MILOUD BETTAYEB

MODÉLISATION D'UN USAGER DE JEU VIDÉO AVEC UN MODÈLE DE MARKOV CACHÉ

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en informatique
pour l'obtention du grade de Maître ès sciences (M. Sc.)

Département d'informatique et de génie logiciel
FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2011

Remerciements

Je souhaiterais adresser mes remerciements les plus sincères aux personnes qui m'ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de ce projet.

Je commencerai par remercier mes parents (Bettayeb Mohamed et Boussouar Rym), pour tous les sacrifices qu'ils ont faits pour moi, pour leur contribution, leur soutien et leur patience. C'est grâce à eux que j'ai réussi à réaliser tous mes projets.

Un grand merci à mes frères et mes sœurs et de manière très spéciale mon frère Said Bettayeb, Ph.D en informatique à l'Université de Houston, Texas, USA, qui m'a aidé beaucoup, et qui a facilité mon intégration dans le domaine de l'informatique.

Je tiens à remercier sincèrement monsieur Luc Lamontagne, mon directeur de mémoire, qui s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi que pour l'inspiration, l'aide et le temps qu'il a bien voulu me consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Mes remerciements s'adressent également à mes collègues Houcine Romdhane, Aicha Rhazi, Marouan Ben jabeur, Ghina Besbes, Mourad Sellami et Ahmed Chakouch.

Enfin, j'adresse mes plus sincères remerciements à tous mes proches et amis, qui m'ont toujours soutenu et encouragé au cours de la réalisation de ce mémoire.

Merci à tous et à toutes.

Résumé

Le succès de l'utilisation du modèle de Markov caché dans des domaines comme le traitement des images, la biologie, la médecine et la robotique, est principalement dû à la possibilité qu'il offre d'obtenir des traitements efficaces et de construire des modèles par apprentissage automatique, même pour d'importantes masses de données. L'objectif de cette mémoire est de d'évaluer l'adéquation et l'efficacité de ce modèle pour modéliser les activités d'utilisateurs de jeux vidéo.

Dans ce mémoire, nous avons choisi le jeu Pacman pour mener notre étude. Ce jeu présente un intérêt particulier car les décisions de déplacement et les stratégies utilisées par les joueurs sont basées sur des contraintes liées à l'environnement du jeu (les fantômes, les points, les pastilles, les fruits...). Nous avons choisi d'appliquer le modèle de Markov caché pour modéliser le contrôle du Pacman par un joueur.

Notre premier objectif est de prédire la stratégie utilisée par le joueur pendant des parties de jeu. Pour cette tâche, nous avons utilisé seulement le modèle du Markov caché,

Notre deuxième objectif est de tenter d'identifier un joueur à partir d'épisodes de jeux. Pour cette deuxième tâche, nous avons combiné un modèle de Markov caché avec une méthode de classification pour obtenir nos résultats.

D'après les résultats obtenus dans nos travaux, nous pouvons affirmer que ces modèles se révèlent efficaces pour la reconnaissance d'activités dans des jeux vidéo.

Abstract

The successful usage of hidden Markov models in areas such as image processing, biology, medicine and robotics, is mainly due to the possibility it offers to obtain effective treatment and to automatically learn models, even for large masses of data. The objective of this thesis is to evaluate the importance and effectiveness of these models for modeling activities in video games.

For this thesis, we chose the game Pacman to conduct our study. This game is of particular interest because the travel decisions and strategies used by either the players are based on constraints from the game environment (the ghost points, pellets, fruits....). In our case, we chose to apply hidden Markov models to represent the control of Pacman moves by a human player.

Our first goal is to predict the strategy used by the player during game plays. For this task, we only used a hidden Markov model.

Our second goal is to identify the player based on game episodes. To accomplish this second task, we combined HMM model with a classification method to obtain our results.

From the results obtained in this thesis, we can confirm that this model is an effective way to apply artificial intelligence for activity recognition in video games.

Table des matières

Liste des tableaux

Liste des figures

Chapitre 1

Introduction

Dans ce mémoire, nous abordons le problème de la reconnaissance automatique d'un joueur (*Automatic User Recognition*). Nous abordons ce problème dans le cadre des jeux vidéo. L'objectif consiste à proposer une technologie informatique qui permet de modéliser les actions et les décisions des joueurs pendant des séances de jeu. Le modèle résultant servirait par la suite à analyser des séquences d'observations d'une partie de jeu vidéo et à reconnaître un joueur à partir de ces séquences.

Ce genre de modèle peut avoir différentes applications. Par exemple, il peut permettre de prédire la stratégie utilisée par le joueur à partir des observations sans connaître l'état de ce dernier. Un modèle peut permettre d'ajuster le niveau du jeu afin de l'adapter aux aptitudes du joueur et de lui offrir une opposition plus intéressante. Finalement, le modèle pourrait être réutilisé dans la conception du jeu soit en transférant certaines connaissances aux personnages animés du jeu (*non-player character* - NPC) ou pour simuler des comportements de joueur afin de tester le logiciel.

Plusieurs techniques ont été utilisées dans la littérature pour modéliser un joueur. On compte parmi celles-ci les réseaux de neurones, les n-grams et les scripts. L'application de ces techniques est bien détaillée dans le chapitre 3. Pour notre travail de recherche, nous nous intéressons à la construction de modèles probabilistes. Plus précisément, nous utilisons les modèles de Markov caché. Nous visons à déterminer dans quelle mesure un modèle s'appuyant sur une représentation incertaine du

comportement du joueur permet de bien prédire certaines activités. Pour vérifier l'adéquation de cette approche, nous avons choisi les deux tâches suivantes :

- Reconnaître la stratégie d'un joueur à partir de l'état du jeu et des actions qu'il choisit;
- Reconnaître l'identité d'un joueur lorsqu'on dispose de plusieurs modèles.

Pour reconnaître différentes facettes de jeu à partir d'une séquence d'observations, nous devons construire des modèles pour mener nos analyses. Nous adoptons les modèles de Markov cachés (en anglais, HMM - *Hidden Models Markov*) qui sont les modèles fréquemment employés pour décrire les générations de séquences.

Un HMM est la combinaison de deux processus stochastiques, une chaîne de Markov et une densité de probabilité associée à chaque état de la chaîne. Un HMM modélise la production d'une séquence de la façon suivante :

- En commençant dans un état de départ, on passe d'état en état suivant une matrice de transition de la chaîne markovienne,
- En émettant, chaque fois qu'on arrive en un état une valeur jusqu'à ce que l'état final soit atteint.

Si on note $S = \{S_1, S_2, ...S_p\}$ l'ensemble des états et X l'espace des observations, $b_k(x) = p(x \mid S_k)$ sera la probabilité d'émettre x dans l'état S_k, $a_{ij}(x) = p(S_i \mid S_j)$ sera la probabilité de transition de l'état S_j à l'état S_i. Pour rendre les calculs et l'estimation possibles, on fait des hypothèses, dont les plus courantes, sont :

- L'hypothèse markovienne, qui dit que a_{ij} dépend uniquement des états S_i et S_j,
- L'hypothèse d'indépendance des sorties qui dit que $b_k(x)$ dépend uniquement de l'état courant S_k et de l'émission x.

Les probabilités de transition et d'émission sont apprises par un algorithme itératif du type EM (Algorithme Espérance-Maximisation) qui permet d'apprendre les probabilités sans supervision.

Pour accomplir une tâche de reconnaissance, un HMM prendra en entrée une séquence d'observations provenant d'épisodes de jeux et fournira en sortie la séquence d'états la plus probable ainsi que son score (sa valeur de probabilité).

Dans ce mémoire, nous essayons de prédire le comportement d'un joueur et nous utilisons le jeu Pacman pour mener nos essais. Le comportement, représenté à l'aide d'un modèle de Markov caché, est basé sur des observations enregistrées durant une partie du jeu. Par l'utilisation du HMM, nous essayons de prédire la stratégie utilisée à partir des observations enregistrées et par la suite, nous tentons de reconnaitre le joueur par une méthode de classification. Par exemple, dans les cas des jeux vidéo multi-joueurs en ligne où un certain nombre de joueurs sont simultanément branchés au jeu, l'état du joueur est caché et l'identité même du joueur est parfois inconnue. Par exemple, dans les jeux de combat, on observe seulement les gestes et les actions du joueur, mais on ne sait pas s'il est dans un état d'attaque ou de fuite. Mais par l'utilisation de modèles de prédiction, on peut prédire l'état du joueur. Et dans le cas où il y a plusieurs joueurs dans le même jeu, l'identité du joueur peut être inconnue et nous avons donc besoin de prédire quel joueur est en train de jouer. L'objectif est simple, mais l'utilisation de l'HMM s'avère efficace.

La structure de ce mémoire est la suivante. Dans le deuxième chapitre, nous allons présenter le modèle de Markov observable, c'est-à-dire le modèle de Markov standard basé sur des états et des observations totalement observables. Par la suite, nous allons expliquer l'ajout de termes cachés permettant de présenter le modèle de Markov basé sur des états cachés.

Dans le troisième chapitre, nous présentons quelques exemples d'utilisation du HMM dans différents domaines tels que la reconnaissance d'écriture, la reconnaissance du geste et la reconnaissance de trajectoires de robot. Nous allons

également donner un aperçu des principales techniques utilisées pour modéliser un joueur dans le cadre d'un jeu vidéo.

Dans le quatrième chapitre, nous introduisons une définition détaillée du jeu Pacman que nous utilisons pour mener nos expérimentations. Nous présentons également les différentes astuces du jeu que nous avons préconisées pour mener nos essais.

Dans le cinquième chapitre, nous allons expliquer les travaux que nous avons menés dans le cadre de nos expérimentations qui visent à prédire la stratégie utilisée par le joueur et ensuite l'identité du joueur lui-même.

Enfin, un dernier chapitre permet de conclure et de présenter le fruit de notre travail qui aboutira sur la conclusion que les modèles de Markov cachés sont relativement efficaces pour la modélisation de joueurs en contexte de jeux vidéo.

Chapitre 2

Modèle de Markov

2.1. Modèle de Markov observable

Un modèle de Markov est un modèle qui permet de représenter un processus aléatoire, dont l'état S à l'instant t dépends de son état à l'instant $t-1$. Ce modèle peut être représenté selon deux façons :

- Soit par une matrice transition.
- Soit par un graphe orienté.

2.1.1. Matrice de transition

Pour un processus, dont l'ensemble des états possibles, est $S=\{S_1,S_2,...S_n\}$, la matrice de transition A permet de représenter les probabilités de changement d'état $S_{(t)}$ à l'état $S_{(t+1)}$. Par exemple, pour un processus ayant deux états possibles, la transition entre chaque état peut donc se présenter comme suit :

$$A=\begin{pmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} \end{pmatrix}$$

Les colonnes représentent les états du système à l'instant t, et les lignes représentent les états du système à l'instant $t+1$. Par exemple, étant à l'état 1 au temps t, la matrice indique que nous avons 1 chance sur 3 de retomber dans le même état et 2

chances sur 3 de changer d'état. Il est important de noter que la somme des probabilités sur une ligne sera toujours égale à 1.

La probabilité initiale que le processus soit dans un état particulier est donnée par la distribution π. Cette distribution est soit équivalente ou non équivalente :

- La distribution est équivalente dans le cas où le point de départ pour tout le monde est le même. La probabilité est donc répartie uniformément sur le nombre d'états. Par exemple, si on a 4 états, la distribution pour chaque état est « ¼ ».
- La distribution est non équivalente dans le cas où le point de départ pour tout le monde n'est pas le même. Par exemple, si on a deux états pour le joueur et on sait bien que le joueur commence tout le temps avec le premier état, alors la probabilité initiale est égale à « 1 » pour le premier état et à « 0 » pour le deuxième état.

Le point le plus important est que la somme des probabilités est égale à 1 dans les deux distributions.

2.1.2. Graphe orienté

Un graphe permet de représenter les probabilités de changement d'état de façon plus visuelle qu'avec une matrice. Par exemple, considérons le graphe de la figure 1:

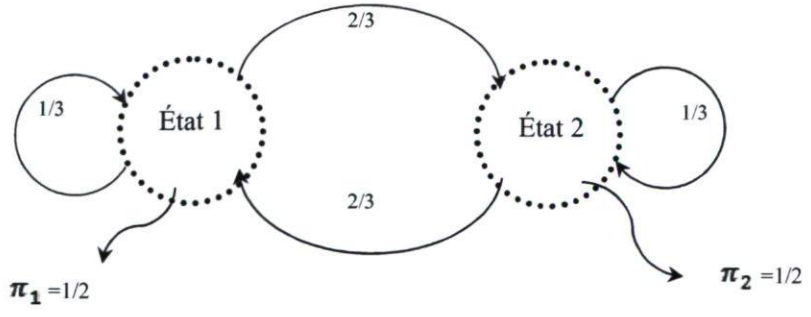

Figure 1 : Graphe orienté

Chaque nœud du graphe correspond à un état possible du système. Les arcs représentent la probabilité de changer d'état ou de rester dans le même état lorsqu'il s'agit de boucles.

2.1.3. Utilisation

Les modèles de Markov observables ne sont pas suffisants pour faire des opérations sur des séquences complexes et partiellement observables, car, dans la plupart des cas, nous ne connaissons pas l'état du système. Dans certains modèles, les états sont cachés et le modèle de Markov observable n'est plus utilisable. Donc la solution est d'utiliser le modèle de Markov caché.

2.2. Modèle de Markov caché

Un modèle de Markov caché est un modèle de Markov avec 2 suites de variables :

- Une séquence observée (les observations).
- Une séquence cachée (les états).

On suppose dans un tel modèle que lorsque le processus change d'état, il émet un symbole qui peut être observé. Cependant, les états du processus ne sont pas accessibles à l'observateur.

Pratiquement, la construction d'un tel modèle consiste à créer deux matrices. Une matrice représente la séquence observée et l'autre représente la séquence cachée.

L'observation d'une séquence n'est donc plus uniquement liée à une suite d'états, comme le modèle de Markov observable, mais aussi liée à une suite d'observations.

2.2.1. Définition

Un modèle de Markov caché peut se définir comme un triplet (A, B, π) avec la matrice de transition A, la matrice d'observation B et le vecteur de probabilités initiales π.

a. Vecteur de probabilités initiales

π_i est la probabilité que le système commence dans l'état i. Nous devons respecter la contrainte que $\pi_1 + \pi_2 + ... + \pi_n = 1$, puisque le système doit commencer dans un certain état. Il est important de prendre note que $\pi_i(\mathrm{k}) \geq 0$.

b. Matrice des probabilités de transition

A_{ij} est la probabilité que, si le système est dans l'état i à un moment donné, le prochain état se retrouve dans un état j. Il est important de noter que le système peut se retrouve de nouveau dans le même état, c.-à-d. que la valeur d'a_{ii} peut être différente de zéro.

Cependant, le graphe des états et des transitions n'a pas besoin d'être complet, c.-à-d. que certains a_{ij} peuvent être zéro.

La somme des probabilités sur une ligne sera toujours égale à 1 et est donnée par la formule suivante :

$a_{i1} + a_{i2} + ... + a_{ik} = 1$ Pour tout état i avec $a_{ij} \geq 0.$

c. Matrice des probabilités d'émission

Avant de transiter vers le prochain état, le système émet une observation. La probabilité d'émission $b_j(k)$ est la probabilité d'émettre l'observation k étant donné que le système est dans l'état j

$$b_j(k) = P(obs = k \text{ à l'instant t} \mid \mathrm{q_t} = j)$$

Puisque le système émet toujours une observation avant de changer l'état,

$$b_j(k_1) + b_j(k_2) + ... + b_j(k_m) = 1 \quad \text{avec} \quad b_j(k) \geq 0$$

2.2.2. Types de HMM

Selon la dépendance de la probabilité d'émission, il existe deux types de HMM,

a. HMM état-émission (machine de Moore)

Tel qu'illustré à la figure 2, la probabilité d'émission dépend seulement de l'état.

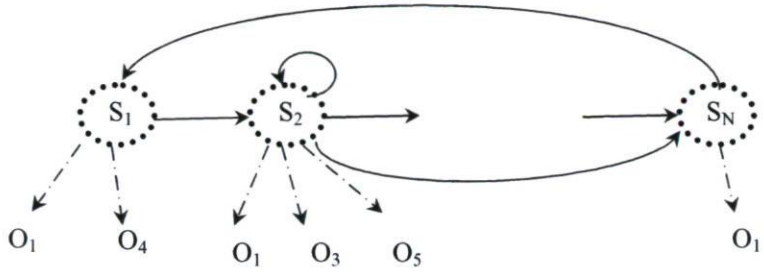

Figure 2 : Machine de Moore.

b. HMM arc-émission (machine de Mealy)

La probabilité dépend de la transition entre les deux états (voir figure 3).

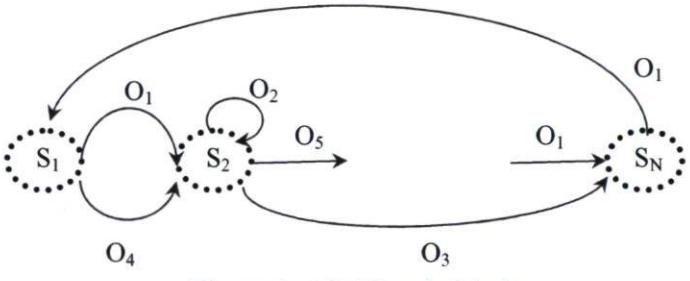

Figure 3 : Machine de Mealy

2.2.3. Topologie des HMMs

a. Modèle ergodique

Dans ce modèle (figure 4), tout état est atteignable depuis tous les autres états du modèle. Ainsi la somme des probabilités en sortie est égale à 1 et la somme de probabilité en entrée est égale à 1 également.

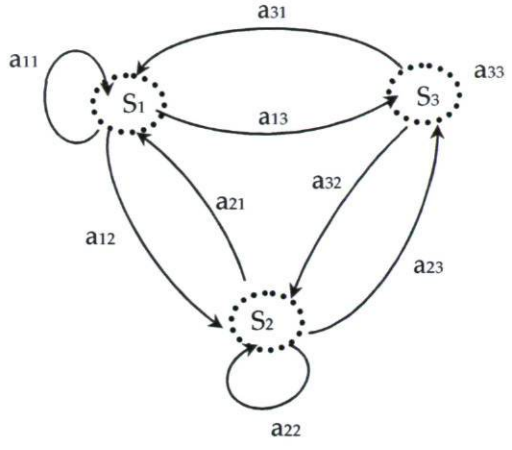

Figure 4 : Modèle ergodique

b. Modèle gauche-droite

Dans ce modèle, la somme des probabilités en sortie est égale à 1 et il n'existe plus de relation (probabilité=0) qui mène de l'état actuel vers les états précédents (figure 5).

Formellement : $a_{ij} = 0$ si j<i;

$$\sum a_{ij} = 1 \quad \text{pour } j \in [0..4]$$

Dans notre exemple : $a_{21} = a_{31} = a_{41} = 0$

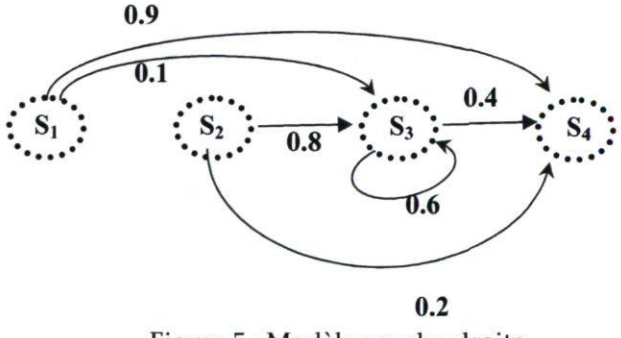

Figure 5 : Modèle gauche-droite

2.3. Les trois "grandes questions" des HMMs

Il y a trois questions fondamentales au sujet des HMMs que nous aborderons comme étant l'évaluation, le décodage et l'apprentissage.

2.3.1. Évaluation d'un HMM

L'évaluation est définie par le calcul de la probabilité d'une séquence d'observation O étant donné un modèle d'HMM. Ceci correspond à la vraisemblance $P(O \mid \lambda)$ que le modèle λ génère la séquence O.

Il existe deux méthodes pour l'évaluation :

Évaluation directe de la probabilité d'observation

La probabilité de la suite d'observations O, étant donné le modèle λ, est égale à la somme sur tous les chemins d'états possible S des probabilités conjointes de O et de S :

$$P(O \mid \lambda) = \sum_{S} P(O, S \mid \lambda) = \sum_{S} P(O \mid S, \lambda) P(S \mid \lambda)$$

Tel que :

$$P(S \mid \lambda) = \pi_{S_1} \; a_{S_1 S_2} \; a_{S_2 S_3} \ldots \; a_{S_{(T-1)} S_T}$$

$$P(O \mid S, \lambda) = b_{S_1}(O_1) \; b_S \; (O_2) \ldots \; b_S(O_T)$$

D'où :

$$P(O \mid \lambda) = \sum_{S_1 S_2 \ldots S_T} \pi_{S_1} \; b_{S_1}(O_1) a_{S_1 S_2} b_{S_2}(O_2) \ldots \; a_{S_{(T-1)} S_T} b_{S_T}(O_T)$$

Cette formule directe nécessite $n^T - 1$ addition et $(2T-1)n^T$ multiplication

(n^T étant le nombre de chemins possible de longueur T), soit $2T \times n^T$ opérations.

Évaluation par les fonctions Forward-Backward.

Une astuce utilisée pour simplifier l'évaluation d'une séquence d'observations est de découper le calcul en deux parties :

- On estime la probabilité d'émettre le début de la séquence d'observations O (1 : t), qui correspond aux observations sur du temps 1 au temps t, et de se retrouvé dans l'état S_i au temps t. Cette quantité est désignée par $\alpha(t, S_i)$.
- Et on estime également la probabilité de la fin de la séquence d'observations O(t+1 : T) sachant que l'on part de S_i au temps t. Cette quantité est désignée par $\beta(t, S_i)$.

En combinant les deux parties, l'évaluation de la séquence d'observations est égale à : $P(O \mid \lambda) = \sum_{S_i} \alpha(t, S_i)\beta(t, S_i)$

Avec $\alpha(t, S_i)$ qui est estimé par la formule du Forward et $\beta(t, S_i)$ qui est estimée par la formule du Backward. Ces deux formules sont définies dans la section suivante.

Algorithme Forward

Cette partie de l'algorithme consiste à calculer la probabilité d'émettre la suite d'observations $\{O_1, O_2, ...O_T\}$ et d'aboutir à S_t à l'instant t avec t croissant. On notera $\alpha(t, S_i)$ par $\alpha_t(i)$ (« la probabilité de l'observation S_i quand le temps égale à t ») qui désigne le Forward.

$$\alpha_t(i) = P(\{O_1, O_2, ...O_T\}, S_t = S_i \mid \lambda)$$

$\alpha_t(i)$ est calculée de manière récursive telle que décrite à la figure 6.

Ce calcul est basé sur le fait que pour émettre le début de l'observation O (1 : t + 1) et aboutir dans l'état S_t au temps $t+1$, on doit nécessairement être dans l'un des états à l'instant t. Ce calcul nécessite $n + n \times (n+1) \times (T-1)$ multiplications et $(n-1) \times n \times (T-1)$ additions, soit une complexité en $2 \times n^2 \times T$.

Algorithme Backward

Cet algorithme permet de calculer la probabilité d'émettre la suite $O = \{O_1, O_2, ...O_T\}$ En partant de l'état S_i à l'instant t avec t décroissant. $\beta(t, S_t)$ sera noté $\beta_t(i)$ dans le reste de ce document:

$$\beta_t(i) = P(\{O_{t+1}, O_{t+2}, ...O_T\} | S_t = S_i, \lambda)$$

On déduit β_t de β_{t+1} par l'algorithme présenté à la figure 7. Les calculs de β est aussi en $2 \times n^2 \times T$.

Algorithme 1: Forward

Entrées : π , a, b

Sorties : α, $P(O|\lambda)$

pour t=1 **faire**

pour tout i ∈ {1,2,.....n} **faire**

$\alpha_t(i) \leftarrow \pi_i * b_i(O_1)$

pour tout t ∈ {1,2,.....T-1} **faire**

pour tout j ∈ {1,2,.....n} **faire**

$\alpha_{t+1}(j) \leftarrow [\sum_{i=1}^{n} \alpha_t(i) * a_{ij}] * b_j(O_{t+1})$.

pour tout i ∈ {1,2,.....n} **faire**

Calculer $P(O|\lambda) \leftarrow \sum_{i=1}^{n} \alpha_T(i)$.

retourner α, $P(O|\lambda)$

Figure 6 : Algorithme Forward pour l'évaluation de probabilités

Algorithme 2: Backward.

Entrées : π , a, b

Sorties : β, $P(O|\lambda)$

pour t=1 **faire**

pour tout i ∈ {1,2,.....n} **faire**

$\beta_T(i) \leftarrow 1$

pour tout t ∈ {T-1 ..., 2, 1} **faire**

pour tout i ∈ {1, 2 ... n} **faire**

$\beta_t(i) \leftarrow \sum_{j=1}^{n} a_{ij} * b_j(O_{t+1}) * \beta_{t+1}(j)$

pour tout i ∈ {1,2,.....n} **faire**

Calculer $P(O|\lambda) \leftarrow \sum_{i=1}^{n} \beta_t(i)$ pour t=1

retourner β, $P(O|\lambda)$

Figure 7 : Algorithme Backward

2.3.2. Le problème du décodage

Étant donnés une suite d'observations $O = \{O_1, O_2, ...O_T\}$ et un modèle λ = {Π, A, B}, comment peut-on choisir une suite d'états $S = \{S_1, S_2, ...S_T\}$ qui est optimale selon un critère convenable ? La difficulté réside dans la suite optimale d'états, car il existe plusieurs méthodes : le critère local, le critère global et l'algorithme de Viterbi. [2].

Le critère local

Cette technique consiste à choisir l'état S_t qui est le plus probable et ceci indépendamment des autres états individuellement pour chaque t. [2].

Le critère global

On cherche à trouver l'unique trajectoire optimale de la suite d'états, donc de maximiser P (S|O,λ). Pour cela, on peut définir la probabilité maximale d'une séquence d'état au temps t qui se termine dans l'état S_t. [2].

$$\delta_t(j) = \max_{S_1 S_2 ... S_{t-1}} P(S_1 S_2 ... S_t = S_i, O_1 O_2 ... O_t \mid \lambda)$$

$$\delta_{t+1}(j) = \max_i \left[\delta_t(i) a_{ij}\right] \times b_j(O_{t+1})$$

En conservant pour chaque t et chaque i l'état ayant amené au maximum : $\delta_t(j)$, ψ_t (j) on obtient l'algorithme de Viterbi tel que :

$\delta_t(j)$: la probabilité de Viterbi

ψ_t (j) : défini la trace de la probabilité maximale

Algorithme Viterbi

Il s'agit de trouver dans le modèle la suite d'états qui maximise P (S,O|λ). Pour trouver le meilleur chemin $S = (S_1, S_2 ... S_T)$ pour une suite d'observations $O = (O_1, O_2, ... O_T)$ on définit $\delta_t(i)$ qui est la probabilité du meilleur chemin menant à l'état S_i à l'instant t, en étant guidé par les t premières observations :

$$\delta_t(j) = \max_{S_1 S_2 .... S_{t-1}} P(S_1 S_2 ... S_t = S_i, O_1 O_2 ... O_t \mid \lambda)$$

Par récurrence, on calcule $\delta_{t+1}(j) = \max_i \left[\delta_t(i) a_{ij}\right] \times b_j(O_{t+1})$ On garde la trace de la suite d'états qui donne le meilleur chemin amenant à l'état S_i à t dans un tableau. Le pseudo-code de l'algorithme de Viterbi est décrit à la figure 8.

Algorithme 3 : Viterbi.

Entrées : π , a, b

Sorties : S_T^*

pour t=1 **faire** // initialiser les probabilités et les chemins pour la première
// Observation

pour tout i ∈ {1,2,.....n} **faire**

$\delta_1(i) \leftarrow \pi_i b_i(O_1)$

$\psi_1(i) \leftarrow 0$;

pour tout t ∈ {1,2,.....T-1 } **faire** // déterminer les transitions optimales

pour tout j ∈ {1,2,.....n} **faire**

$\delta_t(j) \leftarrow max_{1 \le i \le n} [\, \delta_{t-1}(i)\, a_{ij}\,]\, b_j(O_t)$

$\psi_t(i) \leftarrow argmax_{1 \le i \le n} [\, \delta_{t-1}(i)\, a_{ij}\,]$

pour tout i ∈ {1,2,.....n} **faire** // conserver les résultats dans le tableau

$P^*(j) \leftarrow max_{1 \le i \le n} [\delta_T(i)]$,

$S_T^* \leftarrow argmax_{1 \le i \le n} [\delta_T(i)]$;

pour tout i ∈ {T-1,.....2, 1} **faire**

Calculer $S_T^* \leftarrow \psi_{t+1}(S_{t+1}^*)$

retourner S_T^*

Figure 8 : Algorithme Viterbi

2.3.3. Apprentissage

Le but de l'apprentissage est de déterminer les paramètres (A, B, π) qui maximisent la probabilité de la suite d'observations $P(O \mid \lambda)$. L'idée employée ici est d'utiliser des procédures de ré-estimation par punition-récompense :

- choisir un ensemble initial de paramètres λ_0 ;
- calculer λ_1 à partir de λ_0 ;
- répéter ce processus jusqu'à ce qu'un critère de terminaison soit satisfait.

On cherche donc à définir une fonction F telle que : $\lambda_{n+1} = F(\lambda_n)$. L'approche la plus simple pour définir F consiste à faire des statistiques sur l'utilisation des transitions et des distributions. Ceci revient à calculer des fréquences d'utilisation à partir de l'ensemble d'apprentissages.

Si l'ensemble est important, ces fréquences fournissent une bonne approximation des probabilités a posteriori utilisables alors comme paramètres du modèle pour l'itération suivante. La méthode d'apprentissage va donc consister, à partir d'un modèle initial aléatoire, à estimer ses paramètres comme indiqué ci-dessus, puis à recommencer cette estimation ("ré-estimation") jusqu'à obtenir une certaine convergence.

$\overline{\pi_i}$ = nombre de fois que HMM s'est trouvé dans S_i à l'instant 1

$$\overline{a_{ij}} = \frac{\text{nombre de fois où la transition de } S_i \text{ à } S_j \text{ a été utilisée}}{\text{nombre de transitions effectuées à partir de } S_i}$$

$$\overline{b_j}(k) = \frac{\text{nombre de fois où le HMM s'est trouvé dans l'état } S_j \text{ en observant } O_k}{\text{nombre de fois où le HMM s'est trouvé dans l'état } S_j}$$

De façon expérimentale, ces quantités sont obtenues par les équations suivantes

$$\overline{\pi_i} = \frac{1}{n}\sum_{k=1}^{N} \gamma_t^k(i)$$

$$\overline{a_{ij}} = P(q_{t+1} = S_j \mid q_t = S_i) = \frac{\sum_{k=1}^{N}(\sum_{t=1}^{|O^k|-1} \xi_t^k(i,j))}{\sum_{k=1}^{N}\sum_{t=1}^{|O^k|-1} \gamma_t^k(i)}$$

$$\overline{b_j}(k) = P(q_{t+1} = S_j \mid q_t = S_i) = \frac{\sum_{k=1}^{N}(\sum_{t=1,|O^k|-1 \text{ avec } O_t^k = v_t} \gamma_t^k(j))}{\sum_{k=1}^{N}\sum_{t=1}^{|O^k|-1} \gamma_t^k(j)}$$

Où $\gamma_t^k(i)$ est la probabilité que le $t^{ème}$ symbole émis dans la séquence O^k soit émis par l'état S_i

$$\gamma_t^k(i) = P(q_t = S_i \mid O^k, \lambda)$$

$$\gamma_t^k(i) = \sum_{j=1}^{N} P(q_t = S_i, q_{t+1} = S_j \mid O^k, \lambda)$$

$\xi_t^k(i,j)$ est la probabilité d'utiliser la transition allant de l'état S_i (en émettant le Symbole (O_t^k) à l'état S_j (en émettant O_{t+1}^k) de la $k^{ème}$ séquence d'observations.

$$\xi_t^k(i,j) = P(q_t = S_i, q_{t+1} = S_j \mid O^k, \lambda)$$

L'algorithme de Baum Welch, qui opérationnalise ces étapes, est décrit à la figure 9.

Algorithme 4 : Baum Welch.

Entrées : π , a, b

Sorties : $\overline{\lambda}$

1. Fixer les valeurs initiales

   pour tout i , $j, k \in$ {1,2,.....n} **faire**

   Calculer π_i^0 , $a_{i\bar{j}}^0$, $b_{\bar{j}}^0(k)$

   fin pour()

2. Calculer à l'aide des fonctions Forward-Backward :

   pour tout i , $j, k \in$ {1,2,.....n} **faire**

   pour tout $t \in$ {1,2,.....T-1} **faire**

   $\xi_t^k(i,j)$, $\gamma_t^k(i)$

   Calculer $\overline{\lambda}$ en utilisant les formules de ré-estimation.

3. Recommencer en 2 jusqu'à ce qu'un certain critère de convergence soit rempli.

retourner $\overline{\lambda}$

Figure 9 : Algorithme Baum-Welch

2.4. Conclusion

Nous avons vu dans ce chapitre le modèle de Markov observable, et par la suite nous avons introduit le modèle caché. Ce dernier est pertinent à nos travaux, car nous allons montrer qu'il peut résoudre notre problème de modélisation. Dans le prochain chapitre, nous allons présenter des utilisations du HMM dans quelques domaines d'applications.

Chapitre 3

Des applications du modèle de Markov caché

Dans ce chapitre, nous présentons quelques applications qui utilisent le modèle de Markov caché pour modéliser et reconnaître différentes activités, tels que la reconnaissance de gestes humains et la reconnaissance de trajectoires de robot. Nous décrivons également les principales techniques qui sont utilisées pour modéliser le comportement d'un joueur de jeu.

3.1. HMM pour la reconnaissance de la trajectoire du robot

S.K.Tso et K.P.Liu [8] proposent un système de sélection de la meilleure trajectoire d'un robot à partir des 12 trajectoires qui représentent les états du modèle, et de 128 symboles d'observation déduite par la méthode de quantification vectorielle sur 588 vecteurs. La quantification vectorielle est une technique souvent utilisée dans la compression de données avec pertes de données dont l'idée de base est de coder ou de remplacer une clé des valeurs d'un espace vectoriel multidimensionnel par des valeurs d'un sous-espace discret de plus petite dimension. [13]

La sélection de la trajectoire est basée sur un modèle de Markov caché de 6 états et de type gauche-droite. La modélisation est faite sur 12 trajectoires qui ont été capturées pour la sélection. Le taux d'échantillonnage a été 40Hz et la durée a été 10s.

L'apprentissage du HMM est fait par l'utilisation des algorithmes Forward et Backward pour faire l'évaluation du modèle et par l'utilisation de l'algorithme Baum-

Welch pour mesurer de la qualité de chaque trajectoire. L'indice du maximum de la vraisemblance est utilisé pour déterminer la meilleure trajectoire du robot.

3.2. HMM pour la reconnaissance de gestes de tennis

Petkovic et al. [9] présentent un modèle pour reconnaître les différentes classes des coups enregistrées avec une caméra durant des matchs de tennis. Dans ce travail, les auteurs étudient les parties de plusieurs joueurs dans des tournois différents et les analyses ensuite par l'utilisation d'un modèle de Markov caché.

L'approche technique est basée sur un algorithme qui se compose en 3 étapes :

1) La première étape est de segmenter l'arrière-plan du joueur.
2) La deuxième étape est d'extraire un certain nombre de caractéristiques de la représentation binaire de joueur.
3) La troisième est d'utiliser le modèle de Markov caché pour trouver la solution optimale.

Étape 1 : Segmentation d'image

La segmentation de l'arrière-plan du joueur est faite par les étapes suivantes :

1- On capture une image à partir de la vidéo enregistrée.
2- On utiliser la distance de Mahalanobis, pour adapter le modèle des lignes de tennis (voir figure 10.a).
3- Le modèle 3D est construit en tenant compte de la visibilité des lignes (figure 10.b).
4- Après la construction du modèle 3D, on obtient des connaissances de la scène à un plus haut niveau sémantique (figure 10.c).

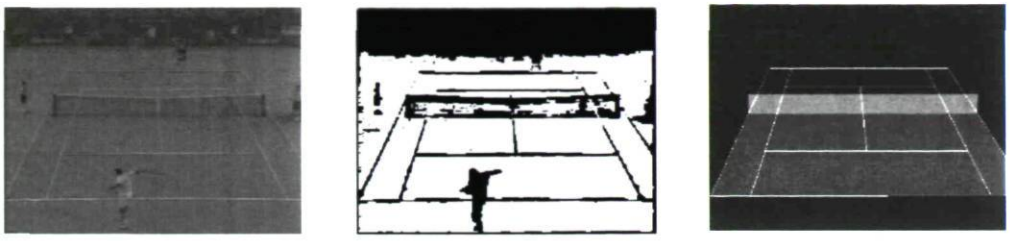

Figure 10 : a) image originale, b) segmentation initiale, c) modélisé l'image en 3D.

(Source : [9])

Étape 2 : Extraction les caractéristiques d'image

La deuxième étape est d'extraire un certain nombre de caractéristiques de la représentation binaire de joueur.

Figure 11 : a) segmentation final du joueur, b) extraction des caractéristiques

(Source : [9]).

Étape 3 : Utilisation du HMM

La dernière étape consiste à trouver le meilleur coup de joueur, c.-à-d. reconnaitre le geste le plus probable. Comme les gestes correspondent aux états du HMM, les auteurs ont mené deux expérimentations avec des modèles de 6 gestes (états) et de 11 gestes (états).

Pour la reconnaissance, ils ont utilisé un HMM discret de type gauche-droite de 4 à 48 états. Le HMM discret exige la transformation des paramètres continus en un ensemble prédéfini des symboles.

Dans le processus d'apprentissage, ils ont mené un certain nombre d'itérations de l'algorithme de Baum-Welch avec la formule de ré-estimation.

Dans une première expérience, ils ont tenté de déterminer le meilleur ensemble des caractéristiques et d'examiner l'indépendance de la personne de son ensemble des caractéristiques. Dans cette expérience, ils ont choisi six événements à reconnaître (droit, revers, service, smash, volée droite et volée de revers) et six HMMS ont été construits - un pour chaque type d'événement.

Dans la deuxième expérience, ils ont étudié les taux de reconnaissance de combinaisons de différentes caractéristiques. Pour cela, ils ont choisi 11 coups différents. L'apprentissage et l'évaluation sont restés les mêmes que dans la première expérience.

Les résultats expérimentaux du classement de coups de tennis ont montré que l'apprentissage avec HMM est une bonne méthode pour catégoriser automatiquement des statistiques des jeux de tennis à l'aide de vidéos diffusées à la télévision.

3.3. HMM pour la classification des joueurs dans un jeu multi-joueurs « MMOG»

Yoshitaka Matsumoto et Ruck Thawonmas [7] proposent de faire la classification des joueurs dans les jeux en ligne massivement multi-joueurs (MMOG) par l'utilisation de modèles de Markov caché. Un jeu MMOG est un type de jeu vidéo qui n'est accessible qu'en réseau, qui est persistant (c'est-à-dire qu'il est en marche tout le temps, que des joueurs y soient connectés ou non) et qui a un très grand nombre de joueurs simultanément (au moins une centaine).

Les auteurs ont basé leurs travaux sur un modèle de Markov caché de 8 états correspondant à des stratégies utilisées dans le jeu (combattre, parler, poursuivre, transiter, fuir…) et 9 observations qui sont les actions suivantes : (marcher, attaquer, clavarder, ramasser une potion, prendre la nourriture, ramasser des clés, abandonner, recommencer, éliminer). Et une probabilité initiale équivalente égale à 1 / N = 0,125, a été attribuée à chaque élément de Π et A.

	w	a	c	p	f	k	e	l	r
combattre	0.00	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.25
parler	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
chasser	0.70	0.00	0.00	0.10	0.10	0.10	0.00	0.00	0.00
en transit	0.75	0.00	0.00	0.00	0.00	025	0.00	0.00	0.00
chercher l'énergie	0.00	0.00	0.00	0.50	0.50	0.00	0.00	0.00	0.00
fuir	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.25
s'ennuyer	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
transporter	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.50	0.00

Figure 12: Matrice des observations [7]

L'algorithme de Baum-Welch a été utilisé pour former les HMMs, un pour chaque type de joueur, à l'aide les données d'apprentissage. L'ensemble de données de d'apprentissage se compose de la séquence d'action et le type de chaque agent de joueurs connus. Pour identifier le type d'un agent d'un joueur inconnu, l'algorithme de Viterbi a été utilisé. Cet algorithme calcule les probabilités de logarithmes en utilisant l'HMM. L'agent du joueur inconnu se verra attribué le type du HMM avec la plus grande probabilité.

	w	a	c	p	f	k	e	l	r	longueur
K	157.79	29.08	0.00	0.46	1.96	0.78	0.12	0.12	0.06	190
IMK	223.69	2.52	22.09	1.96	1.91	0.33	0.11	0.11	0.01	253
EMK	219.27	6.62	19.83	4.31	4.09	0.40	0.10	0.10	0.00	255

Figure 13: Fréquence d'apparition moyenne de chaque action, et la durée moyenne de l'action [7]

Des expérimentations sont faites pour comparer deux méthodes, les HMM et une méthode de classification par l'approche AMBR (Adaptive Memory Based Reasoning). Et les résultats de classification montrent que le taux de reconnaissance du joueur avec un HMM est toujours meilleur qu'avec l'approche AMBR. [7]

3.4. HMM pour l'analyse des activités alimentaires des personnes dans une maison de retraite

Cet article [11] propose un algorithme qui permet d'analyser les activités alimentaires des personnes dans une maison de retraite basée sur le modèle de Markov cachée. L'objectif est de surveiller les personnes âgées, via un algorithme basé sur trois étapes :

1) La détection des personnes : on utilise un algorithme de segmentation qui permet de localiser toutes les personnes.
2) La détection des visages de chaque personne: cette tâche est basée sur l'algorithme de (Schneiderman et Kanade, 2000).
3) La détection des activités : elle est basée sur un modèle de Markov caché dont les paramètres sont :
   - Les états :
     - ✓ Q1 : la main vers la tête qui correspond au début du mouvement de manger.
     - ✓ Q2 : la main s'éloigne de la tête (fin du mouvement de manger).
     - ✓ Q3 : les mouvements de la main ne sont pas directement liés à l'alimentation, tels que le déplacement entre les plats.

   - Les observations : elles correspondent aux mouvements et la distance entre la tête, le bras et la main.
     - ✓ Les mouvements entre la tête, le bras et la main,
     - ✓ Les mouvements de la main droite,

✓ Les mouvements de la main gauche.

- Les probabilités des états initiaux sont égales, c.-à-d.

$$P(Q1) = P(Q2) = P(Q3) = 1/3.$$

Les résultats de la comparaison entre la méthode de détection par HMM et la méthode basée sur les mouvements relatifs entre la tête et les mains pour 10 résidents, capturés dans des jours différents. Comme la longueur totale des vidéos est de 30 minutes, les auteurs ont montré que le taux de détection par HMM est élevé avec un taux de faux positif faible par rapport à la deuxième méthode.

Méthode	détections correctes	détections ratées	fausses alarmes
les mouvements relatifs	50	6	26
HMM	50	6	9

Figure 14 : Comparaison entre la méthode de détection par HMM et la méthode basée sur les mouvements relatifs entre la tête et les mains [11]

Cette approche pourrait aider les soignants pour faire l'évaluation du niveau d'activité des résidents, et même pour faciliter le contrôle des personnes âgées ou des personnes malades à partir des caméras à distance.

3.5. HMM pour reconnaissance de l'action humaine pour des séquences d'images

Dans cet article [10], les auteurs proposent une méthode de reconnaissance des actions humaines basées sur un modèle de Markov caché (HMM). Pour appliquer le HMM, un ensemble d'images en temps séquentiel est transformé en séquence de vecteur qui caractérise l'image. La séquence est convertie en une séquence des symboles par la méthode de quantification vectorielle (figure 15).

Pour reconnaître une séquence observée, on choisit le HMM qui donne la probabilité la plus élevée pour la séquence choisie. Les résultats expérimentaux montrent que le taux de reconnaissance est supérieur à 90%. Le taux de reconnaissance peut être

amélioré par l'augmentation du nombre de personne utilisé pour générer les données d'entrainement.

Dans l'apprentissage de l'action du joueur, les paramètres du HMM, un par catégorie d'action humaine, sont optimisés de façon à mieux décrire les séquences de l'action humaine. Les principaux résultats obtenus à partir des deux expérimentations faites dans cet article sont les suivants.

- Lorsque les données de formation et les données d'essai sont celles d'un même sujet, on obtient un taux de reconnaissance de plus de 90%.
- Lorsque les données de formation et les données d'essai sont celles de différents sujets, la performance est réduite.
- Le taux de reconnaissance a été amélioré en mélangeant les données de deux sujets d'apprentissage.

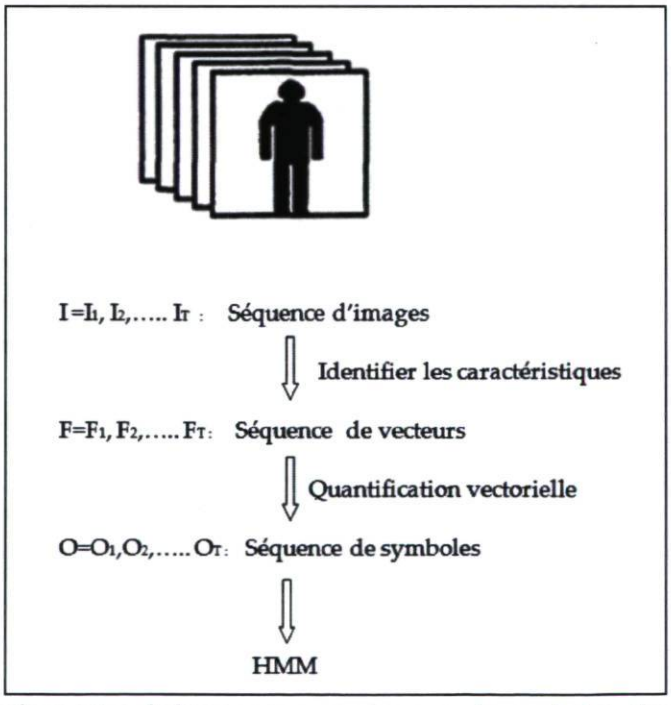

Figure 15 : Transformation de l'image en une séquence de symboles. (Source : [10])

3.6. Modèle d'utilisateur dans un jeu (Joueur)

Dans cette section nous abordons le sujet de la modélisation d'un utilisateur de jeu.

Le modèle de l'utilisateur correspond à l'ensemble des informations nécessaires pour prédire des comportements d'utilisateur soumis à un ensemble de stimulations possibles.

Le modèle de l'utilisateur devrait prendre en compte les motivations générales du joueur. Ainsi, il dépend d'un ensemble de paramètres qui peuvent être soit statiquement définis par le concepteur de jeu, ou bien modifiés par les changements d'états de l'utilisateur. Les paramètres pourraient :

1- Être une conséquence d'une hypothèse proposée par le concepteur du jeu et des informations sur les joueurs, ou
2- Découler des actions et des réponses du joueur en temps réel, ou
3- Provenir d'une analyse complète de l'action du joueur dans le jeu. [14]

Ces données peuvent être plus ou moins personnalisées, ce qui conduit à trois niveaux de modèles d'utilisateur.

3.6.1 Modèle générique (GM)

Un modèle générique de l'utilisateur ne distingue pas un joueur d'un autre. Il repose sur une hypothèse générale d'un joueur (par exemple, 65% hommes, 35% femmes, 34 ans d'âge moyen, occidental...). [14]

3.6.2 Modèle localisé (LM)

Un modèle localisé d'utilisateur (LM) ajoute au modèle générique de la connaissance de localisation de l'utilisateur. Cette information est interprétée par le jeu grâce à d'autres variables contextuelles du monde réel connu par le jeu. Par exemple, le jeu peut savoir, à partir de l'emplacement de l'utilisateur et une carte du monde réel, que le joueur est dans une cafétéria. [14]

3.6.3 Modèle personnalisé (PM)

Le troisième niveau de modèle utilisateur est le modèle personnalisé d'utilisateur (PM) qui garde les variables d'état complexes sur chaque utilisateur. Certaines de ces variables sont déjà utilisées dans les jeux classiques. Par exemple, on utilise parfois différents types de défis qui correspondent au niveau de compétence de l'utilisateur. Dans les MMORPG (*Massively Multiplayer Online Role-Playing Games*), chaque utilisateur a un profil qui définit son type d'avatar, son niveau dans le jeu et certaines données utilisées pour faciliter les relations sociales. Mais dans des environnements de réalité mixte, des données beaucoup plus complexes peuvent être utilisées: état civil, les habitudes personnelles, les relations sociales ...

Certaines de ces données doivent être fournies par l'utilisateur lui-même, et certaines peuvent être déduites de ses actions dans le jeu. [14]

3.7. Techniques de modélisation de l'utilisateur

Les principales techniques que l'on retrouve dans la littérature pour modéliser un utilisateur de jeu vidéo sont les suivantes.

3.7.1 Fonctions de l'évaluation

La fonction d'évaluation est utilisée dans la recherche heuristique. Le modèle de l'adversaire peut se concentrer sur les préférences du joueur. [15] Chaque position du jeu a une valeur en fonction de ces préférences. Cette valeur est codée dans une fonction d'évaluation statique qui peut être fabriquée à la main (sur la base des connaissances du domaine) ou apprise en utilisant les techniques d'apprentissage automatique.

3.7.2 Réseau de neurones

Un réseau de neurones est un modèle de calcul dont la conception est inspirée du fonctionnement des neurones biologiques. Les préférences de l'adversaire peuvent être, par exemple, représentées par le réseau de neurones. Elles peuvent être tirées d'épisodes de jeu. Le réseau de neurones pourrait également représenter l'ordre préféré de mouvements et de la difficulté des positions pour le joueur réel. [15]

3.7.3 Scripting

Les développeurs des jeux vidéo ont souvent recours à des techniques à base de règles sous forme de scripts. Les programmeurs définissent les ensembles de règles pour contrôler le comportement de l'IA.

Le principal avantage de cette approche est que les scripts sont faciles à comprendre. D'autre part, ces ensembles règles sont souvent vastes et complexes. En outre, parce que les scripts sont statiques, les agents font souvent des erreurs. Pour éviter ces problèmes, le jeu devrait avoir des capacités d'apprentissage automatique.

Les agents devraient être en mesure de corriger leurs erreurs, de changer leur comportement, une solution prometteuse est le script dynamique. [15]

3.7.4 Scripting dynamique

Cette technique utilise des règles de bases de connaissances du domaine, un ensemble pour chaque type d'adversaire. À partir de ces règles de bases, les nouveaux scripts sont générés. Quand l'adversaire apparaît, les règles sont extraites. Les probabilités de sélection plus élevées sont affectées à ces règles. Les chances d'être sélectionnés sont mises à jour après chaque rencontre de l'adversaire. Pour cette raison, la base de règle permet d'optimiser rapidement la génération des scripts. [15]

Chapitre 4

Pacman : Un banc d'essai pour nos expérimentations

Dans cette section, nous présentons le jeu que nous avons utilisé pour mener notre expérimentation. Il s'agit de Pacman, un jeu que nous avons choisi, car il est en temps réel et il comporte des changements d'état des principaux personnages (le Pacman et les fantômes). Dans ce projet, nous entreprenons les tâches de a) prédire la stratégie utilisée pour contrôler le Pacman et b) prédire l'identité du joueur qui joue une partie de Pacman.

4.1. Introduction

Pacman est un personnage en forme de rond jaune doté d'une bouche qui se trouve dans un labyrinthe. Pour accumuler des points, il doit manger toutes les pastilles avant que les fantômes ne l'attrapent. Certaines pastilles rendent Pacman invulnérable et lui permettent de détruire les fantômes. De plus, des bonus sous forme de fruits et d'une clé apparaissent au fur et à mesure de l'avancement du jeu.

Le but du jeu Pacman est de manger 244 pastilles de la même couleur et 4 Pac-gommes, tout en évitant les quatre fantômes. Ceci permet de passer de niveau à l'autre jusqu'au dernier.

Le tableau suivant présente les différents fruits, leur valeur et les niveaux dans lesquels on peut les trouver dans le jeu standard :

Fruits	Cerise	Fraise	Orange	Pomme	Melon	Galaxian	Cloche	Clé
Points	100	300	500	700	1000	2000	3000	5000
Niveaux	1	2	3 et 4	5 et 6	7 et 8	9 et 10	11 et 12	13 et...

Tableau 1: Les différents fruits et leurs valeurs pour chaque niveau (Wikipédia, 2010)

Pour minimiser la complexité du jeu, nous avons utilisé dans notre banc d'essai un seul fruit (l'orange). Tel qu'illustré dans le tableau 2, sa valeur est de 750 points multipliés par le niveau de jeu. Si le joueur dépasse les 10 000 points, il obtient une vie supplémentaire.

Fruits	Orange	Orange	Orange	.....	Orange
Points	750	1500	2250		750 × n
Niveaux	1	2	3		n (niveau)

Tableau 2 : Les points pour chaque niveau

4.2. Description du jeu

4.2.1. Le Pacman

Le Pacman est représenté par 5 formes qui désignent les différentes directions qu'il peut prendre.

				
Gauche	Droite	Haut	Bas	Standard (en arrêt)

Tableau 3 : Les formes du Pacman

4.2.2. Les monstres

Les monstres utilisent 2 formes selon leur état :

- Normal, lorsqu'ils poursuivent et mangent le Pacman.

- Bleu, lorsqu'ils peuvent se faire manger par le Pacman.

Les autres éléments utilisés au cours du jeu

- Le point, ▪ qui rapporte des points lorsque le Pacman lui passe dessus.
- Le mur,  qui sert pour le décor.
- La pièce (les Pac-gommes)  , qui rend les monstres vulnérables.
- Le bonus,  qui rapporte une vie supplémentaire.

4.3. Labyrinthe de Pacman

On représente le labyrinthe sous la forme d'un tableau de 28 colonnes et de 31 lignes, c.-à-d 868 cellules. La figure 16 montre une structure possible de labyrinthe de Pacman.

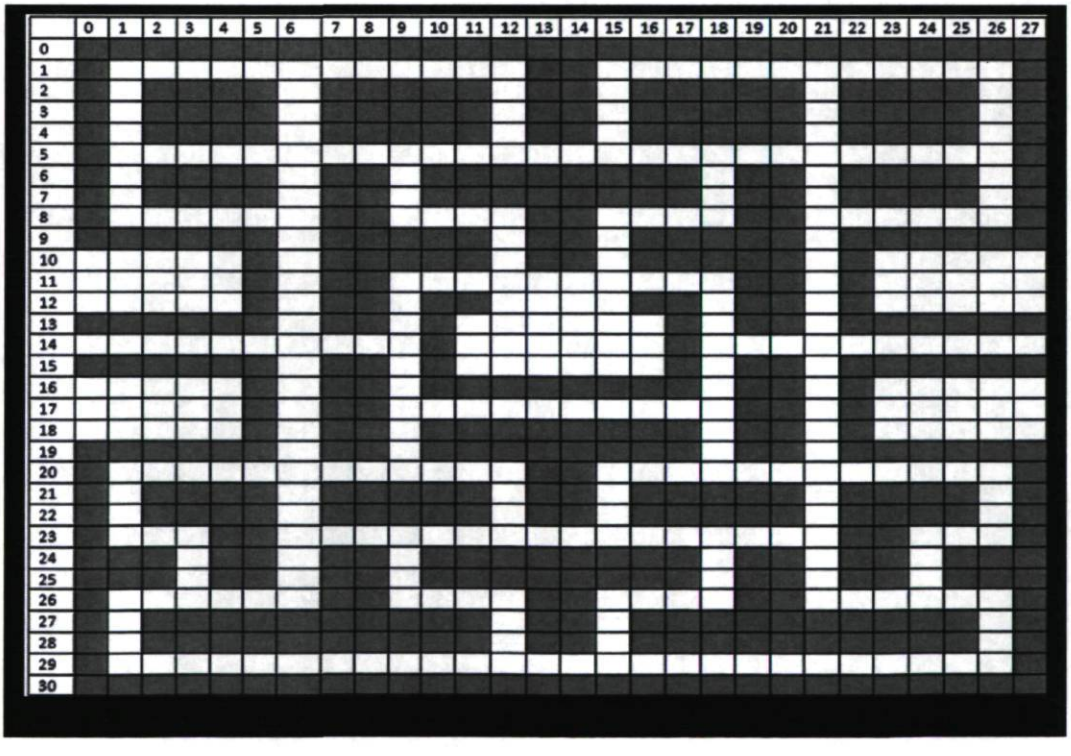

Figure 16 : Labyrinthe de Pacman

4.4. Description générale du jeu Pacman

4.5.1 La modélisation du jeu de Pacman

Dans ce jeu, le rôle du joueur est de déplacer le Pacman dans un labyrinthe et de lui faire ramasser les Pac-gommes et les fruits qui s'y trouvent en évitant d'être touchés par des fantômes.

Les paramètres principaux qui décrivent l'état du jeu sont les suivants :

- La position de chacun des personnages du jeu;
- L'orientation de chacun des personnages du jeu;
- L'état des monstres;
- Le nombre de points restant et leur position;
- Le nombre de Pac-gommes et leur position;
- Le nombre de bonus et leur position.

Pour faciliter la modélisation du jeu et permettre le contrôle des déplacements du Pacman dans le labyrinthe, nous avons adopté une représentation relative du jeu. Une représentation relative n'utilise pas les positions absolues des objets sur le plateau de jeu, mais repose plutôt sur La distance de Manhattan entre les objets. Ceci permet d'exploiter les symétries du jeu et de généraliser différentes configurations similaires.

Dans ce mode de représentation, nous avons retenu les attributs suivants :

- la distance entre Pacman et le plus proche fantôme en fuite;
- la distance entre Pacman et le plus proche fantôme dangereux;
- la position du Pacman par rapport aux Pac-gommes;
- le nombre des Pac-gommes restantes;
- la direction du Pacman.

Avant de choisir ces paramètres, nous avons fait plusieurs essais pour obtenir ces derniers.

4.5.2 Les stratégies pour jouer à Pacman

Les différentes stratégies dans le jeu Pacman sont :

- **L'exploration (*Explore*) :** le Pacman fonce vers la Pac-gomme (super-Pac-gomme ou la pastille) la plus proche. Donc le Pacman doit essayer de manger les Pac-gommes tout en gardant une distance entre lui-même et les fantômes.

- **L'attaque (*Attack*) :** il fonce vers le fantôme le plus proche dans le labyrinthe. Quand Pacman mange une super-Pac-gomme, les fantômes changent leur couleur en bleu, ils deviennent vulnérables et le Pacman peut changer sa stratégie vers l'attaque. C'est le seul état dans lequel le Pacman peut manger les fantômes.

- **La fuite (*Retreat*) :** le Pacman s'éloigne des fantômes. Quand la distance entre Pacman et les fantômes est petite, l'objectif de Pacman est de trouver une solution pour s'éloigner le plus vite possible. Sinon les fantômes réussissent à le manger et il perd une vie.

L'application de ces stratégies varie d'un joueur à l'autre. Par exemple, un joueur expérimenté aura plus de facilité à faire de l'exploration et éviter de devoir fuir à l'approche des fantômes. Donc nous supposons que chaque joueur aura des patrons particuliers de stratégies qui le distinguent des autres joueurs.

4.5.3 Modélisation des décisions du Pacman à l'aide d'un diagramme état-transition

Un diagramme état-transition sert à représenter les états d'un objet ainsi que les transitions entre ces états. Dans le cadre du jeu Pacman, ce type de diagramme nous donne la possibilité de modéliser le processus décisionnel du Pacman.

Dans la formulation de diagramme que nous avons retenue, les états correspondent aux stratégies (figure 17) Donc trois états sont possibles : exploration, attaque et fuite. Et, le changement d'un état vers un autre correspond à certaines conditions de jeu.

Entre chaque paire d'états, nous avons deux transitions (une dans chaque direction). En tenant compte de la possibilité de rester dans un même état (les transitions récurrentes T7, T8 et T9), nous obtenons neuf transitions possibles au total. Les transitions possibles du Pacman sont définis par des conditions suite à une situation précise, c'est-à-dire il faut certaines conditions pour le Pacman déplace d'exploration vers l'attaque, ou d'exploration vers la fuite ou de rester dans d'exploration, parmi ces conditions : la position de Pacman par rapport les fantômes, la position du Pacman par rapport aux grandes pastilles, le nombre de grosses pastilles restantes.

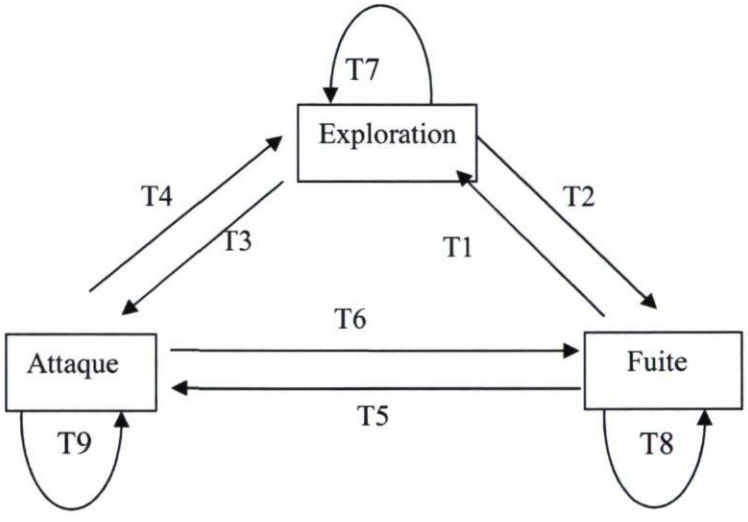

Figure 17 : Modélisation du Pacman par un FSM

4.5.4 Modélisation des décisions du Pacman à l'aide d'un HMM

Nous proposons dans ce qui suit un modèle de Markov caché qui représente le comportement d'un joueur. Comme les stratégies du joueur ne sont pas explicites durant le déroulement d'une partie, elles sont représentées par les états cachés. Et pour chaque état (stratégie), nous avons des observations qui correspondent à des configurations de jeu et à la position du Pacman dans cette configuration.

Les transitions entre les stratégies du modèle HMM sont illustrées à la figure 18. Nous avons retenu une topologie de modèle totalement connecté, afin de permettre les transitions entre toutes les stratégies. L'approche pour définir les probabilités de chacune des transitions sera abordée au chapitre 5 de ce document. Le lecteur remarquera que les mouvements possibles du Pacman ne sont pas décrits explicitement dans le modèle HMM mais sont plutôt représentés par des probabilités de transition d'un état vers un autre.

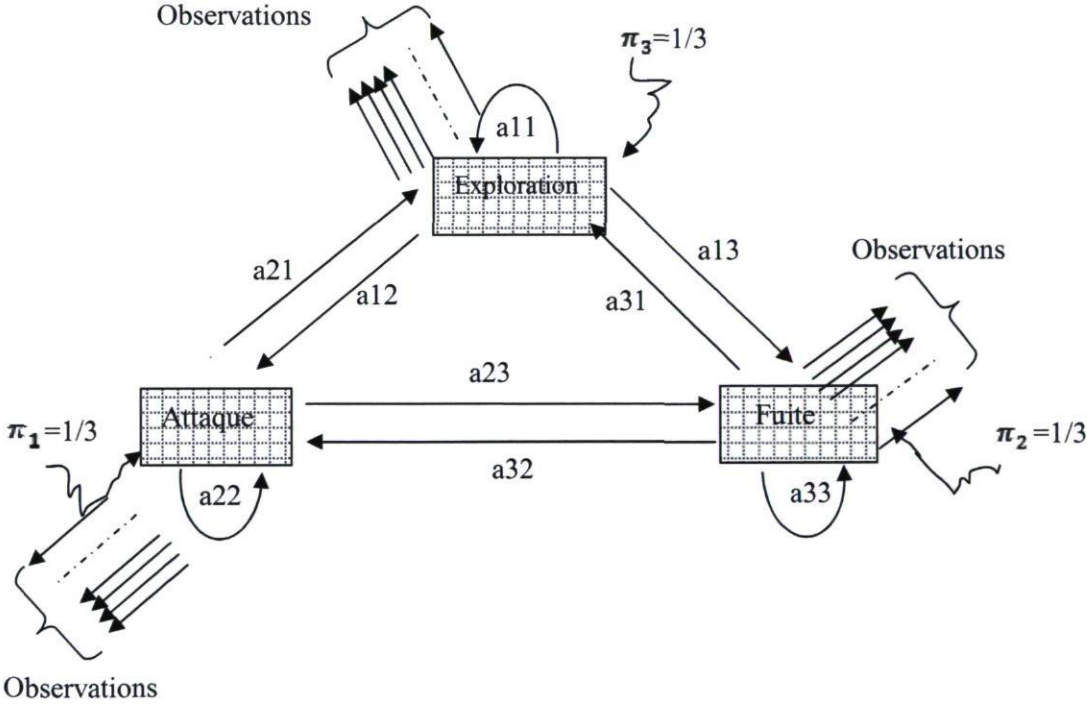

Figure 18 : Modélisation du Pacman par le modèle HMM

Les paramètres du modèle sont les suivantes :

- Probabilité initiale : π_i.
- Matrice de transition : est définie par la probabilité de passage (a_{ij}) d'une stratégie à l'autre (état à l'autre).
  (Exploration, Attaque et Fuite : sont les stratégies utilisées)

- Matrice d'observations : elles désignent la probabilité des éléments observables dans le modèle. Ces éléments sont décrits dans les prochains paragraphes.

Les observations

Une observation représente l'état du jeu et est décrite à l'aide d'un vecteur de six paramètres. Ces paramètres sont définis dans le Tableau 5. Pour déterminer cette représentation d'observations, nous avons fait plusieurs tests sur le jeu Pacman pour identifier les éléments qui ont un effet significatif sur l'état du jeu.

Attribut	Exemple de valeurs	Description
Distance_Fantôme_Pacman_Attaque	-1 : INDÉFINI Valeurs : 0, 1…	Distance entre Pacman et le fantôme le plus dangereux et le plus proche.
Distance_Fantôme_Pacman_Fuite	-1 : INDÉFINI Valeurs : 0, 1…	Distance entre Pacman et le fantôme le plus proche en fuite.
Distance_Pacman_Pacgum	-128 : INDÉFINI Valeurs : 0, 1…	Distance à la plus proche grosse pastille
Nombre_Pacgum	Valeurs : 0, 1, 2, 3, 4	Nombre de grosses pastilles restantes
Nombre_Pastilles	Valeurs : 0,..244	Nombre de pastilles restantes
Direction_Pacman	Valeurs : 0, 1, 2,3, 4	Direction possible pour Pacman

Tableau 4 : description détaillée des observations.

La description de chacun des paramètres est comme suit :

- *Distance_Fantôme_Pacman_Attaque* : dans le cas où il existe un fantôme dangereux qui est proche du Pacman, on calcule la distance de ce dernier par rapport au Pacman.

- *Distance_Fantôme_Pacman_Fuite:* dans le cas où il existe un fantôme vulnérable qui est proche du Pacman, on calcule la distance de ce dernier par rapport au Pacman.
- *Distance_Pacman_Pacgum* : c'est la distance entre Pacman et la grosse pastille la plus proche de lui.
- *Nombre_Pacgum:* c'est le nombre des grosses pastilles qui reste dans le jeu. La valeur varie entre 0 et 4.
- *Nombre_Pastilles* : c'est le nombre de pastilles qui reste dans le jeu, qui varie entre 0 et 244.
- *Direction_Pacman:* ce sont les directions possibles pour le Pacman.

  0 : Pacman reste dans la même position (stable)

  1 : direction du Pacman vers le haut

  2 : direction du Pacman vers le bas

  3 : direction du Pacman vers la gauche

  4 : direction du Pacman vers la droite

4.5. Conclusion

Dans ce chapitre, nous avons présenté le jeu Pacman que nous utilisons comme banc pour mener nos essais. Nous avons également donné une description détaillée du jeu et de la modélisation de comportement de Pacman par un diagramme états-transitions et par un HMM.

Dans le prochain chapitre, nous décrivons les expérimentations que nous avons menées à l'aide de notre modèle HMM et les résultats que nous avons obtenus.

Chapitre 5

Modélisation d'un joueur de jeu vidéo à l'aide de HMM

5.1. Introduction

Nous abordons dans ce chapitre le problème de la reconnaissance d'un joueur par l'utilisation de modèles HMM. Tel qu'indiqué au chapitre précédent, nous allons utiliser Pacman comme banc d'essai.

Les objectifs que nous tentons d'atteindre sont :

- La construction de modèles de Markov caché à partir d'épisodes de jeux;
- L'apprentissage des modèles via l'algorithme Baum Welch; et
- La reconnaissance d'un joueur à partir d'épisodes anonymes de jeu.

Nous présentons dans ce qui suit la modélisation d'épisodes de jeu sous forme d'HMM. Nous décrivons les expérimentations que nous avons menées sur plusieurs parties de jeux. Et finalement, nous présentons des résultats empiriques qui indiquent le taux de reconnaissance d'un joueur que nous avons atteint pour différentes parties de jeu.

5.2. Approche expérimentale

Tel qu'illustré à la figure 19, l'approche que nous préconisons est constituée de deux parties. Une première vise à récolter des données qui sont utilisées pour modéliser le comportement décisionnel d'un joueur. Ces données sont saisies à partir d'épisodes de jeux pour un même joueur. La deuxième partie consiste à utiliser ces modèles pour tenter de reconnaitre la stratégie et l'identité d'un joueur.

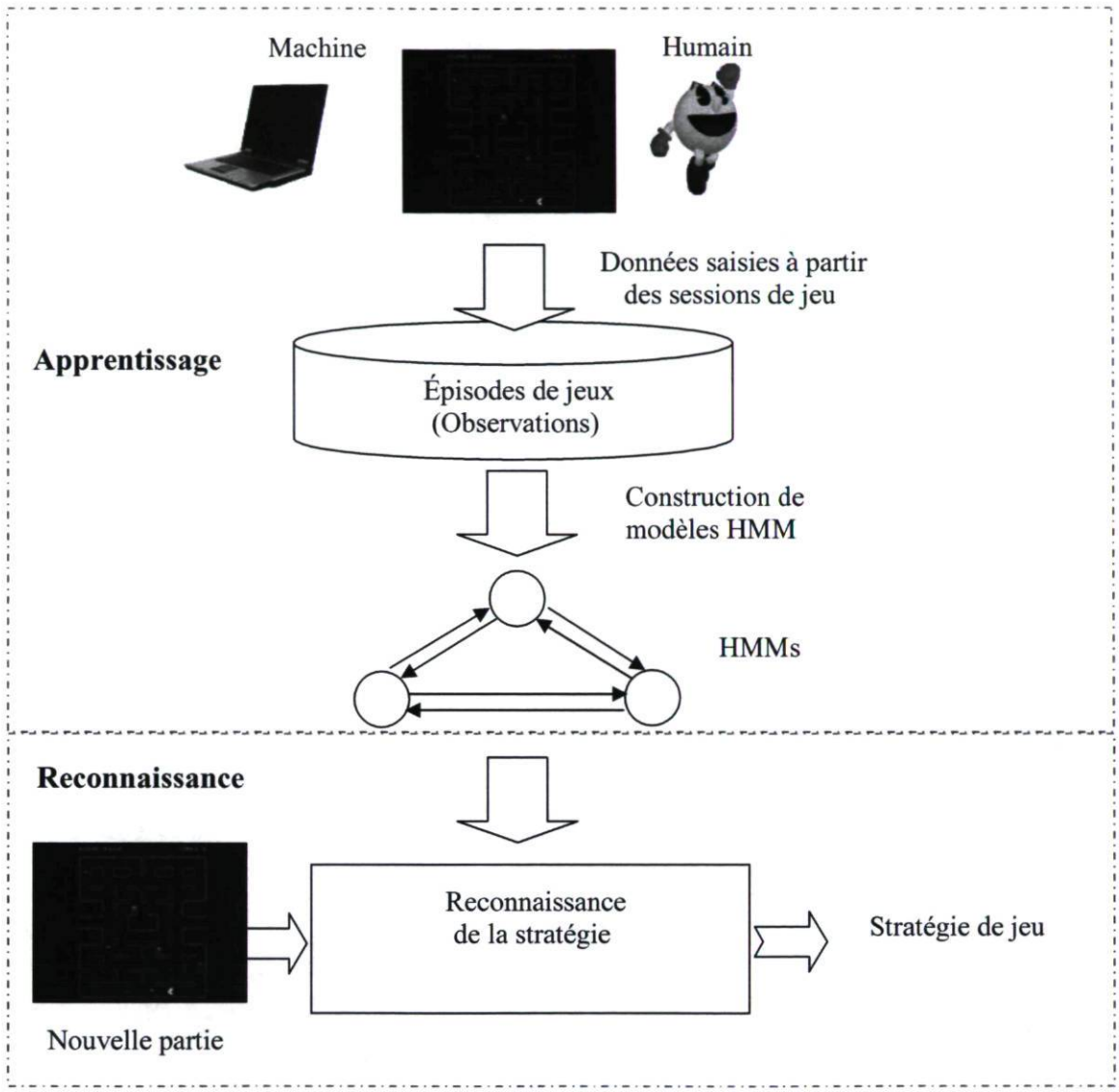

Figure 19 : Système de reconnaissance de stratégie quelque soit l'utilisateur

Pour l'apprentissage de modèles HMM, nous avons considéré deux scénarios :

- La construction de modèles avec des informations complètes où les joueurs indiquent leur intention à chaque mouvement. Ainsi, pour chaque décision, nous avons l'état exact du jeu, c.-à-d. la position des agents/objets sur le labyrinthe et la stratégie adoptée par le joueur qui contrôle le Pacman.

- La construction de modèle lorsque la stratégie du joueur est inconnue. Dans ce cas-ci, il est nécessaire d'extrapoler la stratégie préconisée afin de décrire complètement un HMM.

Le premier scénario, avec informations complètes, permet de construire des modèles qui devraient représenter fidèlement le contexte de décision d'un joueur. Ceci permet de valider l'approche HMM dans des conditions favorables à l'apprentissage. Cependant, comme il est difficile en pratique d'obtenir des informations exactes sur l'intention d'un joueur, les modèles obtenus dans le deuxième scénario nous permettent de tester nos approches dans un contexte de jeu plus réaliste.
Pour la partie portant sur les tâches de reconnaissance, nous avons adopté deux techniques différentes pour la reconnaissance de la stratégie d'un joueur et pour déterminer son identité. Ces techniques sont décrites dans les dernières sections de ce chapitre.

5.3. Construction de HMM avec informations complètes

Dans cette section, nous présentons les principales étapes pour produire les HMM à partir d'épisodes de jeux.

5.3.1. Cueillette des données

Pour construire un modèle HMM, nous utilisons des traces obtenues lors de parties jouées par un joueur humain ou par une machine. On laisse l'automate ou le joueur humain jouer des parties et on sauvegarde les observations dans un fichier. Dans le scénario d'informations complètes, c'est le joueur qui définit la stratégie utilisée. Ainsi, les données recueillies pour chaque changement de position du Pacman sont :

- la distance entre Pacman et le plus proche fantôme en fuite,
- la distance entre Pacman et le plus proche fantôme dangereux,

- la position du Pacman par rapport aux Pac-gommes,
- le nombre des Pac-gommes restantes,
- la direction du Pacman,
- la stratégie du Pacman (soit l'exploration, l'attaque, la fuite).

Nous avons mené des essais avec deux types de joueur :

- un Pacman contrôlé par un FSM (diagramme état-transition);
- des joueurs humains de différents niveaux.

Dans le cas du FSM, l'information sur la stratégie est directement accessible puisqu'elle est encodée dans la structure du diagramme. Il n'y a donc pas de traitement supplémentaire à effectuer pour compléter les données.

Cependant, dans le cas du joueur humain, il a fallu ajouter des fonctionnalités au jeu pour que le joueur puisse indiquer la stratégie qu'il adopte. Pour cela, 3 touches du clavier ont été redéfinies pour correspondre à chacune des stratégies. Ainsi, pour contrôler le Pacman, le joueur doit indiquer la direction choisie et ainsi que la stratégie utilisée. Bien que ce schéma puisse comporter des erreurs, il donne quand même une bonne indication des changements d'état dans l'attitude du joueur.

5.3.2. Élimination des redondances dans les données

Comme les traces de jeu comportent plusieurs milliers d'étapes, on retrouve plusieurs fois les mêmes situations dans les données. Nous devons donc faire un prétraitement pour éliminer les doublons et ainsi construire un ensemble d'observations uniques.

Nous faisons donc un compactage de nos données. L'objectif est de générer un ensemble d'observations qui servira de référence pour la construction des modèles.

Pour chaque observation du fichier d'observations initiales, nous vérifions si elle est présente dans le nouveau fichier. Si elle n'y est pas, alors on l'ajoute (figure 20.) On élimine donc les redondances et nous nous assurons que chaque observation n'existe

qu'une fois seulement dans le nouveau fichier d'observations étiquetées qui est utilisé pour la construction des modèles.

Cette première étape permet d'obtenir un ensemble d'observations de taille suffisamment raisonnable pour permettre d'exécuter efficacement les différents algorithmes HMM nécessaires au projet.

Pour la plupart des essais que nous avons menés, on note que la taille de l'ensemble d'observations est réduite de plus de 65%. Certains fichiers comptent un ensemble de plus de 800 observations. Il est important de noter qu'à partir de différents fichiers de données, nous obtenons des ensembles d'observations différents.

observation	*stratégie courante*
1 9 2 8 2 2	*Explore*
5 1 6 1 0 2	Attack
2 1 3 1 0 1	Explore
1 9 2 8 2 2	Attack
2 5 3 1 0 2	Retreat
1 9 2 8 2 2	*Explore*
2 5 3 1 0 2	Retreat
1 9 2 8 2 2	**Retreat**
2 5 3 1 0 2	Retreat

Redondance dans le fichier d'observation

N° d'observation	*vecteur observation*
Obs-0	-1 -1 3 14 1 2
Obs-1	1 4 2 8 4 2
…	…
Obs-x	**1 9 2 8 2 1**
…	…
Obs-873	1 10 2 5 2 1

Fichier d'observation étiquetées et compactées

Figure 20 : Illustration du compactage d'observations

5.3.3. Calcul des paramètres du modèle HMM

Par la suite, à partir des données recueillies et nettoyées, nous devons construire le modèle HMM qui sera utilisé pour reconnaître la stratégie de jeu.

Le jeu de données contient des séquences d'observations O qui serviront à déterminer les paramètres du modèle HMM. Tel qu'indiqué précédemment, les observations du modèle HMM correspondent à la description de la situation sur le labyrinthe à chaque pas de temps. Dans le scénario avec données complètes, les stratégies sont indiquées dans les données recueillies et elles correspondent aux trois états S de notre modèle (Exploration, Attaque et Fuite).

Tel que présenté au chapitre 2, les paramètres nécessaires pour construire un HMM sont :

- Les probabilités de transition A, qui correspondent à la probabilité de passer d'une stratégie à une autre, c.-à-d. P(stratégie_courante | stratégie_précédente) (figure 21).

Stratégie Précédente \ Stratégie Courante	*Explore*	*Attack*	*Retreat*	
Explore	0.956	0.020	0.024	Σ =1
Attack	0.096	0.889	0.015	Σ =1
Retreat	0.102	0.008	0.890	Σ =1

Figure 21 : Matrice de transition (A)

- Les probabilités initiales du modèle π (figure 22). Le jeu Pacman a la particularité que le joueur commence toujours par la stratégie *Exploration*. Donc la probabilité initiale d'exploration est égale à 1. Le vecteur résultant est $\pi = [1, 0, 0]$.

	Probabilité initiale
Explore	1
Attack	0
Retreat	0

$\Sigma = 1$

Figure 22 : Probabilités initiales

- Les probabilités d'émissions B : elles définissent la probabilité d'avoir obtenu l'observation « O »sachant que nous avons adopté une stratégie (figure 23), donc $P = (\pi \mid \text{stratégie})$.

	Explore	*Attack*	*Retreat*
Obs 0	0.16	0.051	0.03
Obs 1	0.22	0.0	0.016
Obs 2	0.0	0.15	0.0
…	..	..	..
Obs 873	0.042	0.04	0.006

Figure 23 : Matrice d'observations (B)

Comme toutes les informations sont disponibles, nous construisons ces matrices en comptant les occurrences d'états et d'observations dans nos données. Les formules pour déterminer les paramètres du modèle sont: où

$$\pi_i = \text{Probabilité d'être dans l'état } S_i \text{ à l'instant } t = 1$$

$$a_{i,j} = \frac{\text{nombre de transition de l'état } S_i \text{ vers l'état } S_j}{\text{nombre de fois où l'on quitte l'état } S_i}$$

$$b_i(j) = \frac{\text{nombre d'apparitions simultanées de l'état } S_j \text{ et du symbole } v_j}{\text{nombre d'apparitions de l'état } S_j}$$

stratégie précédente		*observation*	*stratégie courante*
NB (Attack) = 1 -->	Attack	2 1 0 1 0 1	Explore
	Explore	5 2 6 1 0 3	Attack
NB (Attack) = 2 -->	Attack	5 1 6 1 0 2	Attack
NB (Attack) = 3 -->	Attack	2 1 3 1 0 1	Explore
	Explore	2 5 3 1 0 1	Retreat
	Retreat	2 1 -3 1 0 2	Retreat

NB (Explore | Attack) = 1

NB (Attack | Attack) = 1

NB (Explore | Attack) = 2

NB (Retreat | Attack) = 0

Figure 24 : Calcul de la matrice de transition à partir du fichier d'observation

Par exemple, la figure 24 illustre le calcul de la matrice de transition A. La probabilité de transition $P = (Explore \mid \text{Attaque})$ est égale au nombre de fois que la stratégie précédente était « Attaque » et que nous sommes passées à la stratégie courante « Explore », le tout pondéré par le nombre total de fois que la stratégie précédente était « Attaque ». Dans cet exemple, la valeur de probabilité serait de 2/3.

Le calcul de la matrice d'émission B est similaire. Cette matrice contient les probabilités que le joueur émette l'observation X lorsqu'il adopte une stratégie particulière.

5.4. Apprentissage de HMM avec information incomplète

Afin de tenir compte du contexte plus réaliste où les intentions du joueur ne sont pas connues lors de la cueillette des données, nous avons également exploré comment construire un HMM lorsque nous ne connaissons pas la stratégie préconisée.

Les étapes pour construire le HMM sont les mêmes que celle utilisée dans le cas où nous avons des informations complètes. Cependant, comme nous n'avons pas d'informations sur la stratégie, il n'est pas possible de construire directement le HMM simplement par un compte des occurrences d'observations et d'états comme nous l'avons fait à la section 5.3.3.

Donc nous avons utilisé l'algorithme de Baum Welch pour apprendre les paramètres du modèle HMM. Tel que décrit au chapitre 2, cet algorithme permet de faire de l'apprentissage non supervisé et de déterminer les paramètres du modèle uniquement à partir des observations.

Ainsi, l'approche que nous avons préconisée pour construire les HMMs à partir d'informations incomplètes est la suivante:

- Nous avons demandé à différents joueurs humains de jouer 10 parties de Pacman. Nous avons également fait jouer 10 parties à un diagramme état-transition (FSM) qui contrôle le personnage de Pacman. Les séquences d'observations ont été enregistrées dans des fichiers différents pour chacune des parties.
- Pour chaque joueur, nous avons retenu la première partie comme partie de référence et nous avons construit le fichier d'observations étiquetées à partir des observations (voir section 5.3.2);
- Nous avons par la suite appliqué l'algorithme Baum Welch à des parties de chaque joueur pour construire un modèle HMM correspondant à chacun des joueurs. Nous utilisons pour nos expérimentations le logiciel Jahmm[1] qui comporte une implantation de cet algorithme.

Nous avons rencontré quelques difficultés dans l'application de Baum Welch à nos séquences de jeu. Une première difficulté est que le résultat est sensible à la longueur des séquences des observations. Comme nous avons plusieurs observations pour

[1] Jahmm est une librarie de différents algorithmes reliés au HMM. Ce logiciel, développé en Java, est disponible à l'adresse suivante : http://www.run.montefiore.ulg.ac.be/~francois/software/jahmm/

chaque séquence et que la somme des probabilités de ces observations doit être égale à 1, nous obtenons des petites valeurs. Et comme le calcul de certaines étapes de Baum Welch nécessite la multiplication de valeurs probabilités, les résultats obtenus sont de très petites valeurs proches de zéro qui mettent à l'épreuve le calcul de précision du langage java. Nous avons réussi à surmonter cette difficulté en modifiant la longueur des séquences utilisées pour l'entraînement et en modifiant le traitement des valeurs réelles de quelques opérations de l'algorithme.

Une autre difficulté est de traiter les observations qui ne sont pas présentes dans le fichier d'observations étiquetées. Ceci est possible, car ce fichier ne comporte que les observations de la première partie de chaque joueur. Et ces observations ne sont qu'un sous-ensemble des quelques millions de situations possibles du jeu Pacman. Nous pourrions assigner une probabilité de 0 à une observation non présente dans le ficher, mais le résultat de l'apprentissage sera également 0, car il repose sur des multiplications de probabilités. Nous pourrions également considérer mettre toutes les observations possibles dans le fichier. Mais cette solution s'avère impossible, tant par les contraintes d'espace mémoire que par les temps de traitement qui seraient nécessaires pour exploiter cette structure.

Nous avons donc choisi d'adopter une approche où nous approximons l'espace d'observation en extrapolant sur les instances contenues dans le fichier d'observations étiquetées. Cette approche est décrite dans la prochaine section.

5.5. Réduction de l'espace d'observations par KNN

Puisque l'espace des observations est très grand, et pour essayer de réduire la complexité du problème, nous utilisons une représentation approximative des observations. Pour diminuer cet espace d'observations, nous faisons l'utilisation de l'algorithme KNN.

Étant donnée une nouvelle observation, l'algorithme KNN consiste à chercher dans le fichier d'observations étiquetées les k instances les plus similaires à cette observation. Il est alors possible de remplacer dans nos traitements la nouvelle observation par son plus proche voisin. Également, lorsqu'une classe est nécessaire (par exemple la stratégie associée à une observation), il est possible d'affecter la classe qui est la plus représentée dans les k instances les plus similaires.

Cette approche nécessite une mesure de similarité ou de distance entre les instances. Les mesures les plus souvent utilisées pour la distance $dist(x_i, x_j)$ entre la nouvelle observation et un voisin sont :

- La distance euclidienne (valeurs continues)

$$dist(x_i, x_j) = \sqrt{\sum_{r=1}^{d} (a_r(x_i) - a_r(x_j))^2}$$

- La distance de Manhattan (valeurs continues)

$$dist(x_i, x_j) = \sum_{r=1}^{d} \left| a_r(x_i) - a_r(x_j) \right|$$

- La distance de Hamming (valeurs discrètes)

$$dist(x_i, x_j) = \#\left\{ r \in d : a_r(x_i) \neq a_r(x_j) \right\}$$

Les valeurs a_r sont celles des différents attributs des instances. Dans notre cas, elles correspondent aux attributs décrivant une situation de jeu (voir section 5.3.1).

Dans la version que nous avons implantée, nous estimons la similarité entre deux observations comme une fonction inverse de la distance. Plus précisément, la formule utilisée est la suivante :

$$Similarité = 1/(1 + \text{distance})$$

Et la distance entre deux observations est donnée par la mesure de Manhattan.

5.6. Reconnaissance de la stratégie d'un joueur

Dans cette partie de nos expérimentations, notre objectif est de reconnaitre la stratégie utilisée par un joueur dans une nouvelle partie, et ce, à partir d'une séquence d'observations tirées de cette partie. Nous présumons à cette étape que nous connaissons l'identité du joueur. Nous détenons un modèle λ, construit par comptage (information complète) ou entraînement (information incomplète), qui décrit son comportement de jeu.

5.6.1 Approche de reconnaissance de stratégie

La méthode de reconnaissance de stratégie est basée sur l'algorithme Viterbi. Cet algorithme permet de trouver la séquence d'état qui est la plus probable étant donnée une séquence d'observations. Ainsi, nous utilisons Viterbi pour trouver la séquence optimale de stratégies qui correspond aux différentes situations de jeux rencontrées dans une partie.

La procédure que nous avons préconisée pour estimer la stratégie correspondant à une observation est la suivante (figure 25):

1. On fait la lecture d'une nouvelle observation à partir de l'environnement du jeu.
2. On trouve l'observation la plus similaire dans le fichier étiqueté à l'aide du KNN.
3. On place cette valeur dans la séquence d'observations et on utilise l'algorithme Viterbi (avec le HMM du joueur) pour évaluer la vraisemblance de chacune des stratégies pour la nouvelle observation.
4. On choisit la stratégie la plus probable.
5. On enregistre l'observation pour l'utiliser dans l'analyse des prochaines observations.

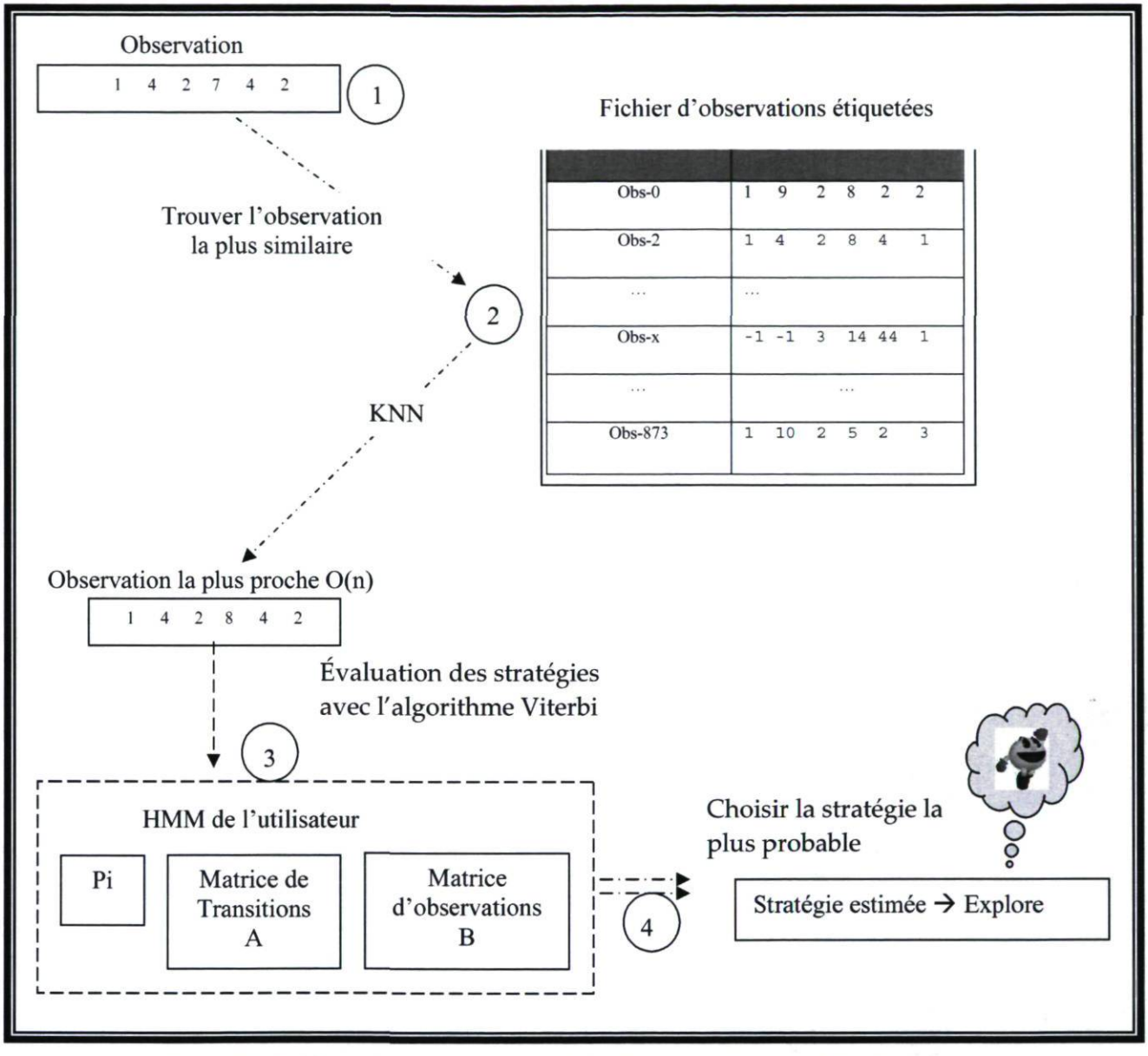

Figure 25 : Système de reconnaissance de la stratégie quelque soit l'utilisateur

La figure 26 illustre les résultats de la méthode de reconnaissance de stratégie. Nous prenons comme exemple la première observation (9 -1 7 3 2 3). La prochaine étape est de tester s'il existe dans le tableau des observations étiquetées. Sinon on cherche la plus similaire dans le fichier d'observations compacté. Dans notre cas, l'observation n'existe pas, et la plus similaire est l'observation (9 -1 7 4 2 3). Par la suite, on utilise

l'algorithme Viterbi pour savoir quelle stratégie est possible pour cette observation. Dans notre exemple les résultats nous donnent :

- Viterbi(*Explore*)=2.888E-27,
- Viterbi(*Retrait*)=0.0,
- Viterbi(*Attack*)=0.0,

Donc la stratégie choisie est *explore*, c.-à-d. celle qui a la valeur de probabilité la plus grande.

Finalement on teste la performance de reconnaissance pour déterminer si chaque stratégie sélectionnée est exacte et on cumule le taux de succès de reconnaissance. Dans cet exemple, le taux de reconnaissance est égal à 0.85 et il mesure la performance totale sur l 'ensemble des prédictions d'une partie.

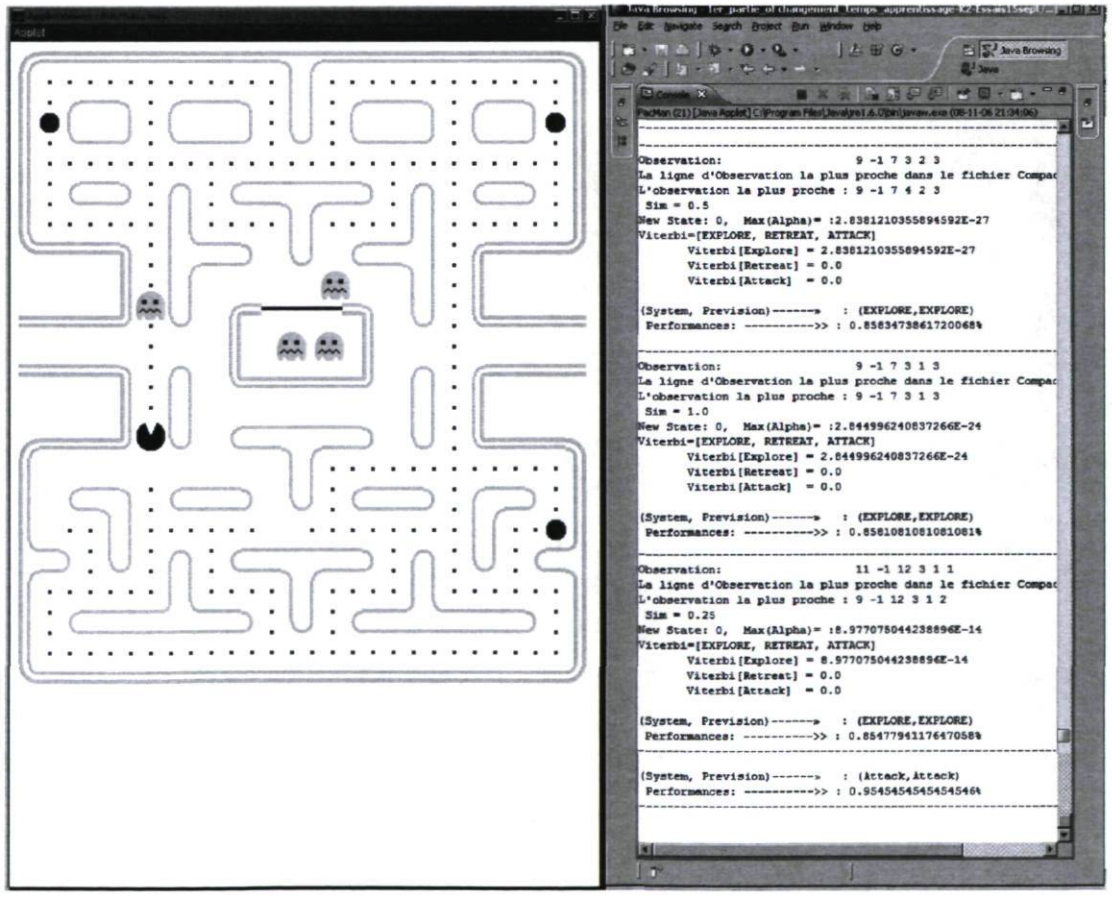

Figure 26 : Environnement du jeu

5.6.2 Expérimentations pour la reconnaissance de la stratégie

Dans cette partie nous présentons les résultats obtenus avec l'algorithme Viterbi pour prédire la stratégie utilisée par un joueur.

Pour mener nos expérimentations, nous avons fait jouer des parties par trois (3) joueurs :

- Machine : le diagramme état-transition (FSM) que nous avons décrit au chapitre 4.
- Joueur1 : un joueur humain qui est expérimenté à ce jeu
- Joueur2 : un joueur humain qui a un niveau différent (niveau inférieur) que le joueur1.

Pour mesurer l'efficacité de cette méthode, nous avons fait 30 parties pour chaque joueur, et pour chaque partie du jeu, nous avons pris le taux de reconnaissance moyen.

Pour plus de détails concernant les parties utilisées dans nos expérimentations, nous avons utilisé une longueur moyenne des parties de 3 à 5 minutes. Pour la machine, la comparaison est faite entre la reconnaissance de la stratégie par HMM et le FSM. Et pour les deux autres joueurs (les joueurs humains), la comparaison est faite entre l'HMM et la stratégie saisie par le joueur sur le clavier.

5.6.3 Résultats des expérimentations avec informations complètes

Dans cette section, nous présentons les résultats obtenus avec les modèles HMM estimé à partir de données complètes. Nous rappelons que ces modèles ont été construits par comptage des fréquences observés dans les données d'entraînement.

Les figures ci-dessous donnent un aperçu du taux de reconnaissance obtenu dans divers parties pour chacun de nos joueurs. Le taux de reconnaissance correspond au nombre de fois que la stratégie a été reconnue correctement sur le nombre totale

d'observations. Ceci correspond à la proportion de stratégies correctement prédites par Viterbi depuis le début d'une partie.

On note que le taux, pour l'ensemble des parties, se situe toujours entre 0,64 et 1. On note que la diminution est assez constante pour la plupart des parties. On remarque cependant qu'il y a une dégradation importante de performance vers la 100ième observation pour la partie 1 et vers la 300ème observation pour la partie 2 et vers la 80ième observation pour la partie 3. Ceci correspond à la stratégie utilisée au début qui est la plupart du temps la stratégie « *Explore* » pour le joueur et pour le HMM. C'est pour cela que le taux de performance est presque égal à 1 au début de la partie. Mais à partir la 100ième observation, le taux de reconnaissance diminué pendant un certain intervalle de temps, et il reste stable entre 0.82 et 0.64 pour le reste de la partie.

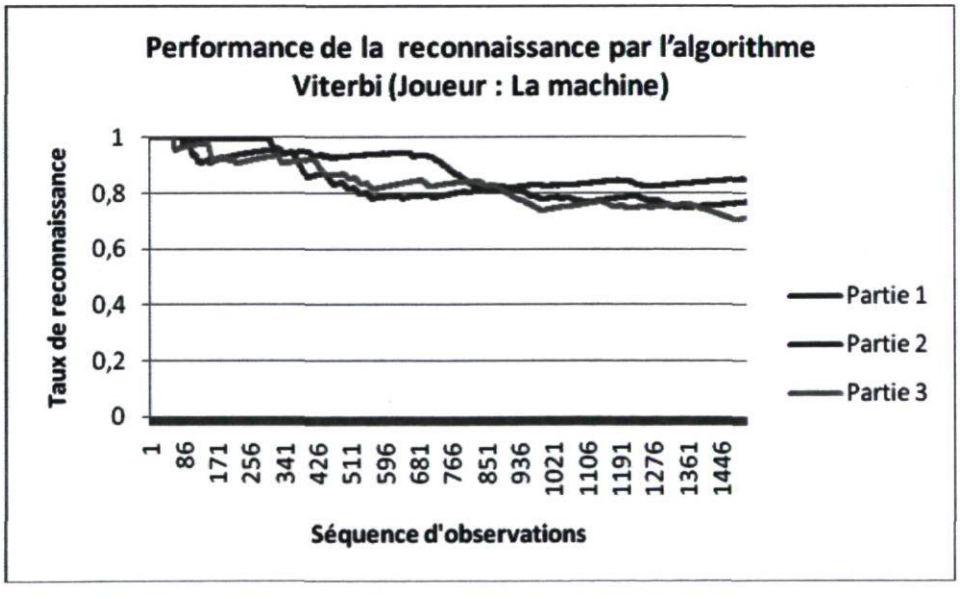

Figure 27 : Performance de la reconnaissance par l'algorithme Viterbi (machine)

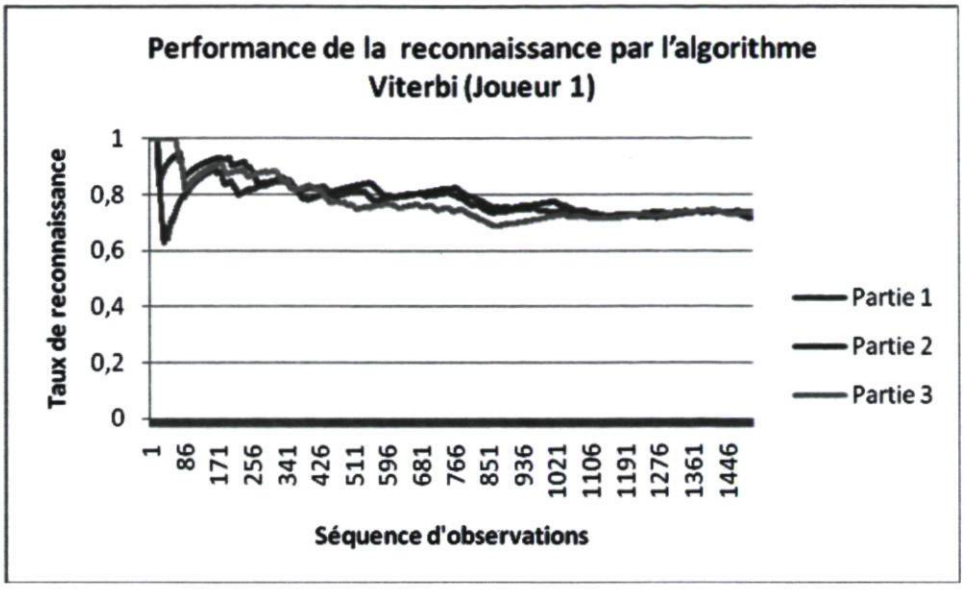

Figure 28 : Performance de la reconnaissance par l'algorithme Viterbi (Joueur 1)

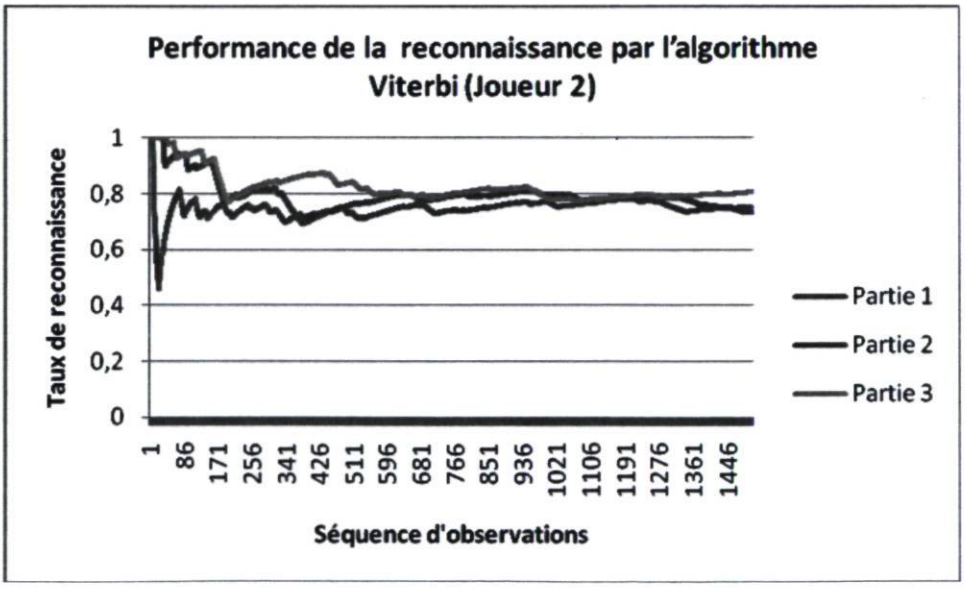

Figure 29 : Performance de la reconnaissance par l'algorithme Viterbi (Joueur 2)

Les tableaux (5,6 et 7) nous donnent une idée de la performance de l'algorithme pour chacune des stratégies. On y présente les 4 valeurs suivantes pour chaque stratégie :

- Le nombre de fois que la stratégie a été reconnue correctement (correct positif);

- Le nombre de fois que l'algorithme prédit alors que c'est une autre stratégie (faux positif);
- Le nombre de fois que l'algorithme prédit une autre stratégie alors que c'est la stratégie (faux négatif);
- Le nombre de fois que l'algorithme prédit correctement que ce n'est pas la stratégie (correct négatif);

Stratégie	Correct positif	Faux positif	Faux négatif	Correct négatif
Explore	26 %	41 %	6 %	25%
Fuite	16 %	59 %	22 %	2 %
Attaque	23 %	65%	4 %	6 %

Tableau 5 : Performance de reconnaissance de la stratégie pour Joueur Machine

Stratégie	Correct positif	Faux positif	Faux négatif	Correct négatif
Explore	51 %	16 %	19 %	13%
Fuite	4 %	77%	11 %	6 %
Attaque	10 %	70%	3 %	14 %

Tableau 6 : Performance de reconnaissance de la stratégie pour Joueur 1

Stratégie	Correct positif	Faux positif	Faux négatif	Correct négatif
Explore	43 %	41 %	10 %	6 %
Fuite	9 %	81 %	6 %	2 %
Attaque	31 %	60%	1 %	8 %

Tableau 7 : Performance de reconnaissance de la stratégie pour Joueur 2

Pour mesurer l'efficacité de cette méthode, nous avons fait jouer 50 parties pour chaque joueur. Et pour chaque partie du jeu, nous avons pris le taux de reconnaissance moyen. Les figures suivantes résument le taux de reconnaissance oscille toujours entre 0,64 et 1. Nous considérons que c'est un bon résultat.

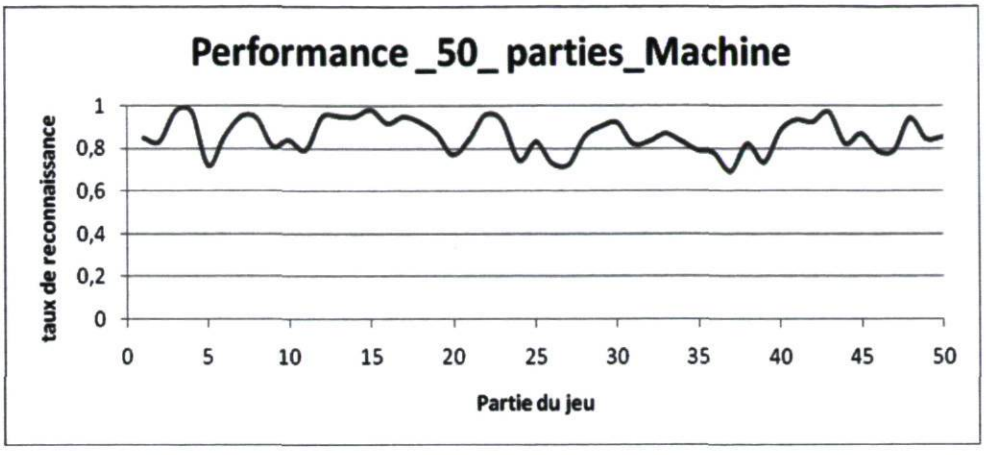

Figure 30 : Performance de la reconnaissance pour 50 parties jouées par la machine.

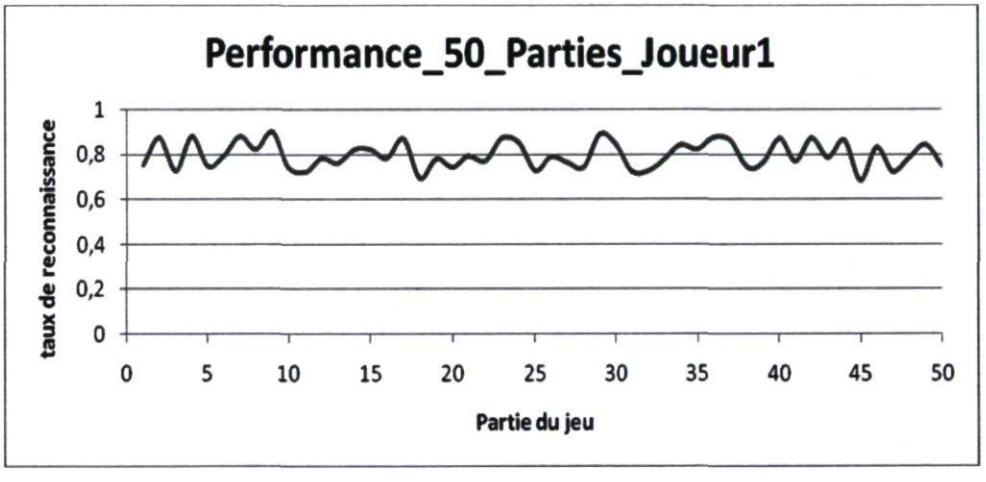

Figure 31 : Performance de la reconnaissance pour 50 parties jouées par Joueur 1

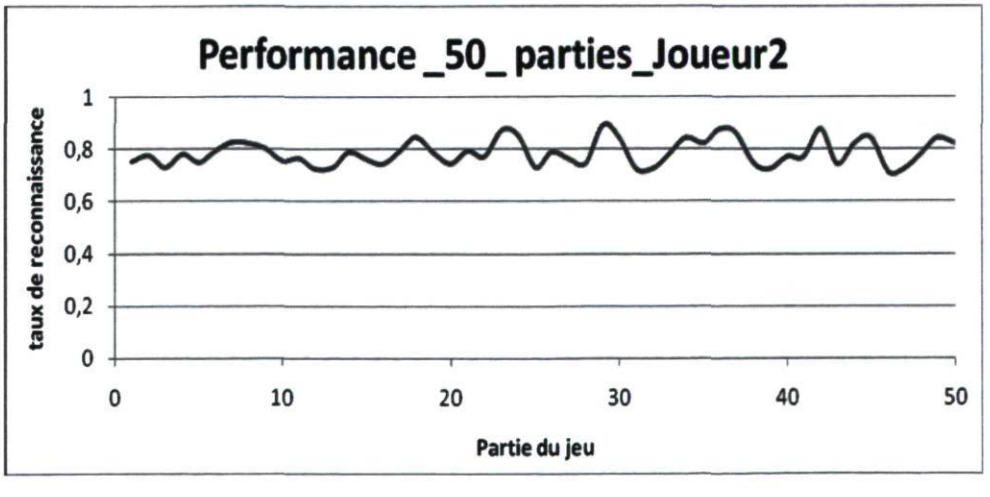

Figure 32 : Performance de la reconnaissance pour 50 parties jouées par Joueur 2

Le tableau suivant donne un aperçu général pour chaque joueur pour l'ensemble des parties jouées:

Joueur	Taux moyen de reconnaissance
Machine avec FSM	81%
Joueur humain1	79%
Joueur humain 2	74%

Tableau 8 : Taux de reconnaissance pour 50 parties

5.6.4 Conclusion de la méthode de la reconnaissance de la stratégie

D'après les résultats obtenus, nous pouvons affirmer que le modèle HMM est un outil efficace, d'un point de vue global, pour la reconnaissance dans les jeux vidéo. Cependant, les résultats obtenus indiquent également que les HMMs éprouvent des problèmes à reconnaître l'état de fuite, qui est un état que l'on retrouve moins fréquemment dans les parties.

5.7. Utilisation de l'algorithme Baum-Welch

La deuxième étape de notre approche est d'utiliser l'algorithme Baum-Welch pour faire l'apprentissage du modèle pour une séquence d'observation, c.-à-d. d'obtenir une matrice de transition et une matrice d'observation pour chaque séquence.

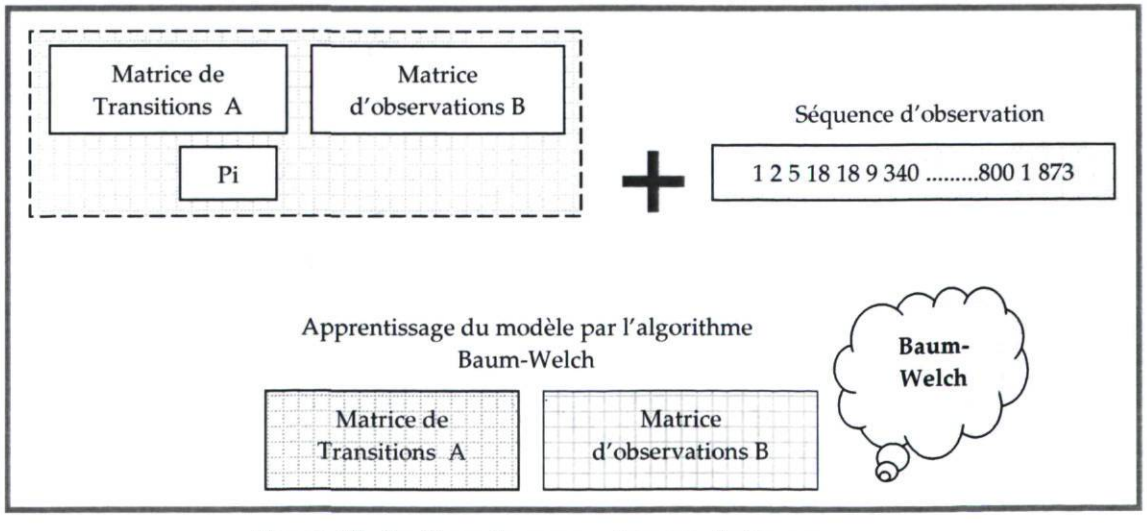

Figure 33 : Système de reconnaissance du joueur

5.8. Méthode de la reconnaissance de l'identité d'un joueur

Dans cette section, nous présentons les expérimentations que nous avons menées pour identifier le joueur d'un jeu. Nos expériences ont été menés avec deux joueurs humains et un contrôleur de type FSM (que nous appellerons machine dans ce texte). La démarche que nous avons utilisée est la suivante :

a. On utilise le modèle HMM construit dans la partie précédente.
b. On utilise l'algorithme Baum Welch pour faire l'apprentissage du HMM.
c. On construit la matrice d'observations pour chaque partie de jeu.
d. On utilise une méthode de classification pour détecter la ressemblance entre les matrices des observations.
e. On prend une décision quand à l'identité du joueur.

Cette démarche est illustrée dans la figure 34.

Pour permettre la construction des modèles, nous enregistrons des séquences d'observations tirées de parties de jeu pour chaque joueur. Pour notre expérimentation, nous avons utilisé une séquence de 300 observations et une séquence de 500 observations pour trois joueurs.

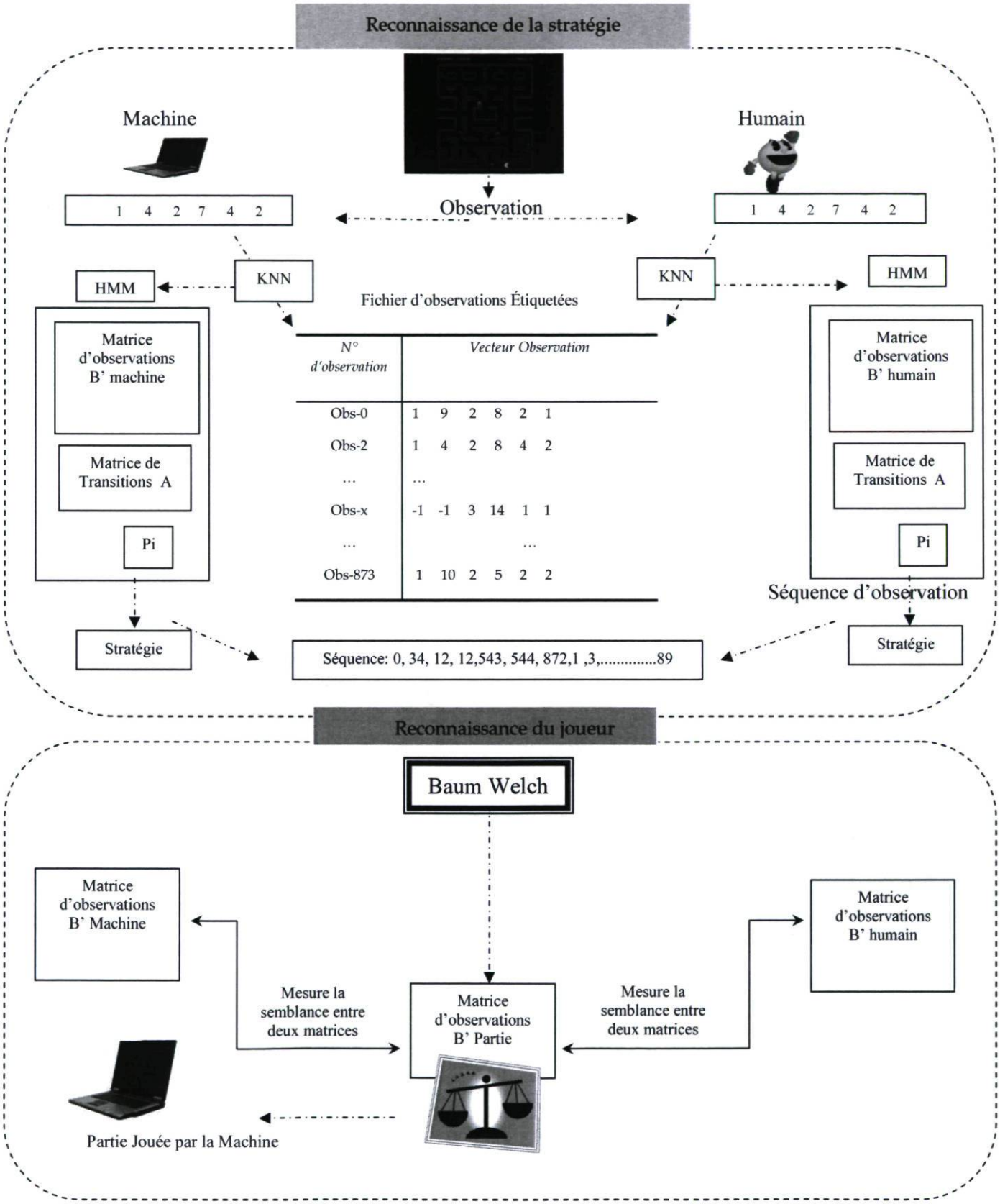

Figure 34 : Système de reconnaissance du joueur

5.9. Reconnaissance d'un utilisateur

Pour reconnaître un utilisateur, plusieurs algorithmes de calcul de similarité peuvent être adoptés et nous proposons une méthode basée la différence entre les paramètres qui caractérisent une partie avec les différents modèles de joueurs que nous détenons. Plus précisément, nous calculons la matrice d'observations B correspondant à une partie et nous estimons la similarité entre cette matrice et celles des différents modèles de joueurs.

Pour illustrer cette approche, nous donnons un exemple. Soit la matrice B obtenue dans le cadre d'une nouvelle partie et la matrice B provenant d'un modèle de joueur. On débute par calculer la différence entre les deux matrices telles qu'illustré pour les deux matrices suivantes.

B d'une nouvelle Partie B du Joueur 1 Différence joueur 1

$$\begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.1 & 0.1 & 0.6 \\ 0.4 & 0.7 & 0.1 \end{bmatrix} - \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.2 \\ 0.1 & 0.3 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ -0.3 & 0.0 & 0.4 \\ 0.3 & 0.4 & -0.4 \end{bmatrix}$$

On répète cette étape pour chacun des modèles que nous détenons. Pour comparer les modèles, nous déterminons une nouvelle matrice qui contient la différence entre deux matrices pour qu'on puisse faire la comparaison par la suite.

Différence_ Joueur_1 Différence_ Joueur_2

$$\begin{bmatrix} 0.1 & 0.0 & |-0.1| \\ 0.1 & |-0.4| & 0.2 \\ |-0.2| & 0.4 & |-0.1| \end{bmatrix} < \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ |-0.3| & 0.0 & 0.4 \\ 0.3 & 0.4 & |-0.4| \end{bmatrix} =$$

La comparaison est faite entre chaque valeur absolue de chaque élément de la matrice. Dans l'exemple précédent, si la valeur absolue de l'élément de la matrice Différence_joueur_1 est inférieure à l'élément de la matrice Différence_joueur_2, le résultat est de retenir le joueur 1. Dans le cas contraire, le résultat est joueur 2 et dans le cas que les éléments sont égaux, on ne le compte pas.

Donc obtient à la fin une matrice de la forme suivante :

$$\begin{bmatrix} Joueur2 & & Joueur2 \\ Joueur1 & Joueur2 & Joueur1 \\ Joueur1 & & Joueur1 \end{bmatrix}$$

Pour la prise de décision, on compte le nombre de fois que le joueur 1 se répète dans la matrice, et la même chose pour le joueur 2. Et on divise les deux sur le nombre total d'éléments dans la matrice On fini par choisir celui qui a le grand pourcentage. Par notre exemple :

- Nb_Joueur2 = 3, Nb_Joueur1= 4, Nb_Total=7
- Pourcentage pour le joueur 1 = 4 / 7 = 57%.
- Pourcentage pour le joueur 2 = 3 / 7 = 43%.

  Prise de décision ----------> Partie Jouée par le joueur 1

5.10. Expérimentations

Nous présentons dans les figures ci-dessous 2 expérimentations parmi 20 expérimentations faites pour 3 joueurs (la machine, Miloud et Fafa) et les résultats que nous avons obtenus.

L'exemple présenté à la figure 35 montre bien le résultat de l'exécution d'une partie jouée par « la machine » et indique le taux pour chaque joueur. Nous avons les résultats suivants :

- 80,27 % de probabilité que la partie est jouée par la machine, et
- 11,72% que la partie est jouée par joueur 1 («Miloud»).

Donc nous pouvons conclure à partir de ces valeurs que la partie est jouée par la machine.

La figure 36 présente un autre exemple de résultat d'une partie jouée par le joueur 2 (« Fafa »).

Des essais menés sur 20 parties ont donné un taux de succès de 80%, c.-à-d. 16 identifications qui sont correctes et 4 autres qui sont erronées. Bien qu'il soit difficile

d'identifier les causes qui expliquent les 4 erreurs, nous remarquons que le niveau des joueurs qui se confrontaient pour chacune de ces parties était comparable. Ce facteur pourrait avoir une influence sur les résultats.

Expérimentation numéro 1

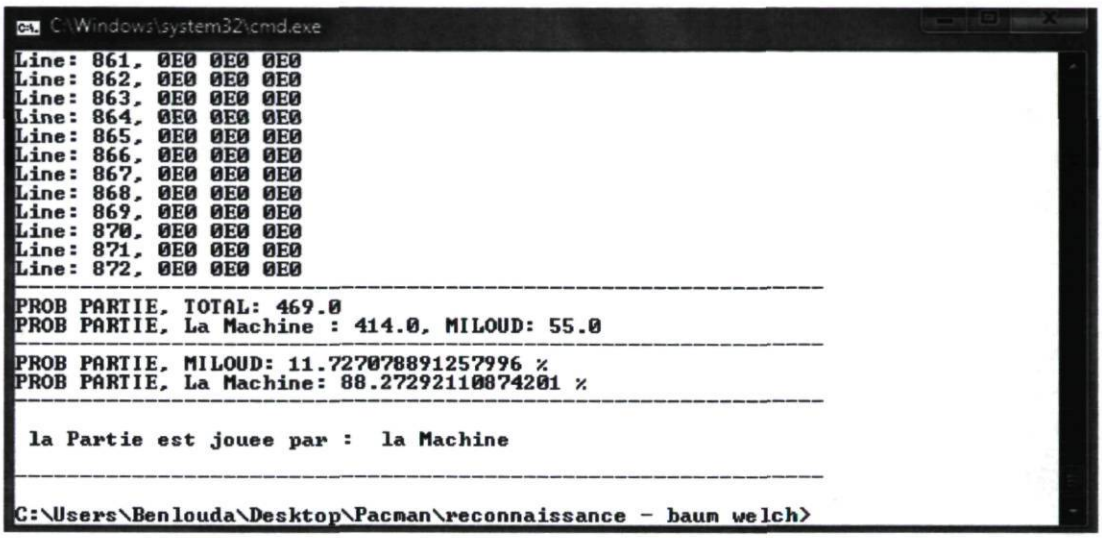

Figure 35 : Résultat de l'exécution d'une partie jouée par « la machine »

Expérimentation numéro 2

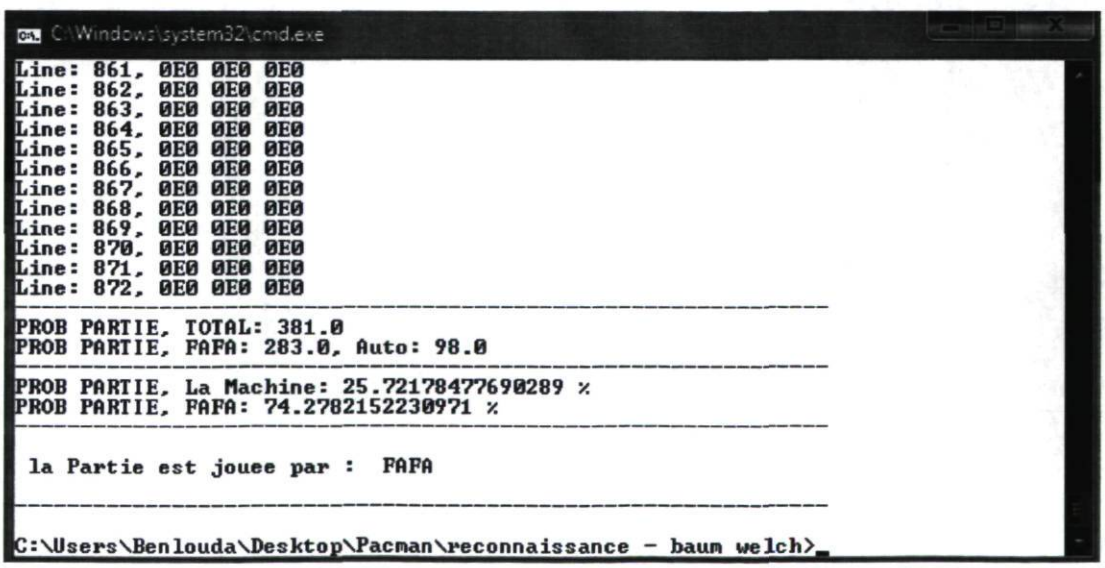

Figure 36 : Résultat de l'exécution d'une partie jouée par Joueur 2« Fafa »

5.11. Conclusion

Nous avons présenté dans cette partie une méthode pour faire la distinction entre deux joueurs (la machine et un joueur humain) par l'utilisation de la similarité entre deux matrices. Sachant que les matrices sont construites par l'algorithme Baum-Welch, nous sommes arrivés à montrer l'efficacité de cette méthode par les résultats obtenus. Pour 20 expérimentations, nous avons réussi à détecter le vrai joueur dans 16 parties. Et pour les 4 qui restent, le niveau entre les deux joueurs était semblable, ce qui pourrait expliquer en parties les résultats négatifs.

Chapitre 6

Conclusion

Avant de commencer la conclusion, nous aimerions d'abord faire un résumé des raisons qui nous ont amenés à préparer ce mémoire. L'idée générale de ce travail était d'appliquer le modèle de Markov caché sur le jeu vidéo Pacman pour prédire la stratégie utilisée par le joueur et cela par l'utilisation de l'algorithme Viterbi. Par la suite, nous avons appliqué l'algorithme Baum-Welch pour l'apprentissage du modèle. Pour la reconnaissance des personnages du jeu, nous avons utilisé une méthode de classification basée sur la comparaison entre les matrices. Les résultats obtenus répondent, en partie, à nos objectifs initiaux.

Les résultats obtenus par l'utilisation de l'algorithme Viterbi pour la reconnaissance de la stratégie utilisée par le joueur donnent un taux de succès entre 0,64 et 1. Nous considérons que c'est un bon résultat global. L'approche montre cependant des limitations pour les états moins fréquents.

Pour la deuxième partie de ce mémoire, les résultats obtenus illustrent l'efficacité de la méthode de reconnaissance par l'utilisation de l'algorithme de Baum-Welch. Dans 20 expérimentations, nous avons réussi à détecter le vrai joueur dans 16 parties, ce qui représente un résultat satisfaisant.

6.1. Perspectives liées à ces travaux

Notre approche de modélisation du jeu vidéo par l'utilisation du modèle de Markov caché permet d'ouvrir de nouvelles perspectives en termes de recherches.

Premièrement, au niveau de classification, la comparaison entre les matrices appliquée dans ce mémoire doit être améliorée dans le but d'obtenir des résultats plus près d'une efficacité de 100%.

Deuxièmement, il faudrait faire une comparaison entre les méthodes appliquées dans le domaine de l'intelligence artificielle, afin de trouver un moyen qui nous permettent de combiner les méthodes existantes dans le but d'obtenir une meilleure performance. Ces variations pourront faire l'objet d'études supplémentaires.

BIBLIOGRAPHIE

[1] Rabiner, L. R.; "A tutorial on hidden Markov models and selected application in speech recognition", Proc. IEEE, Vol. 77, No. 2, February 1989, pp. 267-296.

[2] Dequier, J.; "Chaînes de Markov et applications, Conservatoire nationale des arts métiers, Centre d'enseignement de Grenoble", 1 juillet 2005. < http://www-mrim.imag.fr/publications/2005/DEQ05/dequier05.pdf >.

[3] Rabin, S.; "AI Game Programming Wisdom 2", Charles River Media, INC., 2003.

[4] Blunsom, P.;"Hidden Markov Models" <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.1016&rep=rep1&type=pdf >, August 19, 2004.

[5] Le Roux, J.; "Modèles de Markov Cachés" <http://users.polytech.unice.fr/~leroux/parolehmm.pdf >, 14 mars 2003.

[6] Alani, T., Guellif, H.; "Modèles de Markov Cachés théorie et techniques de base Partie I", < http://hal.archives-ouvertes.fr/docs/00/07/44/75/PDF/RR-2196.pdf >, Février 1994.

[7] Matsumoto, Y , Thawonmas, R .; "MMOG player classification using hidden markov models", Congrès: Entertainment computing:(Eindhoven, 1-3 September 2004) ICEC 2004 : international conference on entertainment computing No3, Eindhoven , PAYS-BAS .

[8] Tso, S.K., Liu , K. P.; "Hidden Markov Model For Intelligent Extraction Of Robot Trajectory Command from Demonstrated Trajectory", Industrial Technology, (ICIT '96), Proceedings of The IEEE International Conference on, 1996.

[9] Petkovic, M. and Jonker, W. and Zivkovic, Z.;"Recognizing Strokes in Tennis Videos Using Hidden Markov Models". In: IASTED International Conference on Visualization, Imaging and Image Processing, VIIP 2001, 3-5 Sep 2001, Marbella, Spain.

[10] YAMATO, J. OHY, J. ISHIII, K.;"Recognizing human action in time-sequential images using hidden Model Markov", *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92*, 1992.

[11] Gao, J., Wactlar, H., Bharucha, A., Hauptmann, A., "Dining Activity Analysis Using a Hidden Markov Model", 17th International Conference on Pattern Recognition (ICPR'04), Cambridge, United Kingdom, August 23-26, 2004.

[12] http://fr.wikipedia.org/wiki/Pac-Man, 2010.

[13] http://fr.wikipedia.org/wiki/Quantification_vectorielle, 2010.

[14] Natkin, S., Yan, C.; "User Model in Multiplayer Mixed Reality Entertainment Applications", Proceeding ACE '06 Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology, 2006.

[15] Ko´Sciuk, K.; "User Modelling in Games, 2007. <http://www.fccs.wshe.lodz.pl/fccs2007/artykuly/kosciuk.pdf >".