

## STROKE LEVEL MODELING OF ON LINE HANDWRITING THROUGH MULTI-MODAL SEGMENTAL MODELS

T. ARTIÈRES , J-M. MARCHAND, P. GALLINARI  
*LIP6*  
*E-mail: [Thierry.Artieres@lip6.fr](mailto:Thierry.Artieres@lip6.fr), [Patrick.Gallinari@lip6.fr](mailto:Patrick.Gallinari@lip6.fr)*

B. DORIZZI  
*INT*  
*E-mail: [dorizzi@int-evry.fr](mailto:dorizzi@int-evry.fr)*

Hidden Markov Models (HMMs) have become within a few years the main technology for on line handwritten word recognition (HWR). We consider here segment models which generalize HMMs, these models aim at modeling the signal at a global level rather than at the frame level and have been shown to overcome standard HMMs in their modeling ability. We propose a new segment model which allows to automatically handle different writing styles. We compare our system on the isolated character set of the UNIPEN database to a reference system and a baseline segment model.

### 1 Introduction

Building an interface where anyone can enter cursive words and have them recognized on-line automatically is a challenging problem today. Indeed, the recent development of electronic notepads, the appearance of new types of devices like tablets and pens and the emergence of electronic ink software, increase the need for this type of system.

The design of such a recognizer relies on two related steps: elementary entities (points, strokes, portions of the drawing) are first coded in feature vectors which are further used as inputs to the recognition system.

If recognition is analytical (the word is recognized through the identification of the letters which compose it), one convenient and efficient approach is to perform segmentation of the words into letters and recognition at the same time. A complex segmentation algorithm is not necessary in this case, contrary to what happens with procedures which use a priori segmentation. Moreover statistical methods can be used for the design of this "segmentation-recognition" step, like Hidden Markov Models (HMM) [2] or hybrid neural and markovian models [4,9,11].

When these systems act at the point level, they are generally composed of more states and they present a richer topology than those relying on the stroke level [2]. The ability of the markovian approach to efficiently deal with the variability inherent to the omni-scriptor framework is thus enhanced. However, using such a fine level of description has its drawbacks: it is not easy to take into account the contextual information in the trajectory nor to give an interpretation of the different states in terms of allographs or portions of allographs.

Segmental models (SM) provide an alternative which allows to keep the HMM formalism while dealing with segments composed of several points. These models have first been introduced in speech recognition (see [3,10] for a review).

In this approach, a segment is not defined "a priori". It will correspond to a portion of the signal which is homogeneous in some sense. In our case, we are interested in on line character modeling and a segment will typically correspond to a stroke in a character. Such a segment will be modeled as a whole and the trajectory of the corresponding signal will be considered as an explicit function of time.

We explore in this article, the potential of segmental models for on line HWR. Starting from a basic SM we then propose a new segmental model which is well suited for handling the variability of handwriting styles. Both models have been tested for the recognition of isolated characters on a large data set with a large number of writers and styles. This dataset is part of the Unipen Database [6]. We have tested our systems for multi and omni-writer recognition. For each task, SMs are compared to a reference system which has been shown to perform well for handwriting word recognition [4].

In section 2 we describe a general framework for sequence modeling and show how basic HMMs fit into this formalism. Segment models are introduced in section 3. We discuss the merits of different implementations and present a basic SM model. An extension of this system, designed for handling different writing styles, is described in section 4. Finally sections 5 and 6 present experimental results.

## 2 Sequence modeling

We focus here on isolated character recognition which is an intermediate complexity task but the extension of the following to word recognition is straight-forward.

Let us denote  $o_1^T = (o_1, \dots, o_T)$  a *sequence* of observations (or frames) where  $o_t$  is a P-dimensional features vector (see §5), and  $\theta$  an M-states character model whose states are  $(s_1, \dots, s_M)$ . For simplicity, we will consider only left to right Bakis models

without skips. For such models, the segmentation of  $o_1^T$  into states is completely defined by the sequence of states,  $s_1^M$  and their respective duration  $d_1^M$  :

$$p(o_1^T / \theta) = p(o_1^T / s_1^M) = \sum_{d_1^M} p(o_1^T / d_1^M, s_1^M) p(d_1^M / s_1^M) \quad (1)$$

Let us now assume that the observations between different states are independent and that the duration model for one state depends only on this state, then:

$$p(o_1^T / d_1^M, s_1^M) = \prod_{i=1}^M p(o_{i+1}^{i-1} / d_i, s_i) \quad (2)$$

$$p(d_1^M / s_1^M) = \prod_{i=1}^M p(d_i / s_i) \quad (3)$$

where  $t_1, t_2, \dots, t_M$  stand for the transition times between two successive states, and  $d_i = t_{i+1} - t_i$ . We use by convention  $t_1 = 1$  and  $t_{M+1} = T + 1$ .

Equation (2) represents the emission probability of a sequence of observations given the segmentation into states while (3) represents the probability of the segmentation. We will now show that this formulation encompasses classical HMMs. First, using the independence assumption between observations, (2) writes:

$$p(o_{t_i}^{t_{i+1}-1} / d_i, s_i) = \prod_{t=t_i}^{t_{i+1}-1} p(o_t / s_i) \quad (4)$$

Second, for first order Markov Chains (3) reduces to:

$$p(d_i / s_i) = [p(s_i / s_i)]^{d_i-1} \cdot (1 - p(s_i / s_i)) \quad (5)$$

where  $p(s_i / s_i)$  is the transition probability from state  $s_i$  to itself. This duration probability is the geometric distribution.

An alternative to standard HMMs consists in using auto-regressive (AR) predictive models for modeling emission probabilities. This allows to slightly relax the independence assumption. Considering an order  $q$  AR model, (4) becomes:

$$p(o_{t_i}^{t_{i+1}-1} / d_i, s_i) = \prod_{t=t_i}^{t_{i+1}-1} p(o_t / o_{t-q}^{t-1}, s_i) \quad (6)$$

Our baseline system is based on (5) and (6)<sup>1</sup>, where the probabilities  $p(o_t / o_{t-q}^{t-1}, s_i)$  are implemented with predictive Multi-Layer Perceptrons (MLP) [4], i.e. with non linear regressors.

### 3 Segment models

#### 3.1 General framework

Segment models attempt to model the signal at a segment level rather than at the observation level. An observation sequence is assumed to be generated by a succession of SM "states" or segments, each being responsible for a subsequence. Each segment generates a random length sub-sequence of observations:

$$p(o_{t_i}^{t_{i+1}-1}, d_i / s_i) = p(o_{t_i}^{t_{i+1}-1} / d_i, s_i) p(d_i / s_i) \quad (7)$$

A segment is thus defined by the segment label and its duration, i.e. instead of a state label  $s$  in a HMM, we have now a couple of random variables  $(s, d)$ . Different forms have been proposed for the two terms in (7). We have approximated duration probabilities via histograms. However, their contribution is small relative to the observation probabilities. Their main utility is to provide bounds on the allowed

<sup>1</sup> An on-line demo is available at: <http://www-poleia.lip6.fr/CONNEX/HWR/>

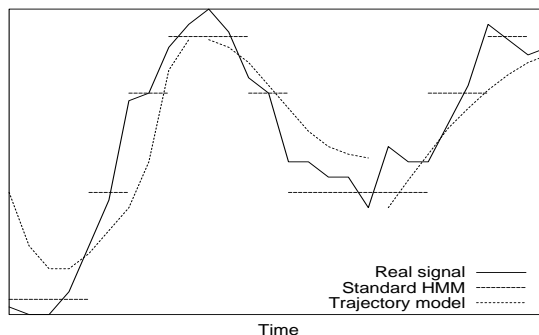
duration for each state. These bounds are then used during decoding for reducing the search space.

The emission probability for an observation will depend on the segment, the segment duration and the entering time in the segment, i.e. on its relative position in the sequence. We will introduce this dependency explicitly by denoting the conditional frame emission probability by:  $p(o_t / t, d_i, t_i, s_i)$ . We will then assume that observations are conditionally independent given  $t, d_i, t_i, s_i$ , so that:

$$p(o_i^{t_{i+1}-1} / d_i, s_i) = \prod_{t=t_i}^{t_{i+1}-1} p(o_t / t, d_i, t_i, s_i) \quad (8)$$

$p(o_i^{t_{i+1}-1} / d_i, s_i)$  represents a  $d_i$  length trajectory in the observation space. In SMs, each  $p(o_t / t, d_i, t_i, s_i)$  is implemented as a continuous and smooth function of time and thus models the correlation between successive observations.

The motivation for using segment models is illustrated in figure 1, where we have plotted two curves corresponding to the modeling of a real signal (the third curve) with a standard gaussian HMM model and a basic trajectory model (see §3.4). One can see that using a classical gaussian HMM assumes that the signal is piece wise constant whereas using a trajectory model assumes a global shape of the sequence which may correspond much better to the real signal. In the figure the classical HMM is assumed to have six states whereas the trajectory model has three states.



**Figure 1:** Comparison of the modeling of a signal with a local model (standard gaussian HMM) and a segment model (trajectory model of section §3.4 corresponding to (12)). The curves represent the assumed time-varying mean of the process (piecewise constant for the standard HMM).

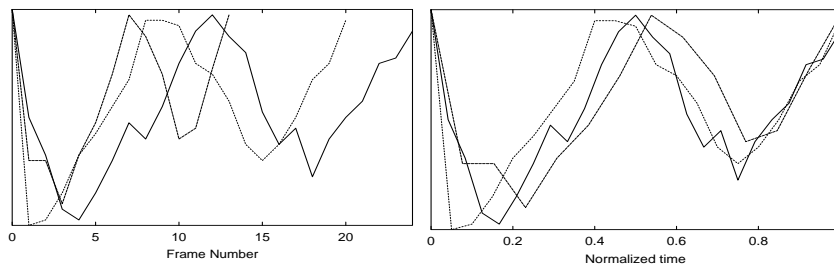
### 3.2 Time Normalization

We will make two changes to the formulation (8). First, absolute time has no meaning and will be replaced by the relative time spent in the state,  $(t - t_i)$ . Second, since we deal with varying-length sequences, a time warping transformation is required for

mapping observations to a segment  $s_i$ . We have used here a linear warping by normalizing time in a state between 0 (entering time) and 1 (exit time) as proposed for example in [7]. Thus, the effective time input to our models is a *normalized* time  $\bar{t}$  given by  $\bar{t}_{t_i, d_i} = \frac{t-t_i}{d_i}$ . In the following, we will denote the normalized time  $\bar{t}_{t_i, d_i}$  by  $\bar{t}$  when there is no ambiguity. With this notation:

$$p(o_t / t, d_i, t_i, s_i) = p\left(o_t / \frac{t-t_i}{d_i}, s_i\right) = p(o_t / \bar{t}, s_i) \quad (9)$$

Figures 2 below illustrate this warping strategy. It shows the evolution of a component of the frames as a function of time, for three different writings of the letter ‘‘a’’. One can see that these writings are very similar in their shape but differ by their lengths when using absolute time. Using a time normalized between 0 and 1 results in curves which are much similar.



**Figures 2:** Curves of the  $\Delta x$  feature corresponding to three writings of an isolated letter ‘‘a’’. In the left figure, absolute time is used whereas normalized time is used in the right figure.

### 3.3 Training and Recognition

During the recognition, we want to find the likelihood of a sequence of observations given a character model (eq. (1)). This computation is expensive for SMs since dynamic programming takes place in a 3-dimensional space (time, state, duration) instead of a 2-dimensional one for classical HMMs. Viterbi-like algorithms are then generally used and the summation in (1) is replaced by the maximum operator:

$$p(o_1^T / \theta) \approx \max_{d_1^M} \left( p(o_1^T / d_1^M, s_1^M) p(d_1^M / s_1^M) \right) \quad (10)$$

The optimal segmentation is simultaneously computed by:

$$\hat{d}_1^M = \arg \max_{d_1^M} \left( p(o_1^T / d_1^M, s_1^M) p(d_1^M / s_1^M) \right) \quad (11)$$

For training SMs, algorithms alternating segmentation and sequence likelihood increase are used. These algorithms are extensions of Baum-Welch or segmental k-means algorithms. We will use here the last one: models are re-estimated so as to maximize the likelihood of the observations along the optimal path.

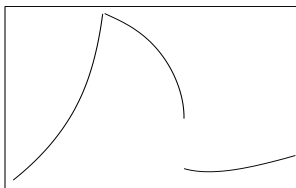
### 3.4 Trajectory model

We now describe a trajectory model that explicitly models the observations as time dependent functions conditionally on  $d_i, t_i, s_i$ . This set of models will then be used to estimate the  $\{p(o_i^{t_i+1}/d_i, s_i); d_i \in \mathbf{D}, s_i \in \mathbf{S}\}$  family where  $\mathbf{D}$  and  $\mathbf{S}$  are the set of allowable duration and segment labels. In a trajectory model, the frames of a subsequence  $o_i^{t_i+1}$  produced in a state  $i$ , are assumed to follow:

$$\begin{cases} o_t = \mu_i(\bar{t}) + \varepsilon_i(\bar{t}) \\ p(o_t / t, d_i, t_i, s_i, \theta) = p_{\varepsilon_i(\bar{t})}(o_t - \mu_i(\bar{t})) \end{cases} \quad \bar{t} \in [0..1] \quad (12)$$

where  $\mu_i(\bar{t})$  is a time dependent mean and  $\varepsilon_i(\bar{t})$  is assumed to be an independently distributed residual (generally a white noise) which may also depend on time. The idea is to approximate  $\mu_i(\bar{t})$  by a continuous function of time with smooth variations, which allows to take into account a certain kind of dependence between frames. Let us denote  $F_i(\bar{t}) = \hat{\mu}_i(\bar{t})$  this approximation.

Different trajectory models have been proposed which differ by the class of functions  $F_i(\bar{t})$  is assumed to belong to. For example, [7] uses a linear function of time and [1,5] a polynomial function. As an example, Figure 3 shows a letter ‘‘r’’ generated with a model (12) implemented with a MLP whose architecture is shown in Figure 4.



**Figure 3:** an example of a character ‘‘r’’ generated by a 3-state letter model obtained after training. One can see the three strokes generated by the three successive states, implemented with trajectory models (12).

Some authors [5] have also extended (12) by considering both relative time and past observations for inputs, leading to:

$$o_t = \mu_i(\bar{t}, o_{t-1}) + \varepsilon_i(\bar{t}) \quad \bar{t} \in [0..1] \quad (13)$$

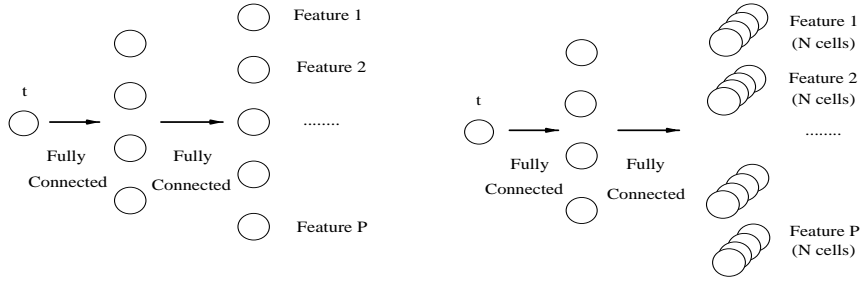
This model may offer better performance at a price of an increased complexity. We have implemented models corresponding to (12) and (13) where  $F_i(\bar{t})$  is implemented by a Multi-Layer Perceptron (MLP). Figure 4 (left) shows the architecture of a 3-state letter-model with an associated trajectory model of the form (12) in each state. These models are flexible enough to learn a wide variety of trajectories, while the control of their complexity (through the number of hidden cells for example) allows to choose the smoothness of the trajectory model.

In the decoding step, assuming a white noise residual in (12), the logarithm of the probability of a frame is approximated by:  $-\log[\hat{p}(o_t / \bar{t}, s_i)] = \|F_i(\bar{t}) - o_t\|^2$

where  $F_i$  is the function implemented by the MLP.

For training, we use a segmental k-means algorithm which alternates segmentation and sequence likelihood increase: for observation  $o_t$  assigned to state  $s_i$  after segmentation, a gradient step is performed on  $\|F_i(\bar{t}) - o_t\|^2$ .

In our experiments, we have found that model (13) performs significantly better than model (12), so that we provide results only for model (13) in the following.



**Figure 4:** MLP architecture of the two segment models used in this paper. The basic trajectory model (12) (on the left) and a new multi-modal model (on the right).

#### 4 A multi-modal model

The model of section 3.4 assumes that the observations follow a unimodal distribution around the time-varying mean. Such a model is not well suited for handling the different writing styles present in multi and omni-writer applications. One solution could be to extend the model by adding multiple branches for each letter or by considering a mixture of trajectory models within each segment. The complexity of such models in the SM context clearly forbids large applications. We therefore developed a more economic solution.

The idea is to use a segment model for each letter segment as in section 3, but instead of learning a time-dependent mean-value, we learn a time-dependent probability distribution. This allows to learn automatically multiple trajectories for a given character segment. In order to implement this idea we made two more assumptions.

First, the features are considered to be conditionally independent so that a *one dimensional distribution* may be learned for each feature:

$$p(o_t / \bar{t}, s_i) = \prod_{j=1}^P p(o_t^j / \bar{t}, s_i) \quad (14)$$

A second assumption is that this one dimensional continuous distribution will be approximated by a *discrete distribution*. For that, the range of values for each feature is divided into  $N$  intervals of equal width  $I_1, \dots, I_N$  (all the features have a limited range). The probabilities in (14) are then approximated through:

$$p(o_i^j / \bar{t}, s_i) = p(I_{k(o_i^j)} / \bar{t}, s_i) \quad (15)$$

where  $k(o_i^j)$  denotes the interval  $I$  corresponding to the value of  $o_i^j$ . Using these assumptions, (14) is computed with:

$$p(o_i / \bar{t}, s_i) = \prod_{j=1}^P p(I_{k(o_i^j)} / \bar{t}, s_i)$$

As for the model of section 3.4, we implemented the multimodal trajectory model using a MLP with one input, the normalized time  $\bar{t}$ . The difference is that we have now  $P*N$  outputs corresponding to the  $N$  intervals  $I_1, \dots, I_N$  for each of the  $P$  features. This implementation is illustrated in figure 4 (on the right). The MLP for state  $s_i$  is trained to output approximations of the probabilities  $p(o_i^j \in I_k / \bar{t}, s_i)$  for each  $j=1..P, k=1..N$ . We note  $F_{i,j,k}(\bar{t})$  this trajectory function.

Training proceeds as in section 3 but the MLP is trained in classification mode rather than in prediction mode as it is the case for the model of section 3.4. This is done as follows. After decoding, each observation is assigned to a particular state at a particular normalized time. For frame  $o_t$  assigned to state  $s_i$  at normalized time  $\bar{t}$ , desired outputs are defined for each of the  $P*N$  output cells of the MLP corresponding to state  $s_i$ . Among the  $N$  output cells corresponding to feature  $j$ , the one corresponding to interval  $k(o_i^j)$  is assigned the desired value "1", and the remaining  $(N-1)$  the desired value "0". Training is then performed as before by stochastic E.M. The local error function for observation  $o_t$  is the quadratic error between the desired output and the output computed by the model for this observation. It is known that after training, the MLP approximates  $p(o_i^j \in I_k / \bar{t})_{j=1..P, k=1..N}$ .

Recognition proceeds as in §3.1. where the probability (7) is approximated by:

$$\hat{p}(o_i / \bar{t}, s_i) = \prod_{j=1}^P F_{i,j,k(o_i^j)}(\bar{t})$$

This model does not assume any prior on the trajectory distributions and allows to handle automatically multi-modal densities when needed.

## 5 Experiments

We performed experiments on the UNIPEN database [6] which has been developed for benchmarking HWR systems. We used directory  $Ic$  which contains about 60000 isolated lowercase characters, some of these characters are shown in Figure 5.



**Figure 5:** Some characters (a, b, c, d, m, n, r, u, v, w), normalized in size, of the Unipen database.

We used 30000 characters for training. They have been written by 265 writers from 12 different contributing organisms. For testing, we used 10000 characters from the same 265 writers in multi-writer mode and 10000 characters from 34 other writers from 8 other organisms in omni-writer mode.

The input signal, a sequence of coordinates (x,y), is first unslanted using base-line detection. The signal is then spatially resampled, which eliminates the speed information. Then a frame of 15 coefficients (i.e.  $P=15$ ) is computed for each sample point. These include 6 “temporal” coefficients ( $\Delta x$ ,  $y$ , sine and cosine of slope and curvature), and nine additional coefficients which represent grey-levels of a local bitmap centered on the sample point. This processing is detailed in [4].

## 6 Recognition results

Table 1 shows comparative results on the two test sets for the baseline model and the two segment models. One must notice that that the proportions of the 26 letters in the data sets are very different so that the overall recognition rate may be biased. Thus, we provide results of the experiments with two different performance measurements. The first one is the overall recognition rate, the second one is the average recognition rate per letter.

**Table 1:** Recognition rate for the three models on the two test sets, corresponding to multi-writer and omni-writer mode (95% confidence interval are  $\pm 0,9\%$ ). Two performance criteria are used, the overall recognition rate (Overall RR) and the average recognition rate per letter (RR per letter).

Dataset	Criteria	Reference (§3.1)	Unimodal SM (§3.4)	MultiModal SM (§4)
Multi	Overall RR	79.5	74.0	77.9
Multi	RR per letter	74.6	69.0	76.3
Omni	Overall RR	67	60.9	69.8
Omni	RR per letter	64.8	59.4	68.8

In the baseline system, letter model has 7 states whereas in the segmental models letter models have 3 states. All the models show a drop of performance between the multi and omni-writer modes. The simple trajectory model performs worse than the baseline model and the multi-modal SM performs similarly to the baseline system for the multi-writer experiments and slightly better (3% or 4%) in omni-writer mode. Note however that both segment models give acceptable performance which shows that this approach may indeed offer an alternative to more classical models. These results are especially encouraging since our mature baseline model has good performances on HWR [4].

Slightly better results have been published on Unipen database (85% accuracy for isolated lower characters [8]). However, there is no standard benchmark and different parts of the database have been used so that results cannot be compared directly.

## 7 Conclusion

Trajectory models are a possible issue to improve recognition accuracy of on-line HWR systems. We have proposed a new segmental model which allows to handle multiple handwriting styles without any prior hypothesis.

Preliminary results are very promising since this model has been shown to perform similarly and sometimes better than a mature reference system.

## 8 References

1. Deng L., Aksamovic M., Sun X., Wu C.W., speech recognition using hidden Markov models with polynomial regression functions as nonstationary states, 1994, IEEE Trans. On Speech & Audio Proc., Vol. 2, n° 4.
2. El Yacoubi A., Gilloux M., Sabourin R., Suen C.Y., An HMM-Based approach for off-line unconstrained handwritten word modeling and recognition, IEEE-TPAMI, vol.21, 1999.
3. Gallinari P., 1998, Predictive models for sequence modelling, application to speech and character recognition, in Adaptive Processing of Sequences and Data Structures, Lee Giles C., Gori M. eds, Springer.
4. Garcia Salicetti S., Dorizzi, Gallinari P., Wimmer Z., Adaptive discrimination in an HMM-based neural predictive system for on line word recognition, ICPR 96.
5. Gish H., Ng K., a segmental speech model with applications to word spotting, ICASSP, 1993.
6. Guyon I., Schomaker L., Plamondon R., Liberman M., Janet S., UNIPEN project of on-line data exchange and recognizer benchmarks, ICPR'94, October 1994.
7. Holmes W.J. , Russel M.J.: Linear dynamic segmental HMMs: variability representation and training procedure, ICASSP, 1997.
8. Hu J., Gek Lim, S., Brown M.K., Writer independent on-line handwriting recognition using an HMM approach, Pattern Recognition, Vol 33, pp 133-147, 1999.
9. Manke S., Finke M., Waibel A., NPen++: a writer independent, large vocabulary on-line cursive handwriting recognition system, ICDAR 95, Montreal.
10. Ostendorf M., Digalakis V.V., Kimball O.A., From HMMs to segment models: a unified view of stochastic modeling for speech recognition, IEEE Trans. Speech and Audio proc., Vol 4., N° 5., 360-378, 1996.
11. Schenkel M., Guyon I., Henderson D., On-line cursive script recognition using time delay neural networks and hidden markov models, Machine Vision and Applications, 1995.