Approches génératives pour le traitement de séquences d'images: application à la reconnaissance dynamique des gestes de la main

Sébastien Marcel
IDIAP, Institut Dalle Molle d'Intelligence Artificielle Perceptive
rue du Simplon 4, 1920 Martigny, Suisse
marcel@idiap.ch

Résumé

Cet article propose deux approches génératives pour le traitement de séquences d'images appliquées à la reconnaissance dynamique des gestes de la main. Dans un premier temps, un modèle probabiliste gaussien de régions homogènes de teinte chair (blob) est présenté. Les paramètres des blobs sont calculés par un algorithme EM (Expectation-Maximisation). Les blobs obtenus sont moins nombreux et plus stables que les zones de pixels de teinte chair connexes dont les blobs sont issus. Dans un second temps, l'article décrit un modèle hybride à base de réseaux de neurones et de modèles de Markov cachés pour traiter des séquences de données et ainsi reconnaitre les trajectoires formées par les blobs comme des gestes. Les paramètres du modèle sont calculés également par un algorithme EM. Le modèle obtenu est capable d'effectuer des tâches de prédiction et de classification. Une extension générative de ce modèle est proposée pour prendre en compte la probabilité d'observation des entrées. Ainsi, le nouveau modèle génératif est capable de rejeter une séquence d'entrée qui n'a jamais été apprise.

Mots Clef

vision par ordinateur, teinte chair, blobs, modèle statistique, algorithme EM, réseaux de neurones, modèles de Markov cachés.

1 Introduction

La détection et l'analyse des personnes est un problème fondamental en vision par ordinateur et plus spécifiquement dans la conception des interfaces gestuelles. Les interfaces gestuelles basées sur l'image constituent la voie la plus naturelle pour la construction d'interfaces homme-machine évoluées.

Dans cet article, nous nous intéressons plus particulièrement aux gestes de la main. Ainsi, avant de pouvoir reconnaître un geste de la main dans des images, il convient tout d'abord d'y repérer la main.

Généralement, les objets importants tels que le visage ou les mains sont détectés à l'aide de la couleur de la peau. Il existe de nombreux modèles pour représenter et segmenter la couleur de la peau, mais encore faut-il regrouper les pixels pour former des régions pertinentes autour des objets (blobs) et éviter que ces dernières ne fusionnent entre elles ou avec le fond.

Cet article décrit donc, dans sa première partie, une formalisation des blobs définissant un modèle génératif (probabiliste gaussien) dont les paramètres sont ajustés aux données de l'image par une procédure EM.

Dans une seconde partie, une nouvelle approche générative pour le traitement des séquences de données basée sur les Input-Output Hidden Markov Models est présentée aprés un rappel sur les modèles de Markov cachés. Cette nouvelle approche est appliquée à la reconnaissance des gestes de la main et plus exactement à la reconnaissance des trajectoires que les blobs de la main forment dans une séquence d'images.

2 Le traitement de l'image

Les séquences d'images que nous traitons sont issues d'une caméra (Figure 1) et sont au format CIF (384x288 pixels).

FIG. 1 – *Image de type visiophone à analyser.*

Dans ces images, nous nous intéressons à des objets particuliers, les visages et les mains. Ces objets possèdent une teinte caractéristique qu'il est possible de séparer du reste de l'image.

2.1 Le filtrage de la teinte chair

Il existe de nombreuses méthodes pour modéliser la teinte chair, principalement à l'aide de gaussiennes ou de mélanges de gaussiennes [8]. De plus, divers espaces colorimétriques RVB, HSV ou Lab peuvent être utilisés. Nous avons choisi l'espace colorimétrique YUV, car c'est le format issu de la caméra et il est relativement robuste au changement de luminosité grâce à la séparation de la luminance (Y) et de la chrominance (UV).

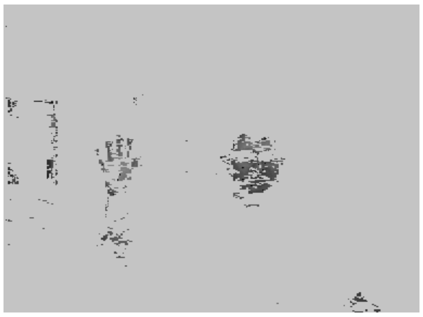

FIG. 2 – *Image des pixels de teinte chair.*

Nous avons choisi de filtrer l'image en utilisant une table de correspondance des pixels de teinte chair (Figure 2). Cette technique bien que rudimentaire est efficace et très rapide. Une fois les pixels de teinte chair séparés du reste de l'image, nous cherchons à les regrouper en régions.

2.2 Zonage par regroupement des pixels connexes

Une première approche pour regrouper les pixels est de tenir compte de leur connexité. Un algorithme d'expansion permet de déterminer des zones de pixels de teinte chair connexes (Figure 3).

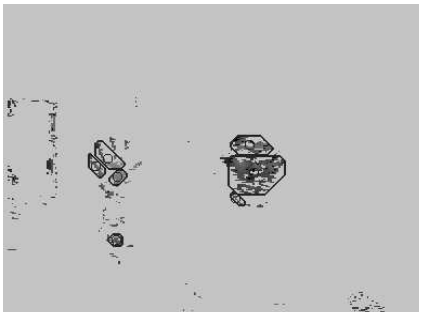

FIG. 3 – *Zones connexes de teinte chair à l'image t.*

De plus, on élimine les zones de hauteur ou de largeur inférieure à 10 pixels. Ceci explique pourquoi certaines zones n'apparaissent pas dans le fond. Cependant, le bruit de la caméra et la variation de l'illumination dans la scène rend ces zones géométriquement instables. Certaines zones apparaissent ou disparaissent, d'autres peuvent changer de taille ou fusionner ensemble (Figure 4). Nous allons utiliser une méthode non plus basée uniquement sur la connexité mais sur les composantes spatiales et chromatiques des pixels pour déterminer des régions homogènes.

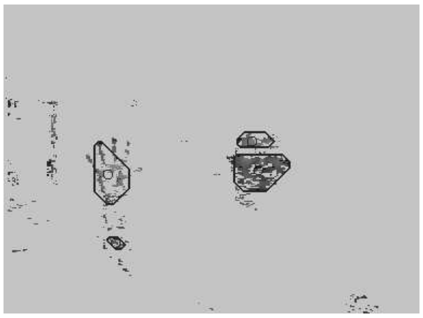

FIG. 4 – *Zones connexes de teinte chair à l'image $t+1$.*

3 Les blobs

De nombreux modèles de blobs ont déjà été utilisés pour la détection des personnes [15] ou pour la reconnaissance des gestes de la main [10].

3.1 Le modèle des blobs

Dans cette section, nous définissons un blob comme un modèle probabiliste gaussien que nous cherchons à ajuster aux pixels de teinte chair de composantes spatiales (x,y) et chromatiques (U,V). Le nombre de blobs est impossible a estimer car les objets (visage et mains) sont inconnus. Nous supposerons donc que le nombre de blobs est fixe, puis nous introduirons une méthode pour initialiser et fusionner les blobs.
Introduisons les notations suivantes :

- P : nombre de pixels de teinte chair dans l'image,
- $\mathcal{I} = \{\mathbf{x}_i\}$: l'ensemble des pixels de teinte chair de l'image, avec $i = 1 \ldots P$
- $\mathbf{x}_i = {}^T[x_i,y_i,u_i,v_i]$
  - x_i,y_i : position du pixel dans l'image,
  - u_i,v_i : chrominance du pixel dans l'image.
- B : nombre de blobs fixé au départ,
- $\mathcal{B} = (\{\boldsymbol{\mu}_j\},\{\boldsymbol{\sigma}_j\})$: ensemble des blobs, avec $j = 1 \ldots B$
  - $\boldsymbol{\mu}_j$: vecteur centre des pixels du blob j,
  - $\boldsymbol{\sigma}_j$: inverse de la matrice de covariance des pixels du blob j.

Dans ce modèle, on suppose connu a priori le nombre de blobs B. On recherche les paramètres $\{\boldsymbol{\mu}_j\}$ et $\{\boldsymbol{\sigma}_j\}$ qui permettent d'obtenir les meilleurs blobs étant donnés les pixels de teinte chair obtenus par le filtrage de l'image, c'est-à-dire ceux qui maximisent la vraisemblance des pixels observés.

3.2 La vraisemblance des blobs

Soit Θ la collection des paramètres $\{\boldsymbol{\mu}_j\}$ et $\{\boldsymbol{\sigma}_j\}$. On peut alors introduire la fonction de vraisemblance L (Equation 1) des paramètres des blobs Θ sur l'ensemble des données $\mathcal{I}$, c'est-à-dire les pixels de teinte chair.

$$L(\Theta,\mathcal{I}) \quad = \quad \Pr(\mathcal{I} \mid \Theta) = \prod_{i=1}^{P} \Pr(\mathbf{x}_i \mid \Theta) \qquad (1)$$

L'équation 1 peut être maximisée par l'algorithme EM.

3.3 L'algorithme EM

L'algorithme EM (Expectation Maximisation) [4] cherche à maximiser la fonction de log-vraisemblance (Equation 2) sur les paramètres Θ, l'ensemble des données $\mathcal{I}$ étant donné.

$$l(\Theta,\mathcal{I}) = logL(\Theta,\mathcal{I}) \qquad (2)$$

Pour simplifier ce problème, l'hypothèse EM est d'introduire un nouvel ensemble de paramètres $\mathcal{H}$ dits cachés. Ainsi, nous obtenons un nouvel ensemble de données $\mathcal{D} = (\mathcal{I},\mathcal{H})$, appelé l'ensemble complet des données, de fonction de log-vraisemblance $l(\Theta,\mathcal{D})$. Cependant, cette fonction ne peut pas être maximisée directement car $\mathcal{H}$ est inconnu. Il a été déjà démontré [4] que l'estimation itérative de la fonction auxiliaire Q (Equation 3) maximise $l(\Theta,\mathcal{D})$ en utilisant les paramètres $\hat{\Theta}$ de l'itération précédente.

$$Q(\Theta,\hat{\Theta}) = E_{\mathcal{H}}[l(\Theta,\mathcal{D}) \mid \mathcal{I},\hat{\Theta}] \qquad (3)$$

L'algorithme EM est le suivant :

- Pour $k = 1 \ldots K$
  - Estimation : calcul de $Q(\Theta,\Theta^{(k-1)}) = E_{\mathcal{H}}[l(\Theta,\mathcal{D}) \mid \mathcal{I},\Theta^{(k-1)}]$
  - Maximisation : $\Theta^{(k)} = \arg\max_{\Theta} Q(\Theta,\Theta^{(k-1)})$

3.4 Ajustement des blobs

Les variables cachées. L'ajout de variables cachées doit faciliter la résolution du problème. Dans notre cas, ces variables cachées vont nous permettre d'associer un pixel à un et un seul blob. Soit les paramètres cachés $\mathcal{H} = h_1 \ldots h_P$, où h_i est le blob associé au pixel i.

Nous avons l'ensemble complet des données $\mathcal{D}$ défini par :

$$\mathcal{D} = (\mathcal{I},\mathcal{H}) = (\mathbf{x}_1 \ldots \mathbf{x}_P, h_1 \ldots h_P)$$

La fonction de vraisemblance sur $\mathcal{D}$ est :

$$\begin{aligned} L(\Theta,\mathcal{D}) &= \Pr(\mathcal{I},\mathcal{H} \mid \Theta) = \prod_{i=1}^{P} \Pr(\mathbf{x}_i, h_i \mid \Theta) \\ &= \prod_{i=1}^{P} \Pr(\mathbf{x}_i \mid h_i,\Theta) \Pr(h_i \mid \Theta) \qquad (4) \end{aligned}$$

Posons z_{ij} la variable aléatoire définie par :

$$z_{ij} = \begin{cases} 1 & : \quad h_i = j \\ 0 & : \quad h_i \neq j \end{cases}$$

alors on peut réécrire l'équation 4 comme :

$$L(\Theta,\mathcal{D}) \quad = \quad \prod_{i=1}^{P} \prod_{j=1}^{B} \Pr(\mathbf{x}_i \mid h_i = j,\Theta)^{z_{ij}} \Pr(h_i = j \mid \Theta)^{z_{ij}}$$

Etape d'Estimation. L'équation 3 nous permet d'écrire la fonction auxiliaire :

$$\begin{aligned} Q(\Theta,\hat{\Theta}) &= E_{\mathcal{H}}[logL(\Theta,\mathcal{D}) \mid \mathcal{I},\hat{\Theta}] \\ &= \sum_{i=1}^{P} \sum_{j=1}^{B} \Pr(h_i = j \mid \mathcal{I},\hat{\Theta}) \\ &\quad [\log \Pr(\mathbf{x}_i \mid h_i = j,\Theta) + \log \Pr(h_i = j \mid \Theta)] \end{aligned}$$

Nous avons supposé précédemment qu'un pixel appartient à un et un seul blob. Nous supposons que tous les blobs sont équiprobables d'où :

$$\Pr(h_i = j \mid \Theta) = \frac{1}{B}$$

De plus, on pose que la fonction de densité de probabilité d'un pixel de teinte chair $\mathbf{x}_i$ connaissant le blob j est une gaussienne, ainsi :

$$\begin{aligned} \Pr(\mathbf{x}_i \mid h_i = j,\Theta) &= \mathcal{G}(\mathbf{x}_i,\boldsymbol{\mu}_j,\boldsymbol{\sigma}_j) \\ &= \sqrt{\frac{\mid \boldsymbol{\sigma}_j \mid}{(2\pi)^4}}\, e^{-\frac{1}{2}\,{}^{T}(\mathbf{x}_i - \boldsymbol{\mu}_j)\boldsymbol{\sigma}_j(\mathbf{x}_i - \boldsymbol{\mu}_j)} \end{aligned}$$

D'autre part, étant donné qu'un pixel appartient à un et un seul blob, on a:

$$Pr(h_i = j \mid \mathcal{I},\hat{\Theta}) \quad = \quad \frac{1}{B} \frac{\Pr(\mathbf{x}_i \mid h_i = j,\hat{\Theta})}{\sum_{l=1}^{B} \Pr(\mathbf{x}_i \mid h_i = l,\hat{\Theta})}$$

Finalement, nous avons :

$$\begin{aligned} Q(\Theta,\hat{\Theta}) &= \sum_{i=1}^{P} \sum_{j=1}^{B} E_{ij} \log \mathcal{G}(\mathbf{x}_i,\boldsymbol{\mu}_j,\boldsymbol{\sigma}_j) \\ &+ \sum_{i=1}^{P} \sum_{j=1}^{B} E_{ij} \log \frac{1}{B} \end{aligned}$$

avec :

$$E_{ij} = \frac{\mathcal{G}(\mathbf{x}_i,\hat{\boldsymbol{\mu}}_j,\hat{\boldsymbol{\sigma}}_j)}{\sum_{l=1}^{B} \mathcal{G}(\mathbf{x}_i,\hat{\boldsymbol{\mu}}_l,\hat{\boldsymbol{\sigma}}_l)} \tag{5}$$

L'étape d'estimation de l'algorithme EM s'achève sur le calcul des E_{ij} en utilisant les données $\mathcal{I}$ et les paramètres précédents $\hat{\Theta}$.

Etape de Maximisation. Dans cet article, nous avons choisi de développer une maximisation analytique des blobs. On recherche les paramètres Θ de sorte à maximiser la fonction auxiliaire. Pour cela, on calcule les paramètres $\boldsymbol{\mu}_j$ et $\boldsymbol{\sigma}_j$ tels que $\frac{\partial Q(\Theta,\hat{\Theta})}{\partial \boldsymbol{\mu}_j} = 0$ et $\frac{\partial Q(\Theta,\hat{\Theta})}{\partial \boldsymbol{\sigma}_j} = 0$.

Commençons par dériver $Q(\Theta,\hat{\Theta})$ par rapport à $\boldsymbol{\mu}_j$:

$$\frac{\partial Q(\Theta,\hat{\Theta})}{\partial \boldsymbol{\mu}_j} = 0 \Leftrightarrow \boldsymbol{\mu}_j = \frac{\sum_{i=1}^{P} E_{ij}\,\mathbf{x}_i}{\sum_{i=1}^{P} E_{ij}} \tag{6}$$

Ainsi, si l'on cherche à annuler les dérivées, on obtient une somme des pixels pondérés par l'espérance E_{ij}.

A présent, dérivons $Q(\Theta,\hat{\Theta})$ par rapport à $\boldsymbol{\sigma}_j$:

$$\frac{\partial Q(\Theta,\hat{\Theta})}{\partial \boldsymbol{\sigma}_j} = 0$$

$$\Leftrightarrow \quad \boldsymbol{\sigma}_j = \left(\frac{\sum_{i=1}^{P} E_{ij}\,(\mathbf{x}_i - \boldsymbol{\mu}_j)^{\,T}(\mathbf{x}_i - \boldsymbol{\mu}_j)}{\sum_{i=1}^{P} E_{ij}} \right)^{-1} \tag{7}$$

On obtient pour $\boldsymbol{\sigma}_j$ (Equation 7) l'inverse de la matrice de corrélation pondérée par les espérances E_{ij}.

L'algorithme EM appliqué aux blobs devient à présent :

- Pour k = 1 . . . K, avec K étant le nombre d'itérations EM
  - Calculer les E_{ij} d'après l'équation 5,
  - Calculer les $\{\boldsymbol{\mu}_j\}$ d'après l'équation 6,
  - Calculer les $\{\boldsymbol{\sigma}_j\}$ d'après l'équation 7.

A présent, il nous manque une procédure pour initialiser les blobs avant de débuter l'algorithme EM.

3.5 Initialisation des blobs

On suppose que le nombre de blobs, qui est constant durant l'algorithme EM, doit être convenablement initialisé. Or, on ne le connaît pas a priori. Ainsi, les blobs sont initialisés à partir des zones connexes de teinte chair. Puis, on fusionne les nouveaux blobs avec les blobs de l'image précédente. Cette approche permet le plus souvent de sur-estimer le nombre de blobs. Une fois l'ensemble des blobs ci-dessus obtenu, on applique l'algorithme EM. A l'issue de celui-ci, les blobs ont convergé vers un état stable et se sont répartis sur les objets. Cependant, comme il est possible d'avoir deux blobs sur le même objet (Figure 5), il faut introduire un mécanisme pour fusionner les blobs.

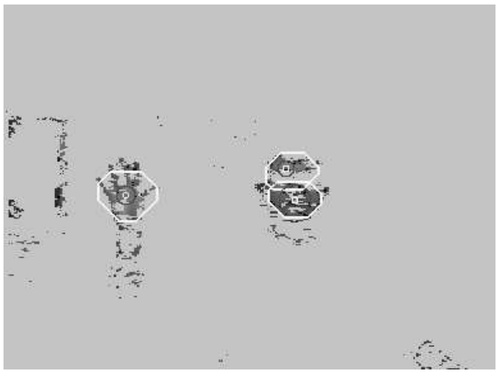

FIG. 5 – *Blobs avant la fusion.*

3.6 Fusion des blobs

La fusion s'effectue sur un critère spatial et chromatique à l'aide d'un "rayon d'attraction" intra-blob ς recalculé après la maximisation finale des $\{\boldsymbol{\mu}_j\}$. On pose ς_j comme étant l'écart moyen spatio-chromatique (Equation 8) des pixels au centre du blob $\boldsymbol{\mu}_j$.

$$\varsigma_j = \frac{1}{Card\,\Omega_j} \sum_{\mathbf{x}_k \in \Omega_j} \| \mathbf{x}_k - \boldsymbol{\mu}_j \| \tag{8}$$

On décide de fusionner deux blobs j et k (Figure 6) si $\| \boldsymbol{\mu}_j - \boldsymbol{\mu}_k \| \leq \varsigma_j + \varsigma_k$. En utilisant cette méthode, on permet à deux blobs suffisamment proches dans l'espace spatio-chromatique de fusionner ensemble. On peut éviter également qu'un blob-main ne fusionne avec un blob-visage lorsque la main passe près du visage ou vient toucher le menton. Cependant, la fusion n'est pas garantie, on peut très bien observer temporairement deux blobs sur le même objet. Mais, lorsque les objets bougent, on constate que, le plus souvent, l'image apporte de nouvelles informations de teinte chair pour permettre aux blobs de fusionner.

FIG. 6 – *Blobs après la fusion.*

3.7 Résultats

Nous obtenons des blobs (Figures 7 et 8) plus robustes que les zones de pixels de teinte chair connexes. En effet, on constate que les blobs sont géométriquement plus stables que les zones.

FIG. 7 – *Exemple de blobs du visage et de la main sur fond uniforme.*

FIG. 8 – *Exemple de blobs du visage et de la main sur fond complexe.*

Prenons l'exemple d'une séquence ne contenant que le visage, la variation de la surface d'un blob-visage est moindre que celle d'une zone-visage (Figure 9).

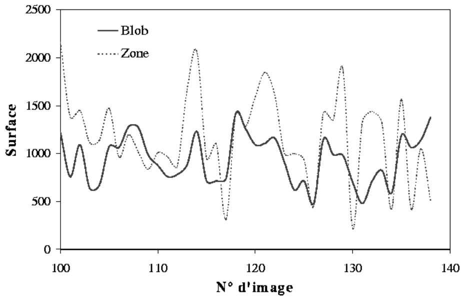

FIG. 9 – *Variation de la surface d'une zone et d'un blob sur une séquence visage seul.*

4 Les "Input-Ouput Hidden Markov Models"

Les Input-Output Hidden Markov Models (IOHMM) ont été introduits par Bengio et Frasconi [2] pour des problèmes d'apprentissage mettant en jeux des données de nature séquentielle. Les IOHMM ont des similarités avec les modèles de Markov cachés (Hidden Markov Models), mais permettent d'associer des séquences d'entrées à des séquences de sorties. En effet, dans de nombreux problèmes d'apprentissage, les données sont de nature séquentielle et les réseaux de neurones multicouches (MLP) ne sont pas naturellement adaptés à cause du manque d'un mécanisme de mémorisation pour retenir l'information passée. Certains réseaux de neurones permettent de capturer les relations temporelles en utilisant des délais dans leurs connexions (Time Delay Neural Networks) [13]. Néanmoins, les relations temporelles sont fixées a priori par l'architecture du réseau et non par les données elles-mêmes qui ont généralement des fenêtres temporelles de taille variable.

Les réseaux de neurones récurrents (RNN) modélisent la dynamique du système en capturant l'information contextuelle d'une observation à l'autre. L'apprentissage supervisé dans les RNN est principalement basé sur des méthodes de descente du gradient: Back-Propagation Through Time [12], Real Time Recurrent Learning [14] et Local Feedback Recurrent Learning [9]. Néanmoins, l'apprentissage avec la descente du gradient est difficile lorsque la durée des dépendances temporelles est grande. De précédents travaux sur des algorithmes d'apprentissage alternatifs [3], comme les IOHMM, suggèrent que le problème réside dans la nature essentiellement discrète du processus de stockage de l'information contextuelle pour une durée de temps indéfinie.

A présent, introduisons les modèles de Markov cachés pour nous familiariser avec les mécanisme du traitement des données séquentielles.

4.1 Les modèles de Markov cachés

Les modèles de Markov cachés (HMM) modélisent les observations de nature séquentielle (Figure 10).

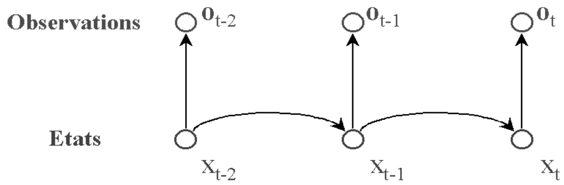

FIG. 10 – *Le modèle du HMM.*

Ils sont utilisés dans une grande variété d'applications, comme la reconnaissance de la parole [11] ou de l'écriture manuscrite. Soit $\mathbf{o}_1^T = \mathbf{o}_1 \ldots \mathbf{o}_T$ une séquence d'observation d'apprentissage. L'ensemble des séquences d'apprentissage est défini par $\mathcal{O} = (\mathbf{o}_1^{T_p}(p))$ $(\forall p = 1 \ldots P)$, où P est le nombre de séquences et T_p est la longueur de la séquence observée p.

Un HMM à N états discrets $X = x_1 \ldots x_N$ où q_t est l'état du modèle à l'instant t, est défini par l'ensemble des paramètres $\lambda = (A,B,\Pi)$. $\forall i,j = 1 \ldots N$: $A = (a_{ij})$ est la matrice des probabilités de transitions entre états, $a_{ij} = \Pr(q_{t+1} = x_j \mid q_t = x_i)$; $B = (b_j(\mathbf{o}))$ est la probabilité d'observation dans l'état j, $b_j(\mathbf{o}) = \Pr(\mathbf{o} \mid q_t = x_j)$; $\Pi = (\pi_i)$ est la distribution des états initiaux, $\pi_i = \Pr(q_1 = x_i)$. Un HMM nous permet de calculer la probabilité $\Pr(\mathbf{o}_1^T \mid \lambda)$ d'une séquence d'observation finie étant donné le modèle λ (Equation 9).

$$\Pr(\mathbf{o}_1^T \mid \lambda) = \sum_{i=1}^{N} \alpha_{i,T} \tag{9}$$

La probabilité $\alpha_{i,t} = Pr(\mathbf{o}_1^t, q_t = x_i \mid \lambda)$ d'une séquence d'observation partielle finissant à l'instant t dans l'état x_i, est définie par induction par $\alpha_{j,t+1} = [\sum_{i=1}^{N} \alpha_{i,t} a_{ij}] b_j(\mathbf{o}_{t+1})$ $(\forall t = 1 \ldots T-1,\ \forall j = 1 \ldots N)$ et initialisée par $\alpha_{i,1} = \pi_i b_j(\mathbf{o}_1)$ $(\forall i = 1 \ldots N)$.

Afin d'ajuster les paramètres λ du modèle pour maximiser $\Pr(\mathcal{O} \mid \lambda)$, on utilise une procédure itérative basée sur l'algorithme EM [4]: l'algorithme de Baum-Welch [11]. De cette manière, un HMM est capable d'ajuster ses paramètres pour modéliser des séquences d'observations. Néanmoins, il n'y a aucune garantie de pouvoir convenablement discriminer deux classes d'observations $\mathcal{O}$ et $\mathcal{O}'$ étant donné deux modèles λ et λ'. En effet, $\Pr(\mathcal{Z} \mid \lambda) > \Pr(\mathcal{Z} \mid \lambda')$ signifie seulement que le modèle λ est mieux adapté aux observations $\mathcal{Z}$ que le modèle λ', mais ne signifie pas que $\mathcal{Z}$ est $\mathcal{O}$.

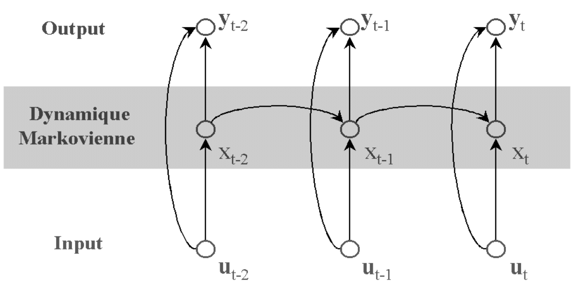

FIG. 11 – *Représentation du IOHMM.*

4.2 Modélisation d'un IOHMM

Le principe d'un IOHMM est d'aligner une séquence d'entrée à une séquence d'états discrets, et de générer une séquence de sortie. Soit $\mathbf{u}_1^T = \mathbf{u}_1 \ldots \mathbf{u}_T$ la séquence d'entrée et $\mathbf{y}_1^T = \mathbf{y}_1 \ldots \mathbf{y}_T$ la séquence de sortie associée. $\mathbf{u}$ est le vecteur d'entrée de dimension m $(\mathbf{u} \in \mathbb{R}^m)$ et $\mathbf{y}$ est le vecteur de sortie de dimension r $(\mathbf{y} \in \mathbb{R}^r)$. P est le nombre de séquences d'entrée/sortie et T est la longueur de la séquence observée. L'ensemble des séquences d'entrée/sortie est défini par $\mathcal{D} = (\mathcal{U},\mathcal{Y}) = (\mathbf{u}_1^{T_p}(p), \mathbf{y}_1^{T_p}(p))$, avec $p = 1 \ldots P$.

Un IOHMM repose sur une chaîne de n états discrets, où x_t est l'état du modèle à l'instant t (Figure 11). Chaque état j est associé à un réseau d'état $\mathcal{N}_j$ et à un réseau de sortie $\mathcal{O}_j$. Le vecteur $\mathbf{u}_t$ est l'entrée de chacun des réseaux à l'instant t (Figure 12). Un réseau d'état $\mathcal{N}_j$ a un nombre de sorties égal au nombre d'états. Chacune de ces sorties donne la probabilité de transition de l'état j vers un nouvel état.

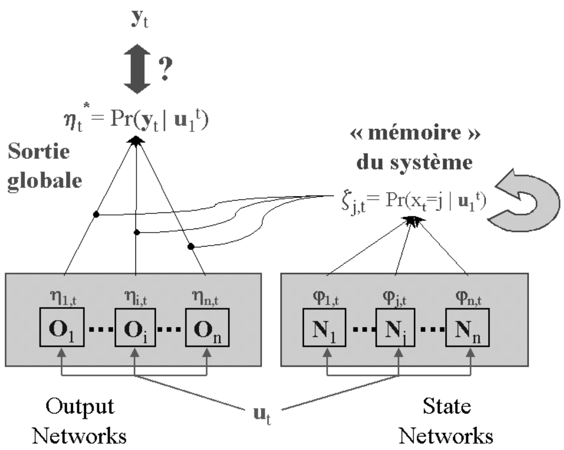

FIG. 12 – *Architecture du IOHMM*

$\boldsymbol{\theta}_j$ est l'ensemble des paramètres du réseau d'état $\mathcal{N}_j$ $(\forall j = 1 \ldots n)$, où $\boldsymbol{\varphi}_{j,t} = {}^T[\varphi_{1j,t} \ldots \varphi_{nj,t}]$ est la sortie du réseau d'état $\mathcal{N}_j$ à l'intant t, avec la relation $\varphi_{ij,t} = Pr(x_t = i \mid x_{t-1} = j, \mathbf{u}_t)$, i.e. la probabilité de transition depuis l'état j vers l'état i, avec $\sum_{i=1}^{n} \varphi_{ij,t} = 1$. $\boldsymbol{\vartheta}_j$ est l'ensemble des paramètres du réseau de sortie $\mathcal{O}_j$ $(\forall j = 1 \ldots n)$, où $\boldsymbol{\eta}_{j,t}$ est la sortie du réseau de sortie $\mathcal{O}_j$ à l'instant t, avec la relation $\eta_{ij,t} = Pr(y_{i,t} \mid x_t = j, \mathbf{u}_t)$.

A présent, nous introduisons deux variables importantes dans le modèle:

- $\boldsymbol{\zeta}_t$ représente la mémoire du système à l'instant t, $\boldsymbol{\zeta}_t \in \mathbb{R}^n$:

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

où $\zeta_{j,t} = Pr(x_t = j \mid \mathbf{u}_1^t)$ et $\boldsymbol{\zeta}_0$ est aléatoirement choisi avec la contrainte $\sum_{j=1}^{n} \zeta_{j,0} = 1$. Cette variable permet d'intégrer dans le temps toutes les probabilités de transitions.

- $\boldsymbol{\eta}_t^*$ constitue la sortie globale du système à l'instant t, $\boldsymbol{\eta}_t^* \in \mathbb{R}^r$:

$$\boldsymbol{\eta}_t^* = \sum_{j=1}^{n} \zeta_{j,t} \boldsymbol{\eta}_{j,t} \tag{10}$$

avec la relation $\boldsymbol{\eta}_t^* = Pr(\mathbf{y}_t \mid \mathbf{u}_1^t)$, i.e. la probabilité d'avoir la sortie désirée $\mathbf{y}_t$ connaissant la séquence d'entrée $\mathbf{u}_1^t$.

Il est également nécessaire de connaître la fonction de densité de probabilité (**pdf**) des sorties $f_Y(\mathbf{y}_t;\boldsymbol{\eta}_{i,t})$, i.e. la probabilité $Pr(\mathbf{y}_t \mid x_t = i,\mathbf{u}_t)$ d'avoir la sortie désirée $\mathbf{y}_t$ connaissant le vecteur d'entrée courant $\mathbf{u}_t$ et l'état courant x_t.

Nous formulons le problème d'apprentissage comme un problème de maximisation de la vraisemblance de l'ensemble des paramètres du modèle sur l'ensemble des séquences d'apprentissage. La vraisemblance des séquences d'entrée/sortie $\mathcal{D}$ (Equation 11) est, comme dans les HMM, la probabilité qu'une séquence finie d'observations ait pu être générée par le IOHMM.

$$\begin{aligned} L(\Theta,\mathcal{D}) &= Pr(\mathcal{Y} \mid \mathcal{U},\Theta) \\ &= \prod_{p=1}^{P} Pr(\mathbf{y}_1^{T_p} \mid \mathbf{u}_1^{T_p},\Theta) \end{aligned} \tag{11}$$

Θ est le vecteur de paramètres donné par la concaténation des $\{\boldsymbol{\vartheta}_j\}$ et des $\{\boldsymbol{\theta}_j\}$. Nous introduisons l'algorithme EM comme une méthode itérative pour estimer le maximum de vraisemblance.

4.3 Apprendre les IOHMM avec EM

Soit $\mathcal{X}$ l'ensemble des séquences d'états, $\mathcal{X} = (\mathbf{x}_1^{T_p}(p))$ avec $p = 1 \ldots P$. Ces derniers jouent le rôle de variables cachées. Ainsi, l'ensemble complet des données est :

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

et la vraisemblance sur $\mathcal{D}_c$ est:

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

Afin de simplifier la notation, nous avons choisi d'omettre la variable p. Les dépendances conditionnelles des variables du système nous permettent d'écrire la vraisemblance ci-dessus comme:

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

L'ensemble des séquences d'états $\mathcal{X}$ est inconnu, on ne peut donc pas maximiser $logL(\Theta,\mathcal{D}_c)$ directement.

Estimation. La fonction auxiliaire Q (Equation 3) doit être calculée:

$$\begin{aligned} Q(\Theta,\hat{\Theta}) &= E_{\mathcal{X}}[logL(\Theta,\mathcal{D}_c) \mid \mathcal{U},\mathcal{Y},\hat{\Theta}] \\ =\ & \sum_{p=1}^{P} \sum_{t=1}^{T_p} \sum_{i=1}^{n} Pr(x_t = i \mid \mathbf{u}_1^T,\hat{\Theta}) \, log\, f_Y(\mathbf{y}_t;\boldsymbol{\eta}_{i,t}) \\ +\ & \sum_{j=1}^{n} Pr(x_t = i,x_{t-1} = j \mid \mathbf{u}_1^T,\mathbf{y}_1^T,\hat{\Theta}) \, log\, \varphi_{ij,t} \end{aligned}$$

Pour chaque séquence $(\mathbf{u}_1^T,\mathbf{y}_1^T)$ et pour chaque état $j = 1 \ldots n$, on calcule $\boldsymbol{\varphi}_{j,t}$, $\boldsymbol{\eta}_{j,t}$, puis on estime la fonction Q. On cherche alors à ajuster les paramètres $\boldsymbol{\theta}_j$ du réseau d'état $\mathcal{N}_j$ et les paramètres $\boldsymbol{\vartheta}_j$ du réseau de sortie $\mathcal{O}_j$ pour maximiser $Q(\Theta,\hat{\Theta})$.

Maximisation. Les dérivées partielles $\frac{\partial Q(\Theta,\hat{\Theta})}{\partial \theta_{jk}}$ nous permettent de calculer par rétro-propagation dans le réseau d'état $\mathcal{N}_j$ les nouveaux paramètres θ_{jk} (poids du réseau d'état) qui maximisent $Q(\Theta,\hat{\Theta})$. Les dérivées partielles $\frac{\partial Q(\Theta,\hat{\Theta})}{\partial \vartheta_{ik}}$ nous permettent de calculer par rétro-propagation dans le réseau de sortie $\mathcal{O}_i$ les nouveaux paramètres ϑ_{ik} (poids du réseau de sortie) qui maximisent $Q(\Theta,\hat{\Theta})$.

4.4 Approche générative des IOHMM

Nous proposons une extension au modèle des IOHMM par une approche générative. L'idée principale est d'estimer la probabilité d'observation des entrées depuis un état donné.

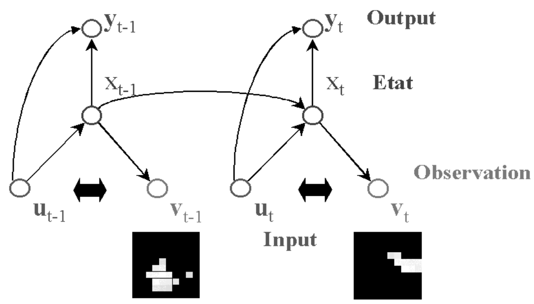

FIG. 13 – *Représentation du IOHMM génératif.*

Cette probabilité est donnée par une nouvelle sortie $\mathbf{v}_t$ (Figure 13) qui nous permet de calculer la probabilité d'observer l'entrée courante $\mathbf{u}_t$. Ainsi, nous pouvons calculer une vraisemblance L_t^V (Equation 12), appelée Vlikelihood. Le lecteur intéressé trouvera de plus amples détails sur les calculs dans [6].

$$L_t^V = \Pr(\mathbf{v}_1^t \mid \mathbf{u}_1^t,\Theta) = \sum_{i=1}^{n} Pr(\mathbf{v}_1^t, x_t = i \mid \mathbf{u}_1^t,\Theta) \quad (12)$$

4.5 Application à la reconnaissance des gestes de la main

Reconnaissance des gestes appris par le IOHMM.

Nous voulons discriminer deux classes de gestes (déictique et symbolique). Les trajectoires gestuelles (Figure 14) sont des séquences de points $[x_t, y_t]$ (coordonnées du blob de la main dans une image à l'instant t).

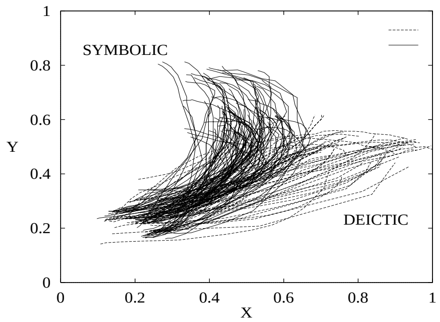

FIG. 14 – *Trajectoires gestuelles de la main.*

Par conséquent, la dimension d'entrée est $m = 2$ et la dimension de sortie est $r = 1$. Nous avons choisi d'apprendre $y_1 = 1$ comme sortie pour les gestes déictiques et $y_1 = 0$ comme sortie pour les gestes symboliques.

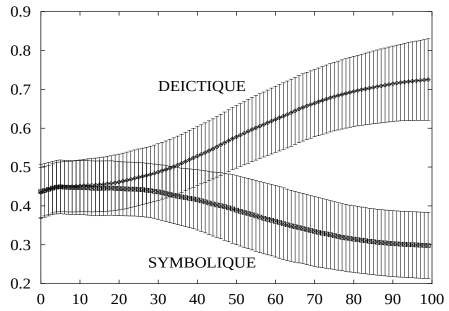

FIG. 15 – *Distribution de la sortie globale (η_t^*) du IOHMM en fonction de la durée t de la séquence.*

Pour déterminer la classe de la séquence d'entrée, on observe la sortie η_t^* (Equation 10) du IOHMM au cours du temps t (Figure 15).

Le IOHMM peut discriminer un geste déictique d'un geste symbolique après que 60% de la séquence a été présentée. Malheureusement, les gestes non-appris ne peuvent pas être classifiés par la sortie globale du IOHMM. Cependant, l'extension générative du modèle (GIOHMM) nous permet de calculer une vraisemblance pour différencier les gestes appris des autres.

Rejet des gestes non-appris par le GIOHMM.

En l'absence de gestes supplémentaires pour évaluer les performances de rejet, on peut utiliser des trajectoires de chiffres manuscrits [1] (Figure 16). Ces dernières possèdent une grande variété de formes et de mouvements.

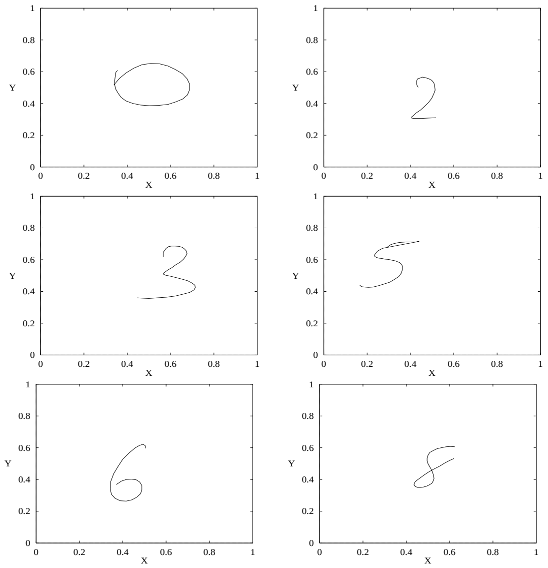

FIG. 16 – *Exemples de chiffres manuscrits.*

Pour chaque classe de chiffres, toutes les séquences sont présentées au GIOHMM ayant appris les gestes. Le GIOHMM calcule, à chaque instant t pour une séquence d'entrée $\mathbf{u}_1^t$, la Vlikelihood (Equation 12).

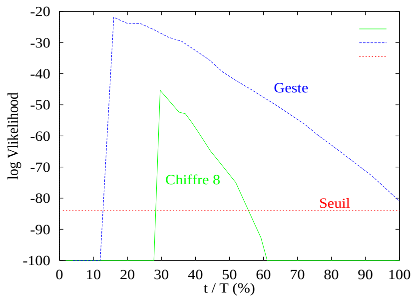

FIG. 17 – *Le* log *de la Vlikelihood d'un geste et d'un chiffre manuscrit.*

1. PenDigits, Machine Learning Databases ftp://ftp.ics.uci.edu/pub/machine-learning-databases

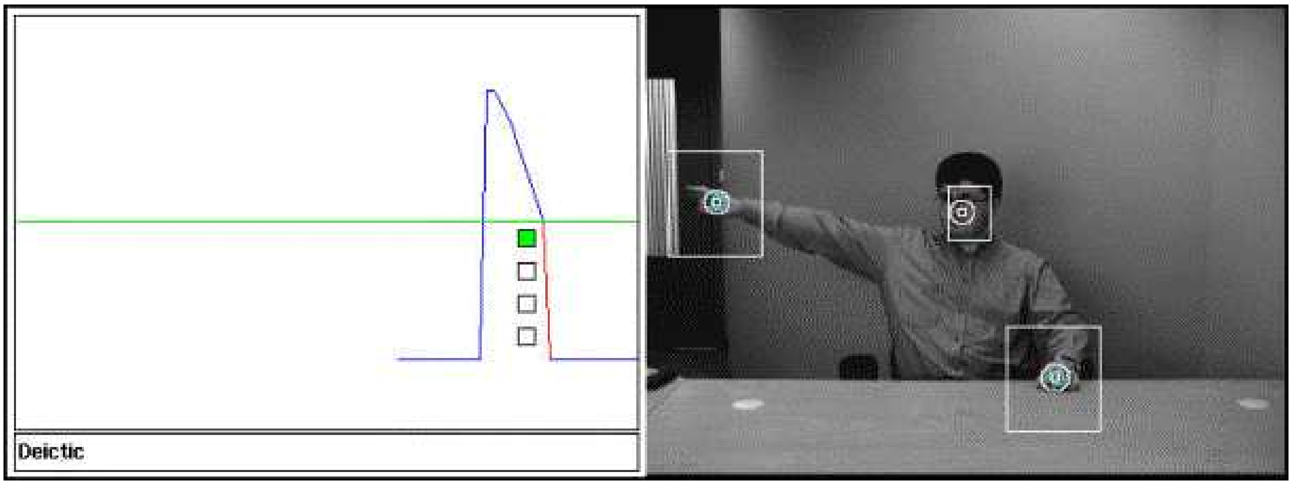

FIG. 18 – *Reconnaissance d'un geste déictique. La Vlikelihood (à gauche) indique la fin du geste et le marqueur l'identifie.*

La détection d'un geste déictique est montrée. La séquence de données issue de la trajectoire du blob de la main (Figure 18) est fournie au GIOHMM. La classe du geste est prise en compte lorsque le log de la Vlikelihood est sous le seuil.

Les séquences, dont la Vlikelihood est nulle ou dont le log de la Vlikelihood est inférieur à un seuil, sont rejetées. Le seuil est déterminé sur les trajectoires de gestes de l'ensemble de test. On remarque que le log de la Vlikelihood décroît plus lentement pour un geste que pour un chiffre manuscrit (Figure 17). Il est donc possible de discriminer une séquence observée d'une séquence inconnue.

TAB. 1 – *Taux de rejet des chiffres manuscrits par le GIOHMM à la fin de la séquence.*

0	97,3%
1	79,78%
2	87,7%
3	95,69%
4	96,56%
5	95,6%
6	87,64%
7	98,55%
8	100%
9	94,9%

Le résultat du rejet des chiffres manuscrits est présenté (Table 1). Les résultats de rejet des chiffres manuscrits sont, dans l'ensemble, très satisfaisants. On constate que les plus mauvais résultats sont obtenus pour les chiffres 1, 2 et 6. La principale cause à cela est certainement leur ressemblance avec l'une des trajectoires de geste appris. Cependant, nous ne possédons pas de résultats comparables pour pouvoir apprécier réellement ceux obtenus.

5 Conclusion

Dans la première partie de cet article, nous avons présenté un modèle probabiliste gaussien de blobs dont les paramètres sont obtenus par un algorithme EM. Les blobs sont initialisés à partir des zones de pixels de teinte chair connexes. On observe que les blobs sont géométriquement plus stables que les zones. Actuellement, les blobs sont utilisés pour la détection des visages [5] et la reconnaissance des postures de la main [7]. Cependant, les travaux futurs s'orientent vers l'analyse des gestes de la main au travers de leurs propriétés géométriques. Il nous faut donc des blobs toujours plus stables. Aussi, nous envisageons d'étudier un nouveau modèle, où le nombre de blobs n'est plus fixé a priori mais estimé dynamiquement.

La deuxième partie de cet article a pour ambition de présenter un modèle génératif neuro-markovien comme une approche très générale pour le traitement des séquences de données. Ce modèle a les propriétés des modèles de Markov cachés et l'efficacité de discrimination des réseaux de neurones. Il utilise seulement l'entrée courante et pas une fenêtre temporelle fixée a priori. Lorsque les séquences apprises sont rencontrées, la classification est possible, et lorsque des séquences non-apprises sont rencontrées, l'aspect génératif du modèle permet de les rejeter. Il a notamment été possible de différencier les gestes de la main des chiffres manuscrits.

Remerciements

Ce travail n'aurait pas été possible sans le soutien de France Telecom R&D. Je remercie tout particulièrement Daniel Collobert, Olivier Bernier et tous les membres de l'équipe de recherche DTL/DLI/TNT.

Références

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

[2] Bengio, Y. et Frasconi, P.: An Input/Output HMM architecture. *Advances in Neural Information Processing Systems*, MIT Press, Cambridge MA, vol. 7, 427-434, 1995

[3] Bengio, Y. et Simard, P. et Frasconi, P., Learning longterm dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, vol. 5(2), 157-166, 1994

[4] Dempster, A.P. et Laird, N.M. et Rubin, D.B.: Maximum-likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society*, vol. 39, 1-938, 1977

[5] Feraud, R.: PCA, Neural Networks and Estimation for Face Detection. NATO ASI, Face Recognition: from Theory to Applications (1997), 424-432

[6] Marcel, S.: Une approche générative neuro-markovienne du traitement de séquences d'images: Application à la reconnaissance statique et dynamique des gestes de la main. Thèse de Doctorat, Université de Rennes I, 4 octobre 2000

[7] Marcel, S.: Hand Posture Recognition in a Body-Face centered space. CHI'99, Extended Abstracts, 302-303, 15-20 mai, Pittsburgh PA, USA, 1999

[8] Ming-Hsuan Yang et Narenda Ahuja: Gaussian Mixture Model for Human Skin Color and Its applications in Image and Video Databases. SPIE, Conference on Storage and Retrieval for Image and Video Databases, San Jose, 1999

[9] Mozer, M., A focused back-propagation algorithm for temporal pattern recognition, *Complex Systems*, vol. 3, 349-381, 1989

[10] Pavlovic, V.I. et Sharma, R. et Huang, Thomas S.: Visual Interpretation of Hand Gestures for Human-Computer Interation: A Review. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, 677-695, 1997

[11] 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.

[12] Rumelhart, D. et Hinton, G. et Williams, R., Learning internal representations by error propagation. *Parallel Distributed Processing*, MIT Press, Cambridge, vol. 1(8), 318-362, 1986

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

[14] Williams, R. et Zipser, D., A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 270-280, vol. 1, 1989

[15] Wren, C. et Azarbayejani, A. et Darrell, T. et Pentland, A.: Pfinder: Real-Time Tracking of the Human Body. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, 780-785, 1997