Hand Gesture Recognition using Input–Output Hidden Markov Models

Sebastien Marcel, Olivier Bernier, Jean–Emmanuel Viallet and Daniel Collobert
France Telecom CNET
2 avenue Pierre Marzin
22307 Lannion, FRANCE
{sebastien.marcel, olivier.bernier, jeanemmanuel.viallet, daniel.collobert}@cnet.francetelecom.fr

Abstract

A new hand gesture recognition method based on Input–Output Hidden Markov Models is presented. This method deals with the dynamic aspects of gestures. 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

Persons 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 coming from a camera. In this system, faces are detected by a modular neural network in skin color zones [3]. In [5], 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 supply the system with a gesture recognition kernel in order to detect the intention of the user to execute a command. This paper describe 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 [1] for learning problems involving sequential structured data. They have similarities to hidden markov models but allows to map input sequences to output 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 memory mechanism to retain past information. Some neural networks models allow to capture the temporal relations by using times in their connections (Time Delay Neural Networks) [11]. 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 for RNN is primarily focused on methods of gradient descent: Back–Propagation Through Time [9], Real Time Recurrent Learning [13] and Local Feedback Recurrent Learning [7]. However, training with gradient descent is difficult when the duration of the temporal dependencies is large. Previous work on alternative training algorithms [2], such as Input/Output Hidden Markov Models, suggest 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 (Figure 1) are gathered into blobs [14]. Blobs (Figure 2) are statistical objects 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 belong to the blob which have the same location and colorimetry component.

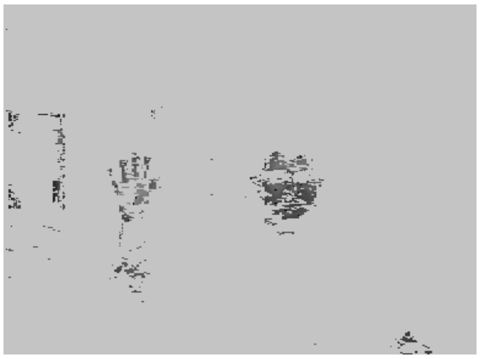

Figure 1. Image with skin color pixels

Figure 2. Example of blobs on the face and the hand represented by polygons

2.2. Extracting gestures

We map over the user a body–face space based on a discrete space for hand location [6] 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 method for hand gesture recognition have been proposed: neural networks (NN), such as recurrent models [8], hidden markov models (HMM)[10] or gesture eigenspaces [12]. On one hand, HMM allow to closely compute the probability that observations could be generated by the model. On the other hand, RNN achieve good classification performance by capturing the temporal relations from one observation to another. However, they cannot compute the likelihood of observation. In this paper, we use IOHMM which have HMM properties and NN discrimination efficiency.

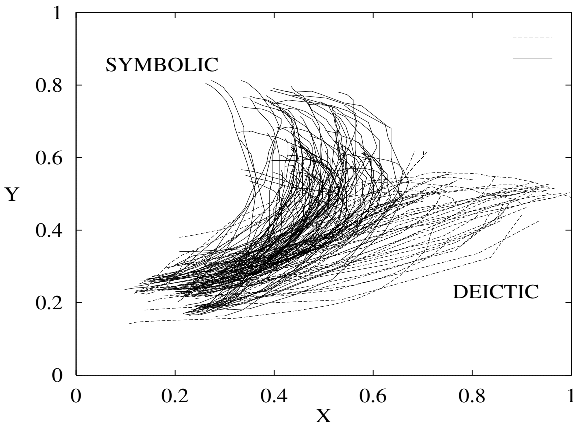

Figure 3. Deictic and Symbolic gesture paths in the body-face space

Our goal is to recognize two classes of gestures: deictic and symbolic gestures (Figure 3). 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 manual video indexing and automatic blob tracking.

4. 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.

4.1. 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 $\mathcal{O}_x$ where 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.

4.2. Modeling

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.

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

$\mathbf{u}$ is the input vector ($\mathbf{u} \in \mathbb{R}^m$) with m the input vector size and $\mathbf{y}$ is the output vector ($\mathbf{y} \in \mathbb{R}^r$) 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 $p = 1 \dots P$. The IOHMM model is described as follows:

- x_t : state of the model at time t where $x_t \in \mathcal{X}$, $\mathcal{X} = 1 \dots n$ 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 :

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

$\boldsymbol{\theta}_j$ is the set of parameters of the state network $\mathcal{N}_j$ ($\forall j = 1 \dots n$), where $\boldsymbol{\varphi}_{j,t} = {}^T[\varphi_{1j,t} \dots \varphi_{nj,t}]$ is the output of the state network $\mathcal{N}_j$ at time t, with the relation $\varphi_{ij,t} = Pr(x_t = i \mid x_{t-1} = j, \mathbf{u}_t)$, i.e. the probability of transition from state j to state i, with $\sum_{i=1}^{n} \varphi_{ij,t} = 1$. $\boldsymbol{\vartheta}_j$ is the set of parameters of output network $\mathcal{O}_j$ ($\forall j = 1 \dots n$), where $\boldsymbol{\eta}_{j,t}$ is the output of the output network $\mathcal{O}_j$ at time t, with the relation $\eta_{ij,t} = Pr(y_{i,t} \mid x_t = j, \mathbf{u}_t)$. 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 $\zeta_{j,t} = Pr(x_t = j \mid \mathbf{u}_1^t)$ and $\boldsymbol{\zeta}_0$ is randomly chosen with $\sum_{j=1}^{n} \zeta_{j,0} = 1$,
- $\boldsymbol{\eta}_t^*$: global output of the system at time t, $\boldsymbol{\eta}_t^* \in \mathbb{R}^r$ is:

  $$\boldsymbol{\eta}_t^* = \sum_{j=1}^{n} \zeta_{j,t} \boldsymbol{\eta}_{j,t} \tag{2}$$

  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 3) is, as in HMM, the probability that a finite observation sequence could be generated by the IOHMM.

$$\begin{aligned} L(\Theta, \mathcal{D}) &= Pr(\mathcal{Y} \mid \mathcal{U}, \Theta) \\ &= \prod_{p=1}^{P} Pr(\mathbf{y}_1^{T_p} \mid \mathbf{u}_1^{T_p}, \Theta) \end{aligned} \tag{3}$$

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.

4.3. The EM algorithm

The goal of the EM algorithm (Expectation Maximization) [4] is to maximize the function of log–likelihood (Equation 4) on the parameters Θ of the model given the data $\mathcal{D}$.

$$l(\Theta, \mathcal{D}) = log L(\Theta, \mathcal{D}) \tag{4}$$

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 [4] that the iterative estimation of the auxiliary function Q (Equation 5), using the parameters $\hat{\Theta}$ of the previous iteration, maximizes $l(\Theta, \mathcal{D}_c)$.

$$Q(\Theta, \hat{\Theta}) = E_{\mathcal{H}}[l(\Theta, \mathcal{D}_c) \mid \mathcal{D}, \hat{\Theta}] \tag{5}$$

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 $k = 1 \dots K$, where K is a local maxima

  Estimation step: computation of
  $Q(\Theta, \Theta^{(k-1)}) = E_{\mathcal{H}}[l(\Theta, \mathcal{D}_c) \mid \mathcal{D}, \Theta^{(k-1)}]$

  Maximization step:
  $\Theta^{(k)} = \arg\max_{\Theta} Q(\Theta, \Theta^{(k-1)})$

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

4.4. Training IOHMM using EM

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

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

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

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

For convenience, we choose to omit the p variable in order to simplify the notation. Furthermore, the conditional dependency of the variables of the system (Equation 1) 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 5):

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

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 [1] for details) using equations (6) and (7).

$$\begin{aligned} \alpha_{i,t} &= Pr(\mathbf{y}_1^t, x_t = i \mid \mathbf{u}_1^t) \\ &= f_Y(\mathbf{y}_t; \boldsymbol{\eta}_{i,t}) \sum_{j=1}^{n} \varphi_{ij,t} \alpha_{j,t-1} \qquad (6) \\ \beta_{i,t} &= Pr(\mathbf{y}_{t+1}^T \mid x_t = i, \mathbf{u}_t^T) \\ &= \sum_{j=1}^{n} f_Y(\mathbf{y}_{t+1}; \boldsymbol{\eta}_{j,t+1}) \beta_{j,t+1} \varphi_{ji,t+1} \qquad (7) \end{aligned}$$

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 $j = 1 \dots n$, 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 adjust $\boldsymbol{\theta}_j$ parameters of the state networks $\mathcal{N}_j$ to maximize the equation (8).

$$\sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{i=1}^{n} \sum_{j=1}^{n} \hat{h}_{ij,t} log\, \varphi_{ij,t} \qquad (8)$$

We also adjust $\boldsymbol{\vartheta}_j$ parameters of the output networks $\mathcal{O}_j$ to maximize the equation (9).

$$\sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{i=1}^{n} \hat{\zeta}_{i,t} log\, f_Y(\mathbf{y}_t; \boldsymbol{\eta}_{i,t}) \qquad (9)$$

Let θ_{jk} be the set of parameters of state networks $\mathcal{N}_j$. The partial derivatives of the equation (8) 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 the equation (9) 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.

4.5. Applying IOHMM to gesture recognition

We want to discriminate a deictic gesture from a symbolic gesture. Gesture paths are sequences of $[\Delta_t, x_t, y_t]$ observations, where x_y, y_t are the coordinate at time t and Δ_t is the sampling interval. Therefore, the input size is $m = 3$, 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 $f_Y(\mathbf{y}_t;\boldsymbol{\eta}_{i,t}) = e^{-\frac{1}{2}\sum_{l=1}^{r}(y_{l,t}-\eta_{l i,t})^2}$, i.e. an exponential Mean Square Error. Then, partial derivatives of the equation (9) 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 sub–sets: the learning set, the validation set and the test set. The learning set is used for training the IOHMM, the valida–tion 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

5. Results

We compare this IOHMM method to another method based on multi–layer neural networks (MLP) with fixed in–put size. Since the gesture database contains sequences of variable duration, sequences are interpolated, before pre–sentation to the neural network, in order to have the same number of observations. We choose to interpolate all se–quences to the mean number of observations $T_{mean} = 16$. Then, the input vector size is $m = 48$ for the MLP based on interpolated gesture paths.

Classification rates on test sets for the MLP based on interpolated gestures and the IOHMM are presented (Table 2). Classification rate for the IOHMM is determine by observing the global output $\boldsymbol{\eta}_t^*$ (Equation 2) over the time t expressed as a percentage of the length of the sequence. The figure 4 presents, for all sequences of both learning class, the mean and the standard deviation of the global output.

Table 2. Classification rate with neural net-works using interpolated gestures, and IOHMM between 90% and 100% of the sequence

	Deictic	Symbolic
NN using interpolated gestures	98.2%	98.9%
IOHMM	97.6%	98.9%

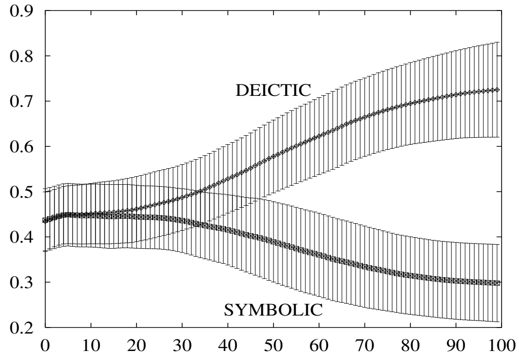

Figure 4. Global output (η_t^*) distribution of IOHMM in function of time sequence t

The IOHMM can discriminate a deictic gesture from a symbolic gesture using the current observation after 60% of the sequence is presented. It achieves the best recog–nition rate between 90% and 100% of the sequence. In this case, IOHMM give equivalent results to MLP based on interpolated gestures. Nevertheless, IOHMM are more advantageous than the MLP used. The temporal window is not fixed a priori and the input is the current observation vector $[\Delta_t, x_t, y_t]$.

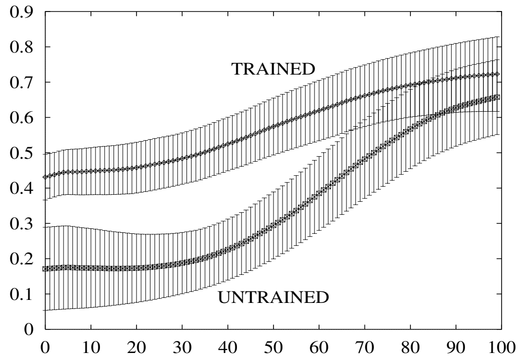

Figure 5. Global output distribution of IOHMM on Trained and Untrained gestures

Unfortunately, untrained gestures, i.e. the deictic and symbolic retractation gestures, cannot be classified neither by the output of the MLP based on interpolated gestures nor by the global output of the IOHMM (Figure 5).

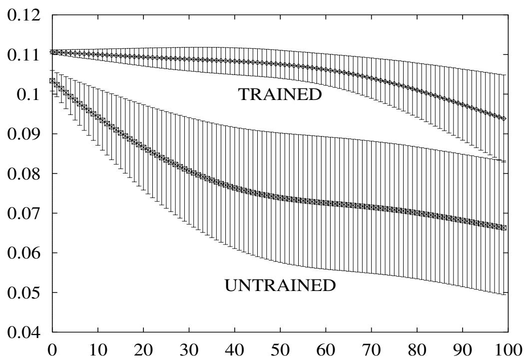

Figure 6. "Likelihood cue" of IOHMM on Trained and Untrained gestures

Nevertheless, it is possible to estimate for the IOHMM, a "likelihood cue" that can be used to stand out trained gestures from untrained gestures (Figure 6). This "likelihood cue" can be computed in a HMM way by adding to each state of the model an observation probability of the input $\mathbf{u}_t$.

6. 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 Hidden Markov Models properties and Neural Networks discrimination efficiency. When trained gestures are encountered the classification is as powerful as the neural network used. The IOHMM use the current observation only and not a temporal windows fixed a priori. Furthermore, when untrained gestures are encountered, the "likelihood cue" is more discriminant than the global output.

Future work is in progress 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] Y. Bengio and P. Frasconi. An Input/Output HMM architecture. In *Advances in Neural Information Processing Systems*, page 427–434, 1995.

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

[3] M. Collobert, R. Feraud, G. LeTourneur, O. Bernier, J. Viallet, Y. Mahieux, and D. Collobert. LISTEN: A system for locating and tracking individual speakers. In *2nd Int. Conf. on Automatic Face and Gesture Recognition*, page 283–288, 1996.

[4] A. Dempster, N. Laird, and D. Rubin. Maximum–likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, 39:1–938, 1977.

[5] S. Marcel. Hand posture recognition in a body–face centered space. In *CHI'99*, page 302–303, 1999. Extended Abstracts.

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

[7] M. Mozer. A focused back–propagation algorithm for temporal pattern recognition. *Complex Systems*, 3:349–381, 1989.

[8] K. Murakami and H. Taguchi. Gesture recognition using recurrent neural networks. In *Conference on Human Interaction*, page 237–242, 1991.

[9] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1, page 318–362. MIT Press, Cambridge, 1986.

[10] A. Starner and T. Pentland. Visual recognition of American Sign Language using Hidden Markov Models. In *Int. Conf. on Automatic Face and Gesture Recognition*, page 189–194, 1995.

[11] A. Waibel, T. Hanazawa, H. G., K. Shikano, and K. Lang. Phoneme recognition using time–delay neural netwoks. *IEEE transactions on Acoustics, Speech and Signal Processing*, 37:328–339, 1989.

[12] T. Watanabe and M. Yachida. Real–time gesture recognition using eigenspace from multi input image sequences. In *Int. Conf. on Automatic Face and Gesture Recognition*, page 428–433, 1998.

[13] R. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

[14] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real–time tracking of the human body. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 19 of 7, page 780–785, 1997.