frames, boundary points, and block of frames for $M = 3$.

$\vec{C}$ be obtained as: $\vec{C} = c_1 \cdots c_M$, where each c_j is an individual character. In general, each such c_j can be associated with a block of frames delimited by two boundary points. Namely, there exist $M+1$ boundary points $\{b_j\}$, $0 \leq j \leq M$, such that: 1) $b_0 = 0$ and $b_M = N$; and 2) each frame f_n such that $b_{j-1} + 1 \leq n \leq b_j$ is associated with c_j . To fix ideas, an example is presented in Fig. 3 with $M = 3$, assuming non-overlapping frames.

Note that in the case of discretely written data, the boundary points $\{b_j\}$ are uniquely determined, since each character is written in isolation. In this case the handwriting channel parameters can be simply expressed as:

$$\Pr(\vec{S} | \vec{C}) = \prod_{j=1}^M \Pr(\Phi_j | c_j) \quad (3)$$

where $\Phi_j = f_{b_{j-1}+1} \cdots f_{b_j}$ represents a block of frames associated with the character c_j . In the case of connected or unconstrained handwriting, however, the boundary points $\{b_j\}$ are, in general, arbitrary. In that case the handwriting channel parameters can be expressed as:

$$\Pr(\vec{S} | \vec{C}) = \sum_{\{b_j\}} \prod_{j=1}^M \Pr(\Phi_j | c_j) \quad (4)$$

where Φ_j assumes the same definition as before for a given set of boundary points. The summation in (4) is taken over all possible segmentations of the frame sequence into M segments for each hypothesis considered.

The issue primarily addressed in this paper is how to compute the term $\Pr(\Phi_j | c_j)$. The strategy is to establish a hidden Markov model (HMM), also referred to as a *baseform*, for each character c_j . Let $\mathcal{B}_j$ denote the baseform representing c_j ; in general, $\mathcal{B}_j$ is determined by a set of states $\{s_n\}$, a set of transition probabilities from one state to the other, and a set of output probabilities governing the production

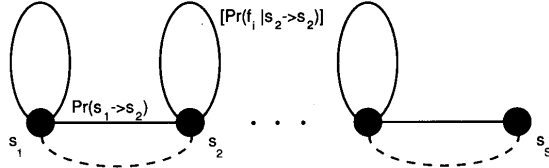


Fig. 4. Typical left-to-right hidden Markov model topology.

of output symbols. In such a set up, the computation of $\Pr(\Phi_j | c_j)$ amounts to computing the probability of the block Φ_j given a model for character c_j , i.e., $\Pr_{\mathcal{B}_j}(\Phi_j)$.

This quantity can be computed by introducing the concept of a *path* through the model, i.e., a particular traversal of the states of the baseform starting at the initial state and ending at the final state. Thus

$$\Pr(\Phi_j) = \sum_{\{\text{paths}\}} \Pr(\Phi_j | \text{path}) \Pr(\text{path}) \quad (5)$$

where $\Pr(\text{path})$ refers to the likelihood of a particular traversal and $\Pr(\Phi_j | \text{path})$ is the overall output probability associated with the block Φ_j given this traversal. The choice of a particular topology for the baseform $\mathcal{B}_j$ will clearly influence the cost-performance behavior of the system. For example, a left-to-right model such as depicted in Fig. 4 will provide detailed modeling of each character at the expense of the computational complexity of going through many states. In this figure, each arc represents a transition from one state to another. The direct path through the model corresponds to the average instantiation of the character. Self loops allow for lengthening of the realization, while state skipping (dotted lines) is permitted to account for shortening of the realization.

To make the computation of (5) efficient, we will place the following assumptions on the model. The transition probability of moving to state s_n given that one is in state s_{n-1} is assumed to follow a first-order Markov process, i.e., $\Pr(s_n | \mathcal{B}_j, \{s_k\}_{k=1}^{n-1}) = \Pr(s_n | s_{n-1})$. The (output) probability to emit frame f_i as one moves from state s_{n-1} to state s_n is assumed to follow a zeroth-order Markov process, i.e., $\Pr(f_i | s_n, s_{n-1}, \mathcal{B}_j, \{f_k\}_{k=1}^{i-1}) = \Pr(f_i | s_n, s_{n-1})$. With these assumptions, $\Pr(\text{path})$ and $\Pr(\Phi_j | \text{path})$ in (5) can be simply expressed as products of terms over the states of $\mathcal{B}_j$. Clearly, these assumptions will only be reasonable if the frames are not heavily correlated with each other. The issue is how to determine each frame to ensure that this condition is met. This is the object of the next section.

III. CHIROGRAPHIC REPRESENTATION

The sequence of frames in Fig. 3 is determined by the front-end of the system and constitutes what we refer to as the chirographic representation of handwriting. The idea is to isolate the most informative constituents of handwriting in the form of individual frames. Although the concept of frames departs from conventional character- and stroke-based templates, it provides a similar kind of parametrization at a much finer level of detail. The complete front-end processing comprises four steps to be described in detail below: 1) data preprocessing, 2) parameter vector selection, 3) feature event extraction, and 4) elimination of redundancy in the data.

A. Preprocessing

We have found that, for the purpose of recognition, resampling the pen trajectory so as to obtain equispaced points is better than keeping the time-dependent representation captured by the tablet (where the spacing is ballistic in nature). This is especially true when building

a writer-independent recognition system, since different writers write with different speed characteristics. In addition, we use a wild point correction algorithm to correct for occasional spurious points, as well as a dehooking algorithm to eliminate extraneous data at pen touch downs and lift ups. This preprocessing is described in detail in [5].

B. Parameter Vectors

At each (equispaced) point on the pen trajectory, a number of measurements are gathered to form a parameter vector that offers a localized view of the handwriting phenomenon. Let P_n be defined by the horizontal coordinate x_n and the vertical coordinate y_n at this point. Due to the poor reproducibility of these measurements, we select instead: 1) the incremental changes in the above quantities from one point to the next ($\Delta x_n = x_n - x_{n-1}$ and $\Delta y_n = y_n - y_{n-1}$); 2) the sine and cosine of the angle of the tangent to the pen trajectory at P_n ; and 3) the sine and cosine of the differential in angle at P_n [6]. These parameters provide both 1st order information (slope) and 2nd order information (curvature) about the pen trajectory at point P_n . Furthermore, these measurements are equally adequate to describe a point in a discretely written character or in a cursively written word.

Additional measurements may be possible depending on the mode of handwriting considered. For example, if the writer is asked to write on lines displayed on the tablet, the vertical distance to the baseline can be used. In the case of *discrete* handwriting, it is also of interest to gather the horizontal displacement from the first point in the trace, and the distance between pen up and pen down whenever it occurs [6]. Note that these additional parameters are more global in nature than those previously mentioned.

All the above measurements need not be grouped into a single parameter vector. In the example above, for instance, the slope and curvature information could be placed into separate, lower dimensional parameter vectors. Assume that parameters are placed into L parameter vectors, each set of which defines a distinct parameter space. If $L > 1$, recognition is said to use multiple information streams. The advantages are twofold: 1) the spaces are more homogeneous, and 2) they can be processed in parallel. In our experiments, we used $L = 2$, with the corresponding parameter spaces related to local and global parameters, respectively (cf. Section VI). For the sake of clarity, however, the following assumes that there is only one P -dimensional parameter space. Extension to multiple streams is straightforward and recognition using multiple streams is discussed in Section V.

C. Windowing

The next step is to extract feature events, i.e., consistent trends in the handwriting that can be observed over several points. To get this less localized perspective, we concatenate (at predetermined intervals along the handwriting data) the parameter vectors corresponding to a fixed number of points. The points in question are said to fall within a splicing window. Information relative to the handwriting contained within this window constitutes a frame. The number of points contributing to a frame (or, equivalently, the size of the window) is important: too small a window will generate noisy and/or uninformative frames, and too large a window will hamper generalization.

For each training token in the training data, we determine a subset of the points P_n in that token, say Q_i , with the property that the Q_i are also approximately equally spaced. This set typically includes the first and the last points of the trace, and the spacing interval is a reasonable fraction of the line height. At each point Q_i , we construct a Q -D spliced vector $\vec{q}_i$ by concatenating together the H parameter vectors $\vec{p}_{k-H}$ preceding Q_i , the parameter vector $\vec{p}_k$, corresponding

to Q_i , and the H parameter vectors $\vec{p}_{k+H}$ following Q_i . This is realizable provided $Q = P(2H + 1)$. At the beginning and end of the trace, the parameter vectors are repeated to complete the window. This can be loosely interpreted as keeping the pen stationary at the endpoints for as long as necessary.

D. Principal Components Analysis

While windowing transforms the 2-D sequence of coordinates (x_n, y_n) into a Q -D ($Q \gg 2$) sequence of spliced vectors representative of the scribble, it also introduces some redundancy in the spliced data due to window overlapping and inter-point correlation. To extract features most useful to the recognition process, we now have to reduce this redundancy. We first compute the mean vector $\vec{\mu}$ and the covariance matrix S of all the Q -D spliced vectors. We then perform a singular value decomposition (SVD) of S , as $S = E\Lambda E^T$, where E is the eigenvector matrix of S , Λ is the diagonal matrix of associated eigenvalues, and ^{T} denotes matrix transpose. By definition, the leading eigenvectors in E correspond to the leading eigenvalues in Λ .

Thus, using the R leading eigenvectors, we can project the Q -D spliced vector onto a space of dimension R . As a result, if $E = (E_R \ E_{Q-R})$, where E_R comprises the first R columns of E and E_{Q-R} the $Q - R$ remaining columns, the transformation matrix is E_R^T and the resulting projected feature vectors $\vec{r}_i$ are obtained as: $\vec{r}_i = E_R^T \vec{q}_i$. At this point the redundancy present in the Q -D spliced vectors has been eliminated by concentrating on the R most informative feature elements. By definition the remaining information constitutes a frame; thus $f_i \equiv \vec{r}_i$. The space spanned by the vectors $\vec{r}_i$ is called a *chirographic space* and will be denoted by $\mathcal{C}$.

IV. MIXTURE MODELING

Once the chirographic space is determined through the above front-end, the prototypes that will be used during the recognition phase, are established through clustering. In this framework, a prototype is simply a region of this space. Partitioning the space into a comparatively small number of such *chirographic prototypes* leads to an alphabet representative of the elementary building blocks used in the writing process. It is clear, however, that since a frame represents only a small portion of a character, in general, several prototypes might contribute to each character. This leads to the concept of mixture model described below. Note that in speech recognition, it has been shown that a mixture of Gaussian distributions typically outperforms single Gaussian distributions [8].

A. Definition

In the mixture model, the probability that a frame $\vec{r}_i$ is part of a character c_j is:

$$\Pr(\vec{r}_i | c_j) = \sum_{k=1}^K \Pr(\vec{r}_i | \pi_k) \Pr(\pi_k | c_j). \quad (6)$$

In (6), π_k represents one of K prototypes, $\Pr(\vec{r}_i | \pi_k)$ is the probability that the frame $\vec{r}_i$ falls within the region of the chirographic space represented by π_k , and $\Pr(\pi_k | c_j)$ are the mixture coefficients, i.e., the probability that each prototype π_k contributes to a given character c_j . For simplicity, $\Pr(\vec{r}_i | \pi_k)$ is modelled as a Gaussian probability density function.

One way to illustrate (6) is in terms of the degenerate hidden Markov model depicted in Fig. 5. In this topology, the initial state coincides with the final state [8] (all the s_i are identical), the mixture coefficients are viewed as the transition probabilities, and the Gaussian densities as the output probability distributions. Since

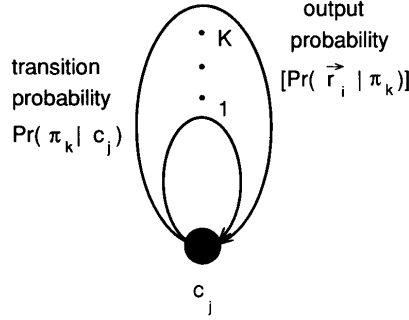


Fig. 5. Degenerate (single state) mixture hidden Markov model topology.

a single state mixture is used in the modeling, the graph must be traversed as many times as there are frames in the character, each loop corresponding to an observation $\vec{r}_i$. Clearly, the model of Fig. 5 cannot pretend to the same level of performance that can be achieved through the general model of Fig. 4, because of its lack of structure. On the other hand, its traversal is greatly simplified and can therefore be implemented extremely fast. Further details on the comparison between the two models can be found in [6].

Note that in (6), there is one set of mixture coefficients per character c_j in the alphabet (81 in our case). However, it would be a simple matter to change c_j to be an allograph as opposed to a character. Then we would have one set of mixture coefficients per allograph, presumably resulting in a better modeling.

B. Training

As a cluster in a chirographic space, a chirographic prototype is defined by a triplet $\pi_k = (\vec{\mu}_k, \Sigma_k, p_k)$ where $\vec{\mu}_k$ is the mean, Σ_k the covariance matrix, and $p_k = \Pr(\pi_k)$, the prior probability of the cluster. Each cluster is therefore represented by a Gaussian probability density function that characterizes the appropriate region of the space. Each triplet is obtained automatically from the training data by performing K -means clustering as described in [9]. The clustering is based on a 2 step-procedure. In the first step all the Σ_k are constrained to be the identity matrix, which amounts to performing Euclidean clustering. This allows one to efficiently create a somewhat coarse partition of the chirographic space. In the second step the K -means algorithm is applied again, this time without constraint, to refine the partition through Gaussian clustering. Finally, the cluster sizes are used to estimate the prior probability $\Pr(\pi_k)$ of each prototype distribution in the R -D chirographic space.

Appropriate mixture coefficients can be trained to each individual writer. The mixture coefficients $\Pr(\pi_k | c_j)$ are trained using an Expectation-Maximization procedure called the Baum-Welch, or forward-backward algorithm [10], [11]. For more details the reader is referred to [8].

V. RECOGNITION

A. Single Frame Recognition

The test data are subjected to the identical front-end: preprocessing, feature extraction, splicing, and projection onto a suitable space. For each frame of data $\vec{r}_i$, represented in the space $C^{(1)}$ by $\vec{r}_i^{(1)}$, and in general in the case of L information streams, in the space $C^{(L)}$ by $\vec{r}_i^{(L)}$, ($1 \leq l \leq L$), we use the Gaussian distributions $\pi_k^{(l)}$ and the

mixture coefficients obtained during training to form:

$$\Pr(\vec{r}_i | c_j) = \prod_{l=1}^L [\Pr(\vec{r}_i^{(l)} | c_j)]^{\alpha_l} \quad (7)$$

where

$$\Pr(\vec{r}_i^{(l)} | c_j) = \sum_{k=1}^{K_l} \Pr(\vec{r}_i^{(l)} | \pi_k^{(l)}) \Pr(\pi_k^{(l)} | c_j). \quad (8)$$

The coefficients α_l are introduced to take into account the possible dependence of the streams. They represent the relative contributions of the respective streams, and, in general, sum to 1. Ideally, each stream should be such that it contains fairly independent information from the others. As this is not always the case, the parameters α_l offer a way to adjust for stream interdependence. Taking this into account, (7) represents the weighted product of L Gaussian mixture distributions covering the entire chirographic spaces.

B. Search Issues

In the case of discrete handwriting, where characters are naturally segmented, it suffices to multiply the score of successive frames within each character to obtain the overall score for the associated block of frames under test. To complete the decoding process, it remains to pick the character with maximum score. In other words, the recognized character is simply:

$$c_j^* = \arg \max_{c_j} \prod_{\vec{r}_i} \Pr(\vec{r}_i | c_j). \quad (9)$$

As explained before, (9) assumes independence of the frames, but this assumption is mitigated by the effect of the projection operation performed to eliminate redundancy.

On the other hand, in (run-on), cursive, or more generally in unconstrained handwriting, a search has to be performed to find the most probable character-segmentation points, cf. (4). There are numerous alternatives suitable to carry out such a search. For example, it could be implemented using an A^* algorithm [12] or a stack search [13]. As a discussion of this topic is beyond the scope of this manuscript, the reader is referred to [7] for further details on search. Suffice it to say that for each hypothesis considered by the search, the score of successive frames is evaluated as in (9). The top hypothesis provides the character sequence that has maximum *a posteriori* probability.

VI. EXPERIMENTAL RESULTS

We considered a handwriting recognition task with a 81-character alphabet comprising upper and lower case letters, digits, and 19 special symbols (mathematical, punctuation). Such a large alphabet makes the task quite challenging due to numerous shape confusions (e.g., "quote" vs. "comma"). The data were collected on a transparent electronic tablet that had a resolution of 254 points/in. On the average, 60 coordinate pairs per character were captured by the tablet. To properly evaluate the performance of the (degenerate) mixture model just proposed, we conducted a comparison with a state-of-the-art algorithm of approximately the same computational complexity. For this comparison the baseline results were obtained using the elastic matching algorithm released in the 1992 IBM Thinkpad product.

Two series of experiments were conducted, addressing writer-dependent (WD) and writer-independent (WI) recognition, respectively. Eight writers participated in the WD study. For each writer, the training set consisted of about 1600 characters, and the test set of about 800 characters. For the WI study, the training data from the eight writers were pooled together as WI training data and an additional four writers (unrelated to the previous eight)

TABLE I
WRITER-DEPENDENT STUDY: CHARACTER ERROR RATES FOR
ELASTIC MATCHING AND STATISTICAL MIXTURE ALGORITHM

	Elastic Match	Stat. Mixture
MAR	13.7%	7.1%
HIL	21.0%	21.0%
VIV	14.2%	15.2%
JOA	14.5%	12.0%
ACY	23.7%	23.0%
NAN	16.1%	12.0%
LOY	12.1%	7.7%
SAB	13.8%	9.4%
Average	16.2%	13.4%
CPU index	1.0	1.5

provided data for testing. Again, for each, the test set consisted of about 800 characters. Since our primary goal was to evaluate the goodness of the handwriting channel modeling rather than get absolute recognition numbers, no language model was used for either elastic matching or the statistical mixture approach. Note that, as a result, inherently ambiguous characters such as "zero" and "oh" cannot be distinguished.

In the elastic matching approach, for the WD study, an average of 200 character templates were generated per writer. For the WI study, 3017 character templates were generated for use during decoding. In the statistical mixture approach, we considered 2 information streams, one for local features (incremental position, direction, curvature) and one for global features (distance from the baseline, horizontal displacement, and distance between pen ups and pen downs whenever relevant). For stream 1, the dimension of the parameter space was $P = 6$, while for stream 2 it was $P = 3$. On the average, there were 10 window centers per character. For stream 1, the spliced vector has dimension $Q = 246$, and projected onto a space of dimension $R = 6$. For stream 2, the spliced vector had dimension $Q' = 123$, and projected onto a space of dimension $R' = 15$. Note that $R' > R$ because there is generally less redundancy present in the global features. For the WD study, the total number of clusters K_i ($i = 1, 2$) in each chirographic space was set to 350. For the WI study, the total number of clusters in each chirographic space was set to 750. The streams were assumed to behave independently so we set $\alpha_1 = \alpha_2 = 0.5$.

The character error rates obtained in the WD study are reported in Table I for each of the writers, for both elastic matching (column 1) and the single state mixture HMM (column 2). Comparing column 1 and column 2, it can be seen that the new approach consistently outperforms elastic matching in recognition accuracy. The recognition speed was slightly slower than with elastic matching. The character error rates obtained in the WI study are reported in Table II for each of the four new writers (unrelated to the eight writers used in the training set). Once again, the single state mixture consistently outperforms elastic matching in recognition accuracy. The CPU index is measured relative to the WD elastic matching experiment. Note that in the WI case, the recognition speed was slightly *faster* than with elastic matching. This can be attributed to the fact that in this case elastic matching generates a much larger number of templates.

We can conclude that the statistical mixture approach compares favorably to elastic matching, in both speed and accuracy. If we

TABLE II
WRITER-DEPENDENT STUDY: CHARACTER ERROR RATES FOR
ELASTIC MATCHING AND STATISTICAL MIXTURE ALGORITHM

	Elastic Match	Stat. Mixture
MAL	27.4%	27.4%
BLA	25.0%	21.0%
SAM	27.0%	30.0%
WAR	18.7%	12.7%
Average	24.5%	19.1%
CPU index	5.0	3.5

eliminate confusions between ambiguous characters (on the assumption that the use of a language model would address this problem), trouble spots seem to be characters with dots (i , j , "colon", etc.) and other small strokes ("comma," "semicolon," etc.) We attribute this behavior to the splicing mechanism, which is not well suited to such small strokes. The reader is referred to [6] for further comments on these results.

VII. CONCLUSION

Looking at the handwriting process from an information theoretic viewpoint, we have presented a probabilistic framework that unifies several approaches to on-line handwriting recognition. Within this framework, we have derived a fast statistical mixture algorithm of about the same degree of complexity as elastic matching. The approach relies on a novel front-end processor that articulates around a small handwriting elementary unit called a frame. The algorithm is based on 1) producing parsimonious feature vectors in one (or several) chirographic spaces, 2) Gaussian K -means clustering in these spaces, and 3) mixture modeling taking into account the contributions of all relevant prototypes in each space. We have compared this strategy to elastic matching on a simple task involving an 81-character alphabet. The new approach was found to be competitive with elastic matching in both speed and accuracy.

One drawback of the above approach is that it does not provide any duration modeling. Since there is only one state per character (or allograph), there is no concept of representing the beginning of a character through one set of mixtures and the end through another set for example. This is a direct consequence of the degeneracy of the HMM representation chosen. This issue will be addressed in a subsequent paper, where we relax the degeneracy assumption at the expense of a greater computational complexity. In that respect, we would like to emphasize that the fast algorithm proposed here will prove useful to efficiently produce a short list of candidate characters, and thus reduce the amount of computation to be performed with such a non-degenerate HMM.

REFERENCES

- [1] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in on-line hand-writing recognition," *IEEE Trans. Patt. Anal. Mach. Intel.*, vol. 12, no. 8, pp. 787-808, 1990.
- [2] H. L. Teulings and L. R. Schomaker, "Unsupervised learning of prototype allographs in cursive-script recognition using invariant handwriting features," in *Proc. 2nd Int. Workshop on Frontiers in Handwriting Recognition*, Bonas, France, September 1991, pp. 45-55.
- [3] P. Morasso, J. Kennedy, E. Antonj, S. Di Marco, and M. Dordini, "Self-organization of an allograph lexicon," in *Proc. Int. Neural Network Conf.*, Paris, France, 1990. Dordrecht: Kluwer Academic Publ. 1990, pp. 141-144.

- [4] L. R. Bahl, F. Jelinek and R. L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Trans. Patt. Anal. Mach. Intel.*, Vol. PAMI-5, no. 2, pp. 179–190, Mar. 1983.
- [5] C. C. Tappert, "Speed, Accuracy, and Flexibility Trade-Offs in On-Line Character Recognition," *Intl. Journal of Pattern Recognition and Artificial Intelligence*, Vol. 5, No. 1–2, pp. 79–95, 1991.
- [6] E. J. Bellegarda, J. R. Bellegarda, D. Nahamoo and K. S. Nathan, "A Probabilistic Framework for On-Line Handwriting Recognition," in *Proc. 3rd Intl. Workshop on Frontiers in Handwriting Recognition*, Buffalo, NY, pp. 225–234, May 1993.
- [7] T. Fujisaki, K. Nathan, W. Cho and H. Beigi, "On-Line Unconstrained Hand-writing Recognition by a Probabilistic Method," in *Proc. 3rd Intl. Workshop on Frontiers in Handwriting Recognition*, Buffalo, NY, pp. 235–241, May 1993.
- [8] J. R. Bellegarda and D. Nahamoo, "Tied Mixture Continuous Parameter Modeling for Speech Recognition," in *IEEE Trans. Acoust. Speech, Signal Processing*, Vol. 38, No. 12, pp. 2033–2045, Dec. 1990.
- [9] J. A. Hartigan, *Clustering Algorithms*, J. Wiley: Ed., 1975.
- [10] L. E. Baum, T. Petrie, G. Soules and N. Weiss, "A maximisation technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.
- [11] L. E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, vol. 3, pp. 1–8, 1972.
- [12] N. J. Nilsson, *Problem Solving Methods in Artificial Intelligence*. New York: McGraw-Hill, 1971.
- [13] F. Jelinek, "A fast sequential decoding algorithm using a stack," in *IBM J. Res. Dev.*, vol. 13, pp. 675–685, 1969.

A Metric between Unrooted and Unordered Trees and its Bottom-Up Computing Method

Eiichi Tanaka

Abstract—This correspondence proposes a distance between unrooted and unordered trees based on the strongly structure preserving mapping (SSPM). SSPM can make correspondences between the vertices of similar substructures of given structures more strictly than the already proposed mappings. The time complexity of computing the distance between trees T_a and T_b is $O_T(m_b^3 Na Nb)$, where $Na(Nb)$ and $m_a(m_b)$ are the number of vertices in tree $T_a(T_b)$ and the maximum degree of a vertex in $T_a(T_b)$, respectively, and $m_a \leq m_b$ is assumed. The space complexity of the method is $O_S(Na Nb)$.

Index Terms—Distance, dynamic programming, pattern matching, pattern recognition, similar structure search, similarity, tree.

I. INTRODUCTION

Several tree metrics have already been proposed, but most of them are metrics on rooted and ordered trees. Some of them [1]–[5] are based on the edit operations of the substitution, the insertion and the deletion of a vertex and other [6] is based on the edit operations of the splitting and the merging of vertices. Recently, it was reported that the computation of the Tai distance [2] for rooted and unordered trees is NP-hard [7]. Many papers on metrics between graphs were published [8]–[10]. However, since a graph is far more complicated

than a tree, a method that can solve the graph matching problem by dynamic programming has not yet been proposed. A tree structure is found in many engineering problems. For example, the study of the relationship between the structures of chemical compounds and their properties is one of the most important problems in chemistry [11], [12]. A structural formula is nothing but a labeled graph. The graph isomorphism and the subgraph isomorphism problems and the largest common subgraph problem have been discussed [13]–[15]. To extend the concepts of isomorphism, substructure isomorphism and largest common substructure to the concepts of inexact isomorphism, inexact substructure isomorphism and largest similar substructure, respectively, we must define a metric or a similarity on unrooted and unordered trees before we define these measures on graphs.

II. MAPPINGS AND DISTANCES BETWEEN UNROOTED TREES

A rooted tree is a tree with a root, and an ordered tree is a tree such that for each vertex, its adjacent vertices are ordered. We will consider an unrooted and unordered tree (*tree*) and a rooted and unordered tree (*R-tree*). A different number is assigned to each vertex of a *tree*. Let $T = (V, E)$ be a *tree*, where V and E are the sets of vertices and edges, respectively. The degree of a vertex is the number of vertices adjacent to it. Let $lab(u)$ and $deg(u)$ be the label and the degree of vertex u , respectively. Let $L = \{u \mid u \in V, deg(u) = 1\}$. An element of L is called a leaf. $T(u)$ denotes an *R-tree* with root u . $P(x, y)$ denotes the path from vertex x to vertex y . If vertex x can be reached through m edges from the root, the depth of x is m and is denoted by $dep(x) = m$. The depth of the root is 0. Consider two vertices x and y on path $P(u, \tilde{u})$ of $T(u)$, where $\tilde{u}$ is a leaf. If $dep(x) < dep(y)$, x is a proper ancestor of y and y is a proper descendant of x . $An(x)$ and $De(x)$ denote the sets of proper ancestors of x and proper descendants of x , respectively. If $dep(y) = dep(x) + 1$, y is a child of x . $Ch(x)$ denotes the set of children of x . A part of $T(u)$ that consists of x and $De(x)$ is called a subtree of $T(u)$ and is denoted by $T(u, x) = (V(u, x), E(u, x))$. x is the root of $T(u, x)$. Another notation of a subtree is used. Assume that edge ξ is incident with vertex x . $T^\xi(x) = (V^\xi(x), E^\xi(x))$ denotes the subtree with root x that is obtained by deleting ξ from *tree* T . Let $|A|$ be the number of elements in set A . If the root x of a subtree is a leaf and $|V| = |V^\xi(x)|$, we write $T^\epsilon(u)$, where ϵ indicates the null edge. If x is a child of u , $T(u, x)$ is called a child subtree of u . Let $Ta(u) = (Va(u), Ea(u))$ and $Tb(v) = (Vb(v), Eb(v))$ be two *R-trees*. Consider a mapping from the vertices of $Ta(u)$ to those of $Tb(v)$. If the vertices of $Ta(u, x)$ map to those of $Tb(v, y)$, we call $Ta(u, x)$ maps to $Tb(v, y)$, $Tb(v, y)$ is the image of $Ta(u, x)$ and $Ta(u, x)$ is the inverse image of $Tb(v, y)$. The fact that $Ta(u, x)$ maps to $Tb(v, y)$ does not always mean that x maps to y . A tuple (i, j) indicates that $i \in Va(u)$ maps to $j \in Vb(v)$. Let Mr be the set of (i, j) 's. Let $Ir = \{i \mid (i, j) \in Mr\}$, $Jr = \{j \mid (i, j) \in Mr\}$ and $Ir(x) = Ir \cap Va(u, x)$. Let $Tb(v, r_x)$ be the smallest subtree that covers the set of images of vertices in $Ir(x)$ and does not include other images. Note that r_x cannot always be determined.

Definition 1: Mr is called a structure preserving mapping (SPM) from $Ta(u)$ to $Tb(v)$, if Mr satisfies the following conditions a1) and a2).

- a1) For any (i_1, j_1) and (i_2, j_2) in Mr , i) $i_1 = i_2$ iff $j_1 = j_2$, ii) $i_1 \in An(i_2)$ iff $j_1 \in An(j_2)$.
- a2) If r_{x_1} and r_{x_2} can be determined for x_1 and x_2 , respectively, then $x_1 \notin An(x_2)$ and $x_2 \notin An(x_1)$ iff $r_{x_1} \notin An(r_{x_2})$ and $r_{x_2} \notin An(r_{x_1})$, where $x_1, x_2 \in Va(u)$ and $r_{x_1}, r_{x_2} \in Vb(v)$.

Manuscript received January 22, 1993; revised December 6, 1993. Recommended for acceptance by Editor-in-Chief A. K. Jain.

The author is with the Department of Electrical and Electronics Engineering, Faculty of Engineering, Kobe University, Rokkodai, Nada, Kobe, 657 Japan. IEEE Log Number 9406166.