Modèles de Markov Cachés (HMM) pour de la reconnaissance de gestes humains

Clément Réverdy

Stage Master Recherche

Rapport de stage

Modèles de Markov Cachés (HMM) pour de la reconnaissance de gestes humains

Auteur :
Clément Reverdy

Superviseur :
Pierre-François Marteau

Équipe : Expression

Résumé

La reconnaissance de gestes humains est l'un des nombreux défis posés dans le domaine de la vision par les ordinateurs et dans le domaine des intéractions humains-machines. Cette problématique qui a gagné en attention au cours des dernières décennies, faisant suite aux nouveaux développements technologiques (caméra 3D, capteurs EEG et EMG, etc.) et nouveaux résultats scientifiques obtenus dans les communautés du traitement du signal et de l'apprentissage automatique, entre autres. C'est un sujet qui soulève plusieurs problématiques : l'acquisition et le prétraitement des données et la détection/reconnaissance (classification) des mouvements. Dans le cadre de ce stage, nous avons comparés différents modèles de Markov cachés (HMM) dont un modèle expérimental avec fonction de densité non paramétrique, nous avons également comparé les résultats de ce modèle avec ceux obtenus via un autre type de classifieur (SVM).

Keywords : Hidden Markov Models, Motion Recognition, Kernel Method.

Mots clés : Modèles de Markov Cachés, Reconnaissance de gestes humains, Méthodes à Noyaux.

Table des matières

1 Introduction

1.1 Contexte général de la reconnaissance de gestes humains

La reconnaissance de gestes humains regroupe l'ensemble des techniques visant à capturer des informations caractérisant les mouvements d'un corps humain dans l'espace et à détecter / reconnaitre des gestes significatif d'une action, d'une intention ou caractérisant une expression spécifique. C'est un des nombreux défis posés dans le domaine de la vision par les machines ainsi que l'intéraction humains-machines et qui a gagné en attention au cours des dernières décennies avec les évolutions tant technologiques (caméra à capteur de profondeur par exemple) que les méthodes et outils développés par les communautés issues du traitement du signal et de l'apprentissage artificiel.

Les applications sont nombreuses :

- Reconnaissance automatique des langues signées : reconnaissance des phrases, stockage sous une forme symbolique compacte en vue, par exemple, de les transmettre sur un réseau ou de les synthétiser via un avatar virtuel.
- Intéractions humain-machine : conception de nouvelles interfaces par association geste / commande, réactions pertinentes de la machine vis-à-vis des actions effectuées par un humain, etc.
- Apprentissage d'activité assistée par ordinateur, connaissant une séquence de gestes efficace pour effectuer une certaine activité (musique, sport, ...), la machine peut être à-même de détecter des écarts par rapport à un geste expert ou souhaité.
- Vidéo-surveillance : détection automatique de certaines classes de gestes.
- etc.

Les problèmes posés sont variés et peuvent dépendre du dispositif de capture utilisé (vidéo 2D, MoCap, vidéo à capteur de profondeur type Kinect, gants de données...) ; détecter et extraire la forme du corps humain de son envirronnement, traiter ces informations de façon à extraire ce qui caractérise l'action humaine (extraction de descripteur), détecter / reconnaitre l'action elle-même (classification), assurer une intéraction en temps-réel.

La nature des données initiales dépendent du dispositif de capture utilisé et des pré-traitements déjà effectués sur ces dernières.

Les gants de données (dataglove) permettent de capturer des informations spatiales tridimensionnelles précises sur la main et les doigts au cours du temps. Des combinaisons de données étendent les capacités de captures au corps entier.

Les dispositifs vidéo "classiques" : les données capturées sont une suite (de longueur : T =fréquence$\times temps$) d'images 2D (frame) composées de $l \times h$ pixels suivant la résolution ; des problèmes d'occlusions apparaissent lorsqu'un élément de l'environnement se trouve entre le sujet et le dispositif.

Les dispositifs de MoCap (Motion Capture) permettent de suivre précisément les positions en trois dimensions de marqueurs au cours du temps. Ils sont peu sensibles aux problèmes d'occlusions.

La capture des mouvements d'un acteur, sur le corps duquel un certain nombre de marqueurs visuels ont été disposés, est effectuée au sein d'une salle équipée d'un certain nombre de caméras haute définition. Cette méthode de capture est très intrusive pour l'utilisateur.

Arrivés plus récemment sur le marché, les dispositifs de capture vidéo à capteur de profondeur "grand public" capturent également une suite de frames composées de $l \times h$ pixels auxquels sont associés une dimension de profondeur ; ils sont eux-aussi sensibles aux problèmes d'occlusion. Certains de ces dispositifs de capture à capteurs de profondeur tels que la kinect intègrent des outils logiciels permettant de traquer les positions (x, y, z) de différentes articulations du corps humain. Ce type de détection demeure néanmoins sujet au problème de d'occlusion et sa précision dépend de la résolution du dispositif. Les dispositifs vidéos (avec ou sans capteurs de profondeur) présentent l'intérêt de ne pas être intrusifs pour l'utilisateur.

1.2 Contexte du stage et objectifs

Au cours d'un précédent stage nous avions déjà eu l'occasion de travailler sur les DTW (Dynamic Time Warping) régularisées [24], notamment sur différentes manières (corridor, sous-échantillonnage) de réduire le temps de calcul afin d'appliquer cette méthode comme noyau d'un SVM tout en restant compatible avec du temps réel. Ce précédent stage a notamment mené à deux publications [26] [25].

Les Modèles Morkoviens à états cachés (HMM) ont fait leurs preuves en reconnaissance de la parole et ont déjà été exploités en reconnaissance de gestes humains. Notre but est d'explorer les possibilités offertes par les méthodes à noyaux au sein des HMM. Nos objectifs dans le cadre de ce stage ont donc été de développer une méthode non paramétrique à noyau, à savoir le modèle "Sommes d'exponentielles" (voir section 3.1.4), la comparer à d'autres modèles d'HMM à fonctions de densité paramétriques classiques ainsi qu'à d'autres types de classifieurs mentionnés dans l'état de l'art (notamment les SVM) afin de tester la compétitivité d'une telle méthode.

2 État de l'art

2.1 Prétraitements

Prétraiter les données consiste essentiellement à extraire un descripteur (i.e., un vecteur caractéristique). C'est à dire passer d'une représentation (données initiales) dans un certain espace à une autre représentation dans un autre espace. Cette nouvelle représentation doit satisfaire deux principales caractéristiques ; être discriminant vis-à-vis de la tâche de classification et être d'une dimension aussi réduite que possible afin de diminuer la complexité des futurs traitements.

2.1.1 Caractéristiques désirables pour un descripteur

Caractère discriminant

Le principal objectif recherché est de faciliter le travail de classification. Un descripteur idéal doit représenter les informations relatives au problème posé et uniquement ces informations. Dans un paradigme d'espace métrique, il doit maximiser les distances inter-classes (pouvoir disciminant) et minimiser les distances intra-classe (différence de style au sein d'une même classe). De nombreux descripteurs ont déjà été identifiés et proposés pour la reconnaissance de geste. La section 2.1.3 en présente un certain nombre.

Ces méthodes sont extrêmement variées et leurs niveaux de sophistication divers : covariance[16], distance entre modèles auto-régressifs[39], contours de la silhouette[22], normales[32], position relative entre chaque articulation[40], etc.

Robustesse

Le descripteur doit être robuste au bruit, aux variations d'échelles, aux translations, au déphasage temporel et aux variations de rythme. Certains descripteurs présentent des caractéristiques appréciables, il est possible de réduire le bruit en sélectionnant les principaux composants d'une transformée de Fourier[40] ou bien d'une PCA [27]. Par ailleurs, les données peuvent être traitées (centrées, normalisées) afin d'éliminer les variations liées à l'échelle, à la translation, aux tendances, etc.

Réduction de la dimension de l'espace de représentation

Une représentation compacte est aussi une caractéristique appréciée. La dimension de l'espace de représentation dépend essentiellement du descripteur, néanmoins il existe des méthodes permettant de réduire l'espace de représentation tout en préservant un maximum d'information.

Parmis ces méthodes, la PCA (construction d'un nouvel espace de représentation maximisant la variance et dont les axes sont orthogonaux) est fréquemment employée [13][27].

Les processus gaussiens à variables latentes (PGLVM) permettent d'exprimer la distribution d'un vecteur de dimension p à partir d'un ensemble de variables latentes de taille $q < p$.

Des méthodes de clusterisation (k-means, cartes auto-organisatrices, etc...) permettent également d'exprimer un descripteur dans un espace de dimension inférieur.

Il est aussi possible d'effectuer une sélections sur les variables (expl. Wanqing Li et al. [22] travaillent sur des sous-ensembles d'articulations) ou bien par sous-échantillonnage sur l'axe temporel (Marteau et al.[26] montrent qu'il est possible de réduire considérablement la fréquence d'échantillonnage sans pour autant dégrader fortement la classification par DTW régularisée).

2.1.2 Segmentation du mouvement sur l'axe temporel

Un autre problème posé est celui de la segmentation. Soit pour déterminer un début / fin de mouvement au sein d'une séquence vidéo, soit pour déterminer des sous-séquences au sein d'un même mouvement. Bashir et al. [3] et Sylvain Calinon et Aude Billard [6] (voir section : 2.2.1) ont extrait des sous-séquences en exploitant dans le premier cas des informations sur la vélocité et l'accélération au cours du temps et dans le second cas des points d'inflexion (extrémas locaux). William Chen et Shih-Fu [8] Chang ont implémenté la transformation par ondelettes discrètes (discrete wavelet transform) [23] afin de détecter des pics d'accélération.

2.1.3 Extraction de descripteur

Un sous-ensemble non-exhaustif de méthodes présentées dans cette section permet d'appréhender la diversité des descripteurs et méthodes possibles.

Modèles auto-régressifs Veeraraghavan et al.[39] ont étudié en 2004 les rôles des formes (informations spatiales) et des mouvements (variations temporelles) dans l'analyse des mouvements humains. Différents descripteurs et méthodes de classification ont été testés. Parmis ces méthodes, les auteurs ont étudié la possibilité d'utiliser une distance entre des modèles AR ou ARMA (respectivement Auto-régressif et auto-régressif + moyenne mobile) calculés sur chacune des séries temporelles. L'utilisation de ces descripteurs ne s'est pas révélée très concluante.

Séquence des articulations les plus informatives (Most Informative Joints) Dans un article publié en 2013, Ofli et al. [30] proposent un descripteur basé sur la sélection d'articulations (3D) contenant le plus d'information. Les données de base sont des séries temporelles de positions d'articulations dans l'espace capturées via un dispositif de mocap ou via une caméra à capteurs de profondeur. Les séries sont segmentées sur l'axe temporel et sur chacun des segments la quantité d'information associée à chaque articulation est calculée ; elle sont ensuite ordonnées par quantité d'information décroissante. On obtient un descripteur correspondant à l'ensemble des vecteurs d'articulations ainsi obtenus. Il est possible de réduire la taille de ce descripteur en ne gardant pour chaque segment que les K articulations les plus informatives. Ce descripteur peut ensuite être utilisé en utilisant des algorithmes de calcul de distance entre chaînes de caractères, chaque lettre correspondant à une articulation.

Matrices de covariance (Covariance matrices) Hussein et al. [16] ont proposé en 2013 un descripteur basé sur des matrices de covariance. Les données initiales sont un squelette composé de N articulations dont les positions sont données dans l'espace (x ;y ;z) évoluant dans le temps. En pratique, le descripteur proposé est la matrice de covariance de l'ensemble des coordonnées de toutes les articulations. Afin de conserver l'aspect temporel du mouvement (par exemple pour distinguer 2 classes de mouvements qui seraient l'inverse l'une de l'autre temporellement), une hiérarchie

temporelle inspirée des travaux de Lazebnik et al. [20] composée de sous-séquences temporelles est construite. Une matrice de covariance est calculée par séquence. Dans cette publication, Hussein et al. utilisent un SVM à noyau linéaire (voir section : 2.2.3) pour effectuer la classification.

Sélection de positions sur contours tridimensionnels (Bag of Points) Wanqing Li et al. [22] ont présenté en 2010 une méthode reposant sur un graphe d'action basé sur des points en 3D. Cet article étend en fait une autre publication datant de 2008 [21] proposant une méthode similaire exploitée sur des données 2D. Les données initiales sont capturées à partir d'une caméra à capteurs de profondeur (expl. : Kinect). Les contours de la silhouette humanoïde sont extraits des projections sur chacun des plans (x ;y), (x ;z) et (y ;z), puis un certain nombre de positions sont échantillonnées sur chacun des trois contours obtenus précédemment afin de réduire la complexité (au total 1% seulement des positions de chacune des "frames" est utilisé). Ce descripteur est ensuite utilisé afin d'évaluer via un modèle de mélange gaussien les postures - ou états - d'un graphe d'action (un modèle de Markov à états visibles) exploité pour la classification.

Positions relatives d'articulations, Coefficients de Fourrier, sous ensembles d'échantillons Dans une publication datant de 2012 [40], Wang et al. proposent plusieurs méthodes. Les données de base sont un squelette composé d'articulations positionnées dans l'espace. La première opération consiste à créer pour chaque articulation un vecteur évoluant au cours du temps composé de sa position relative par rapport à chacune des autres articulations ainsi qu'un "Local Occupancy Pattern" (LOP) permettant de synthétiser l'information concernant l'occupation de l'espace autour de l'articulation. Les positions relatives permettent d'obtenir de l'information concernant la configuration du squelette, l'auteur justifie l'utilisation du LOP afin de conserver l'information relative à l'intéraction entre le sujet et les objets de son environnement (par exemple si le sujet tient un verre à la main, l'espace local à l'articulation de la main sera davantage occupé). La seconde opération consiste à créer une hiérarchie temporelle inspirée des travaux de Lazebnik et al. en 2006 [20] et de récupérer sur chacun des segments temporels obtenus les principaux coefficients de la tranformée de Fourier du segment considéré. La segmentation permet de conserver l'aspect temporel tandis que la transformation de Fourier permet d'obtenir une information résistante au bruit ainsi qu'au décalage temporel entre deux séries chronologiques (seul les premiers coefficients sont utilisés). La concaténation des vecteurs ainsi obtenus est le descripteur final.

En ce qui concerne la classification (voir section : 2.2), les auteurs sélectionnent tout d'abord un ensemble d'actionlets (sous ensemble d'articulations) discriminants. Pour ce faire le pouvoir discriminant de chaque articulation est d'abord évalué en entrainant un SVM par articulation, on détermine ensuite l'ensemble des actionlets respectant un certain seuil de confiance et de non-ambiguité. Sur chacun de ces actionlets un nouveau SVM est entrainé. Une fois les actionlets sélectionnés, on entraine un dernier SVM dont le noyau constitué d'une combinaison convexe des noyaux exploités précédemment et correspondant aux différents actionlets.

Histogramme de normales 4D Omar Oreifej et Zicheng Liu [32] ont présenté en 2013 un histogramme des normales en 4D comme descripteur. Les données initiales sont capturées via une caméra à capteur de profondeur (type kinect). Les normales sont calculées dans un espace à 4 dimensions (3 dimensions spatiales + 1 dimension temporelle). Un ensemble de projecteurs sont définis, il s'agit des 120 sommets d'un polychoron (extension d'un polygone dans un espace à

4 dimensions) régulier à 600 cellules. Une optimisation est ensuite effectuée sur ces projecteurs dans le but de capturer au mieux la distribution des normales de façon discriminante. Enfin, un vecteur de 120 valeurs - l'histogramme - est obtenu en sommant pour chacun des 120 projecteurs les projections de chaque normale avec lui-même. La classification est faite par SVM.

Ce descripteur présente l'intérêt d'être une représentation relativement compacte (120 dimensions) tout en prenant en compte l'aspect temporel du mouvement.

2.2 Classification

La tâche de classification consiste selon les cas soit à déterminer si une action (caractérisée par un descripteur) appartient ou non à une classe de mouvement (détection), soit à déterminer la classe la plus probable d'une action parmis un ensemble de classes (reconnaissance).

Plusieurs revues ("survey") couvrent le sujet de la classification : Jain 2000 [17] traite des méthodes statistiques de reconnaissance de motifs (pattern) au sens large ; Ong et Ranganath 2005 [31] traite essentiellement du langage des signes ; Mitra et al. 2007 [28] traite principalement des actions impliquant les bras, les mains, le visage et la tête ; Ronald Poppe 2010 [33] dresse une vue d'ensemble de la reconnaissance de mouvements humains.

Dans le cadre du stage, nous travaillerons principalement sur les classifieurs HMM et SVM (voir respectivement : 2.2.1 et 2.2.3), plusieurs autres méthodes seront néamoins présentées dans cette section (voir : 2.2.4).

2.2.1 Modèles Morkoviens à états cachés (Hidden Markov Model - HMM)

Un HMM modélise un automate à états cachés dans lequel chaque état a une certaine probabilité de transition vers chacun des autres états ; chaque transition engendre une observation, l'observation suit une loi de probabilité associée à l'état courant. Les observations peuvent être discrètes, dans ce cas à chaque état sera associé la probabilité d'effectuer l'observation de chacun des symboles discrêts possibles ; ou elles peuvent être continues, dans ce cas on associe à chaque état une fonction de densité (souvent un modèle de mélange gaussien). Généralement on note un HMM λ sous la forme d'un triplet : $\lambda = (\mathcal{A}, \mathcal{B}, \Pi)$. $\mathcal{A}$ est la matrice de probabilité de transition de chaque état vers chaque état, $\mathcal{B}$ est l'ensemble des fonctions de probabilités des observations associées à chaque état, Π est le vecteur des probabilités d'émission initiales ($\Pi(i)$ est la probabilité d'être à l'état i à l'état initial).

L'emploi des HMM en classification est une méthode générative consistant à entraîner (algorithme de Baum-Welch) dans un premier temps un modèle par classe de mouvement puis, pour une série d'observations donnée, de déterminer quel est le modèle parmis ceux précédemment établis le plus à-même de produire cette séquence d'observations, enfin pour un modèle donné il est possible de retrouver (algorithme de Viterbi) la séquence d'états cachés la plus à-même de produire la séquence d'observations en question.

Le cas général d'un HMM est celui où l'automate des états cachés forme un graphe entièrement connexe. En reconnaissance de mouvements humains, on contraindra le modèle à un graphe où les

états transitent uniquement de gauche à droite avec le temps [28] (voir figure 1).

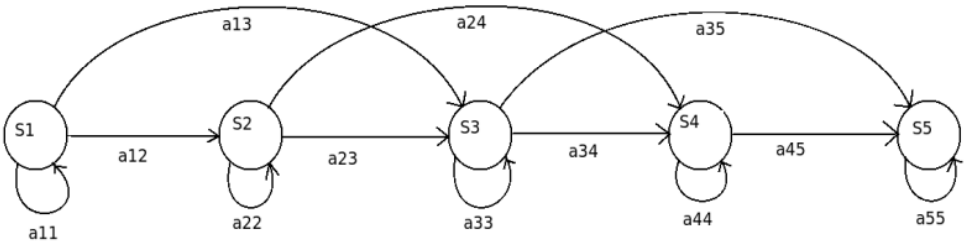

FIGURE 1 – Exemple d'HMM à cinq états de gauche à droite

L'utilisation des HMM a déjà été exploitée avec succès dans un nombre considérable de publications traitant de reconnaissance de mouvements humains.

En 1996, Starner et Pentland [37] ont exploité un HMM afin de reconnaitre en temps réels des phrases exprimées en ASL (American Sign Language). Les données de base sont issues de vidéos 2D. Des ellipses englobant la forme des mains sont extraites à chaque "frame". Un vocabulaire comprenant 6 pronoms, 9 verbes, 20 verbes et 5 adjectifs a été défini, chaque mot correspond à un état. L'algorithme de Viterbi permet de retrouver la séquence de mot la plus probable à partir d'une séquence d'observations.

Sylvain Calinon et Aude Billard [6, 2004], ont exploité des HMM pour apprendre des classes de mouvements puis les faire reconnaitre et imiter par un robot. Les données initiales sont des coordonnées ou des rotations d'articulations en 3D. Des éléments clés (extrema locaux) sont extraits de chaque série temporelle. Lors de la phase d'entraînement, à chaque élément clé est associé un état caché, à chaque état caché est associé une fonction probabiliste des observations qu'il peut générer. L'expérience a été testée avec un modèle d'observations continues et avec un modèle d'observations discrètes. Le robot est capable de détecter une classe jamais observée et d'intégrer son modèle. L'algorithme de Viterbi est utilisé afin de retrouver la meilleur séquence d'états correspondant à une observation donnée, le mouvement est alors reproduit par le robot par interpolation (cosinus pour positions angulaires, splines d'ordre 3 pour coordonnées cartésiennes) des états successifs.

Bashir et al.[3, 2005] utilisent des HMM sur des séries temporelles segmentées et transformées via une PCA. Les données initiales sont des positions (x,y) d'objets extraits d'une vidéo 2D. L'extraction de ces objets est en dehors du domaine d'étude de ce papier. Les séries temporelles sont segmentées en fonction de discontinuités visuelles dans la trajectoire (vitesse, accélération). À chaque segment est associé un ensemble coordonnées (x ;y). Tous les segments sont alors empilés en une matrice. Une PCA est effectuée sur cette matrice et un certain nombre de composantes principales sont utilisées. Afin de réduire le nombre d'états possibles, un algorithme de clusterisation spectrale (spectral clustering) utilisant l'algorithme des k-means est utilisé sur les représentations des trajectoires dans le sous-espace proposé par la PCA. La distribution des observations pour chaque état suit un mélange gaussien.

En 2010, Frédéric Bevilacqua et al. [4] ont proposé un modèle de suivi de mouvement en

temps réel pouvant s'adapter à-priori à n'importe quel type de données multi-dimensionnelles échantillonnées régulièrement. La méthode présente l'avantage de pouvoir s'appliquer à de longues séries temporelles pour lesquelles le nombre d'états cachés serait trop important pour effectuer des calculs en temps réel. La méthode repose sur le calcul en continu ("forward" procédure) de l'état le plus vraissemblable et un système de fenêtre glissante autour de cet état afin de limiter le nombre d'états possibles pris en compte.

Mitra et Acharya citent d'autres études [35][34][5][41] dans leur revue publiée en 2007 [28].

2.2.2 Déformation temporelle dynamique (DTW - Dynamic Time Warping)

DTW est un algorithme de programmation dynamique particulièrement efficace pour comparer des séries temporelles. Il établit une mesure de similarité entre des paires de séries en prenant en compte les variations de rythme au cours du temps. La distance par dtw est définie récursivement ainsi :

$$d_{dtw}(X_p, Y_q) = d_{Euclidienne}(x(p), y(q)) + Min \begin{cases} d_{dtw}(X_{p-1}, Y_q) \\ d_{dtw}(X_{p-1}, Y_{q-1}) \\ d_{dtw}(X_p, Y_{q-1}) \end{cases}$$

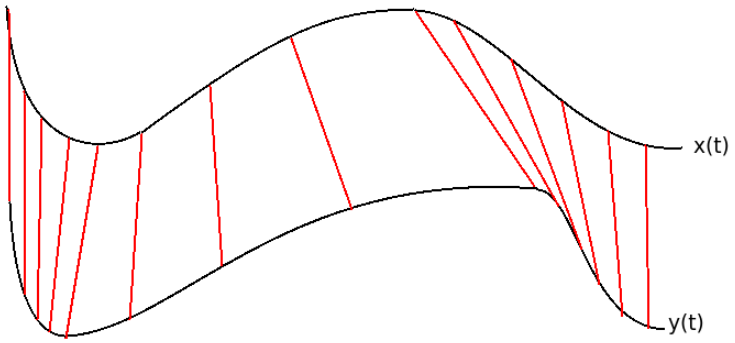

FIGURE 2 – Exemple d'alignement élastique par DTW

L'exploitation de DTW en reconnaissance de mouvement ou de geste a déjà fait l'objet de plusieurs publications [36][15][26] (il s'agit généralement d'une classification par k plus proches voisins exploitant cette mesure).

En l'état, la mesure DTW ne peut toutefois pas être utilisée pour construire un noyau défini positif ce qui peut poser problème pour l'exploiter via une machine à vecteurs supports (SVM, voir section 2.2.3). Des variantes [9] [24] ont été récemment proposées afin de régulariser cette mesure.

2.2.3 Machine à support vecteurs (SVM)

Les SVM (ou séparateurs à vastes marges) ont été introduits par Vapnik [38] et ont été fréquemment exploités en reconnaissance de mouvements humains (voir section 2.1.3). Ils reposent sur deux concepts principaux : les fonctions noyaux permettant de changer d'espace de représentation et la notion de marge maximale pour effectuer une discrimination linéaire.

Initialement le problème est celui de la recherche d'une séparation linéaire entre deux classes de mouvement (ou objet en général). Lorsqu'une telle délimitation est possible, il existe en général plusieurs solutions. Vapnik a démontré que la solution optimale est celle qui maximise la marge, c'est à dire la distance entre la frontière et les objets les plus proches de cette dernière (appelés vecteurs supports). Il existe des algorithmes connus permettant de résoudre ce type de problème.

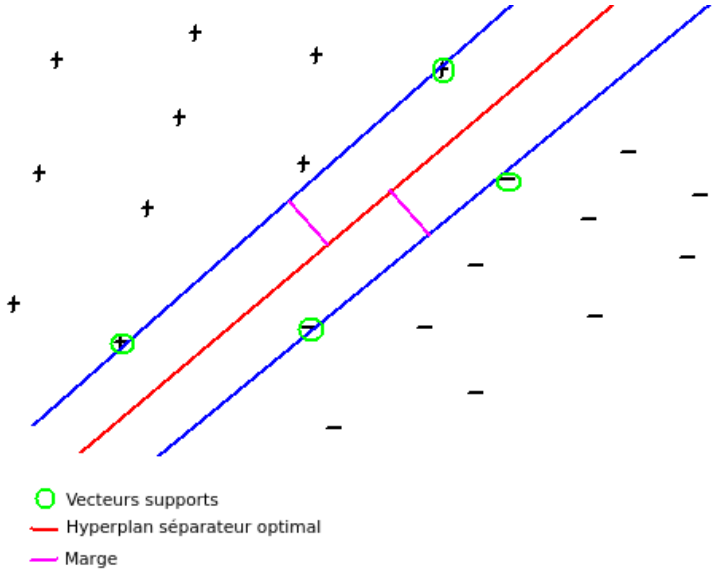

FIGURE 3 – Exemple de séparation linéaire maximisant la marge

Il n'est toutefois pas toujours possible de trouver une solution à ce problème de séparation linéaire. En pratique, on ajoutera d'une part un paramètre C pénalisant chaque objet situé du le mauvais côté de la frontière [1]. D'autre part, on pourra utiliser une fonction noyau afin de transformer l'espace de représentation initial en un espace de représentation de dimension supérieure (voir infini) dans lequel une séparation est possible [2]. Cette fonction noyau doit toutefois respecter les conditions du théorème de Mercer (elle doit être symétrique et semi-définie positive).

Il existe plusieurs noyaux particulièrement connus :

- Le noyau linéaire : $K(x, y) = x^T \cdot y$ (produit scalaire)
- Le noyau polynomial : $K(x, y) = (x \cdot y + c)^d$ [3]
- Le noyau gaussien (ou RBF) : $K(x, y) = exp(\frac{||x-y||^2}{\sigma^2})$

Plusieurs publications [9] [24] récentes ont proposé de régulariser la distance DTW afin de pouvoir exploiter ses caractéristiques d'élasticité temporelle dans un noyau défini positif associé à

1. On parle alors de marge "molle" ou "souple"
2. Ou tout du moins pour lequel les "erreurs" sont moins nombreuses
3. Où c est une constante et $d \in \mathbb{N}$ le degré du polynôme

un SVM.

Initialement les SVM sont des classifieurs binaires (ils ne permettent de classer une observation que dans l'une ou l'autre des deux classes séparées par un hyperplan séparateur comme décrit précédemment), néanmoins il existe des méthodes (non spécifiques aux SVM) pour adapter des classifieurs binaires au cas multi-classe ; notamment [2] :

- Un contre tous : un classifieur binaire est entrainé par classe (la classe en question est évaluée contre toutes les autres classes), le classifieur donnant la valeur de marge la plus importante remporte le vote.
- Un contre un : un classifieur par couple possible entre chacune des classes est entraîné, la classe la plus représentée parmis les décisions remporte le vote.

Parmis les implémentations de SVM, Chih-Chung Chang and Chih-Jen Lin [7] ont développé et mis en ligne un ensemble d'outils SVM (libsvm) très performants. Le code source est disponible (le copyright est disponible sur leur site [7]), il est possible de le modifier afin d'y implémenter des noyaux personnalisés.

2.2.4 Autres méthodes de classification

Réseaux de neurones artificiels (Artificial Neural Network - ANN)

Les ANN regroupent un ensemble de structures de calcul caractérisées par des neurones dont les entrées sont les sorties d'autres neurones, chaque connexion entre neurones est associée à un poid. Il existe différents types de structures possibles, parmis celles-ci :

Perceptrons multi-couches (multi-layer perceptrons) Les perceptrons multi-couches sont une classe de réseaux artificiels de neurones organisés en couches superposées. Il y a une couche de neurones en entrée, un couche en sortie et un ensemble de couches entre les deux. Chaque neurone reçoit un stimulus de la part de chaque neurone de la couche qui leur est immédiatement supérieur et transmet un stimulus à chaque neurone de la couche qui lui est immédiatement inférieur. Un poid est associé à chaque liaison entre deux neurones. Le stimulus transmis par un neurone à tous les neurones de la couche inférieure est fonction des stimulus pondérés reçus de la part des neurones de la couche qui lui est supérieure. L'apprentissage se fait par rétro-propagation, en cas d'erreur (ou de bon résultat), les poids sont modifiés afin de changer ou renforcer le comportement du réseau.

Les perceptrons multi-couches sont une structure d'ANN citée dans plusieurs revues ("survey") [31][28].

Réseaux de neurones récurrents Les réseaux de neurones récurrents sont des réseaux de neurones dans lequel il existe au moins un cycle. Parmis ce type de réseaux, les machines de Boltzmann restreintes peuvent être utilisées pour des problèmes de classification [19].

Arbres de décisions, forêt aléatoire

Un arbre de décision est une structure arborescente pour laquelle chaque noeud est associé à un critère de décision (une caractéristique du descripteur d'entrée qui permet de partitionner un

ensemble d'objets en sous-ensembles associés à des noeuds fils. Les feuilles de l'arbre correspondent à un critère d'arrêt de l'algorithme de construction ou d'élayage. Chaque feuille correspond alors à une prise de décision (affectation du descripteur d'entrée à une classe).

Par exemple, la classe la plus représentée parmis les échantillons d'entraînement ayant suivit cette procédure et ayant abouti à cette feuille sera considérée comme la classe du descripteur testé. Le choix des tests associés à chaque noeud est déterminant pour générer un arbre efficace et discriminant.

Par ailleurs, il est possible d'appliquer une structure floue à un arbre de décision pour permettre de déterminer des appartenances partielles à une ou plusieurs classes. On parle alors d'arbre de décision flou (Fuzzy decision tree).

Enfin, une méthode d'ensemble basée sur les arbres de décisions est appelée Random forest (Forêts aléatoires). Il s'agit de générer aléatoirement un ensemble d'arbres (par exemple en déterminant aléatoirement le test effectué à chaque noeud ou en sélectionnant pour chaque arbre un sous-ensemble de variables explicatives à exploiter). Il est possible en phase d'entraînement de sélectionner un sous-ensemble de ces arbres générés aléatoirement. La décision finale se fait en fonction de l'ensemble des décisions individuelles prises par chacun des arbres (souvent par élection).

En 2004, Fang et al. [12] ont exploité un arbre de décision flou non pas pour déterminer une classe mais un sous-ensemble de classes auxquelles l'échantillon testé pourrait appartenir et ainsi faciliter la tâche d'autres classifieurs utilisés pour affiner la décision. En 2012, Zhao et al. [42] a effectué de la reconnaissance de forme en temps réel à partir d'une forêt d'arbres de décisions, le descripteur en entrée est un vecteur de valeurs binaires (chaque valeur correspond à la comparaison d'une portion d'une image de référence avec la même portion de l'image testée : 1 si similaire, 0 sinon).

3 Travaux effectués au cours du stage

3.1 Modèles de Markov Cachés (HMM)

Au cours de ce stage, nous nous sommes principalement intéressés aux approches HMM en reprenant des modèles classiques (fonction de densité discrète, multinormale) mais aussi en réimplémentant un modèle issu d'un travail existant (fonction de densité "ACP par état" [14]) et en concevant notre propre variante innovante (fonction de densité "Somme d'exponentielles").

3.1.1 Principes de fonctionnement des HMM

L'algorithme repose sur le principe d'optimisation EM (Expectation Maximisation) en alternant une phase d'estimation des probabilités $\alpha_t(i)$, $\beta_t(i)$ et $\gamma_t(i)$ (explicitées ci-dessous) puis une phase d'optimisation des paramètres du modèle λ constitué de : l'ensemble $\mathcal{A}$ (matrice de transitions) des a_{ij}, $\mathcal{B}$ des $b_i(O)$ (collection de fonctions de densité) et Π des π_i (vecteur des probabilités initiales) jusqu'à ce que l'on considère que le modèle ait suffisamment convergé.

Quelques valeurs :

- $\lambda = (\mathcal{A}, \mathcal{B}, \Pi)$ est le triplet caractérisant un HMM.
- $\mathcal{A}$ est la matrice des probabilités a_{ij} de transiter d'un état i vers un état j.
- $\mathcal{B}$ est l'ensemble des fonctions de densité de probabilité $b_i(o)$ d'émission d'une observation o par l'état i.
- Π est la distribution des états initiaux $\pi_i = P(q_t = S_i | t = 0, \lambda)$.
- $\alpha_{kt}(i) = P(o_{k1}, o_{k2}, ..., o_{kt}, q_{kt} = S_i | \lambda)$ est la probabilité d'effectuer une série d'observations donnée jusqu'à l'instant t et d'être dans l'état i à l'instant t connaissant le modèle λ.
- $\beta_{kt}(i) = P(o_{k,t+1}, o_{k,t+2}, ..., o_{kT} | q_{kt} = S_i, \lambda)$ est la probabilité connaissant le modèle λ d'effectuer une série d'observations donnée (à partir de $t+1$) sachant que l'on est dans l'état i à l'instant t.
- $\epsilon_{kt}(i,j) = P(q_{kt} = S_i, q_{k,t+1} = S_j | O_k, \lambda)$ est la probabilité d'être dans l'état i à l'instant t et dans l'état j à l'instant $t+1$ connaissant le modèle λ et une série d'observation O_k.
- $\gamma_{kt}(i) = P(q_{kt} = S_i | O_k, \lambda)$ est la probabilité d'être dans l'état i à l'instant t connaissant le modèle λ et une série d'observations O_k jusqu'à l'instant t.

Initialisation

Il est possible d'initialiser le modèle λ aléatoirement, c'est ce que nous avons fait pour le cas discret. Dans le cas "multinormal", la méthode utilisée a été de déterminer $\mathcal{A}$ aléatoirement mais d'initialiser les paramètres μ_i et Σ_i de chaque fonction de densité $b_i(o)$ de façon "suffisamment large" pour laisser chacun des états se spécialiser lors de la convergence de l'algorithme d'apprentissage.

En ce qui concerne le modèle non-paramétrique "Sommes d'exponentielles", nous avons initialisé le modèle comme suit :

Initialisation $\mathcal{A}$ Le modèle choisi est un modèle gauche-droite, seules les valeurs $a_{i,i+1}$ ne sont pas nulles. Les probabilités de transiter d'un état i à l'état $i+1$ ont été initialisées ainsi :

$$a_{i,i} = 1 - \frac{I}{T}$$

$$a_{i,i+1} = 1 - a_{i,i}$$

Avec T le nombre moyen de "frame" par série temporelle de référence et I le nombre d'états du modèle.

Initialisation $\gamma_{kt}(i)$ Pour chaque frame t de la série temporelle k on calcule la probabilité d'être dans l'état i en ne tenant compte que de la matrice $\mathcal{A}$. Les fonctions de densités seront ensuite naturellement calculées en fonction d'$\mathcal{A}$ et des $\gamma_{kt}(i)$.

Dans l'état de l'art une segmentation sur l'axe temporel est fréquemment effectuée afin d'initialiser le modèle.

La phase "Estimation" - algorithme forward-backward

La phase "Forward" On estime dans un premier temps les probabilités à-*priori* $\alpha_{kt}(i)$:

$$Initialisation : \alpha_{k0}(i) = \pi_i b_i(o_{k0})$$

$$Propagation : \alpha_{k,t+1}(j) = \left(\sum_{i=0}^{I} \alpha_{k,t}(i) a_{ij} \right) b_j(o_{k,t+1})$$

La phase "Backward" Puis les probabilités à-postériori $\beta_{k,t}(i)$:

$$Initialisation : \beta_{kT_k}(i) = 1$$

$$Propagation : \beta_{kt}(i) = \sum_{j=0}^{I} a_{ij} b_j(o_{k,t+1}) \beta_{k,t+1}(j)$$

Le calcul des $\gamma_{kt}(i)$ Il est alors possible d'estimer les probabilités $P(q_{kt} = S_i | O_k, \lambda)$:

$$\gamma_{kt}(i) = \frac{\alpha_{kt}(i)\beta_{kt}(i)}{\sum_{i=0}^{I} \alpha_{kt}(i)\beta_{kt}(i)}$$

Le calcul des $\epsilon_{kt}(i)$ Et les probabilités $P(q_{kt} = S_i, q_{k,t+1} = S_j | O_k, \lambda)$:

$$\epsilon_{kt}(i,j) = \frac{\alpha_{kt}(i) a_{ij} b_j(o_{k,t+1}) \beta_{k,t+1}(j)}{\sum_{i=0}^{I} \sum_{j=0}^{I} \alpha_{kt}(i) a_{ij} b_j(O_{k,t+1}) \beta_{k,t+1}(j)}$$

La phase "Optimisation"

Une fois estimées les valeurs $\gamma_{kt}(i)$ et $\epsilon_{kt}(i,j)$ des K séries temporelles de référence, il est alors possible de ré-estimer les paramètres optimaux du modèle vis-à-vis de ces observations.

La distribution des états initiaux Π Les distributions des états initiaux π_i peuvent être estimées ainsi :

$$\pi_i = \frac{\sum_{k=0}^{K} \gamma_{k0}(i)}{K}$$

Néanmoins dans le cadre de cette étude le modèle est contraint à une topologie gauche-droite et la distribution des états initiaux est fixée :

$$\begin{cases} si\ i = 0\ :\ \pi_i = 1 \\ sinon\ :\ \pi_i = 0 \end{cases}$$

La matrice de transition $\mathcal{A}$ Les probabilités de transition a_{ij} peuvent être estimées ainsi :

$$a_{ij} = \frac{\sum_{k=0}^{K} \sum_{t=0}^{T_k} \epsilon_{kt}(i,j)}{\sum_{k=0}^{K} \sum_{t=0}^{T_k} \gamma_{kt}(i)}$$

Les fonctions de densité $\mathcal{B}$ Les fonctions de densité utilisées varient selon les modèles implémentés, elles sont détaillées dans les sections suivantes.

L'algorithme de Viterbi [4]

Un autre algorithme fréquemment employé dans le cadre des HMM est l'algorithme de Viterbi, il permet de calculer la succession de transition d'états la plus vraisemblable. Cet algorithme sera notamment exploité pour "affecter" des observations à des états (section : 3.1.4).

$$Initialisation :$$

$$H(i,0) = \ln(\pi_i) + \ln(b_i(o_0))$$

$$B(i,0) = 0$$

$$s_T = \arg\max_i(H_i, T)$$

4. Originalement l'algorithme de Viterbi utilise des produits de probabilités. L'algorithme présenté ici exploite la fonction logarithme à des fins purement numériques (erreurs d'arrondis). Dans le cadre du modèle "Somme d'exponentielles" nous avons seulement besoin de connaître le chemin optimal et sa probabilité qui s'exprime comme une somme de log de probabilités locales.

Propagation :

$$H(i,t) = \max_j(H(j,t-1) + \ln(a_{ij}) + \ln(b_i(o_t)))$$

$$B(i,t) = \arg\max_j(H(j,t-1) + \ln(a_{ij}) + \ln(b_i(o_t)))$$

$$s_t = B(s_{t+1}, t+1)$$

Avec $H(i,t)$ le score du meilleur chemin partiel menant à l'état i à l'instant t, $B(i,t)$ le meilleur prédécesseur de l'état i à l'instant t et s_t la trace du meilleur chemin à l'instant t.

3.1.2 Quelques contraintes topologiques

Dans le cadre de ce stage, nous avons contraint le modèle à une topologie gauche-droite sans possibilité de "sauter" un état et nous avons fixé le nombre d'états à 15.

Le modèle gauche-droite est celui employé dans chaque exemple trouvé dans l'état de l'art et convient très bien à la reconnaissance de gestes humains (un début, une fin et une succession d'états transitoires). Le fait de forcer le passage par tous les états (i.e. ne pas permettre le "saut" d'état), nous a permis de contrôler plus efficacement nos modèles durant leurs élaborations.

Enfin en ce qui concerne le nombre d'états, au cours d'une étude précédente [26], nous avions déterminé qu'une quinzaine de *frame* pouvaient suffire pour effectuer une bonne classification. Par ailleurs nous avons effectué plusieurs essais sur MSRAction afin de vérifier empiriquement qu'augmenter davantage le nombre d'états n'améliorait pas sensiblement les taux de bonne classification ; par la suite nous avons laissé ce paramètre inchangé y compris pour le jeu de données HDM05.

3.1.3 Modèles "classiques"

Les deux modèles les plus répandus dans l'exploitation des HMM en reconnaissance de geste sont :

- Le modèle "discret" avec un vocabulaire de symboles discrets associé à une probabilité d'émission de chaque symbole par état.
- Le modèle continu avec une fonction de densité multinormale par état définissant la probabilité d'émission d'une observation donnée par état.

Nous disposions dès le début du stage d'une implémentation du modèle discret (bibliothèque GRT), le modèle continu a quant à lui été implémenté dans le cadre de ce stage.

HMM "discret"

Il s'agit du modèle d'HMM le plus simple. Les données doivent être au préalable quantifiées (3.2.3) afin d'établir un vocabulaire de symbole. Chaque observation o_{kt} correspond donc à l'un des

éléments (v_0, v_1, ..., v_W) de ce vocabulaire. On peut estimer cette fonction de densité discrète de la manière suivante :

$$b_i(v_w) = \frac{\sum_{k=0}^{K}\sum_{t=0}^{T_k}\left\{\begin{array}{rl} si\ v_w = o_{kt} & : 1 \\ sinon & : 0 \end{array}\right\} \times \gamma_{kt}(i)}{\sum_{k=0}^{K}\sum_{t=0}^{T}\gamma_{kt}(i)}$$

HMM "continu, loi multinormale"

Comme nous l'avons vu précédemment, le modèle d'HMM discret n'est pas directement utilisable si nous travaillons avec des données de nature continue. Il est néanmoins possible d'utiliser dans le cadre des HMM des fonctions de densité continues. Parmi celles-ci l'une des loi les plus classiques est probablement la loi gaussienne multidimensionnelle (ou loi multinormale) :

$$b_i(o) = \frac{1}{(2\pi)^{P/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(o-\mu_i)^T\Sigma_i^{-1}(o-\mu_i)}$$

Avec P la dimension du vecteur o, $|\Sigma_i|$ le déterminant de la matrice de covariance de l'état i, Σ_i^{-1} l'inverse de la matrice de covariance de l'état i et μ_i le vecteur moyen de l'état i.

Le vecteur moyen de chaque état μ_i est estimé par :

$$\mu_i = \frac{\sum_{k=0}^{K}\sum_{t=0}^{T_k}\gamma_{kt}(i)o_{kt}}{\sum_{k=0}^{K}\sum_{t=0}^{T_k}\gamma_{kt}(i)}$$

Les éléments de la matrice de covariance de chaque état Σ_i sont quant à eux estimés de cette façon :

$$\Sigma_i(p0,p1) = \frac{\sum_{k=0}^{K}\sum_{t=0}^{T_k}\gamma_{kt}(i)(o_{kt}(p0)-\mu_i(p0))(o_{kt}(p1)-\mu_i(p1))}{\sum_{k=0}^{K}\sum_{t=0}^{T_k}\gamma_{kt}(i)}$$

Ce modèle pose néanmoins certains problèmes notamment en ce qui concerne l'inversion de la matrice de covariance Σ_i surtout lorsque cette dernière est de grande dimension. C'est pourquoi il a été nécessaire de passer par une Analyse en Composantes Principales (ACP) afin de réduire le nombre de dimensions des données initiales.

3.1.4 Modèles "Expérimentaux"

Nous avons implémenté deux modèles "expérimentaux" dans le cadre de ce stage :

HMM "ACP par état"

Cette méthode consiste créer un espace de représentation "optimal" des données sur chacun des états de l'HMM par ACP. L'objectif est double, d'une part en effectuant une ACP sur chaque état,

nous obtenons des espaces sur lesquels nos données sont décorrélées, facilitant ainsi l'inversion de la matrice de covariance. D'autre part, il parait raisonnable de penser que les facteurs caractérisant un geste évoluent au cours du temps (i.e. d'un état à l'autre dans le cadre d'un HMM) et que, de fait, nous pouvons espérer gagner en efficacité en spécialisant chacun des états.

Deux approches ont été testées. La première "naïve" consiste à calculer un espace de représentation par ACP pour chaque état à partir de la covariance estimée pour cet état, puis calculer la représentation $o_{rep,i}$ de chaque observation o dans chacun de ces espaces de représentation. La fonction de densité de chaque état est une loi multinormale dans l'espace de représentation qui lui est associé et prends en entrée les observations représentées dans ce même espace. Cette méthode n'a donné aucun résultat.

La seconde approche a été reprise d'un papier d'Horenko et al. [14], la fonction de densité utilisée est la suivante :

$$b_i(o) = \frac{1}{(2\pi)^{Q/2}|S_i|^{1/2}} e^{-\frac{1}{2}(o-\mu_i)^T T_i S_i^{-1} T_i^T (o-\mu_i)}$$

Avec S_i la matrice diagonale de dimension Q des Q premières valeurs propres du jeu de données, T_i la matrice de transformation composée des Q premiers vecteurs propres de la matrice de convariance du jeu de données sur cet état et T_i^T sa transposée.

Cette méthode n'a pas non plus donné de résultat probant (figure : 5).

Nous ingorons encore pourquoi aucune de ces deux approches n'ont fonctionné correctement. Les taux des succès obtenus (figure : 5) via la méthodes d'Horenko sont suffisamment élevés pour que la classification ne soit pas purement aléatoire. Néanmoins les deux principales hypothèses (qui demandent à être vérifiées) sont que d'une part les états sont trop spécialisés ce qui ne permet pas la prise en compte des observations mals représentées par leur espace de représentation empêchant ainsi une mise à jour efficace de ces états, d'autre part il est probable que ce modèle soit sur-paramétré vis-à-vis de la tâche de reconnaissance de gestes humains et que pour assurer son efficacité, la taille du jeu de données de référence devrait être nettement supérieure.

HMM "Somme d'exponentielles"

Cette approche novatrice que nous avons développée consiste à évaluer la probabilité d'un état i d'émettre une observation o donnée en fonction de la proximité de cette observation vis-à-vis de toutes les observations émises par cet état i. Le seul paramètre à estimer dans ce modèle est la distance moyenne $\mu_{d,i}$ entre chaque observation émise par l'état i.

Il s'agit à ma connaissance d'un modèle original qui n'apparait pas dans l'état de l'art (en tout cas en ce qui concerne la reconnaissance de geste humains). Cette méthode est une adaptation de la méthode non-paramétrique des fenêtres de parzen (Parzen Window [1]) d'estimation d'une fonction de densité. Un état de l'art plus approfondi est en cours. Nous savons que les fenêtres de parzen associées aux HMM sont utilisées en reconnaissance de la parole (Jianing Dai 1995 [11], Dai et al. 1993 [10]) ; Shang L. et Chan K.P. 2009 [18] ont également exploité un modèle d'HMM non paramétrique pour de la reconnaissance d'expression faciale.

Deux méthodes conçues sur ce principe ont été développées au cours de cette étude. La première prend en considération pour chaque état l'ensemble des observations o_{kt} pondérées par la probabilité $\gamma_{kt}(i)$ que chacune de ces observations aient été émises par l'état i. La seconde passe par une étape d'affectation basée sur l'algorithme de viterbi afin de ne prendre en considération qu'un nombre limité d'observations réduisant ainsi les temps de calcul.

1^{ere} approche : somme d'exponentielles sans affectation Considérons la fonction de densité de la loi exponentielle :

$$f(t) = \frac{1}{\mathbb{E}(X)} e^{-\frac{t}{\mathbb{E}(X)}}$$

Cette fonction semble bien adaptée pour décrire la distribution des distances entre chaque observation deux à deux. Le seul paramètre à estimer est l'espérance de la distance entre paires d'observations. Elle est définie pour des distances positives, strictement décroissante et sa pente est adaptative avec l'espérance de la distance ce qui permet d'adapter la fonction à chaque état du modèle HMM.

Pour chaque état i, nous estimerons la moyenne $\mu_{d,i}$ des distances comme étant la somme pondérée par les $\gamma_{kt}(i)$ des distances euclidiennes entre chaque paire d'observations. Les observations appartenant à une même série temporelle ne sont pas comparées entre elles car cela risquerait d'entraîner un biais dans l'estimation (les distances entre les observations successives d'une même série temporelle sont nettement plus réduites) :

$$\mu_{d,i} = \frac{\sum_{k_0=0}^{K} \sum_{t_0=0}^{T_{k_0}} \sum_{k_1=(k_0+1)}^{K} \sum_{t_1=0}^{T_{k_1}} \gamma_{k_0,t_0}(i)\gamma_{k_1,t_1}(i) dist_{eucl}(o_{k_0,t_0}, o_{k_1,t_1})}{\sum_{k_0=0}^{K} \sum_{t_0=0}^{T_{k_0}} \sum_{k_1=(k_0+1)}^{K} \sum_{t_1=0}^{T_{k_1}} \gamma_{k_0,t_0}(i)\gamma_{k_1,t_1}(i)}$$

Partant de là, la fonction de densité associée à l'état i est définie comme étant la somme pondérée par les $\gamma_{kt}(i)$ des fonctions de densité exponentielles prenant en paramètre la distance de l'observation o considérée à l'ensemble des observations de référence et comme estimation de l'espérance la moyenne $\mu_{d,i}$ définie précédemment :

$$b_i(o) = \frac{\sum_{k=0}^{K} \sum_{t=0}^{T_k} \gamma_{kt}(i) \frac{1}{\mu_{d,i}} e^{-\frac{dist_{eucl}(o,o_{kt})}{\mu_{d,i}}}}{\sum_{k=0}^{K} \sum_{t=0}^{T_k} \gamma_{kt}(i)}$$

NB : Durant la phase d'entraînement les observations appartenant à la même série temporelle que l'observation considérée ne seront pas prises en compte dans le calcul de $b_i(o_{kt})$ pour la même raison que celle évoquée précédemment à propos du calcul de $\mu_{d,i}$.

2^{nd} approche : somme d'exponentielles avec affectation via viterbi La méthode évoquée précédemment a donné des résultats intéressants, néanmoins cette dernière est coûteuse en temps de calcul puisqu'elle nécessite d'effectuer des calculs d'exponentielles sur les distances entre chaque paire d'observations disponibles en référence.

Une solution approchée consiste à affecter chaque observation de chaque série temporelle à un et un seul état i à l'aide de l'algorithme de Viterbi (3.1.1). Cette affectation est soumise à deux contraintes : la première observation de chaque série temporelle est affectée au premier état, la dernière observation au dernier état.

Les calculs sont effectués identiquement à ceux de la "Somme d'exponentielle sans affectation" à ceci près que seules les observations affectées à l'état considéré sont prises en compte, les sommes de fonctions exponentielles sont quant-à-elles toujours pondérées par les $\gamma_{kt}(i)$.

3.2 Expériences et résultats

3.2.1 Architecture du projet et outils utilisés

Le code a été entièrement écrit en C++. L'espace de travail (figure : 4) a été configuré via l'outil cmake [5].

L'ensemble des bibliothèques externes ont été placées dans un même dossier *thirdparty*, il s'agit des bibliothèques suivantes :

- La librairie GRT (Gesture Recognition Toolkit [6]) utilisée principalement pour son implémentation d'HMM discrets et son quantificateur par K-means. Elle fournit néanmoins un panel d'autres outils très intéressants (filtres, prétraitements divers, algo de classification, clusterisation, ...).
- Eigen [7] : une library "template" pour de l'algèbre linéaire, surtout utilisée dans ce programme pour gérer les matrices mais aussi pour résoudre certain problèmes (décomposition en valeurs propres / vecteurs propres, inversion de matrices...).
- BOOST [8] qui fournit un panel de fonctions utilitaires en C++. Un certains nombre de ces fonctions ont d'ailleurs été intégrée dans la bibliothèque standard C++11.

Les autres composants du code du code sont répartis en "packages" :

- *dataStruct* : contient plusieurs classes qui me permettent de gérer mes collections de séries temporelles nettement plus simplement et efficacement que celles mises en oeuvre dans la GRT.
- *parsers* : contient l'ensemble des parsers permettant de charger les différents formats de fichiers selon les bases de données rencontrées (msrc12, hdm05, msraction). Il contient également des méthodes permettant de passer du format de données de la GRT à mon propre format et vis versa.
- *pre* : contient exclusivement des fonctions appliquées en prétraitements.
- *works* : contient l'ensemble des procédures utilisées pour l'obtention des résultats.
- *utils* : contient plusieurs ensembles de fonctions usuelles regroupées par espaces de nommages.

5. http ://www.cmake.org/
6. http ://www.nickgillian.com/software/grt
7. http ://eigen.tuxfamily.org/
8. http ://www.boost.org/

La partie "expérimentale" proprement dite du projet est contenue dans deux classes isolées (le fichier *test.cpp* fait simplement office de "*main*") : $CustomHiddenMarkovModel$ et $CustomHmm$. Ces deux classes sont initialement issues de la GRT ($HiddenMarkovModel$ et HMM) néanmoins le code d'origine (sauf le code relatif au modèle HMM discret) a été largement retravaillé et complété afin de développer les différents autres modèles.

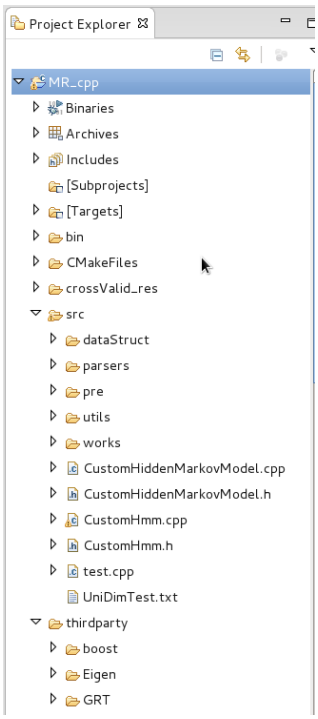

FIGURE 4 – Arbre hiérarchique du projet

3.2.2 Jeux de données

Nos tests ont été effectués sur deux jeux de données reconnus par la communauté. Le choix a été fait d'exploiter uniquement les positions tridimensionnelles de chacune des articulations en tant que descripteur pour chacun des modèles testés afin de pouvoir comparer leurs efficacités respectives. Néanmoins les premiers tests ont été effectués (voir section 3.2.4) avec un autre type de descripteur afin de pouvoir se comparer à l'état de l'art.

HDM05 HDM05 [9] est une base de données capturée par Motion Capture (des données peu bruitées). À l'origine cette base de données contient les rotations sur les axes (x ;y ;z) de 31 articulations au cours du temps au format ASF/AMC. La féquence d'échantillonnage initial est de 240 images par seconde Les positions de ces articulations ont été extraites à l'aide d'un outil interne au laboratoire.

9. http ://resources.mpi-inf.mpg.de/HDM05/

Nous considérerons sur cette base de données 11 classes (depositFloorR, elbowToKnee3RepsLelbowStart, grabHighR, hopBothLegs3hops, jogLeftCircle4StepsRstart, kickRFront1Reps, lieDownFloor, rotateArmsBothBackward3Reps, sneak4StepsRStart, squat1Reps, throwBasketball) similairement aux publications d'Hussein et al. 2013[16] et Ofli et al. 2012[29].

Chaque classe de mouvement a été exécutée environ 4 à 5 fois par chacun des 5 acteurs (ce qui constitue 249 séries temporelles au total).

MSRAction3D MSRAction [10] est une base de données capturée par une caméra à capteur de profondeur (données davantage bruitées). Cette base de données contient les coordonnées (x,y,z) des positions de 20 articulations au cours du temps. La fréquence d'échantillonnage est de 15 images par seconde.

Nous considérerons dans cette base de données les 20 classes de mouvement, chaque classe de mouvement a été exécuté 2 ou 3 fois par chacun des 10 acteurs (567 séries temporelles au total).

3.2.3 Prétraitements

Plusieurs prétraitements ont été effectués sur les données initiales :

- La position de l'articulation "root" pour hdm05 / "shoulderCenter" pour MSR à la première frame a été soustraite des positions de toutes les articulations de chaque frame afin de gommer les différences liées à la position de départ des acteurs.
- Les données ont été normalisées entre 0 et 1 sur chaque variable de chaque série temporelle $newValue = (value - minValue)/(maxValue - minValue)$.
- En ce qui concerne l'exploitation des HMM "discret", les données ont été discrétisée via l'algorithme des K-means déjà implémenté dans la GRT.
- En ce qui concerne l'exploitation des HMM "continu, loi multinormale", la dimension des données a été réduite à 10 par PCA (extraction des valeurs / vecteurs propres via la librairie Eigen) principalement afin de faciliter l'inversion de la matrice de covariance.
- Afin de limiter les temps de calcul pour l'exploitation de certains algorithmes, le jeu de données a été sous-échantillonné linéairement afin de ne garder qu'un maximum de 60 frames par série temporelle (des tests ont été effectués sur le jeu de données non sous-échantillonné afin d'observer l'influence du sous échantillonnement).

Quel que soit le jeu de données (MSR ou HDM), les séries temporelles utilisées en référence ne partagent aucun acteur en commun avec celles utilisées en test (le but étant de passer outre les différences de style entre acteurs). Dans la mesure du possible le choix a été fait d'effectuer des cross-validations sur l'ensemble des combinaisons d'acteurs possibles. Nous avons fait varier le nombre d'acteurs utilisés en référence afin de tester l'influence du nombre de séries temporelles utilisées pour entraîner les modèles sur la qualité de ces derniers :

- 252 combinaisons possibles en prenant 5 acteurs parmi 10 (5c10) sur MSR
- 45 combinaisons possibles en prenant 8 acteurs parmi 10 (8c10) sur MSR
- 8 combinaisons possibles en prenant 3 acteurs parmi 5 (3c5) sur HDM05
- 5 combinaisons possibles en prenant 4 acteurs parmi 5 (4c5) sur HDM05

10. http ://research.microsoft.com/en-us/um/people/zliu/ActionRecoRsrc/

– etc (MSR 1c10, 2c10, 3c10, 4c10 / HDM 1c5, 2c5).

Néanmoins plusieurs cas font exception à la règle :

– HMM continu, loi multinormale : les cas où l'entraînement a échoué (inversion de matrice impossible) ont été retirés (12 cas sur MSRAction 5c10, 2 cas sur hdm 3c5, 1 cas sur hdm 4c5). Il en résulte que les taux de succès moyens sont un peu sur-évalués (les cas défaillants sont en général ceux donnant de moins bon résultats sur les autres modèles).
– HMM somme d'exponentielles sans affectation : pour des raisons de temps de calcul, les tests n'ont été effectués que sur 15 combinaisons d'acteurs sélectionnées aléatoirement sur les 252 possibles avec MSRAction 5c10 et 45 possibles avec MSRAction 8c10 ce qui entraîne une variabilité plus importante du taux de succès moyen.
– HMM somme d'exponentielles avec affectation via viterbi + descripteur positions relatives (voir : 3.2.4) : l'intégralité des calculs est en cours de traitement.

Enfin, les modèles "fonction de densité continue, loi multinormale" et "discret" ont nécessité chacun un prétraitement spécifique :

– Pour le modèle d'HMM discret, les données ont été discrétisées via l'algorithme des K-means implémenté dans la GRT. La taille du vocabulaire a été déterminée empiriquement en augmentant le nombre de symboles jusqu'à ce que le gain au niveau du taux de succès soit mineur ce qui donne un vocabulaire de 75 symboles sur MSRAction et de 150 symboles sur HDM05 (sous échantillonné).
– Concernant le modèle d'HMM avec fonction de densité multinormale, la dimension des données a été au préalable réduite par ACP. Le nombre de dimensions a été détérminé empiriquement en fonction du nombre d'échecs lors de l'entraînement des modèles et du taux de succès [11]. Le nombre de dimensions retenu est 10.

3.2.4 Résultats

Au cours de travaux antérieurs, nous avons réimplémenté la solution proposée par Hussein et al.[16] nous avons notamment effectué une cross-validation sur MSRAction sur les 20 classes du jeu de données (contre 8 dans la publication en question). Les résultats présentés dans cet article nous serviront de point de comparaison en ce qui concerne la base de données HDM05, en ce qui concerne MSRAction, nous reprendrons les résultats que nous avions obtenu en réimplémentant cette méthode et en la testant sur les 20 classes (contre 8 dans le papier) et sur l'ensemble des 252 combinaisons d'acteurs possibles (contre seulement 1 dans le papier en question).

Par ailleurs au cours de ce stage nous avons principalement travaillé sur les HMM en tant que classifieur et non sur les descripteurs en entrée de ce classifieur. Or la reconnaissance de gestes à

11. Bien qu'un nombre extrêmement réduis d'axes concentrent l'essentiel de l'information, il faut garder à l'esprit que la position des articulations est régie par la hiérarchie du squelette si bien que la position de chaque articulation est extrêment dépendante de celles de ses articulations parentes. Il est donc nécessaire de conserver un nombre supérieur d'axes. Cela aurait pu être arrangé en prenant en descripteur la position relative de chaque articulation par rapport à l'articulation parente, néanmoins le objectif du stage était de comparer différents modèles de classifieurs d'où la nécessité de conserver l'unicité du descripteur quel que soit le modèle

partir d'un "squelette" revêt 2 principaux aspects :

- La configuration spatiale des articulations du squelette les unes part rapport aux autres (Matrices de covariance chez Hussein et al.).
- L'évolution de cette configuration au cours du temps (Hiérarchie temporelle pyramidale ches Hussein et al.).

Nous avons effectué des tests préliminaires en prenant en entrée les positions relatives des articulations les unes par rapport aux autres (Wang et al. [40]) soit un vecteur pour chaque frame de longueur : $(3\ (x; y; z)\ \times 20\ (articulations)\ \times\ 19\ (autres\ articulations))/2 = 570$ sur MSRAction, 1395 sur HDM. Ces tests ont été effectués sur 30 combinaisons de 5 acteurs parmi 10 choisies aléatoirement en exploitant le modèle "HMM Somme d'exponentielles avec affectations via Viterbi". Ces premiers résultats sont présentés en figure 5 (dist_viterbi, pos. rel.).

Enfin, les temps de calculs [12] sont compatibles avec du temps réel (< 1s / série temporelle à classer) pour chaque modèle d'HMM sauf le modèle d'HMM "Somme d'exponentielles sans affectation".

Méthode	MSRAction 5c10	MSRAction 8c10	HDM05_sous_ech 3c5	HDM05_sous_ech 4c5
dist_viterbi	65.21 % (66.28%)	66.85 %	74.60 % (81.65%)	75.75 %
dist	65.60 % (68.58%)	70.48 %	64.11 % (68.80%)	63.19 %
discret	58.84 % (61.30%)	62.96 %	69.80 % (72.48%)	72.77 %
densité multinormale	54.51 (60.91%) %	67.08 %	64.98 % (61.46%)	78.42 %
"PCA par état"	32.38 % (34.86%)	42.22 %	49.06 % (60.55%)	51.81 %
Hussein et al.[16]	72,33 % (71,27%)	à tester	(95.41 %)	à tester
Wang et al.[40]	88 %	?	?	?
Ofli et al.[29]	(33.33)* %	?	(84.40 %)	?
dist_viterbi, pos. rel.	74,30 % (77.01%)	à tester	88.70 (93.58 %)	à tester

FIGURE 5 – Taux de bonnes classifications moyens obtenus selon les différentes méthodes. Entre parenthèses le résultat obtenu pour - la - combinaison d'acteurs utilisée par Hussein et al.[16].. En ce qui concerne Ofli, nous ignorons quels acteurs ont été sélectionnés sur MSRAction (5 en entraînement, 3 en test).

Concrètement si l'on s'en réfère aux résultats (figure : 5) nos modèles par "somme d'exponentielles" surpassent les autres modèles d'HMM (figure : 7). Par ailleurs, avec un vecteur descripteur adéquat, le modèle élaboré semble donner des résultats concurenciels vis-à-vis du classifieur SVM. Les résultats sont encore améliorables [13], ils démontrent la viabilité du modèle par somme d'exponentielles avec affectation.

12. 4 coeurs × 2.00 GHz, la plupart du temps de multiples tâches (parfois > 4) lancées simultanément et exploitant chacune 100% du temps de l'un des coeurs.

13. Wang et al. exploitent un certain nombre d'autres traitements (local occupancy Pattern, Fourier Temporal Pyramid, ...)

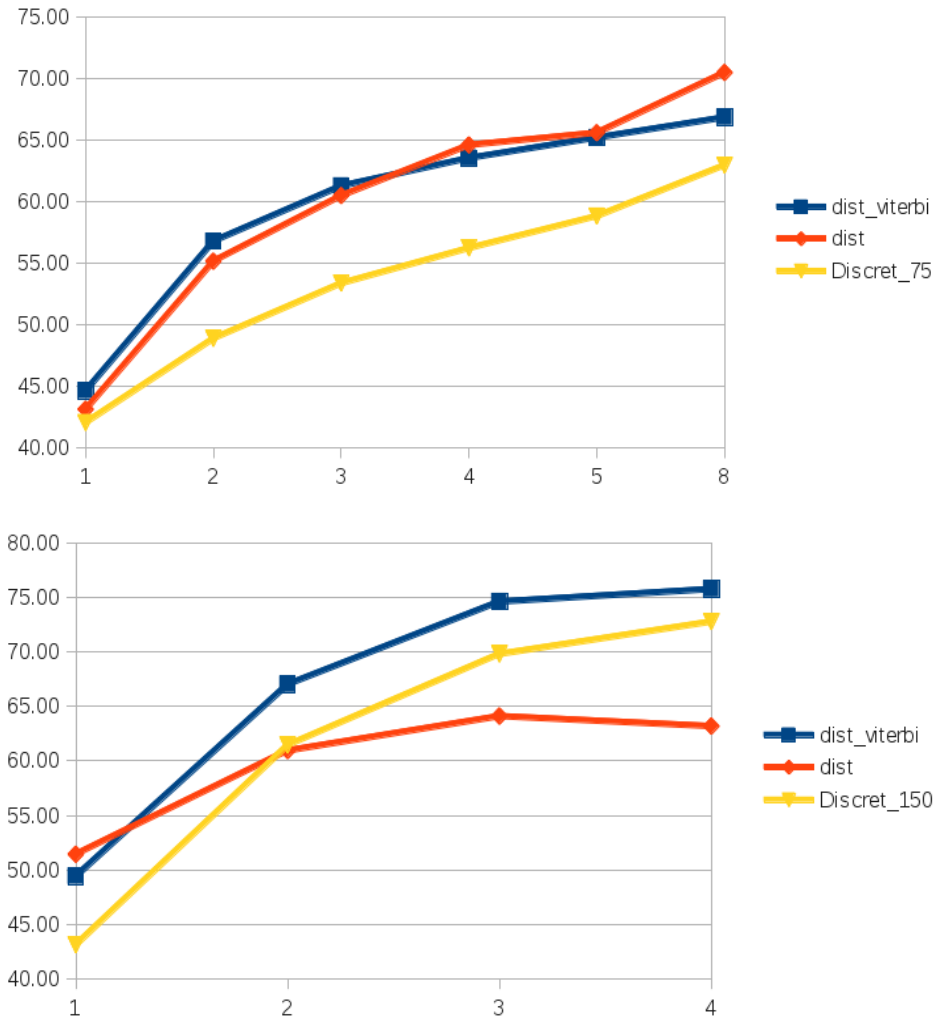

FIGURE 6 – Taux de succès moyen en fonction du nombre de série temporelles de référence, MSR en haut, HDM en bas.

Influence du nombre de séries temporelles de référence

Nous avons fait varier le nombre de séries temporelles utilisées lors de la phase d'entraînement des modèles en modifiant le nombre d'acteurs pris en référence (figure : 6).

Les résultats obtenus via les modèles "sommes d'exponentielles" donnent de meilleurs résultats pour un jeu de donnée de référence de même dimension. Par ailleurs ces modèles semblent converger plus rapidement vers un optimum.

Modèle HMM "somme d'exponentielles" avec ou sans affectation

Si l'on compare les résultats obtenus (figure : 6) via nos modèles d'HMM "sommes d'exponentielles" avec ou sans affectation, on remarque que le fait de limiter le nombre de comparaisons sur

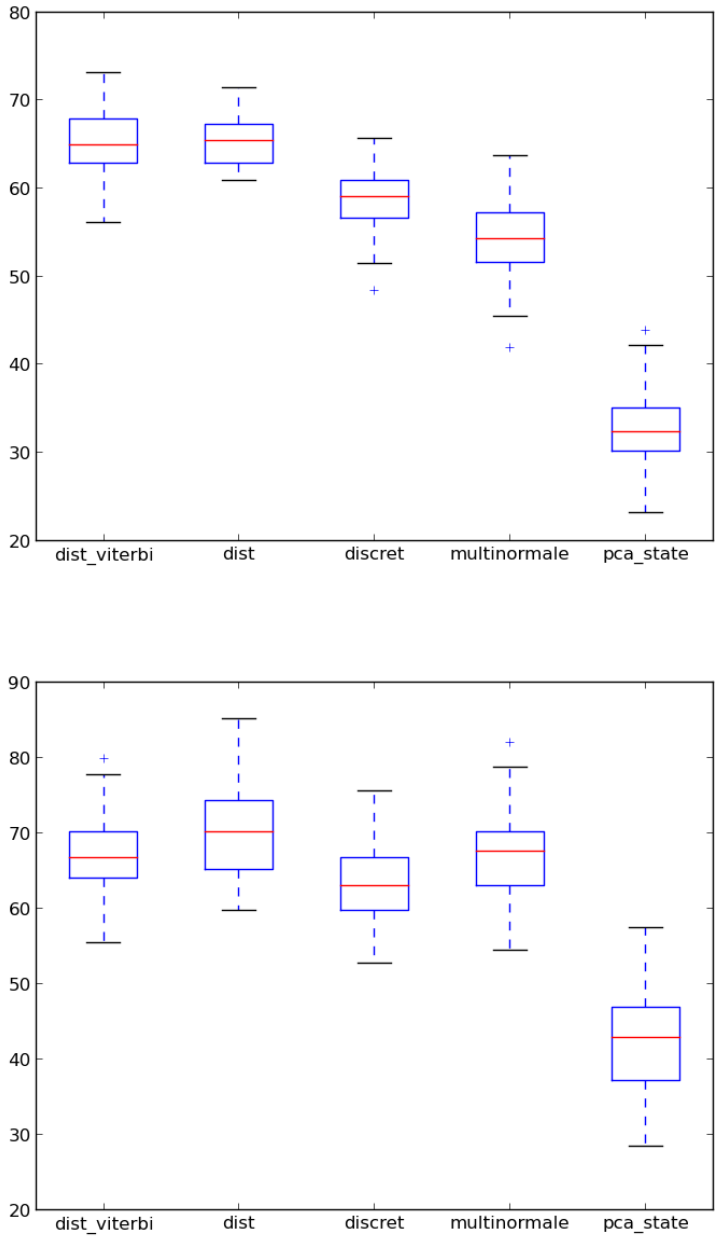

FIGURE 7 – Taux de succès moyen en fonction des modèles d'HMM sur MSRAction, 5c10 en haut, 8c10 en bas.

chaque état, en plus de réduire les temps de calcul, ne détériore pas les résultats voir les améliore lorsque les données sont peu bruitées (HDM05).

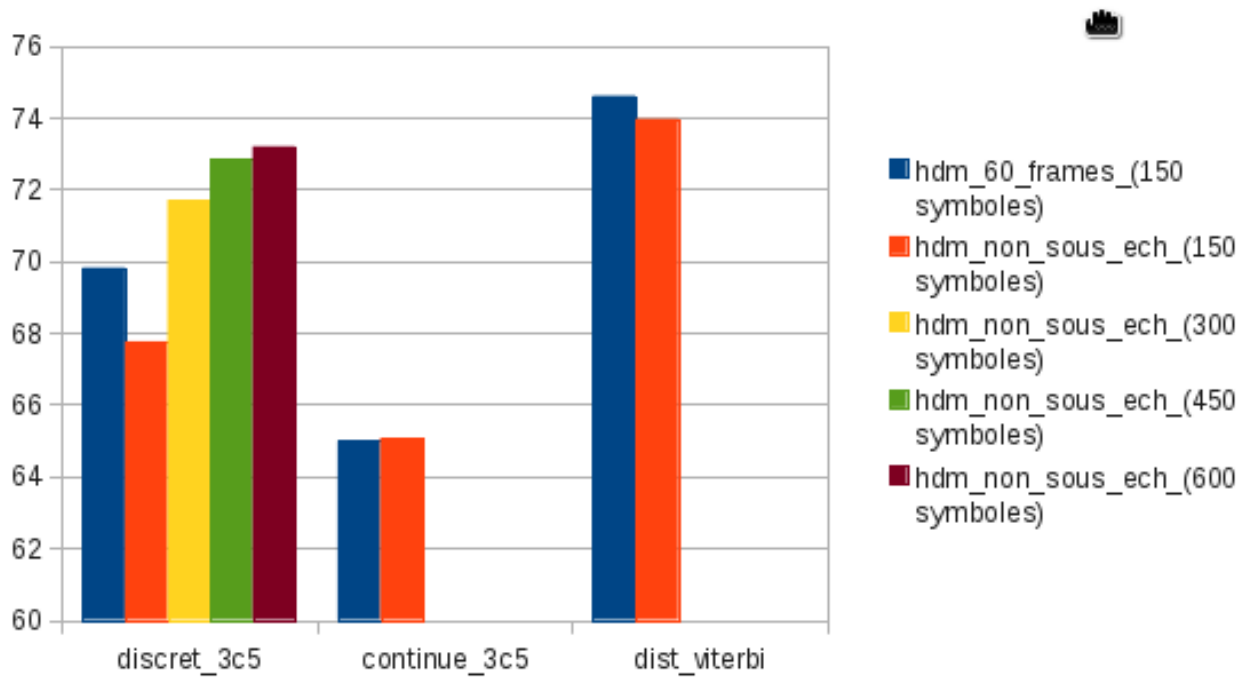

FIGURE 8 – Taux de succès moyen en fonction du sous-échantillonnage ou non d'HDM (nombre de symboles du vocabulaire variant pour le modèle discret)

HDM05, influence du sous échantillonnage

On sait([26]) que l'information d'une frame sur l'autre est suffisamment redondante pour qu'il soit possible de sous-échantillonner une série temporelle sans dégrader déraisonnablement les résultats. L'objectif initiale de ce traitement effectué sur HDM05 était de réduire les temps de calcul (notamment pour que le modèle avec "sommes d'exponentielles" soit compatible avec du temps réel). Néanmoins nous nous sommes rendu compte (figure : 8) que les taux de bonne classification s'en trouvaient légèrement améliorés.

Des tests sont en cours afin d'analyser plus en détails le phénomène, sur un modèle discret avec 75 symboles (150 habituellement sur hdm), le taux de succès passe de 67.0% en sous échantillonné 60 frames à 61.1%. La première hypothèse retenue est que le fait de ramener chaque série temporelle au même nombre de frame a pour effet de "normaliser la matrice de transitions $\mathcal{A}$ du modèle". C'est à dire que si les séries temporelles d'une même classe n'ont pas la même longueur, le risque est que les probabilité de transition de chaque état vers l'état suivant ne soit pas comparable d'une série temporelle à l'autre (par exemple une série temporelle plus lente aura des probabilité de transiter vers l'état suivant globalement plus faibles ce qui posera problème si l'on cherche à estimer un modèle à partir des cette série et d'autres plus rapides même si elles appartiennent à la même classe).

Un second élément entre en compte pour le cas discret ; le fait d'augmenter le nombre d'échantillons en ne sous échantillonnant pas hdm permet à l'algorithme des k-means d'atteindre moins rapidement sa saturation en ce qui concerne la taille du vocabulaire. Aussi est-il possible d'augmenter le nombre

de symboles voulus, d'améliorer les résultats et donc de compenser l'effet constaté précédemment. On suppose que c'est également l'effet observé en ce qui concerne le modèle avec loi multinormale, un nombre d'échantillon supérieur permet d'estimer plus efficacement les paramètres du modèle. Le modèle avec somme d'exponentielles et affectation via Viterbi est quant-à-lui très peu paramétré ce qui explique un effet moindre.

Matrice de confusion obtenue par la méthode "somme d'exponentielles avec affectation viterbi"
pour le jeu de données MSRAction et la combinaison d'acteurs testée par Hussein et al. :

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0.13	0	0.86	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0.06	0.06	0.86	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0.93	0	0	0	0	0.06	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0.25	0	0	0	0	0	0	0.75	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0.33	0	0.06	0	0	0	0.53	0	0	0	0	0	0	0	0.06	0	0	0
10	0.06	0	0	0	0	0	0	0	0	0.80	0	0	0	0	0	0	0	0	0	0.13
11	0	0	0	0	0	0	0	0	0	0.20	0.66	0	0.06	0	0	0	0	0	0.06	0
12	0	0	0	0	0.13	0	0	0	0	0	0	0.20	0.06	0.59	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0.26	0.73	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0.07	0.23	0.53	0	0.07	0	0	0	0.07
15	0	0	0.27	0	0	0	0	0	0	0	0	0.09	0	0	0.63	0	0	0	0	0
16	0	0	0.18	0.09	0	0	0	0	0	0	0	0.36	0.09	0	0	0.27	0	0	0	0
17	0	0	0	0	0	0	0	0	0.08	0	0	0.08	0	0.08	0.16	0.08	0.41	0	0	0.08
18	0	0	0.25	0	0	0	0	0	0	0.16	0	0.25	0.08	0.08	0	0	0	0.08	0	0.08
19	0	0	0.08	0	0	0	0	0	0	0.08	0	0.16	0.08	0	0	0	0	0	0.33	0.25
20	0.08	0	0	0	0	0	0	0	0	0.25	0	0.08	0	0	0	0	0	0	0.08	0.5

Effectifs :

train:	15	15	15	14	15	15	14	15	15	15	15	15	12	14	9	15	15	15	15	13
test:	12	12	12	12	11	11	13	15	15	15	15	15	8	15	11	15	15	15	15	9

Acteurs : 5_3_1_9_7, success rate : 66.2835%

Matrice de confusion obtenue par la méthode "somme d'exponentielles avec affectation viterbi"
pour le jeu de données HDM05 et la combinaison d'acteurs testée par Hussein et al. :

	1	2	3	4	5	6	7	8	9	10	11
1	0.75	0	0	0	0	0	0	0	0	0.25	0
2	0	1	0	0	0	0	0	0	0	0	0
3	0.08	0	0.66	0	0	0.08	0	0.08	0	0.08	0
4	0	0	0	0.33	0.33	0	0	0	0.16	0	0.16
5	0	0	0	0	0.62	0	0	0	0.37	0	0
6	0	0	0	0	0	0.91	0	0	0	0	0.08
7	0	0	0	0	0.10	0	0.90	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0
9	0	0	0	0	0.14	0.14	0	0	0.57	0	0.14
10	0	0	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0.16	0	0	0	0	0.83

Effectifs :

train:	20	7	17	6	9	18	10	10	9	28	8
test:	12	6	12	6	8	12	10	6	7	24	6

Acteurs : tr_mm_bd, success rate : 81.6514%

FIGURE 9 – Matrices de confusion données par la méthode "Somme d'exponentielles avec affectations via Viterbi" pour les combinaisons d'acteurs testées par Hussein et al., MSRAction en haut, HDM en bas.

Enfin, une analyse détaillée des matrices de confusions (figure : 9) obtenues montrent des disparités sensibles entre les taux de succès selon les différentes classes de mouvement. Cela suggère soit que les données pour ces classes sont de mauvaise qualité, soit qu'un classifieur HMM n'est pas le plus adapté en ce qui concerne ces classes en particulier. Il pourrait donc être intéressant de s'orienter vers une démarche de fusion de classifieurs (forêts d'arbres aléatoires de décision, ...) afin de tirer un parti optimal des qualités respectives de chaque type de classifieurs (SVM, HMM, réseaux de neurones, ...).

3.3 Travaux en cours et ouverture

Le stage ne se terminera qu'à la fin du mois de Juin et plusieurs travaux sont actuellement en cours :

Vecteur Descripteur : Nous n'avons jusqu'à maintenant pas - ou très peu - travaillé sur le vecteur descripteur passé en entré du classifieur. L'exemple donné en section 3.2.4 est lourd et loin d'être optimum par rapport à ce qu'il est possible de faire. Par ailleurs, il convient d'effectuer des tests sur les autres modèles d'HMM afin de pouvoir s'y référer. Il ne s'agit que d'un exemple préliminaire permettant de montrer que notre modèle peut être une alternative viable à d'autres types de classifieurs.

Variantes d'affectations : L'algorithme de viterbi est la méthode d'affectation qui nous a paru la plus simple et pertinente mais d'autres techniques sont envisageables. Par exemple, nous testerons prochainement l'effet de l'affectation d'une même observation à plusieurs états différents (les état adjacents à celui qui a été désigné par Viterbi). L'algorithme du K-D tree a également été envisagé (reste à réfléchir au moyen de l'adapter au problème).

Etat de l'art : Un état de l'art plus approfondi de l'emploi des fenêtres de Parzen et autres méthodes non-paramétrique comme fonction de densité d'un HMM reste encore à faire et pourrait mener à d'autres idées innovantes.

Modèles à fonction de densité continues : Nous avons passé beaucoups plus de temps que prévu sur le modèle à fonction de densité multinormale à cause de son instabilité. Si bien que le modèle de mixture gaussiennes n'a pas encore été implémenté (il le sera peut-être à titre de référence au cours des travaux à venir mais n'est pas la première des priorités).

4 Conclusion

Au cours de ce stage, nous avons comparé plusieurs types d'HMM (fonctions de densité discrète, multinormale, "ACP par état", non-paramétrique "Somme d'exponentielles") sur deux jeux de données reconnus par la communauté [14]. Nous avons montré que la méthode à noyau non-paramétrique (Somme d'exponentielles) développée au cours de ce stage est plus efficace que les autres méthodes HMM testées, nécessite un jeu de données de taille moins grande pour être efficace et est compatible avec du temps réel [15] si l'on réduit intelligemment l'espace des comparaisons [16].

Par ailleurs, nous avons montré qu'en employant un bon descripteur [17], notre modèle d'HMM "Somme d'exponentielles avec affectation via Viterbi" est compétitif vis-à-vis des autres types de classifieurs (principalement SVM) que l'on peut trouver dans l'état de l'art

Il reste encore des travaux à effectuer avant la fin de ce stage, néanmoins la preuve de concept nous semble acquise ; ces premiers résultats déjà très compétitifs peuvent être améliorés, ce que nous envisageons de montrer dans les semaines à venir.

14. HDM05 et MSRAction
15. Temps inférieurs à 1 seconde par série temporelle durant la phase de test
16. Ici en affectant chaque observation à un seul état via l'algorithme de Viterbi
17. Ici la position relative de chaque articulation vis-à-vis de chaque autre articulation

Références

[1] Wikipédia - estimation par noyau. http://fr.wikipedia.org/wiki/Estimation_par_noyau.

[2] Wikipédia - machine à vecteurs de support. http://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support.

[3] Faisal Bashir et al. Hmm-based motion recognition system using segmented pca. *ICIP IEEE*, 2005.

[4] Frédéric Bevilacqua et al. Continuous realtime gesture following and recognition. *Kopp S and Wachsmuth I. editors "Gesture in Embodied Communication and Human-Computer Interaction"*, 5934 :73–84, 2010.

[5] Richard Bowden and David Windridge. A linguistic feature vector for the visual in terpretation of sign language. 2004.

[6] Sylvain Calinon and Aude Billard. Stochastic gesture production and recognition model for a humanoid robot. *IROS IEEE*, 2004.

[7] Chih-Chung Chang and Chih-Jen Lin. Libsvm – a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[8] William Chen and Shih-Fu Chang. Motion trajectory matching of video objects. *IS&T SPIE*, 2000.

[9] Marco Cuturi et al. A kernel for time series based on global alignments. *Proceedings of ICASSP*, 2007.

[10] Dai et al. Fuzzy smoothing of hmm parameters using the parzen window. *Electronics Letters*, 1993.

[11] Jianing Dai. Robust estimation of hmm parameters using fuzzy vector quantization and parzen's window. *Pattern Recognition*, 1995.

[12] Gaolin Fang et al. Large vocabulary sign language recognition based on fuzzy decision trees. *IEEE Transactions on Systems, Man, and Cybernetics*, 34, 2004.

[13] Lei Han et al. Discriminative human action recognition in the learned hierarchical manifold space. *Image and Vision Computing*, 2010.

[14] Horenko et al. Set-oriented dimension reduction : Localizing principal component analysis via hidden markov models. *Computational Life Sciences II Lecture Notes in Computer Science*, 2006.

[15] Shah Muhammed Abid Hussain and A. B. M. Harun ur Rashid. User independent hand gesture recognition by accelerated dtw. *ICIEV*, 2012.

[16] Mohamed E. Hussein et al. Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations. *International Joint Conference on Artificial Intelligence*, 2013.

[17] Anil K. Jain et al. Statistical pattern recognition : a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[18] Shang L. and Chan K.P. Nonparametric discriminant hmm and application to facial expression recognition. *Electronics Letters*, 2009.

[19] Hugo Larochelle. Learning algorithms for the classification restricted boltzmann machine. *Journal of Machine Learning Research*, 2012.

[20] Svetlana Lazebnik et al. Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. *IEEE CVPR*, 2, 2006.

[21] Wanqing Li et al. Expandable data-driven graphical modeling of human actions based on salient postures. *IEEE Transactions On Circuits And Systems For Video Tech*, 18, 2008.

[22] Wanqing Li et al. Action recognition based on a bag of 3d points. *Int'l workshop on CVPR (CVPR4HB)*, 2010.

[23] Stephane Mallat. Zero-crossings of a wavelet transform. 1991.

[24] Pierre-Francois Marteau and Sylvie Gibet. Constructing positive definite elastic kernels with application to time series classification. *Technical report, UMR 6074 IRISA*, 2013.

[25] Pierre-François Marteau et al. Down-sampling coupled to elastic kernel machines for efficient recognition of isolated gestures. *ICPR*, 2014.

[26] Pierre-François Marteau et al. Sous échantillonnage et machine à noyaux élastiques pour la classification de données de mouvement capturé. *EGC*, 2014.

[27] Osama Masoud and Nikos Papanikolopoulos. A method for human action recognition. *Image Vision Computing*, 2003.

[28] Mitra and Tinku Acharya. Gesture recognition : A survey. *IEEE Transactions On Systems, Man, And Cybernetics*, 2007.

[29] Ferda Ofli et al. Sequence of the most informative joints (smij) : A new representation for human skeletal action recognition. *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2012.

[30] Ferda Ofli et al. Sequence of the most informative joints (smij) : A new representation for human skeletal action recognition. *Journal of Visual Communication and Image Representation*, 2013.

[31] Sylvie C.W. Ong and Surendra Ranganath. Automatic sign language analysis : A survey and the future beyond lexical meaning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[32] Omar Oreifej and Zicheng Liu. Histogram of oriented 4d normals for activity recognition from depth sequences. *IEEE ICPR*, 2013.

[33] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 2010.

[34] Aditya Ramamoorthy et al. Recognition of dynamic hand gestures. 2003.

[35] Ferdinando Samaria and Steve Young. Hmm-based architecture for face identification. *Image and Vision Computing*, 1994.

[36] Samsu Sempena et al. Human action recognition using dynamic time warping. *International Conference on Electrical Engineering and Informatics*, 2011.

[37] Thad Starner and Alex Pentland. Real-time american sign language recognition from video using hidden markov models. 1996.

[38] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[39] Ashok Veeraraghavan et al. Role of shape and kinematics in human movement analysis. *CVPR*, 2004.

[40] Jiang Wang et al. Mining actionlet ensemble for action recognition with depth cameras. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1290–1297, 2012.

[41] J. Yamato et al. Recognizing human action in time-sequential images using hidden markov model. 1992.

[42] Xian Zhao et al. Real-time hand gesture detection and recognition by random forest. *Communication and Information Processing*, 289 :747–755, 2012.