Input-Output Hidden Markov Models for Hand Gesture Recognition

Sébastien Marcel, Olivier Bernier and Daniel Collobert
France Télécom CNET
2, avenue Pierre Marzin
22307 Lannion, FRANCE
{sebastien.marcel, olivier.bernier, daniel.collobert}@cnet.francetelecom.fr

Abstract

A new hand gesture recognition method based on Input-Output Hidden Markov Models deals with the dynamic aspects of gestures. This method is compared to standard Hidden Markov Models and to Multi-layer Neural Networks. Gestures are extracted from a sequence of video images by tracking the skin-color blobs corresponding to the hand into a body-face space centered on the face of the user. Our goal is to recognize two classes of gestures: deictic and symbolic.

1 Introduction

Person detection and analysis is a challenging problem in computer vision for human computer interaction. LISTEN is a real-time computer vision system which detects and tracks a face in a sequence of video images. In this system, faces are detected by a modular neural network in skin color zones [6]. In [9], we developed a gesture based LISTEN system integrating skin-color blobs, face detection and hand posture recognition. Hand postures are detected using neural networks in a body-face space centered on the face of the user. Our goal is to recognize two classes of hand gestures (deictic and symbolic), prior to the hand posture recognition (i.e. command gesture). This paper describes a new approach for hand gesture recognition based on Input-Output Hidden Markov Models.

Input-Output Hidden Markov Models (IOHMM) were introduced by Bengio and Frasconi [3] for learning problems involving sequential structured data. They have similarities to Hidden Markov Models and also allow to map input sequences to ouput sequences. Indeed, for many training problems, the data are of sequential nature and Multi-layer Neural Networks (MLP) are often not adapted because of the lack of a memory mechanism to retain past information. Some neural networks models allow to capture the temporal relations by introducing time in their connections (Time Delay Neural Networks [17]). However, the temporal relations are fixed a priori by the network architecture and not by the data themselves which generally have temporal windows of variable input size.

Recurrent Neural Networks (RNN) model the dynamics of a system by capturing contextual information from one observation to another. The supervised training of RNN is primarily focused on methods of gradient descent: Back-Propagation Through Time [15], Real Time Recurrent Learning [19] and Local Feedback Recurrent Learning [11]. However, training with gradient descent is difficult when the duration of the temporal dependencies is large. Previous work on alternative training algorithms [4], such as IOHMM, suggests that the root of the problem lies in the essentially discrete nature of the process of storing contextual information for an indefinite amount of time.

2 Image Processing

We are working on image sequence in CIF format (384x288 pixels). In such images, we are interested in face detection and hand gesture recognition. Consequently, we must segment faces and hands from the image.

2.1 Face and hand segmentation

We filter the image using a fast look-up indexing table of skin color pixels in YUV color space. After filtering, skin color pixels are gathered into blobs [20].

Figure 1: Example of blobs on the face and the hand represented by polygons.

Blobs (Figure 1) are statistical objects [2] based on the location (x,y) and the colorimetry (Y,U,V) of the skin color pixels in order to determine homogeneous areas. A skin color pixel belongs to the blob which have the same location and colorimetry component.

2.2 Gesture extraction

We map, over the user, a body-face space based on a "discrete space for hand location" [10] centered on the face of the user as detected by LISTEN. The body-face space is built using an anthropometric body model expressed as a function of the total height of the user, itself calculated from the face height. Blobs are tracked into the body-face space. The 2D trajectory of the hand-blob[1] during a gesture is called a gesture path.

3 Hand Gesture Recognition

Numerous methods for hand gesture recognition have been proposed: Neural Networks (NN), such as recurrent models [12], Hidden Markov Models (HMM) [16] or gesture eigenspaces [18]. On one hand, NN achieve good classification performance, but are not adapted to observations with various durations. On the other hand, HMM allow to closely compute the probability that observations could be generated by the model, but have not the NN discrimination efficiency.

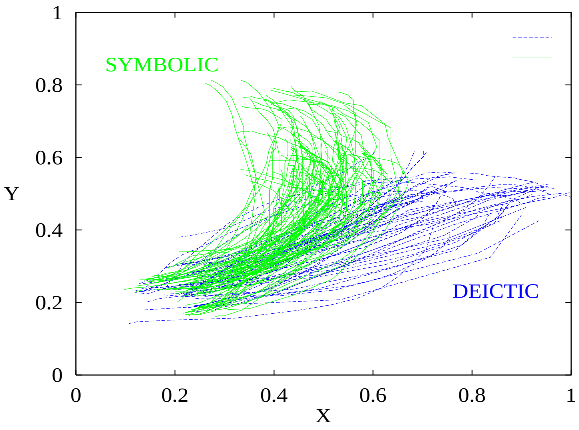

Figure 2: Deictic and Symbolic gesture paths in the body-face space.

Our goal is to recognize two classes of gestures: deictic and symbolic gestures (Figure 2). Deictic gestures are pointing movements towards the left (right) of the body-face space and symbolic gestures are intended to execute commands (grasp, clic, rotate) on the left (right) of shoulders. A video corpus was built using several persons executing several times these two classes of gestures. A database of gesture paths was obtained by automatic blob tracking and by manual video indexing.

4 Standard Hidden Markov Models

HMM are used in a widely range of applications, such as speech recognition [13]. A HMM allow us to compute $Pr(\mathcal{O} \mid \lambda)$ the probability of the finite observation sequence $\mathcal{O}$ given the model λ.

Let $\mathbf{o}_1^T = \mathbf{o}_1 \ldots \mathbf{o}_T$ be the training observation sequence, then the set of training sequences is defined by $\mathcal{O} = (\mathbf{o}_1^{T_p}(p))$ $(\forall p = 1 \ldots P)$, where P is the number of sequences and T_p is the length of the observed sequence p.

A HMM is the set of parameters $\lambda = (A, B, \Pi)$, with N discrete states $X = x_1 \ldots x_N$ and q_t is the state of the model at time t. $A = (a_{ij})$ is the probability transition matrix between states: $a_{ij} = \Pr(q_{t+1} = x_j \mid q_t = x_i)$. $B = (b_j(\mathbf{o}))$ is the observation probability in state j: $b_j(\mathbf{o}) = \Pr(\mathbf{o} \mid q_t = x_j)$. $\Pi = (\pi_i)$ is the initial state distribution: $\pi_i = \Pr(q_1 = x_i)$. We assume that $i, j = 1 \ldots N$. The probability $\Pr(\mathbf{o}_1^T \mid \lambda)$ of the finite observation sequence given the model is computed using the **Forward** step (Equation 1).

$$\Pr(\mathbf{o}_1^T \mid \lambda) = \sum_{i=1}^{N} \alpha_{i,T} \tag{1}$$

The **Forward** step computes the probability $\alpha_{i,t} = Pr(\mathbf{o}_1^t, q_t = x_i \mid \lambda)$ of a partial observation sequence ending at time t in state x_i. $\alpha_{i,t}$ is defined by induction by $\alpha_{j,t+1} = [\sum_{i=1}^{N} \alpha_{i,t} a_{ij}] b_j(\mathbf{o}_{t+1})$ $(\forall t = 1 \ldots T-1, \forall j = 1 \ldots N)$ and initialized by $\alpha_{i,1} = \pi_i b_j(\mathbf{o}_1)$ $(\forall i = 1 \ldots N)$.

In the case of a discrete observation $\mathbf{o} \in \mathcal{V}$, $\mathcal{V}$ is the set of finite observable symbols (i.e. the "codebook"). In the case of a continuous observation, observation probabilities are modeled using mixtures of density functions. Generally, these densities will be assumed to be Gaussian. Then, observation probabilities are given by $b_j(\mathbf{o}) = \sum_{m=1}^{M} c_{jm} \mathcal{M}(\mathbf{o}, \Theta_{jm})$, where M is the number of mixing parameters, c_{jm} is the mixing parameter m of state j $(\sum_{m=1}^{M} c_{jm} = 1)$ and Θ_{jm} is the set of parameters of the probability density function $\mathcal{M}(.)$.

In order to adjust the model parameters λ to maximize $\Pr(\mathcal{O} \mid \lambda)$, we use an iterative procedure such as the **EM** algorithm [7] and more specifically for HMM, the Baum-Welch algorithm [13]. In this way, HMM are able to model different observation sequences. However, there is no guaranty to suitably discriminate two classes of observations $\mathcal{O}$ and $\mathcal{O}'$ given the two models λ and λ'. Indeed, $\Pr(\mathcal{Z} \mid \lambda) > \Pr(\mathcal{Z} \mid \lambda')$ only means that the model λ better suits observations $\mathcal{Z}$ than the model λ', but does not mean that observations $\mathcal{Z}$ are the same as the ones in $\mathcal{O}$.

In this paper, we propose to use a new method for hand gesture recognition based on IOHMM. They have similarities with HMM and allows to discriminate two classes of observations.

5 Input-Output Hidden Markov Models

The aim of IOHMM is to propagate, backward in time, targets in a discrete space of states, rather than the derivatives of the errors, as in NN. The training is simplified and has only to learn the outputs and the next state defining the dynamic behavior.

[1] center of gravity of the blob corresponding to the hand

5.1 Architecture and modeling

The architecture of IOHMM consists of a set of states x, where each state is associated to a state neural network $\mathcal{N}_x$ and to an output neural network $\mathcal{O}_x$. For each network, the input vector $\mathbf{u}_t$ is the input at time t. A state network $\mathcal{N}_j$ has a number of outputs equal to the number of states. Each of these outputs gives the probability of transition from state j to a new state.

Let $\mathbf{u}_1^T = \mathbf{u}_1 \ldots \mathbf{u}_T$ be the input sequence (observation sequence) and $\mathbf{y}_1^T = \mathbf{y}_1 \ldots \mathbf{y}_T$ the output sequence. $\mathbf{u}$ is the input vector ($\mathbf{u} \in \mathcal{R}^m$) with m the input vector size and $\mathbf{y}$ is the output vector ($\mathbf{y} \in \mathcal{R}^r$) with r the output vector size. P is the number of input/output sequences and T is the length of the observed sequence. The set of input/output sequences is defined by $\mathcal{D} = (\mathcal{U}, \mathcal{Y}) = (\mathbf{u}_1^{T_p}(p), \mathbf{y}_1^{T_p}(p))$, with $p = 1 \ldots P$. The IOHMM model is described as follows:

- x_t : state of the model at time t where $x_t \in \mathcal{X}$, $\mathcal{X} = 1 \ldots n$ and n is the number of states of the model,
- S_i : set of successor states for state i, $S_i \subset \mathcal{X}$,
- $\mathcal{F}$: set of final states, $\mathcal{F} \subset \mathcal{X}$.

The dynamic of the model is defined by:

$$\begin{aligned} x_t &= f(x_{t-1}, \mathbf{u}_t) \\ \mathbf{y}_t &= g(x_t, \mathbf{u}_t) \end{aligned} \tag{2}$$

$\boldsymbol{\theta}_j$ is the set of parameters of the state network $\mathcal{N}_j$ ($\forall j = 1 \ldots n$). $\boldsymbol{\varphi}_{j,t} = {}^T[\varphi_{1j,t} \ldots \varphi_{nj,t}]$ is the output of the state network $\mathcal{N}_j$ at time t: $\varphi_{ij,t} = Pr(x_t = i \mid x_{t-1} = j, \mathbf{u}_t)$, i.e. the probability of transition from state j to state i ($\sum_{i=1}^n \varphi_{ij,t} = 1$). $\boldsymbol{\vartheta}_j$ is the set of parameters of output network $\mathcal{O}_j$ ($\forall j = 1 \ldots n$), where $\boldsymbol{\eta}_{j,t}$ is the output of the output network $\mathcal{O}_j$ at time t: $\eta_{ij,t} = Pr(y_{i,t} \mid x_t = j, \mathbf{u}_t)$. Let us introduce the following variables in the model:

- $\boldsymbol{\zeta}_t$: "memory" of the system at time t, $\boldsymbol{\zeta}_t \in \mathcal{R}^n$:

  $$\boldsymbol{\zeta}_t = \sum_{j=1}^{n} \zeta_{j,t-1} \boldsymbol{\varphi}_{j,t} \text{ for } t \neq 0$$

  where $\zeta_{j,t} = Pr(x_t = i \mid \mathbf{u}_1^t)$ and $\boldsymbol{\zeta}_0$ is randomly choosen with $\sum_{j=1}^n \zeta_{j,0} = 1$,
- $\boldsymbol{\eta}_t^*$: global output of the system at time t, $\boldsymbol{\eta}_t^* \in \mathcal{R}^r$ is:

  $$\boldsymbol{\eta}_t^* = \sum_{j=1}^{n} \zeta_{j,t} \boldsymbol{\eta}_{j,t} \tag{3}$$

  with the relation $\boldsymbol{\eta}_t^* = Pr(\mathbf{y}_t \mid \mathbf{u}_1^t)$, i.e. the probability to have the expected output $\mathbf{y}_t$ knowing the input sequence $\mathbf{u}_1^t$,
- $f_Y(\mathbf{y}_t; \boldsymbol{\eta}_{i,t})$: probability density function (**pdf**) of outputs where $f_Y(\mathbf{y}_t; \boldsymbol{\eta}_{i,t}) = Pr(\mathbf{y}_t \mid x_t = i, \mathbf{u}_t)$, i.e. the probability to have the expected output $\mathbf{y}_t$ knowing the current input vector $\mathbf{u}_t$ and the current state x_t.

We formulate the problem of the training as a problem of maximization of the probability function of the set of parameters of the model on the set of training sequences. The likelihood of input/output sequences $\mathcal{D}$ (Equation 4) is, as in HMM, the probability that a finite observation sequence could be generated by the IOHMM.

$$\begin{aligned} L(\Theta, \mathcal{D}) &= Pr(\mathcal{Y} \mid \mathcal{U}, \Theta) \\ &= \prod_{p=1}^{P} Pr(\mathbf{y}_1^{T_p} \mid \mathbf{u}_1^{T_p}, \Theta) \end{aligned} \tag{4}$$

where Θ is the parameter vector given by the concatenation of $\{\boldsymbol{\vartheta}_j\}$ et $\{\boldsymbol{\theta}_j\}$. We introduce the **EM** algorithm as a iterative method to estimate the maximum of the likelihood.

5.2 The EM algorithm

The goal of the EM algorithm (Expectation Maximization) [7] is to maximize the function of log-likelihood (Equation 5) on the parameters Θ of the model given the data $\mathcal{D}$.

$$l(\Theta, \mathcal{D}) = log L(\Theta, \mathcal{D}) \tag{5}$$

To simplify this problem, the EM assumption is to introduce a new set of parameters $\mathcal{H}$ known as the hidden set of parameters. Thus, we obtain a new set of data $\mathcal{D}_c = (\mathcal{D}, \mathcal{H})$, called the complete set of the data, of log-likelihood function $l(\Theta, \mathcal{D}_c)$. However, this function cannot be maximized directly because $\mathcal{H}$ is unknown. It was already shown [7] that the iterative estimation of the auxiliary function Q (Equation 6), using the parameters $\hat{\Theta}$ of the previous iteration, maximizes $l(\Theta, \mathcal{D}_c)$.

$$Q(\Theta, \hat{\Theta}) = E_{\mathcal{H}}[l(\Theta, \mathcal{D}_c) \mid \mathcal{D}, \hat{\Theta}] \tag{6}$$

Computing Q corresponds to supplement the missing data by using knowledge of the observed data and of the previous parameters. The EM algorithm is the following:

- For $k = 1 \ldots K$, where K is a local maxima
  - **E**stimation step: computation of $Q(\Theta, \Theta^{(k-1)}) = E_{\mathcal{H}}[l(\Theta, \mathcal{D}_c) \mid \mathcal{D}, \Theta^{(k-1)}]$
  - **M**aximization step: $\Theta^{(k)} = \arg\max_{\Theta} Q(\Theta, \Theta^{(k-1)})$

Analytical maximization is done by cancelling the partial derivatives $\frac{\partial Q(\Theta, \hat{\Theta})}{\partial \Theta} = 0$.

5.3 Training IOHMM using EM

Let $\mathcal{X}$ be the set of states sequences, $\mathcal{X} = (\mathbf{x}_1^{T_p}(p))$ with $p = 1 \ldots P$, the complete data set is:

$$\begin{aligned} \mathcal{D}_c &= (\mathcal{U}, \mathcal{Y}, \mathcal{X}) \\ &= (\mathbf{u}_1^{T_p}(p), \mathbf{y}_1^{T_p}(p), \mathbf{x}_1^{T_p}(p)), p = 1 \ldots P \end{aligned}$$

and the likelihood on $\mathcal{D}_c$ is:

$$\begin{aligned} L(\Theta, \mathcal{D}_c) &= Pr(\mathcal{Y}, \mathcal{X} \mid \mathcal{U}, \Theta) \\ &= \prod_{p=1}^{P} Pr(\mathbf{y}_1^{T_p}(p), \mathbf{x}_1^{T_p}(p) \mid \mathbf{u}_1^{T_p}(p), \Theta) \end{aligned}$$

For convenience, we choose to omit the variable p of the current sequence in order to simplify the notation. The conditional dependency of the variables of the system (Equation 2) allows us to write the above likelihood as:

$$L(\Theta, \mathcal{D}_c) = \prod_{p=1}^{P} \prod_{t=1}^{T_p} Pr(\mathbf{y}_t, x_t \mid x_{t-1}, \mathbf{u}_t, \Theta)$$

Let us introduce the variable $\mathbf{z}_t$

$$z_{i,t} = \begin{cases} 1 & : \quad x_t = i \\ 0 & : \quad x_t \neq i \end{cases}$$

the log-likelihood is then:

$$\begin{aligned} l(\Theta, \mathcal{D}_c) &= log\, L(\Theta, \mathcal{D}_c) \\ &= \sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{i=1}^{n} z_{i,t} log\, Pr(\mathbf{y}_t \mid x_t = i, \mathbf{u}_t, \Theta) \\ &+ \sum_{j=1}^{n} z_{i,t} z_{j,t-1} log\, Pr(x_t = i \mid x_{t-1} = j, \mathbf{u}_t, \Theta) \end{aligned}$$

However, the set of states sequences $\mathcal{X}$ is unknown, and $l(\Theta, \mathcal{D}_c)$ cannot be maximize directly. The auxiliary function Q must be computed (Equation 6):

$$\begin{aligned} & Q(\Theta, \hat{\Theta}) = E_{\mathcal{X}}[l_c(\Theta, \mathcal{D}_c) \mid \mathcal{U}, \mathcal{Y}, \hat{\Theta}] \\ &= \sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{i=1}^{n} \hat{\zeta}_{i,t} log\, f_Y(\mathbf{y}_t; \boldsymbol{\eta}_{i,t}) + \sum_{j=1}^{n} \hat{h}_{ij,t} log\, \varphi_{ij,t} \end{aligned}$$

where $\hat{h}_{ij,t}$ is computed using $\hat{\Theta}$ as follows:

$$\begin{aligned} h_{ij,t} &= Pr(x_t = i, x_{t-1} = j \mid \mathbf{u}_1^T, \mathbf{y}_1^T) \\ &= \frac{\alpha_{j,t-1}\, \varphi_{ij,t}\, \beta_{i,t}\, f_Y(\mathbf{y}_t; \boldsymbol{\eta}_{i,t})}{L} \end{aligned}$$

and $L = Pr(\mathbf{y}_1^T \mid \mathbf{u}_1^T)$, $\alpha_{i,t}$ and $\beta_{i,t}$ are computed (see [3] for details) using equations (7) and (8).

$$\begin{aligned} \alpha_{i,t} &= Pr(\mathbf{y}_1^t, x_t = i \mid \mathbf{u}_1^t) \\ &= f_Y(\mathbf{y}_t; \boldsymbol{\eta}_{i,t}) \sum_{j=1}^{n} \varphi_{ij,t} \alpha_{j,t-1} \qquad (7) \\ \beta_{i,t} &= Pr(\mathbf{y}_{t+1}^T \mid x_t = i, \mathbf{u}_t^T) \\ &= \sum_{j=1}^{n} f_Y(\mathbf{y}_{t+1}; \boldsymbol{\eta}_{j,t+1}) \beta_{j,t+1} \varphi_{ji,t+1} \qquad (8) \end{aligned}$$

Then L is given by:

$$\begin{aligned} L &= Pr(\mathbf{y}_1^T \mid \mathbf{u}_1^T) \\ &= \sum_{i \in \mathcal{F}} Pr(\mathbf{y}_1^T, x_T = i \mid \mathbf{u}_1^T) = \sum_{i \in \mathcal{F}} \alpha_{i,T} \end{aligned}$$

The learning algorithm is as follow: for each sequence $(\mathbf{u}_1^T, \mathbf{y}_1^T)$ and for each state $j = 1 \ldots n$, we compute $\boldsymbol{\varphi}_{j,t}$, $\boldsymbol{\eta}_{j,t}$, then $\alpha_{i,t}$, $\beta_{i,t}$ and $h_{ij,t}$ $(\forall i \in S_j)$. Then we ajust $\boldsymbol{\theta}_j$ parameters of the state networks $\mathcal{N}_j$ to maximize equation (9):

$$\sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{i=1}^{n} \sum_{j=1}^{n} \hat{h}_{ij,t} log\, \varphi_{ij,t} \qquad (9)$$

We also ajust $\boldsymbol{\vartheta}_j$ parameters of the output networks $\mathcal{O}_j$ to maximize equation (10).

$$\sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{i=1}^{n} \hat{\zeta}_{i,t} log\, f_Y(\mathbf{y}_t; \boldsymbol{\eta}_{i,t}) \qquad (10)$$

Let θ_{jk} be the set of parameters of state networks $\mathcal{N}_j$. The partial derivatives of equation (9) are given by:

$$\frac{\partial Q(\Theta, \hat{\Theta})}{\partial \theta_{jk}} = \sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{i \in S_j} \hat{h}_{ij,t} \frac{1}{\varphi_{ij,t}} \frac{\partial \varphi_{ij,t}}{\partial \theta_{jk}}$$

where the partial derivatives $\frac{\partial \varphi_{ij,t}}{\partial \theta_{jk}}$ are computed using classic back-propagation in the state network $\mathcal{N}_j$.

Let ϑ_{ik} be the set of parameters of output network $\mathcal{O}_i$. The partial derivatives of equation (10) are given by:

$$\begin{aligned} \frac{\partial Q(\Theta, \hat{\Theta})}{\partial \vartheta_{ik}} &= \sum_{p=1}^{P} \sum_{t=1}^{T_p} \hat{\zeta}_{i,t} \frac{\partial log\, f_Y(\mathbf{y}_t; \boldsymbol{\eta}_{i,t})}{\partial \vartheta_{ik}} \\ &= \sum_{p=1}^{P} \sum_{t=1}^{T_p} \hat{\zeta}_{i,t} \sum_{j=1}^{r} \frac{\partial log\, f_Y(\mathbf{y}_t; \boldsymbol{\eta}_{i,t})}{\partial \eta_{ji,t}} \frac{\partial \eta_{ji,t}}{\partial \vartheta_{ik}} \end{aligned}$$

As before, partial derivative $\frac{\partial \eta_{ji,t}}{\partial \vartheta_{ik}}$ can be computed by back-propagation in output networks $\mathcal{O}_i$. The **pdf** $f_Y(\mathbf{y}_t; \boldsymbol{\eta}_{i,t})$ depends on the problem.

5.4 Applying IOHMM to gesture recognition

Gesture paths are sequences of $[x_t, y_t]$ observations, where x_t, y_t are the coordinates at time t. Therefore, the input size is $m = 2$, and the output size $r = 1$. We choose to learn $y_1 = 1$ as output for deictic gestures and $y_1 = 0$ as output for symbolic gestures. Furthermore, we assume that the **pdf** of the model is $f_Y(\mathbf{y}_t;\boldsymbol{\eta}_{i,t}) = e^{-\frac{1}{2}\sum_{l=1}^{r}(y_{l,t}-\eta_{li,t})^2}$, i.e. an exponential Mean Square Error. Then, partial derivatives of equation (10) becomes:

$$\frac{\partial Q(\Theta,\hat{\Theta})}{\partial \vartheta_{ik}} = \sum_{p=1}^{P}\sum_{t=1}^{T_p}\hat{\zeta}_{i,t}\sum_{j=1}^{r}(y_{j,t}-\eta_{ji,t})\frac{\partial \eta_{ji,t}}{\partial \vartheta_{ik}}$$

Our gesture database (Table 1) is divided into three subsets: The learning set, the validation set and the test set. The learning set is used for training the IOHMM, the validation set is used to tune the model and the test set is used to evaluate the performance. Table 1 indicates, in the first column, the number of sequences. The second, third and fourth columns respectively indicates the minimum number of observations, the mean number of observations and the maximum number of observations.

Table 1: Description of the gesture database.

Deictic gestures				
	P	T_{min}	T_{mean}	T_{max}
Learning set	152	5	13	29
Validation set	76	5	14	29
Test set	57	5	15	28

Symbolic gestures				
	P	T_{min}	T_{mean}	T_{max}
Learning set	196	5	17	36
Validation set	98	5	17	36
Test set	99	8	18	37

6 Results and Discussion

First, we compare this IOHMM method to an other method based on MLP with fixed input size. Since the gesture database contains sequences of variable duration, sequences are interpolated, before presentation to the neural network, in order to have the same number of observations. We choose to interpolate all sequences to the mean number of observations $T_{mean} = 16$. Then, the input vector size is $m = 32$ for the MLP based on interpolated gesture paths.

We also compare the IOHMM based method to the HMM based one. We built two HMM, both in the discrete case and in the continuous case, one for deictic gestures and one for symbolic gestures. In the discrete case, each axis of the observation space $(\mathcal{R}^2)$ is divided into d intervals to obtain a codebook of dimension $D = d^2$. In practice, we have tested several values for $N = 3, 5, 7, 9$, $d = 4, 6, 10$ and $M = 1, 2, 3$, all leading to equivalent results. We observe that discrete HMM have always given better classification rates than continuous HMM. Therefore, only discrete HMM results are presented. The classification rate is computed on the test set by choosing, as for isolated word recognition [13], the model whose likelihood is highest.

Table 2: Classification rates for discrete HMM, IOHMM between **90%** and **100%** of the sequence, and MLP using interpolated gestures.

	Deictic	Symbolic
Discrete HMM	96.8%	65.6%
IOHMM (90%-100%)	97.6%	98.9%
MLP using interpolated gestures	98.2%	98.9%

Classification rates on the test set for discrete HMM, IOHMM and MLP are presented (Table 2). Classification rate for the IOHMM is determine by observing the global ouput $\boldsymbol{\eta}_t^*$ (Equation 3) versus time t expressed as a percentage of the duration of the sequence. Figure 3 presents, for all test sequences of both learning class, the mean and the standard deviation of the global output.

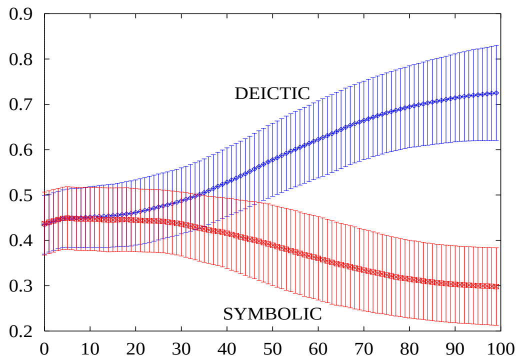

Figure 3: Global output $(\boldsymbol{\eta}_t^*)$ distribution of IOHMM versus time t expressed as a percentage of the duration of the sequence.

The IOHMM discriminates a deictic gesture from a symbolic gesture using the current observation after 60% of the sequence is presented. It acheives its best recognition rate between 90% and 100% of the sequence. In this case, IOHMM give equivalent results to MLP based on interpolated gestures and better results than HMM (Table 2). Nevertheless, IOHMM are more advantageous than the MLP used. The input is the current observation vector $[x_t, y_t]$, and we do not need to know, as for the MLP, the end of the sequence, and also to fix a temporal window a priori.

When trained gestures are encountered the classification is as powerful as the neural network used. But, untrained gestures (for example deictic and symbolic retractation gestures) cannot be classified correctly neither by the global output of the IOHMM nor by the ouput of the MLP based on interpolated gestures.

7 Conclusion

A new hand gesture recognition method based on Input/Output Hidden Markov Models is presented. IOHMM deal with the dynamic aspects of gestures. They have HMM properties and NN discrimination efficiency. When trained gestures are encountered the classification is as powerful as the neural network used.

Furthermore, as in HMM, it is possible to estimate in IOHMM a "likelihood cue" that can be used to stand out trained gestures from others. Future work is in progress in this direction and to integrate the hand gesture recognition, based on IOHMM, into the LISTEN based system. The full system will integrate face detection, hand posture recognition and hand gesture recognition.

References

[1] Baum, L. "An inequality and associated maximization technique in statistical estimation of probabilistic functions of markov processes", *Inequalities*, Vol. 3, pp. 1-8, 1972.

[2] Belongie, S. and Carson, C. and Greenspan, H. and Malik, J. "Color- and texture-based image segmentation using EM and its application to content-based image retrieval", *ICCV'98*, pp. 675-682, Bombay, 1998.

[3] Bengio, Y. and Frasconi, P. "An Input/Output HMM architecture", *Advances in Neural Information Processing Systems*, Vol. 7, pp. 427-434, 1995, Tesauro, G. and Touretzky, D. and Leen, T., MIT Press, Cambridge, MA.

[4] Bengio, Y. and Simard, P. and Frasconi, P., "Learning long-term dependencies with gradient descent is difficult", *IEEE Transactions on Neural Networks*, Vol. 5(2), pp. 157-166, 1994.

[5] Boehm, K. and Broll, W. and Sokolewicz, M. Dynamic Gesture Recognition Using Neural Networks: A Fundament for Advanced Interaction Construction, *SPIE, Conference Electronic Imaging Science and Technology*, pp. 336-346, 1994.

[6] Collobert, M. and Feraud, R. and Le Tourneur G. and Bernier, O. and Viallet, J.E. and Mahieux, Y. and Collobert, D. "LISTEN: A System for Locating and Tracking Individual Speakers", *2nd Int. Conf. on Automatic Face and Gesture Recognition*, pp. 283-288, 1996.

[7] Dempster, A.P. and Laird, N.M. and Rubin, D.B. "Maximum-likelihood from incomplete data via the EM algorithm", *Journal of Royal Statistical Society*, Vol. 39, pp. 1-938, 1977.

[8] Feraud, R. "PCA, Neural Networks and Estimation for Face Detection", *NATO ASI, Face Recognition: from Theory to Applications*, pp. 424-432, 1997.

[9] Marcel, S. "Hand Posture Recognition in a Body-Face centered space", *Conference on Human Factors in Computing Systems*, Extended Abstracts, pp. 302-303, Pittsburgh, PA USA, 1999.

[10] McNeill, D. *Hand and Mind: What gestures reveal about thought*, Chicago Press, 1992.

[11] Mozer, M. "A focused back-propagation algorithm for temporal pattern recognition", *Complex Systems*, Vol. 3, pp. 349-381, 1989.

[12] Murakami, K. and Taguchi, H. "Gesture recognition using Recurrent Neural Networks", *Conference on Human Interaction*, pp. 237-242, New Orlean, Louisiana, 1991.

[13] Rabiner, Lawrence R. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, Vol. 77 (2), pp. 257-285, 1989.

[14] Rubine, D. "Specifying gestures by examples", *Computer Graphics*, Vol. 25, pp. 329-337, 1991.

[15] Rumelhart, D. and Hinton, G. and Williams, R. "Learning internal representations by error propagation", *Parallel Distributed Processing*, Vol. 1(8), pp. 318-362, 1986, Rumelhart, D. and McClelland J., MIT Press, Cambridge.

[16] Starner, A. and Pentland, T. "Visual Recognition of American Sign Language using Hidden Markov Models", *Int. Conf. on Automatic Face and Gesture Recognition*, pp. 189-194, 1995.

[17] Waibel, A. and Hanazawa, T. and Hinton G. and Shikano, K. and Lang, K. "Phoneme recognition using time-delay neural netwoks", *IEEE transactions on Acoustics, Speech and Signal Processing*, Vol. 37, pp 328-339, 1989.

[18] Watanabe, T. and Yachida, M. "Real-Time Gesture Recognition using Eigenspace from Multi Input Image Sequences", *Int. Conf. on Automatic Face and Gesture Recognition*, pp. 428-433, Nara, Japan, 1998.

[19] Williams, R. and Zipser, D. "A learning algorithm for continually running fully recurrent neural networks", *Neural Computation*, Vol. 1, pp. 270-280, 1989.

[20] Wren, Christopher and Azarbayejani, Ali and Darrell, Trevor and Pentland, Alex "Pfinder: Real-Time Tracking of the Human Body", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19(7), pp. 780-785, 1997.