MÉTHODES INFOGRAPHIQUES POUR L'APPRENTISSAGE DES AGENTS AUTONOMES VIRTUELS

THÈSE N^{O} 3387 (2005)

PRÉSENTÉE À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Institut des systèmes informatiques et multimédias

SECTION D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Toni CONDE

ingénieur HES / REG A
et de nationalité espagnole

acceptée sur proposition du jury:

Prof. D. Thalmann, directeur de thèse
Prof. G. Abou Jaoudé, rapporteur
Prof. Y. Duthen, rapporteur
Dr A. Guillot, rapporteur

Lausanne, EPFL
2005

Table des matières

A mes parents, mon fils Antoine, mon frère et à Sophie qui ont illuminé mon chemin par leur affection durant tout ce long parcours de doctorant.

Remerciements

Je tiens à remercier vivement mon directeur de thèse, le Professeur Daniel Thalmann, de m'avoir accueilli dans son prestigieux laboratoire et de m'avoir témoigné de sa confiance tout au long de ces quatre dernières années. Ainsi, qu'en respectant les contraintes liées à mon activité professionnelle à temps partiel (à 70 %) et d'avoir aménagé au mieux mon emploi du temps pour mener à bien mon travail de thèse.

Je tiens aussi à exprimer ma gratitude aux membres du jury de thèse, le Professeur Georges Abou Jaoudé, le Professeur Yves Duthen et le Docteur Agnès Guillot, pour la pertinence de leurs commentaires et suggestions.

Sébastien Schertenleib, Etienne de Sevin, le Docteur Fréderic Vexo et Josiane Bottarelli, pour l'aide précieuse qu'ils m'ont apporté chacun dans son domaine.

Finalement, je tiens à remercier André pour ces précieux conseils dans la rédaction et la présentation de cette thèse.

Abstract

There are two primary approaches to behavioural animation of an Autonomous Virtual Agent (AVA). The first one, or behavioural model, defines how AVA reacts to the current state of its environment. In the second one, or cognitive model, this AVA uses a thought process allowing it to deliberate over its possible actions.

Despite the success of these approaches in several domains, there are two notable limitations which we address in this thesis. First, cognitive models are traditionally very slow to execute, as a tree search, in the form of mapping: states → actions, must be performed. On the one hand, an AVA can only make sub-optimal decisions and, on the other hand, the number of AVAs that can be used simultaneously in real-time is limited. These constraints restrict their applications to a small set of candidate actions. Second, cognitive and behavioural models can act unexpectedly, producing undesirable behaviour in certain regions of the state space. This is because it may be impossible to exhaustively test them for the entire state space, especially if the state space is continuous. This can be worrisome for end-user applications involving AVAs, such as training simulators for cars and aeronautics.

Our contributions include the design of novel learning methods for approximating behavioural and cognitive models. They address the problem of input selection helped by a novel architecture *ALifeE* including virtual sensors and perception, regardless of the machine learning technique utilized. The input dimensionality must be kept as small as possible, this is due to the "curse of dimensionality", well known in machine learning. Thus, *ALifeE* simplifies and speeds up the process for the designer.

La perfection n'est pas atteinte quand il n'y a plus rien à ajouter, mais quand il ne reste plus rien à supprimer.

Antoine de St-Exupéry

Résumé

Il existe deux approches principales pour l'animation comportementale d'un Agent Autonome Virtuel (AAV). La première, ou modèle comportemental, définit la manière dont cet AAV réagit face à l'état courant de son environnement. Dans la seconde, ou modèle cognitif, cet AAV utilise un processus qui lui permet d'agir au-dessus de ses actions potentielles.

En dépit du succès de ces approches dans plusieurs domaines, elles présentent deux limitations importantes que nous relevons dans cette thèse. La première est que les modèles cognitifs étant traditionnellement très lents à exécuter, une recherche arborescente, sous forme d'une association: états → actions doit être effectuée. D'une part, un AAV ne peut prendre que des décisions sous-optimales et, d'autre part, le nombre d'AAVs à même d'être utilisés simultanément en temps réel est limité. Ces contraintes restreignent leur application à un petit nombre d'actions potentielles. La seconde limitation est que les modèles, tant cognitifs que comportementaux, peuvent agir de façon inattendue et présenter un comportement indésirable dans certaines régions de l'espace des états, puisqu'il est impossible de les tester de façon exhaustive pour l'ensemble de ces régions, en particulier lorsque l'espace des états est continu. Ce qui peut être très gênant pour des usagers finaux qui font appel à des AAVs tels que, notamment, les simulateurs d'entraînement pour les voitures et l'aéronautique.

Nos contributions comprennent le développement de nouvelles méthodes d'apprentissage pour approximer les modèles comportementaux et cognitifs. Elles traitent également le problème de la sélection des données d'entrée, grâce à une architecture originale *ALifeE* qui intègre les senseurs virtuels et la perception virtuelle, dans le cadre de la méthodologie de l'apprentissage choisie. La dimension des entrées doit être maintenue aussi petite que possible, en raison de la "malédiction de la dimension", bien connue en apprentissage artificiel. Ainsi, l'utilisation de l'*ALifeE* simplifie et accélère le processus pour le concepteur.

Rien de grand n'a jamais été entrepris sur cette terre sans enthousiasme.

Emerson

Chapitre 1 - Introduction

1.1 Motivations et objectifs

Ce travail de thèse est une contribution à l'apprentissage des Agents Autonomes Virtuels (AAVs) qui repose, en partie, sur les études préalables des domaines perceptifs humains.

La notion d'apprentissage apparaît lorsque l'on aborde la dynamique du comportement et que l'on considère l'évolution des modes de réaction par rapport à l'environnement. Cette évolution peut-être consécutive soit à une modification de l'environnement, soit, au contraire, à un changement de perception du sujet qui va désirer, en conséquence, opérer une modification de son environnement. L'apprentissage peut donc être défini comme un processus de construction et d'assimilation d'une réponse nouvelle, autrement dit comme une démarche d'ajustement du comportement soit à l'environnement, soit à un projet élaboré par le sujet.

L'apprentissage perceptuel constitue une approche importante lorsque l'on veut construire un système complet pour un AAV qui soit, à la fois, autonome et intelligent. Pour se comporter correctement dans un monde virtuel inconnu, cet AAV doit observer son environnement avec un ensemble de senseurs et agir au moyen de l'ensemble d'actuateurs qu'il contrôle. Apprendre comment agir sur un actuateur en réponse à un

ensemble de données acquises par les senseurs virtuels est un problème difficile à optimiser.

A la place de chercher à optimiser ou améliorer les méthodes d'apprentissages qui sont susceptibles d'être appliquées aux AAVs, nous proposons une autre approche. Celle-ci est une architecture qui intègre de façon cohérente l'ensemble des senseurs virtuels d'un AAV pour sa vision, son audition, son toucher et son environnement virtuel couplés à une perception unifiée. L'objectif recherché est de fournir, avant les processus d'apprentissage, qu'ils soient de type comportemental ou de type cognitif, des informations qui ont été au préalable: filtrées, simplifiées et sélectionnées. Ensuite, les processus d'apprentissage se servent de cette architecture pour obtenir un apprentissage perceptif (*perceptual learning*), qui est en mesure de faire évoluer les informations internes et externes de l'AAV ainsi que les processus d'apprentissage proprement dits.

L'objectif étant de permettre à un humain virtuel d'explorer des environnements virtuels inconnus et de construire des modèles et des structures mentales, des « cartes cognitives » ou des plans de son exploration. Une fois sa représentation construite, cette connaissance peut-être communiquée à d'autres humains virtuels. Un humain virtuel autonome perçoit des objets et d'autres humains virtuels avec l'aide de son environnement virtuel, qui lui fournit l'information concernant sa nature et sa position. Le modèle de comportement décide quelles actions l'humain virtuel doit entreprendre: marcher, prendre un objet, etc., et utilise ensuite cette information.

Un problème qui, à notre connaissance, n'a pas encore été véritablement approfondi est celui de l'organisation perceptuelle d'un ensemble de senseurs virtuels pour qu'ils soient à même de fournir la meilleure réponse lors d'un processus d'apprentissage de l'AAV. Ce travail est donc une contribution à l'étude des méthodes infographiques pour l'apprentissage des AAVs qui s'inspire, en partie, des démarches, déjà mentionnées, dans un contexte perceptif humain.

Nous inspirant de la biologie, nous avons donc cherché à construire des AAVs qui disposent de capacités d'autonomie et qui peuvent s'animer eux-mêmes. Et, par conséquent, qui sont en mesure d'apprendre de manière efficace.

De telle sorte que ces humains virtuels, placés dans un environnement également virtuel et munis de senseurs – pour la vision, l'audition et le toucher – soient informés tant de leur environnement que de ses états internes.

1.2 Plan de la thèse

Pour y parvenir, nous avons développé plusieurs méthodes d'apprentissage perceptif, et en particulier :

- L'exploration d'un environnement inconnu,
- L'élaboration de "cartes cognitives",
- L'apprentissage comportemental de bas niveau, et
- L'apprentissage cognitif de haut niveau

Ainsi, le Chapitre 2, "Senseurs virtuels", traite d'un AAV qui, placé dans un environnement de vie artificielle et disposant d'une architecture de contrôle, optimise la gestion de ses senseurs.

Le Chapitre 3, "Perception virtuelle", a pour objet la perception des éléments de l'environnement, une perception essentielle pour donner à l'AAV la "conscience" de ce qui a changé autour de lui.

Le Chapitre 4, "Apprentissage comportemental", est consacré, comme son titre l'indique, à l'apprentissage comportemental de bas niveau.

Le Chapitre 5, "Apprentissage cognitif" se rapporte, lui, à l'apprentissage cognitif de haut niveau.

Enfin, le Chapitre 6, "Etudes de cas" traite des trois cas représentatifs: a) Les déplacements des AAVs dans une ville virtuelle, b) L'environnement de vie artificielle dans un appartement virtuel, et c) La simulation de conduite automobile dans une ville virtuelle.

1.3 Contributions

En résumé, il nous paraît important de préciser que l'essentiel de ce travail de thèse est constitué par un certain nombre de contributions à l'apprentissage perceptif dont, principalement:

- Une nouvelle approche pour l'intégration de perceptions virtuelles,
- Une nouvelle méthode pour la perception prédictive,
- Une nouvelle architecture *Artificial Life Environment (ALifeE)* pour un AAV, et
- Une nouvelle méthode pour l'apprentissage automatique de modèles cognitifs.

A ces contributions, il faut encore ajouter, subsidiairement et pour être complets, celles qui en constituent souvent le départ:

- Une nouvelle approche pour l'intégration de senseurs virtuels,
- Une nouvelle méthode pour l'apprentissage d'un modèle comportemental,
- Une nouvelle méthode pour l'apprentissage automatique de modèles comportementaux,
- Une nouvelle méthode pour l'apprentissage d'un modèle cognitif, et
- Une nouvelle méthode pour l'apprentissage d'un modèle cognitif avec évolution.

L'action semble suivre la pensée, mais en fait action et pensée se produisent simultanément. En réglant l'action qui est sous le contrôle de la volonté, nous pouvons gouverner indirectement les sentiments qui nous échappent.

W. James

L'œil ne voit bien que ce qu'il est préparé à voir.

Paul Valéry

Chapitre 2 - Senseurs virtuels

2.1 Introduction

Ce chapitre traite des senseurs virtuels par une approche originale qui s'inspire des neurosciences, et qui les assimile donc à un "petit système nerveux" ayant une architecture de contrôle simplifiée.

En effet, compte tenu des méthodes d'apprentissage proprement dites, utilisées par l'AAV pour acquérir un ou plusieurs comportements, la valeur *n* (nombre de variables d'entrée) doit rester la plus petite possible. Ceci est dû à la "malédiction de la dimension" (*curse of dimensionality*), postulée en apprentissage artificiel et qui décrit la difficulté d'acquérir l'association: états → actions (*mapping: states → actions*), laquelle augmente exponentiellement avec chaque variable d'entrée supplémentaire.

Selon le choix du modèle d'apprentissage - comportemental ou cognitif d'un AAV (abordés respectivement aux Chapitres 4 et 5) – choix que nous désirons approximer, ce modèle exigera le moins possible d'information sur l'état courant du monde virtuel. Cette information sera fournie sous une forme compacte au système d'apprentissage. Si la

représentation des états ne peut être suffisamment comprimée pour apprendre avec efficacité le comportement souhaité, il sera nécessaire de modifier le modèle que nous désirons approximer. Ceci afin de mieux remplir les critères exigés par la méthode d'apprentissage. Ceci est généralement possible, mais il y a plusieurs types de prises de décision qui, fondamentalement, exigent beaucoup d'information sur les variables d'entrée. Par exemple, le cas d'un simulateur de conduite automobile dans une ville virtuelle, avec un instructeur virtuel sous forme d'un AAV sera étudié en détail aux Chapitres 4, 5 et 6.

Réduire la dimension des données en sélectionnant un sous-ensemble de variables originales peut être avantageux, en raison du coût informatique, pour obtenir, stocker et traiter les mesures. Si l'on ne peut pas le faire, comme dans l'exemple mentionné ci-dessus dans lequel les données d'entrée sont continues et en temps réel, d'autres moyens de réduction de la dimension devront être proposés.

L'art de l'apprentissage commence par une conception appropriée de la représentation des données. Le résultat le meilleur est souvent obtenu en utilisant les éléments (*features*) dérivés des entrées originales. Construire la représentation d'un élément est l'occasion d'incorporer la connaissance du domaine aux données et peut se révéler très caractéristique d'une application. Il y a cependant de nombreuses méthodes de construction d'une caractéristique générique qui incluent, entre autres, le regroupement (*clustering*) et les transformations linéaires des variables d'entrées, telles que:

- L'analyse en composantes principales *Principal Component Analysis (PCA)*, et
- La décomposition en valeurs singulières *Singular Value Decomposition (SVD)*, permettant de former un ensemble de caractéristiques, qui sont des combinaisons linéaires des variables originales fournissant la meilleure reconstruction possible des données, selon la méthode des moindres carrés Duda et *al.* (2001). Il s'agit d'une méthode, non supervisée, de construction d'une caractéristique.

Il y a aussi d'autres méthodes de transformation linéaire, telles que les transformées de Fourrier et les transformées en ondelettes.

La sélection d'entrée (*feature selection*), souvent appelée caractéristique de sélection Guyon et Elisseeff (2003), est un problème connu en apprentissage artificiel. Ainsi que nous l'avons mentionné auparavant, toutes les méthodes d'apprentissage artificiel butent sur la "malédiction de la dimension". Il est donc essentiel de sélectionner

soigneusement et de n'utiliser que les entrées potentielles nécessaires au système d'apprentissage de la "fonction cible". Il s'agit toutefois d'une tâche difficile car il est souvent malaisé de savoir quelles entrées sont critiques, lorsque l'on veut définir une association: états → actions de manière appropriée. Nous avons donc jugé important de proposer une nouvelle approche Conde et Thalmann (2004), qui permette d'automatiser la sélection des entrées avant l'apprentissage proprement dit et ceci dans un contexte propre aux méthodes infographiques et d'animation comportementale d'humains virtuels.

2.2 Objectifs

Pour approximer un modèle d'apprentissage comportemental ou cognitif, nous disposons souvent de beaucoup d'entrées potentielles comme nous l'avons dit auparavant. Ceci est dû au fait que n'importe quelle variable, contenue dans la totalité des états de l'environnement, peut se révéler une donnée utile. Nous avons donc besoin d'une technique robuste de sélection d'entrées qui prenne en charge un large spectre de données potentielles, certaines étant partiellement redondantes, contrairement à beaucoup d'autres sans valeur.

Nous avons vu que de nombreuses techniques de sélection automatique d'entrées ont été développées par les tenants de la statistique et de l'apprentissage artificiel Guyon et Elisseeff (2003). Toutefois, ces techniques ne sont, dans leurs formes traditionnelles, pas très bien adaptées à nos besoins. C'est dû, en partie, à un certain nombre de facteurs. Par exemple, la technique la plus connue d'entre elles, (*PCA*), ne tient pas compte de la "fonction cible" et ne peut donc faire la différence entre une donnée d'entrée valable et le bruit. Beaucoup d'entre elles sont conçues pour la classification plutôt que pour l'approximation (régression), certaines rejetant seulement les entrées bruitées, d'autres n'étant plus robustes lorsqu'il y a beaucoup d'entrées potentielles. En raison de cette constatation, nous avons développé notre propre méthode d'acquisition, de filtrage et de sélection des entrées, au moyen de différents senseurs virtuels.

Avant de décrire, dans les sections suivantes, notre contribution, précisons que nous ne traiterons pas le problème de la création d'éléments (*feature creation*) dans notre approche de la sélection d'entrée. Un élément est un concept de haut niveau, construit à partir de données brutes (*raw data*). Les variables d'entrée sont, elles, de bas niveau car

elles peuvent définir une "fonction cible" mieux apprise (suffisamment continue, par exemple, avec une dimension d'entrée faible). Il n'y a cependant pas de théorie, ou de technique, assez mûre et bien établie, pour la création automatique d'un élément Thornton (2003). Ainsi, la création d'éléments est traditionnellement laissée au concepteur. Nous n'allons pas suivre cette approche standard, laquelle demande au concepteur de spécifier d'abord un ensemble complet, ou un super-ensemble, d'entrées utiles ou d'éléments. Notre technique de sélection d'entrées choisit automatiquement soit un ensemble sous-optimal, soit un sous-ensemble minimal d'entrées. Ceci est fondamental car il est souvent difficile de déterminer quelles entrées sont nécessaires pour une représentation minimale dans une association: états → actions précise.

2.3 Vue d'ensemble

Un AAV, situé dans un Environnement Virtuel (EV) est muni de senseurs pour la vision, l'audition et le toucher, qui l'informent de l'environnement extérieur et de son état interne. Un AAV possède des actuateurs qui lui permettent d'avoir une influence sur l'EV et une architecture de contrôle afin de coordonner ses perceptions et ses actions. Le comportement d'un AAV est adaptatif aussi longtemps que son architecture de contrôle lui permet de maintenir ses variables dans leur domaine de validité

Notre *Artificial Life Environment (ALifeE*) Conde et Thalmann (2004) est basé sur une approche originale, inspirée des neurosciences, et munit un AAV des principaux senseurs virtuels sous la forme d'un petit système nerveux. L'architecture de contrôle est voulue simple, afin d'optimiser la gestion des senseurs et de la perception virtuelle de l'AAV. Les processus de filtrage, de sélection et de simplification sont effectués après l'obtention des informations sensorielles. Cette approche nous permet d'obtenir une certaine persistance sous la forme d'une "carte cognitive".

Le "processus mental" d'un AAV peut être simulé. L'animation comportementale comprend les techniques utilisées pour rendre un AAV intelligent et autonome, le faire réagir à son EV et prendre des décisions conformément à son système perceptif, sa mémoire à court terme et son raisonnement à long terme. L'intelligence est la capacité de planifier et d'effectuer des tâches basées sur la représentation de l'état actuel de l'EV.

Notre objectif est de permettre à l'AAV d'explorer des EVs inconnus et de construire des structures et des modèles mentaux, des "cartes cognitives" ou des plans de cette exploration. Une fois cette représentation créée, sa connaissance peut être communiquée à d'autres AAVs. Chaque AAV perçoit des objets ainsi que d'autres AAVs avec l'aide de son EV, lequel lui fournit des informations concernant leur nature et leurs positions. Le modèle comportemental décide quelle action l'AAV doit entreprendre telle que marcher ou manipuler un objet et utilise alors la connaissance acquise.

2.3.1 Extraction des éléments

Dans les modèles neuromimétiques, les caractéristiques pertinentes sont généralement extraites, de manière non supervisée, par regroupement (*clustering*) ou par analyse, en composantes principales (*PCA*) ou indépendantes, ainsi que nous l'avons décrit auparavant. Nous retrouvons ici, en les exploitant, les techniques classiques de filtrage qui permettent d'extraire, par apprentissage, les filtres (ou prototypes) les plus significatifs du flux de données considéré. Cette batterie de filtres constitue la base du codage de l'information. Nous insistons sur le fait que, dans nos travaux, les techniques de filtrage adaptatif doivent être adaptées aux caractéristiques de l'EV où les données, saisies au vol, ont une distribution qui varie avec le temps et renvoie, simultanément, à des connaissances de manière différente.

Nous observons comment l'apprentissage supervisé permet d'extraire des éléments, tels que les séparateurs à vastes marges *Support Vectors Machine* (*SVM*) Vapnik (1995), à même de sélectionner les prototypes qui déterminent les frontières. De manière similaire, nous nous intéressons à des algorithmes constructifs, dont les uns peuvent effectuer un apprentissage en continu – avec révision si les données changent – et les autres, une mise au point progressive des prototypes.

Tout ceci nous ramène à la notion de mise en forme progressive de l'information, en vue de son utilisation ultérieure. On retrouve cela également en neurosciences, avec la dualité entre l'apprentissage déclaratif d'épisodes et de faits, et l'apprentissage procédural de régularités et de fonctions. Cette dualité soulève aussi la question du transfert de connaissances d'une forme à une autre.

2.3.2 Intégration et combinaison des systèmes synthétiques

Nous désirons, tout d'abord, réaffirmer que nous n'étudierons pas ces mécanismes élémentaires pour eux-mêmes, mais pour les utiliser dans des systèmes complets qui fournissent des capacités d'apprentissage aux AAVs. Ceci nous permettra, surtout, de centrer nos travaux sur les problèmes du monde réel. Ces problèmes sont principalement définis par leurs caractéristiques complexes, car comportant des données multi-modales et dynamiques, et parce qu'ils renvoient généralement à des tâches cognitives typiques: aide à la décision, interprétation des données, pilotage, etc. Ils peuvent, d'autre part, correspondre à la réalisation de systèmes dédiés, qui exploitent le caractère numérique et distribué des modèles.

Notre expérience nous a montré, ensuite, que réaliser de telles tâches avec des systèmes modulaires soulève des problèmes spécifiques, que nous pouvions également aborder en partant des neurosciences, ou des études statistiques. Au niveau systémique, et pour des plate-formes informatiques, le problème se pose de définir une communication adaptée entre les modules et d'en extraire une décision unique. Du côté de l'implémentation, le choix des mécanismes et la manière de les implanter se pose également. Compte tenu de toutes ces contraintes, nous proposons un nouveau modèle qui permet l'intégration et la combinaison des systèmes synthétiques. S'inspirant en partie, des neurosciences.

Un AAV devrait être équipé de senseurs virtuels pour la vision, l'audition et le toucher. Ces senseurs constituent le point de départ de l'implémentation de comportements tels que la vision directe pendant le déplacement, la manipulation d'objets et la réaction aux sons.

Notre *ALifeE* intègre de manière importante les principaux senseurs virtuels d'un AAV, comme dans la *Virtual Vision* proposée par Renault et *al.* (1990), Noser et *al.* (1995), et Kuffner et Latombe (1999), la *Virtual Audition* proposée par Noser et Thalmann (1995) et le *Virtual Touch* proposé par Hudson et *al.* (1997). Après l'acquisition de l'information, la part perceptive essentielle de l'AAV est réalisée par l'approche *Flexible Perception Pipeline*, proposée par Bordeux et *al.* (1999).

2.4 Senseur virtuel de la vision

2.4.1 Introduction

La vision synthétique est le canal d'information principal entre l'EV et l'AAV Reynolds (1993). L'AAV perçoit son EV à partir d'une fenêtre et cette information lui suffit pour des déplacements locaux. La combinaison de la reconnaissance d'images et de la représentation de son EV permet à l'AAV de réagir en temps réel. Noser et *al.* (1995) ont utilisé un *octree* comme représentation interne de la vue qu'avait l'AAV de son EV. Ces approches proposent une mémoire visuelle de l'AAV dans un environnement en 3D composé d'objets statiques et dynamiques.

La perception de l'AAV dans son EV est transmise par la vision et le son, parfois par l'information tactile. Son comportement, comme chez les humains, est fortement influencé par les données fournies par ses senseurs et par sa propre intelligence avec certains objectifs tels que: l'extraction, la simplification et le filtrage, lesquels dépendent des critères de perception associés à chaque modalité sensorielle. Tout ceci est intégré dans la partie perceptive de notre *ALifeE*.

L'AAV explore un environnement inconnu construit aussi bien sur des modèles mentaux que sur des "cartes cognitives", basées sur cette exploration. Le déplacement s'effectue de deux manières: global, avec le modèle préappris de l'EV comprenant peu de changements et la recherche de performance avec un algorithme de type "*Path Planning*", et local avec l'acquisition directe de l'EV. Un modèle géométrique en 3D, sous forme de grille, est implémenté à l'aide d'un *octree*, combiné avec l'approche proposée par Noser et *al.* (1995) et Kuffner et Latombe (1999).

2.4.2 Méthodologie

Nous avons modélisé l'ensemble des fonctionnalités nécessaires à notre senseur virtuel de la vision, ainsi que le montre la Figure 2-1, ensemble qui s'intègre dans notre architecture *ALifeE*.

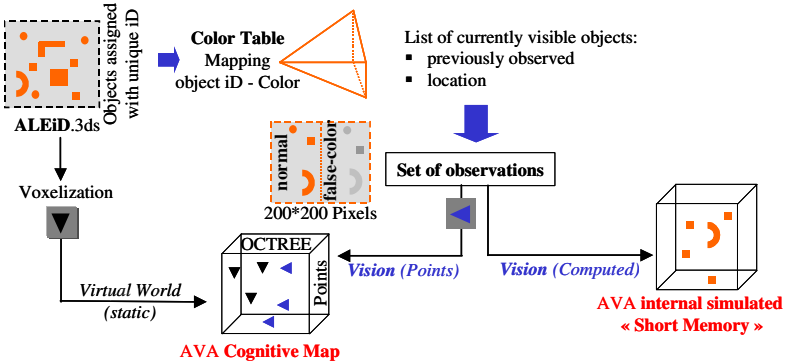


Figure 2-1. *Modélisation du senseur virtuel de la vision.*

Notre senseur virtuel de la vision explore un monde virtuel inconnu (**ALEiD**.3ds) et construit une "carte cognitive" (octree). Cette dernière étant mise à jour, à la fois, par la voxelization du monde virtuel et par l'observation des objets graphiques (mapping: object iD → color) que l'AAV a récemment observés, ainsi que sur leur position (Points). En parallèle à ce processus, le senseur virtuel de l'AAV simule une mémoire à "court terme" (short memory) avec l'ensemble des observations effectuées dans le monde virtuel, permettant entre autres, de traiter les problèmes d'occlusion des objets graphiques.

2.4.3 Implémentation

L'implémentation du senseur virtuel de la vision est conçue selon le diagramme *Unified Modeling Language (UML)* Booch et *al.* (1998) de la Figure 2-2. La méthode *vhdALEVVisionSensor* voxelise l'EV à l'aide de différentes sous-méthodes et met en forme ce dernier au moyen d'un octree. Le senseur virtuel de la vision effectue les opérations de traitement, afin d'obtenir les informations de l'objet virtuel qu'il voit, telles que, notamment, sa position et son identification dans l'EV.

En plus, une mémorisation en temps réel des différents objets virtuels, apparus dans le champ de vision de l'AAV est aussi effectuée. Ceci afin de traiter les problèmes éventuels d'occlusion des objets virtuels déjà vus et mémorisés.

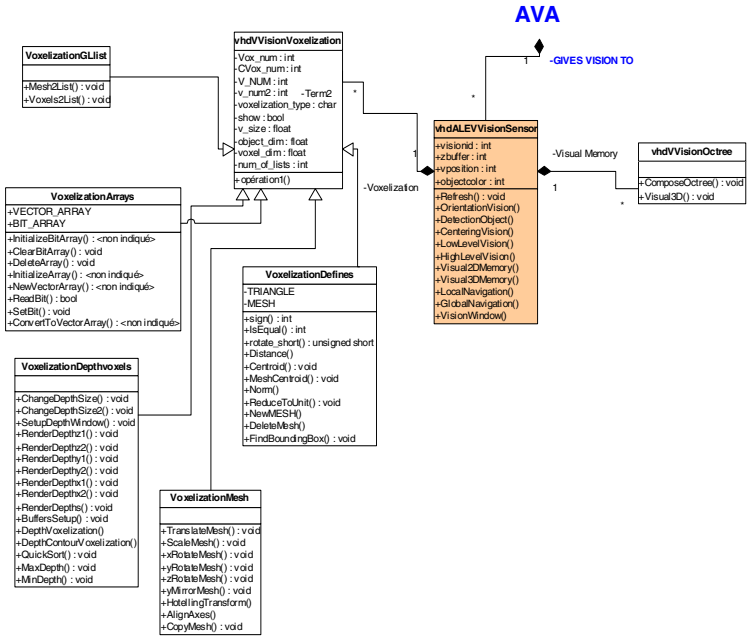


***Figure 2-2**. Diagramme UML du senseur virtuel de la vision.*

2.5 Senseur virtuel de l'audition

2.5.1 Introduction

Dans la vie réelle, le comportement des humains et des animaux peut être fortement influencé par les sons. Wenzel (1992) caractérise cela par: "la fonction des oreilles est de pointer les yeux". L'audition est un senseur temporel, donc très sensible aux modifications des signaux acoustiques. Nous pouvons situer des objets dans l'espace et, encore plus précisément, lorsqu'ils bougent. De plus, les signaux acoustiques transportent une quantité d'informations sémantiques et émotionnelles; ils nous informent sur la situation des sources sonores par rapport à nous aussi bien que sur la propagation et les chemins du son dans un environnement acoustique. Dans notre *ALifeE*, la restitution du son doit être efficace afin de pouvoir réagir aux événements sonores perçus par l'AAV à chaque séquence.

Les caractéristiques les plus importantes d'une source sonore sont, en termes informatiques: la localisation en 3D de la source dans l'environnement, l'orientation de cette source, le cône de propagation, la distance entre celui qui écoute et la source, l'effet Doppler, le volume, la fréquence, l'occlusion et l'obstruction.

Tous ces paramètres peuvent être réglés pour filtrer la source sonore selon les conditions de simulation. Quant au son en 3D, on considère qu'il suffit de placer la source sonore dans un environnement en 3D, sans se préoccuper de son orientation Carollo (2002). Il s'agit cependant d'une limitation trop forte, surtout pour les réverbérations et les réflexions. Dans notre *ALifeE*, nous représentons la propagation du son par un cône, qui correspond à une solution plus flexible pour régler les différents filtres sur chaque source sonore.

2.5.2 Méthodologie

Nous avons modélisé l'ensemble des fonctionnalités nécessaires à notre senseur virtuel de l'audition, ainsi que le montre la Figure 2-3, ensemble qui s'intègre dans notre architecture *ALifeE*.

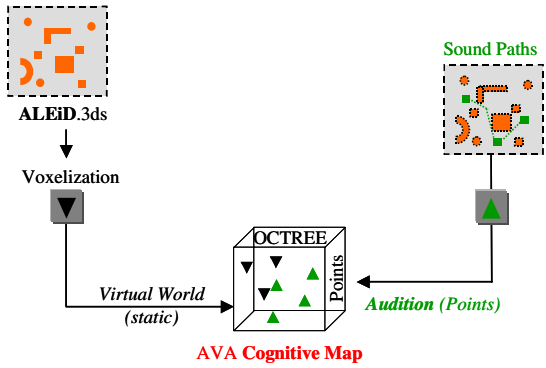


Figure 2-3. *Modélisation du senseur virtuel de l'audition.*

Notre senseur virtuel de l'audition de l'AAV explore un monde virtuel inconnu (**ALEiD**.3ds) et construit une "carte cognitive" (octree). Cette dernière étant mise à jour, à

la fois par la voxelization et par le calcul de la propagation des chemins sonores des objets graphiques que l'AAV a observés, ainsi que par leur position (Points).

2.5.3 Implémentation

L'implémentation du senseur virtuel de l'audition est conçue selon le diagramme UML de la Figure 2-4. Une source sonore, lorsqu'elle émet un son dans l'EV, peut rencontrer des obstacles, pour lesquels elle va être atténuée, absorbée ou réfléchie, selon différents critères. La méthode sonore *vhdALEVAuditionSensor* sépare les obstacles en deux types : ceux d'occlusion et ceux d'obstruction. L'occlusion a lieu lorsque l'AAV est totalement isolé de l'obstacle, et l'obstruction lorsque le son contourne l'obstacle pour arriver à l'AAV.

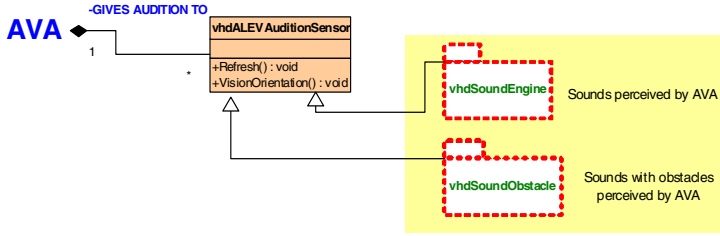


Figure 2-4. *Diagramme UML du senseur virtuel de l'audition.*

2.6 Senseur virtuel du toucher

2.6.1 Introduction

L'information tactile peut être utilisée pour appuyer sur des boutons, ou toucher et manipuler des objets. La simulation de ce type de senseur ressemble à la détection de collisions proposée par Huang et *al.* (1995). Nous avons cependant opté pour le processus décrit par Hudson et *al*. (1997), avec une approche "*V-Collide*" de détection de collisions.

Cette approche effectue des collisions exactes et performantes entre modèles polygonaux triangulés, au moyen d'un procédé hiérarchique à deux niveaux:

- Le niveau élevé ne prend pas en considération les couples d'objets qui ne sont pas proches l'un de l'autre, et
- Le niveau bas détecte avec précision les collisions, jusqu'au niveau des triangles eux-mêmes.

2.6.2 Méthodologie

Nous avons modélisé l'ensemble des fonctionnalités nécessaires à notre senseur virtuel du toucher, ainsi que le montre la Figure 2-5, ensemble qui s'intègre dans notre architecture *ALifeE*.

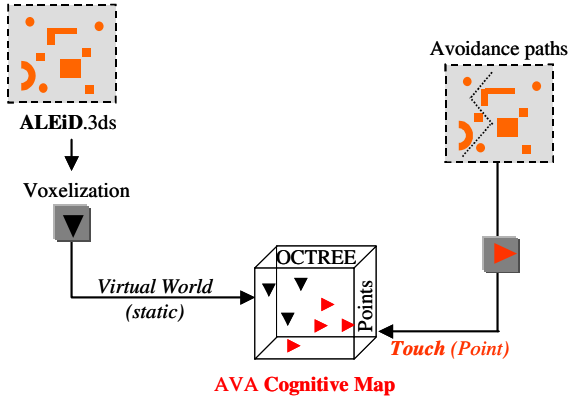


Figure 2-5. *Modélisation du senseur virtuel du toucher.*

Notre senseur virtuel du toucher de l'AAV explore un monde virtuel inconnu (**ALEiD**.3ds) et construit une "carte cognitive" (octree). Cette dernière étant mise à jour à la fois par la voxelization et par le calcul de la projection (trajectoires) des collisions des objets graphiques que l'AAV a observés, ainsi que par leur position (Points).

2.6.3 Implémentation

L'implémentation du senseur virtuel du toucher est conçue selon le diagramme UML de la Figure 2-6. La méthode *vhdVTouchVCollide* est composée de deux couches

distinctes : une détection du contact de paires d'agglomérats de polygones, et une détection de contact de polygones. Chaque agglomérat de polygones est divisé en deux, selon les distances entre les polygones, jusqu'à l'obtention de paires de polygones. Lorsque par la suite le senseur virtuel du toucher interroge la méthode *vhdVTouchVCollide* pour savoir si des collisions ont eu lieu, l'arbre est parcouru récursivement de la racine aux feuilles.

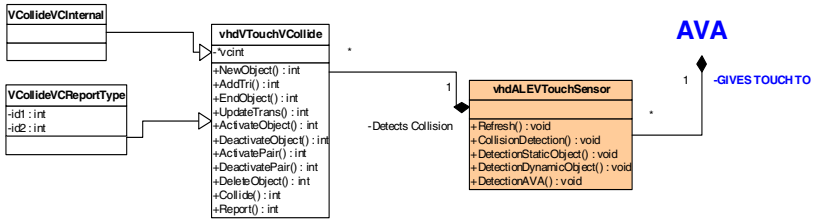


***Figure 2-6**. Diagramme UML du senseur virtuel du toucher.*

2.7 Eléments de l'Environnement virtuel donnés aux senseurs virtuels

2.7.1 Introduction

Les modèles de vision synthétique pour un AAV sont différents de ceux utilisés en robotique comportementale. Par ses senseurs, un robot ne fait qu'acquérir des informations sur son environnement, ce qui limite son comportement en déplacement et en évitement d'obstacles. Dans un EV, des informations supplémentaires peuvent être extraites, et utilisées, conformément au modèle perceptif choisi pour l'AAV. Ceci rend cet AAV plus rapide et plus "intelligent" dans ses actions. Afin d'optimiser le modèle, nous choisissons un type de représentation de l'EV dans lequel l'AAV maintient son système de vision à bas niveau.

Toutefois, par suite des contraintes d'échelle et pour rendre son comportement autonome plausible, un AAV limite sa perception localement, dans un EV en tant qu'ensemble global. Une telle approche est utilisée plus particulièrement lorsqu'il y a une

quantité d'AAVs, comme celle décrite par Reynolds (1993). Mais le plus important est que la méthode choisie reflète un AAV dans un EV proche de la réalité.

2.7.2 Méthodologie

Nous avons modélisé les éléments de l'EV donnés aux senseurs virtuels, ainsi que le montre la Figure 2-7, ensemble qui s'intègre dans notre architecture *ALifeE*.

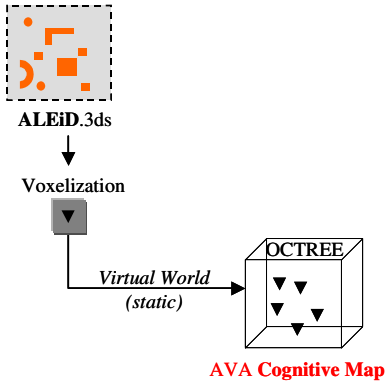


Figure 2-7. *Modélisation des éléments de l'environnement virtuel.*

Notre EV pour les senseurs virtuels de l'AAV explore un monde virtuel inconnu (**ALEiD**.3ds) et construit une "carte cognitive" (octree), celle-ci étant mise à jour par la voxelization.

2.7.3 Implémentation

L'implémentation des éléments de l'EV donnés aux les senseurs virtuels est conçue selon le diagramme UML de la Figure 2-8. De manière globale, l'EV est composé de tous les états définis dans le fichier *xml*. Ce qui permet de garder un EV statique, lequel n'est pas modifié pendant le traitement des informations par les différents senseurs virtuels.

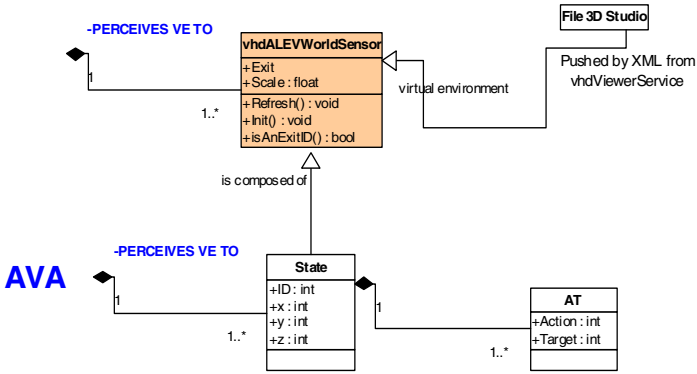


Figure 2-8. *Diagramme UML des éléments de l'environnement virtuel.*

2.8 *Artificial Life Environment (ALifeE)* – Senseurs virtuels

2.8.1 Introduction

Un AAV n'est pas seulement situé dans un environnement équipé de senseurs virtuels, mais il doit disposer de facultés de comportement, de perception et de mémorisation qui le rendent autonome et "intelligent". Notre objectif est de conférer à un AAV la capacité d'explorer un environnement inconnu, afin qu'il puisse construire des modèles mentaux et des "cartes cognitives". Pendant ou après la construction de ces modèles, l'AAV peut exécuter avec succès bien des fonctions, telles que: planification des trajectoires, déplacement ou localisation.

Notre modèle *ALifeE* repose sur l'intégration multi-sensorielle de la théorie typique des neurosciences (voir Figure 2-9). Les signaux qui se rapportent au même objet virtuel, mais proviennent de systèmes sensoriels distincts, sont réunis. Nous allons nous concentrer sur les deux problèmes suivants: 1) celui de l'attribution: déterminer quelles excitations sensorielles appartiennent au même objet virtuel; et 2) celui du recodage sensoriel: recoder les signaux en un format commun, avant de les combiner Pouget (2002).

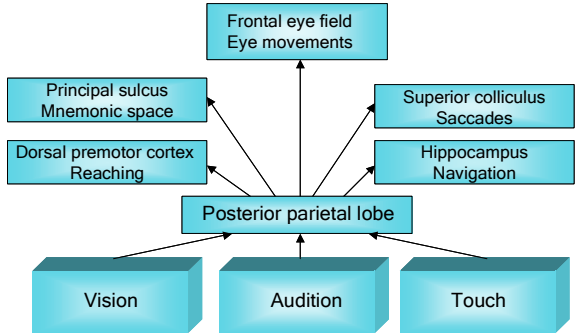


Figure 2-9. *Représentation schématique typique de l'intégration spatiale multi-sensorielle et des transformations sensori-motrices. Les modalités sensorielles encodent l'emplacement des objets dans les référentiels spécifiques de chaque modalité. L'intégration multi-sensorielle a lieu dans de multiples modules du cortex pariétal.*

Nous avons développé une approche multi-sensorielle, basée sur un modèle géométrique en 3D avec une grille qui implémente un *octree*, puisque les humains ne fondent pas le raisonnement spatial sur une carte continue, mais discrète Sowa (1964). Nous avons réalisé notre *octree* en utilisant la méthode de *voxelisation* rapide développée par Karabassi (1999).

Notre objectif ayant été d'introduire l'équivalent d'un "petit système nerveux" dans l'architecture de contrôle, nous avons ainsi relié les senseurs à ses effecteurs (voir Figure 2-10). L'apprentissage peut modifier l'organisation de l'architecture de contrôle et celle, en même temps, des processus évolutifs. Ils seront décrits ultérieurement car ce sont ceux que la nature a inventé pour assurer la survie des êtres vivants.

Un AAV est capable d'explorer son EV et d'établir une représentation mentale de l'organisation spatiale de ce dernier, sous la forme d'une "carte cognitive" qu'il pourra ensuite utiliser pour se situer lui-même et atteindre un objectif donné. Cette capacité repose sur l'utilisation d'un système visuel performant, tel qu'il a été décrit précédemment. Au cours de ses interactions avec l'EV, un AAV se construit un modèle plus général du monde virtuel. Modèle qui l'aide à anticiper les changements de l'EV consécutif aux actions entreprises.

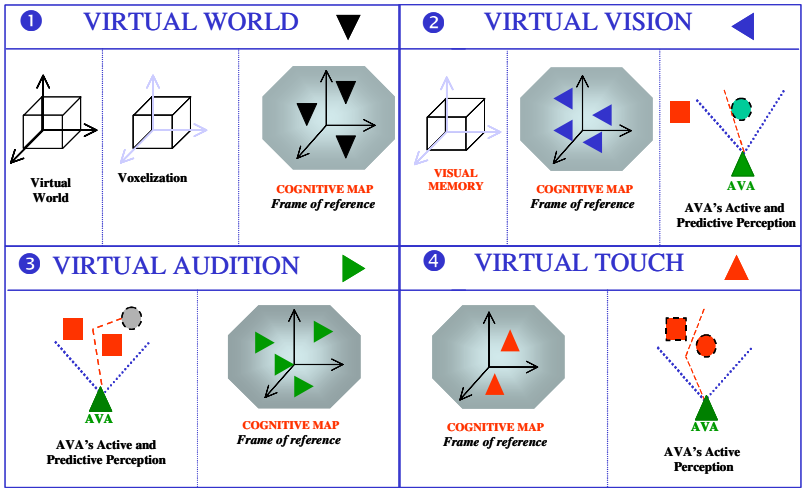


Figure 2-10. *Intégration multi-sensorielle: 1) Virtual World – extraction de l'information de l'EV, voxelisation et insertion dans la carte cognitive; 2) Virtual Vision – extraction de l'information de l'EV vu par l'AAV, en coordination avec la perception virtuelle, et insertion dans la carte cognitive et dans la Mémoire Visuelle; 3) Virtual Audition – extraction de l'information de l'environnement sonore par l'AAV, en coordination avec la perception virtuelle, et insertion dans la carte cognitive; et 4) Virtual Touch – extraction de l'information de l'environnement tactile par l'AAV, en coordination avec la perception virtuelle, et insertion dans la carte cognitive.*

Un AAV est situé dans un EV simulé, avec des senseurs pour la vision, l'audition et le toucher, qui l'informent sur son EV (perception active et prédictive) et sur son état intérieur (proprioception). Il dispose d'effecteurs, qui lui permettent d'agir sur l'EV, ainsi que d'une architecture de contrôle, qui coordonne ses perceptions et ses actions. Le comportement d'un AAV est adaptatif, tant et aussi longtemps que cette architecture de contrôle lui permet de maintenir ses principales variables dans leur domaine de viabilité. Par exemple, une action correctrice est exécutée au point B, afin d'éviter de quitter le domaine de viabilité au point A. L'architecture de contrôle joue le rôle d'un système de motivation lorsqu'elle est utilisée pour le choix des buts successifs que l'AAV essaie d'atteindre, ou pour arbitrer entre des buts contradictoires.

La position auditive d'un objet est prédite à partir de sa position visuelle. Ceci nécessite la transformation du système de référence, dont l'origine est la coordonnée visuelle (position de l'œil), en un système de référence, dont l'origine est la coordonnée auditive (position de la tête). La comparaison des résultats est utilisée pour déterminer si les signaux des deux types de senseurs virtuels appartiennent au même objet.

Notre approche multi-sensorielle (voir Figure 2-11) comprend le modèle comportemental d'un AAV. L'architecture de contrôle est standardisée au moyen de sous-modules qui couvrent les différentes techniques nécessaires à la simulation du comportement de l'AAV.

L'introduction de techniques d'apprentissage automatique dans les systèmes multi-agents constitue un problème majeur de l'apprentissage comportemental. C'est un véritable défi que d'apprendre à des systèmes multi-agents comment se comporter, interagir ou s'organiser pour améliorer leurs performances collectives en exécutant une tâche.

Dans ce domaine, on rencontre deux difficultés majeures:

- Le choix de la technique d'apprentissage qui convienne à la tâche et au niveau de cette dernière, et qui permette la compréhension entre actions similaires, et
- Le choix d'un protocole d'apprentissage.

L'objectif de notre *ALifeE* est de valider ces deux choix afin d'arriver à un type d'apprentissage avec un simulateur de comportement d'un AAV.

2.8.2 Intégration des senseurs virtuels

La modélisation d'un AAV acquérant son indépendance reste, en ce qui concerne sa représentation virtuelle, un sujet important de recherche, et il est très voisin de la robotique comportementale. Il aide également à comprendre le comportement humain. L'AAV recueille les informations uniquement à travers les senseurs virtuels décrits plus haut.

Nous admettons que la vision est le principal canal d'information entre l'AAV et son environnement, ainsi que le postule la théorie typique de l'intégration multi-sensorielle en neurosciences Elfes (1991).

Les modalités sensorielles remettent à jour la "carte cognitive" de l'AAV de manière à la rendre multi-sensorielle. Par exemple, la mémoire visuelle de sa mémoire interne est utilisée pour un déplacement global du point A au point B. S'il y avait des obstacles, elle serait remplacée, pour un déplacement local, par la vision directe.

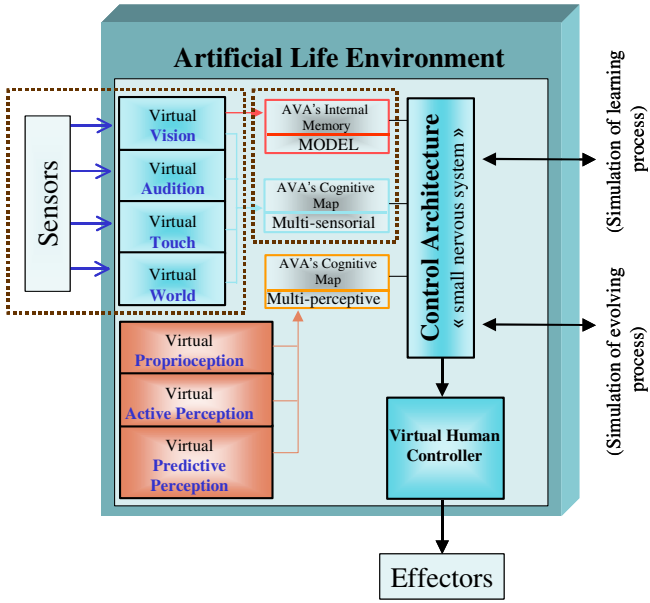


Figure 2-11. *Représentation schématique de notre ALifeE. La vision virtuelle découvre l'EV, construit les différents types de sensation et remet à jour la "carte cognitive" de l'AAV pour en rendre la représentation multi-sensorielle. L'architecture de contrôle utilise alors aussi bien les deux "cartes cognitives" que le "modèle de mémoire", afin de les intégrer avec les processus d'apprentissage, de développement et de contrôle de l'AAV (Virtual Human Controller).*

Dans notre approche de l'*ALifeE*, nous avons réuni toutes les informations multi-sensorielles des senseurs virtuels de l'AAV. En réalité, un AAV, dans son EV, peut avoir différents degrés d'autonomie et différents canaux sensoriels qui dépendent de l'environnement. Par exemple, un AAV se déplaçant dans un EV, représenté par une pièce bien éclairée, va d'abord utiliser l'information sensorielle de la vision. Mais, si la lumière s'éteint, l'AAV va faire appel à l'information auditive, ou tactile, de la même façon qu'une personne humaine le ferait dans une pièce obscure Strösslin et *al.* (2002).

C'est sur cette observation que nous établissons les hypothèses qui sont à l'origine de notre approche structurelle de l'*ALifeE*. Elles s'appuient sur les dernières recherches en neurosciences Pouget (2002), lesquelles décrivent la mise à jour partielle de la carte au niveau du comportement humain, mise à jour qui comprend:

- L'attribution: prédire la position acoustique d'un objet à partir de sa position visuelle exige la transformation de ses coordonnées centrées sur l'œil (senseur visuel) en des coordonnées centrées sur la tête (senseur auditif). La comparaison de ces deux types de résultats est utilisée pour déterminer si les signaux visuels et acoustiques se rapportent au même objet.
- Le recodage: choix du cadre de référence pour l'intégration des signaux sensoriels.

2.8.3 Architecture logicielle et implémentation

L'implémentation de l'ensemble des senseurs virtuels et de l'architecture de contrôle (avec son "petit système nerveux") est conçue selon le diagramme UML de la Figure 2-12.

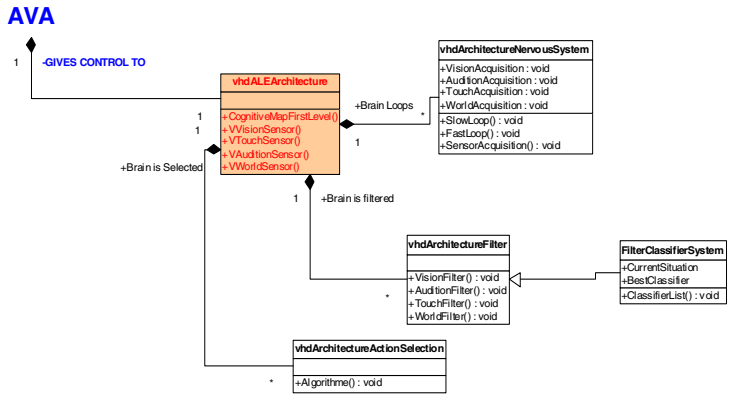


Figure 2-12. *Diagramme UML de l'intégration des senseurs virtuels et de l'architecture de contrôle ALifeE.*

2.8.4 Expérimentation selon *ALifeE*, avec ses senseurs virtuels

A l'aide de différents scénarios, nous avons pu démontrer que notre *ALifeE* réunit les modalités sensorielles et perceptuelles de manière cohérente. La Figure 2-13 montre la représentation d'ensemble de l'architecture à l'origine de nos résultats expérimentaux.

Nous avons atteint nos objectifs en utilisant nos nouvelles méthodologies décrites dans ce chapitre. Nous avons également appliqué cette approche à une méthodologie d'apprentissage de type bayésien non paramétrique, plus particulièrement la méthode des *k*-plus proches voisins (*k*-ppv), en utilisant un algorithme de recherche à grande vitesse Mitchell (1997) ou Cornuéfols et Miclet (2002) pour les chercheurs de langue française.

Information sémantique destinée aux senseurs

Toutes les informations sensorielles utilisées, pour notre application, dans notre plate-forme *ALifeE* sont représentées sur la Figure 2-13.

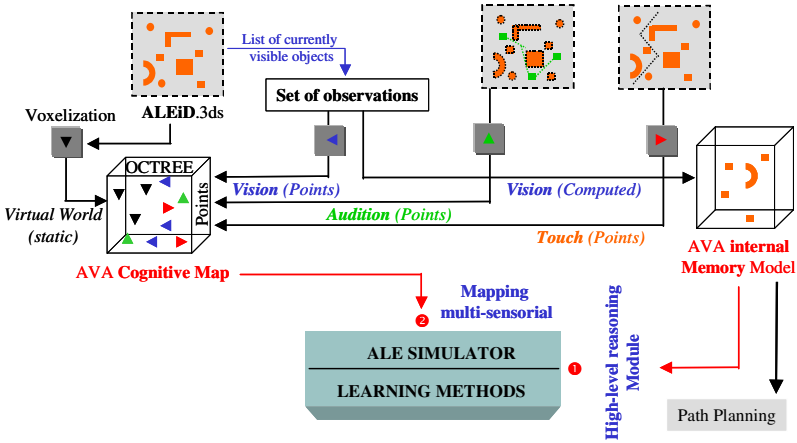


Figure 2-13. *Architecture ALifeE que nous avons utilisée pour nos résultats expérimentaux. L'information sémantique provenant de la représentation multi-sensorielle, multi-perceptive et des modèles de mémoire de l'ALifeE est employée par les processus d'apprentissage pour établir des modules de raisonnement de haut niveau ou des règles de mémoire complexes.*

Exemples d'intégration multi-sensorielle

Nous avons utilisé notre plate-forme *ALifeE* pour l'exploration d'un EV inconnu par un AAV. La mémoire visuelle interne de l'AAV lui permet de se mouvoir aux alentours et de naviguer (voir Figures 2-14, 2-15 et 2-16).

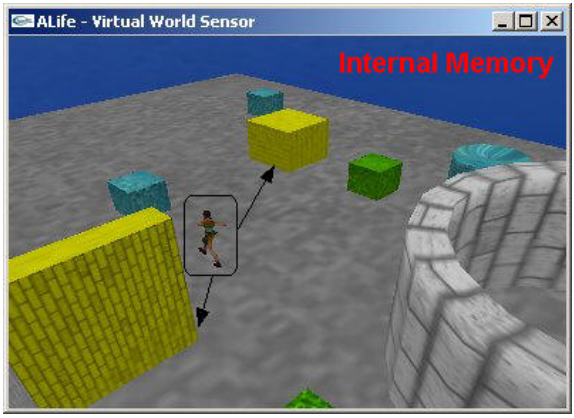


Figure 2-14. *Exemple d'apprentissage d'un EV inconnu par un AAV. Les objets graphiques jaunes sont ceux découverts par l'AAV et mémorisés dans sa mémoire visuelle interne.*

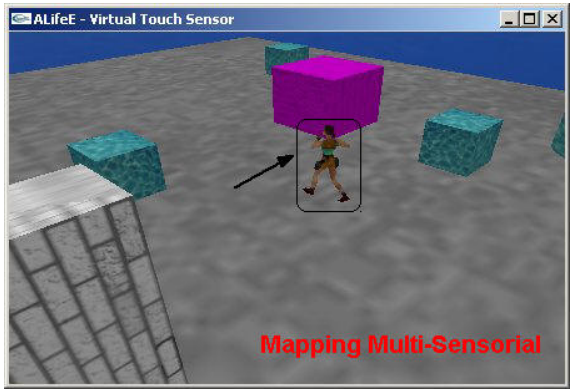
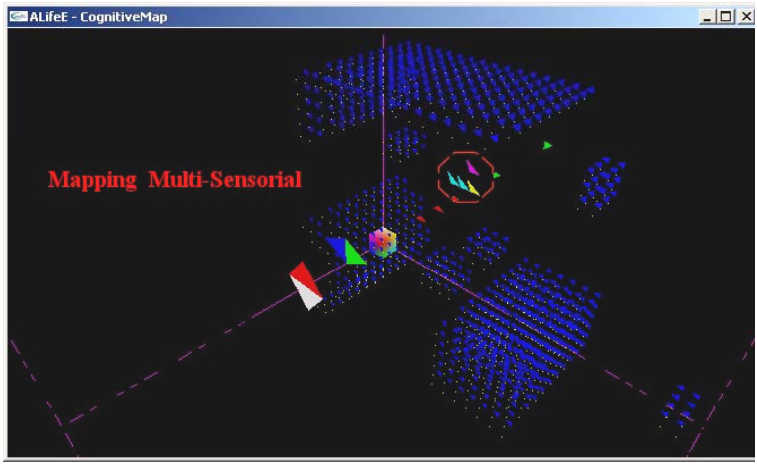


Figure 2-15. *Exemple d'association multi-sensorielle avec deux senseurs virtuels (vision-toucher). L'objet graphique violet est celui détecté par l'AAV.*



Figue 2-16. *Application d'un EV multi-sensoriel (vision-toucher). Instantané d'un AAV (triangle rose) utilisant l'information multi-sensorielle pour se mouvoir dans un EV à l'aide de l'algorithme des k-plus proches voisins (k-ppv). A l'intérieur de l'hexagone rouge: le triangle jaune représente le plus proche voisin (1-ppv) et les triangles cyans les deux plus proches voisins (2-ppv).*

2.9 Discussion

Nous avons présenté, dans ce chapitre, une nouvelle approche pour l'affinement des fonctions sensori-motrices par laquelle, un peu comme dans un système nerveux humain, on sélectionne, filtre ou simplifie l'information, à chaque relais sensoriel. Ce que nous avons obtenu grâce à l'introduction, dans l'architecture de contrôle, d'un module analogue à un "petit système nerveux". Lequel relie les capteurs et les effecteurs, afin de permettre à l'AAV l'acquisition, le filtrage et la sélection des stimuli externes et internes.

Dans ce même chapitre, nous avons aussi proposé une nouvelle méthode automatique de sélection des entrées, conçue spécialement pour approximer les modèles d'apprentissage comportemental et cognitif d'un AAV Conde et Thalmann (2004). Cette méthode simplifie et accélère grandement le processus d'apprentissage pour le concepteur, en sélectionnant les entrées tout aussi bien que lui. Seule l'information pertinente pour l'AAV est retenue, puis convertie en un ensemble compact d'éléments. Dont la plupart sont construits de façon que la translation et la rotation des angles et des distances entre les AAVs - ou entre les objets virtuels - ne varient pas par rapport à la référence de l'AAV.

L'approche proposée dans ce chapitre, nous assure une dimension d'entrée gérable et nous permet une approximation de l'information de l'état dans la plupart des cas. Cette approximation limite un peu la précision d'apprentissage de l'association: états → actions (*mapping: states → actions*) mais, en contrepartie, elle rend possible le processus d'apprentissage dans un EV relativement complexe.

Ainsi que nous l'avons précisé dans notre Introduction (Chapitre 1), il est important de situer les AAVs dans leur environnement, faute de quoi il ne peut y avoir d'intégration à celui-ci, ni d'autonomie. La perception des éléments de l'environnement est essentielle car elle donne à l'AAV la conscience de ce qui a changé autour de lui. Au chapitre suivant, nous présentons une nouvelle approche de l'intégration des différentes perceptions virtuelles à notre architecture *ALifeE*. Nous présentons également une approche de perception prédictive, laquelle joue un rôle fondamental dans l'orientation du regard, ou de l'attention, d'un AAV.

Nous sommes incroyablement légers dans la formation de nos convictions, mais il suffit que nous voulions nous persuader qu'elles sont fausses, pour que nous devenions ardents à les défendre. Ce ne sont évidemment pas les idées elles-mêmes que nous chérissons, mais notre propre estime qui est menacée.

J. H. Robinson

Chapitre 3 - Perception virtuelle

3.1 Introduction

Ce chapitre traite de la perception virtuelle que nous avons imaginée. On y a ajouté la fonction de prédiction, afin que cette perception soit active, notamment pour diriger le regard et l'attention.

Les informations provenant de diverses modalités sensorielles correspondent à des régions spécialisées de l'architecture *ALifeE*, ainsi que nous l'avons décrit au chapitre précédent - Senseurs virtuels. En intégrant différents types de perceptions virtuelles résultant des connexions réciproques entre ces différentes régions, notre approche propose une perception unifiée, avant apprentissage, d'un modèle comportemental ou cognitif de l'AAV.

Dans un système visuel humain, par exemple, c'est dans certaines régions corticales que sont traités les contours, les formes, le mouvement, les distances, la couleur, etc. Il s'agit de processus inconscients dont nous ne voyons qu'un aspect simplifié. L'une des principales caractéristiques du cerveau humain est sa faculté de prédiction, qui joue un rôle fondamental dans la perception active. Nous proposons donc une nouvelle approche de la

perception prédictive, dans laquelle nous obtenons la fonction de prédiction en nous inspirant des propriétés des circuits neuronaux de la vision humaine. Les concepts de perception prédictive et de proprioception sont également explorés à l'aide d'un AAV, ce qui nous permet d'obtenir une persistance et une "carte cognitive" pour les différentes perceptions virtuelles Conde et Thalmann (2004).

Pour ce qui est de l'unité de la perception virtuelle, nous nous sommes inspirés, à nouveau, de la théorie typique des neurosciences. Dans celle-ci la notion de contraction suggère un modèle explicatif pour la convergence des interactions rapides dans le système thalamo-cortical, ainsi que pour la variation de la perception, au fur et à mesure du changement des données sensorielles Edelman et Tonomi (2000). Le principe d'un vaste réseau de systèmes convergents spécialisés, totalement décentralisé, mais néanmoins convergent, peut être utilisé dans un système artificiel qui intègre les informations sensorielles et les algorithmes de traitement. De plus, les boucles d'interaction, que nous avons introduites dans notre architecture *ALifeE*, sont un moyen particulièrement efficace et rapide de partage du traitement de l'information entre de tels systèmes car le temps de réponse de l'ensemble ne dépasse pas celui du système, ou du module sensoriel le plus lent.

3.2 Objectifs

Les techniques de perception constituent les moyens par lesquels l'AAV perçoit son EV. Le type d'information qui peut être détecté dépend surtout de la capacité sensorielle de l'AAV. Reynolds (1987) estime qu'il est inutile de donner à un humain virtuel une information parfaite et complète sur son EV et qu'il est préférable de lui fournir une connaissance incomplète, plus en adéquation avec la représentation de la réalité.

Les techniques individuelles de perception sont limitées dans leur domaine de fonctionnement, ce qui signifie qu'un humain virtuel animé n'est capable de percevoir son environnement que dans les limites de sa technique de perception. La capacité de percevoir un EV doit avoir lieu dans une tranche de temps raisonnable pour que l'AAV puisse détecter rapidement les objets dans cet EV. Plus il fournit rapidement l'information sur le composant virtuel, plus l'animation sera continue. Le choix de la technique de perception est important car l'animation comportementale de l'humain virtuel dépend surtout de la qualité de l'information qu'il reçoit. Les techniques de perception peuvent être groupées

essentiellement en trois catégories: approche par zone, approche sensorielle, et approche par vision synthétique.

3.2.1 Approche perceptive par zone

L'approche par zone englobe les alentours immédiats de l'humain virtuel animé, avec une ou plusieurs régions de perception, et tout objet entrant dans cette zone doit être détecté par cet humain virtuel. Reynolds (1987) utilise une approche par zone dans son travail de modélisation du comportement distribué d'une formation d'oiseaux virtuels en vol, approche dans laquelle chaque oiseau virtuel a une zone sphérique de sensibilité centrée sur sa position. Ainsi, n'importe quel objet entrant dans cette zone sera détecté par cet oiseau virtuel. Avec ce type d'approche, Reynolds (1987) a rendu partiellement disponible la même information à un animal virtuel animé qu'à un animal réel. Toutefois, il n'a pas tenté de modéliser les sens qu'utilisent les animaux réels pendant leur vol en formation. A la place, il a essayé de fournir à l'animal virtuel animé l'information issue des processus perceptuels et cognitifs de l'animal réel.

Dans le système proposé par Reynolds, seule une zone de perception a été utilisée pour chaque animal virtuel animé. Il y a cependant, dans la nature, des zones où la perception d'un objet est précise, et d'autres où elle ne l'est pas. Ce problème a été étudié par Takeuchi et *al.* (1992) avec sa technique de planification du chemin et son application à un système d'animation humaine. Il a modélisé les éléments perceptifs d'un humain virtuel animé, en tenant compte de la distance de l'objet observé. Par exemple, lorsqu'un papillon observe les caractéristiques d'une fleur, il peut percevoir l'élément couleur et l'élément fragrance à distance, mais il doit toucher la fleur pour percevoir l'élément goût. Il existe plus d'une catégorie de perception. L'humain virtuel animé peut, par exemple, détecter le son, voir, ou détecter encore d'autres qualités perceptibles et disposer ainsi de multiples capacités perceptuelles.

3.2.2 Approche sensorielle

L'approche sensorielle entraîne la mise en place de senseurs sur un AAV animé, chacun d'entre eux opérant dans sa zone spécifique de détection. Elle permet, pour

différents types de senseurs, de détecter les qualités des objets perçus. Par exemple, détecter une qualité perceptible, comme le son, résulte du placement d'un senseur approprié sur l'humain virtuel animé. Cependant, percevoir n'importe quel stimulus qui provient de l'EV n'est pas toujours possible par cette approche. Le choix des senseurs est toutefois important, de même que la position et l'orientation de chacun d'entre eux. C'est à partir des stimuli reçus, via ces senseurs, que l'AAV acquiert la connaissance de son environnement, ainsi que nous l'avons décrit dans le chapitre précédent.

Braitenberg (1984) a créé plusieurs types de véhicules motorisés, chacun disposant d'une série de senseurs, connectés aux moteurs électriques, pour détecter les stimuli externes. Différentes réponses, qui dépendent surtout du câblage et des connexions, peuvent être générées. La perception matérielle de Braitenberg a inspiré l'approche logicielle de Wilhelms (1990) qui a, en quelque sorte, étendu le concept de base, en l'utilisant comme réseau d'interaction pour l'animation comportementale. Dans cette approche, il a utilisé une série de senseurs pour détecter les caractéristiques spécifiques des objets de l'environnement. Chaque senseur a une position et une orientation sur la périphérie de l'AAV, qui influencent fortement la manière dont il est stimulé. Il en existe de deux types: a) des senseurs de proximité et d'éloignement, qui détectent, respectivement de combien l'AAV est proche ou éloigné des autres objets de l'EV, et b) des senseurs de qualité, qui détectent les caractéristiques et les propriétés des autres objets.

L'une des limitations de l'approche sensorielle est le niveau de réalisme avec lequel les caractéristiques de perception des objets de l'environnement sont détectées. Par exemple, dans le monde réel, des événements comme le son peuvent n'être actifs que pendant un nombre de secondes insuffisant pour être détecté par un AAV équipé de senseurs pour l'audition.

L'approche sensorielle peut être réalisée dans un contexte sensitif, en commun avec l'approche par zone, en utilisant des senseurs qui peuvent varier leurs zones de détection en fonction de leur environnement. Ce qui peut encore être étendu en attribuant à l'AAV une série de zones de détection qui représentent les régions d'espace personnel associées à un humain ou à un animal. La grandeur de la zone de détection est importante, car si elle est trop petite il se peut que l'évitement de la collision ne soit pas possible, et si, en revanche, elle est trop grande, cela signifie que le senseur aura plus de choses à traiter, d'où une augmentation du temps de calcul. Le temps de planification d'un chemin dépend également

des dimensions de la zone de perception; plus cette zone est grande plus la capacité de planifier en avant sera importante.

Tous les modèles de perception basés sur une approche sensorielle sont assez similaires, en ce sens qu'ils disposent d'un ou de plusieurs senseurs pour détecter une qualité spécifique de perception. Bien que le type de senseur puisse varier, chacun d'entre eux opère uniquement à l'intérieur de ses limites.

3.2.3 Approche par la vision synthétique

La vision synthétique est une technique par laquelle l'AAV animé voit son EV, ce qui entraîne la modélisation de sa capacité de perception de l'information visuelle qu'il en reçoit. Cette modélisation a été introduite pour l'animation comportementale par Renault et *al.* (1990), et elle établit que la vision à partir d'un AAV animé est elle-même synthétique. Toutefois, comme elle ne fonctionne qu'à partir de stimuli visuels, on ne peut l'utiliser pour d'autres stimuli, tels que le son, par exemple.

Renault et *al.* (1990), Noser et *al.* (1995) et Tu et Terzopoulos (1994) se sont approchés de la vision synthétique en interrogeant une base de données contenant la description de tous les objets de l'EV et en extrayant la position de ceux qui sont dans le champ de vision de l'AAV. C'est cet ensemble de positions qui constitue alors la vue synthétique. Ainsi, la vision est un sous-système très important d'un AAV animé, c'est une approche idéale pour modéliser l'animation comportementale. Elle constitue l'approche universelle pour transmettre à l'AAV les informations nécessaires sur son environnement, lors de problèmes de recherche de chemin et d'évitement d'obstacles Noser (1995) et *al.*, ainsi que décrit dans le chapitre précédent.

Nous proposons d'inclure, dans la suite de ce chapitre, les différents types de perception qui s'inspirent des concepts et idées proposés par les pionniers précités, et d'y ajouter les senseurs virtuels que nous avons traités au chapitre précédent. Nous proposons aussi, dans le cadre de cette intégration perceptive, une nouvelle approche de la perception active et prédictive d'un AAV, afin de réduire l'information inutile à son processus d'apprentissage.

3.3 Vue d'ensemble

Le problème de l'interface entre l'élément décisionnaire d'un AAV et son élément animateur constitue un véritable défi. Ces éléments sont montrés de manière différente dans l'EV. Afin de le visualiser et de l'animer, on désigne l'EV par un ensemble d'objets géométriques, dans lequel la plupart des modèles comportementaux utilisent une représentation abstraite et symbolique. La perception est un processus important qui permet de réduire la différence entre les deux éléments précités: la décision et l'animation, ce qui conduit l'AAV à prendre conscience de son environnement. Gilles (2001) a en outre observé que, pour que la simulation du comportement humain soit fidèle, "elle doit inclure l'interaction des humains virtuels avec leur environnement et, pour cela, simuler leur perception de ce dernier".

Un AAV utilise des senseurs virtuels pour la vision, l'audition et le toucher afin d'obtenir sa perception virtuelle. Il doit assurer des actions de complexités différentes, telles que: progresser dans son EV, interagir avec lui et communiquer avec d'autres AAVs. Pour cela, il doit se déplacer aux alentours, ou être à même de saisir des objets qui correspondent à sa perception virtuelle.

Les limitations apparaissent lorsque les actions de l'AAV exigent une connaissance dynamique de l'environnement, donc un système de perception. Cette exigence, jointe à la complexité intrinsèque et aux contraintes cognitives, caractérise la limite entre les actions et les comportements de l'AAV. La liaison avec les actions réactives, qui exige la perception, mais non la mémoire de ce qui a été perçu, apparaît alors. Dans une approche courante, le comportement induit son propre mécanisme de perception, lequel génère une duplication de traitements, chaque fois que plusieurs comportements sont impliqués.

3.3.1 Introduction

De nombreux chercheurs ont étudié l'implémentation d'agents à mécanismes d'animation sensitifs et à mémorisation interne.

Gilles (2001) utilise une approche psychologique pour construire son algorithme de vision. Par cette méthode, il ne prétend pas simuler l'image sur la rétine de l'humain virtuel, ou lui permettre de percevoir des éléments tels que la couleur ou la forme. Chopra et

Badler (2001) ont introduit un cadre avec le comportement d'attention visuelle d'un agent humain virtuel, basé sur les études psychologiques, les facteurs humains et la vision par ordinateur. Itti (2003) a utilisé le mécanisme d'attention visuelle sélective pour diriger rapidement l'oeil vers des objets intéressants de l'environnement.

Hill (2000) a produit un modèle d'attention perceptive, pour créer des pilotes humains virtuels acceptables dans des simulations militaires. Les objets perçus ont été groupés selon divers critères. La finesse de perception de ces derniers dépendait du niveau d'attention et des objectifs du pilote. Peters et O'Sullivan (2002) ont combiné des modules de vision synthétique avec un modèle de mémoire construit sur la théorie des étages (tirée de la psychologie cognitive), afin d'obtenir qu'un humain virtuel prête attention à son environnement. Le modèle de mémoire était utilisé pour mémoriser les informations, perçues et attendues, sur les objets à différentes étapes du processus de filtrage. Peters et O'Sullivan (2002) ont ajouté, dans leur œuvre, un système de production automatique d'opérations d'attention, de bas en haut, des humains virtuels. Par attention de bas en haut on entend la manière dont l'environnement attire l'attention, sans que l'on tienne compte du niveau de la tâche. Courty et Marchand (2003) a introduit un modèle de simulation de la perception visuelle basé sur la détection d'éléments caractéristiques, qui correspond au maximum de vraisemblance visuelle d'une image statique, ou d'une séquence d'images.

Notre approche intégrée d'une perception virtuelle, découle des recherches en neurosciences, lesquelles se rattachent au problème de l'unité de la perception. Berthoz (1997) avance l'hypothèse suivante: "le principe d'un vaste réseau de systèmes compacts spécialisés, complètement décentralisés mais globalement convergents, peut aussi être utilisé dans un ensemble artificiel pour combiner les différentes informations sensorielles et les algorithmes de traitement".

Les problèmes de cohérence et d'unification de la perception ne peuvent être résolus par la géométrie et la dynamique seules. On doit aussi utiliser les mécanismes actifs centraux, qui doivent être ajustés au décalage différentiel dans le temps entre les senseurs, et à l'unification des espaces perceptifs. La perception est une interprétation; sa cohérence dépend des facteurs endogènes et des actions qui doivent être entreprises. Elle est aussi essentiellement multi-sensorielle car elle utilise des référentiels nombreux et instables. Cependant, cette perception est avant tout prédictive étant donné que la mémoire permet d'anticiper les conséquences d'une action future.

Cependant, il n'y a pas que le nombre de senseurs qui soit important, il y a aussi les questions que l'esprit humain curieux pose à un monde établi sur des hypothèses qu'il a élaborées et des tâches qu'il se propose d'accomplir. Les sens constituent la source des hypothèses et contribuent à leur vérification. Diriger le regard est l'une des premières fonctions nécessaires au développement de l'esprit curieux, lequel simule l'action. Berthoz (1997) a dit "aller dans la direction de mon regard" et non "regarder là où je regarde". Nous simulons mentalement le trajet et nous comparons le mouvement que nos pieds vont faire à celui prédit. Nous nous sommes beaucoup inspirés de cette méthode pour modéliser notre approche de la perception prédictive virtuelle.

L'opération de prédiction peut être décrite, chez les humains, pour, par exemple, anticiper la trajectoire d'une balle qu'ils doivent attraper, éviter des obstacles mobiles, ou se préparer au réveil durant les ultimes heures du sommeil. Dans le système visuel seul, certaines zones du cortex traitent des configurations, alors que d'autres s'occupent des formes, du mouvement, des distances et des couleurs. Ces processus sont inconscients.

Plusieurs méthodes ont été proposées pour implémenter la perception. Dans notre approche nous proposons une nouvelle méthodologie de perception prédictive, inspirée en partie par le concept de perception active décrit par Bordeux et *al.* (1999) pour la vision synthétique et l'accès à la base de données de l'EV. Dans ce modèle, un AAV maintient un schéma de perception dans lequel chaque partie correspond à un senseur virtuel spécifique. Un *pipeline* se compose de filtres capables d'extraire l'information appropriée des données fournies par le senseur correspondant. Nous allons décrire, dans les sections suivantes, les idées principales qui ont été introduites dans le cadre de notre *ALifeE* Conde et Thalmann (2004).

3.3.2 Perception unifiée virtuelle

Notre approche d'une perception unifiée virtuelle est incluse dans notre plate-forme *ALifeE* qui contient des senseurs virtuels pour la vision, l'audition et le toucher, ainsi que des perceptions virtuelles pour la proprioception, la perception active et prédictive.

L'information provient de différentes modalités sensorielles et est traitée par régions spécialisées avec l'aide de notre plate-forme *ALifeE* (voir Figure 3-1). Ceci nous conduit à une perception unifiée, qui résulte des connexions réciproques entre ces

différentes régions. De plus, les boucles d'interaction constituent des moyens particulièrement efficaces et rapides de partage du traitement de l'information entre des systèmes variés, puisque le temps de réponse total n'excède jamais celui du système le plus lent. Nous nous sommes inspiré de l'approche décrite par Ledoux (1996).

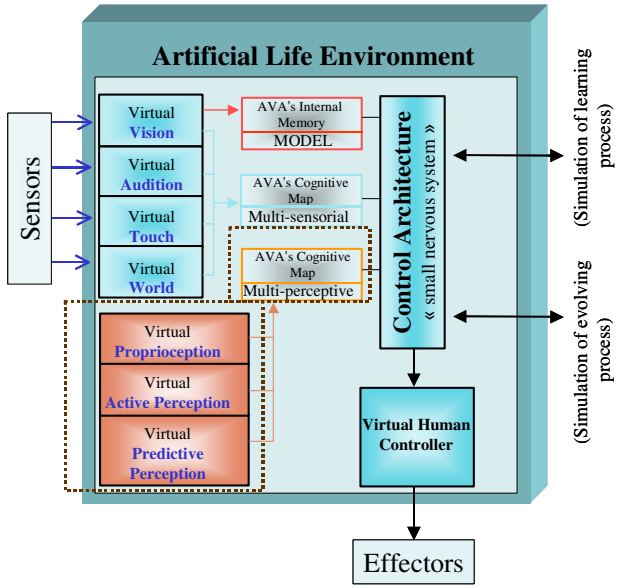


Figure 3-1. *Représentation schématique de notre ALifeE. La vision virtuelle découvre l'EV, construit les différents types de perception et remet à jour la "carte cognitive" de l'AAV pour en rendre la représentation multi-perceptive. L'architecture de contrôle utilise alors aussi bien les deux "cartes cognitives" que le "modèle de mémoire", afin de les intégrer avec les processus d'apprentissage, de développement et de contrôle de l'AAV (Virtual Human Controller).*

3.4 Proprioception virtuelle

3.4.1 Introduction

Notre proprioception repose sur l'intégration d'un seul modèle, mais triple: à variables endogènes et homéostatiques, et à apprentissage renforcé, proposé par Bersini (1994) et adapté pour notre plate-forme *ALifeE*.

La notion de variable endogène se réfère à l'état interne de l'AAV, lequel dépend principalement des perceptions et des actions de ce dernier. Des influences supplémentaires entraînent une différenciation des effets des mêmes perceptions sur les actions résultantes de l'AAV. Les variables endogènes constituent une importante catégorie d'intermédiaires cognitifs entre la sensibilité de ces dernières et les pôles des boucles de comportement du contrôleur humain virtuel (voir Figure 3-1).

La notion d'homéostasie décrit des variables dont la dynamique dans le temps garantit qu'elles restent à l'intérieur des limites préétablies. Puisque dépasser ces limites entraînerait soit un plus grand "inconfort" de l'AAV, soit le malaise de l'être humain, des actions sont entreprises pour prévenir la séparation de ces variables d'avec les valeurs préétablies.

Pour être totalement autonome, l'AAV ne doit pas seulement être capable d'actions intelligentes mais être aussi auto suffisant. Le problème délicat du maintien simultané de la viabilité de toutes les variables a été résolu en implémentant un mécanisme d'apprentissage renforcé, appelé *Q-Learning* Sutton et Barto (1998). Ceci a fourni une séquence d'actions pour maintenir l'AAV viable malgré de fortes contraintes exercées par l'EV et l'inconstance endogène propre de l'AAV.

3.4.2 Méthodologie

Nous avons modélisé, à l'aide de la Figure 3-2, l'ensemble des fonctionnalités nécessaires à notre proprioception virtuelle, et qui s'intègre dans notre architecture *ALifeE.*

Notre proprioperception virtuelle explore un monde virtuel inconnu (**ALEiD**.3ds) et construit une "carte cognitive" (octree). Cette dernière étant mise à jour à la fois par la voxelization et par le maintien des facteurs d'attention ($\boldsymbol{A_f}$) et de distance ($\boldsymbol{D_f}$) des objets graphiques que l'AAV a perçus (Percepts).

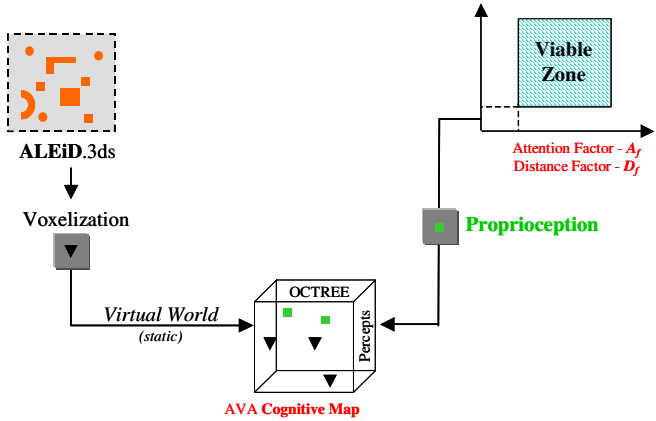


Figure 3-2. *Modélisation de la proprioception virtuelle.*

3.4.3 Implémentation

Etant donné la présence de variables endogènes dans son architecture, le comportement d'un AAV n'est pas uniquement influencé par sa perception Bersini (1994). Ces variables peuvent être associées aux états intérieurs mentaux ou émotionnels, et aux réactions aux stimuli extérieurs. Ils ne doivent pas être confondus avec un autre type de variables internes (forme de mémoire à court terme) qui permet à l'AAV de décider de ses actions sur la base non seulement de ses perceptions et de ses actions courantes, mais également précédentes. Les variables endogènes apprennent à l'AAV à se conformer aux contraintes homéostatiques, telles que, par exemple, se maintenir à l'intérieur des frontières prédéterminées d'une zone de viabilité (voir Figure 3-3).

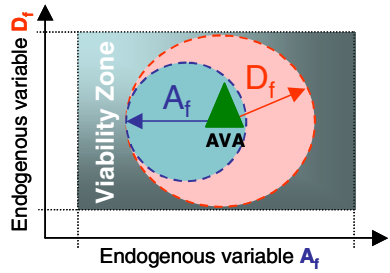


Figure 3-3. *Représentation schématique des deux variables endogènes A_f et D_f utilisées en perception prédictive virtuelle.*

La proprioception génère les deux variables endogènes (voir Figure 3-2) nécessaires au bon fonctionnement de la perception prédictive virtuelle:

- ***Attention Factor*** ***(A_f)*** – réalise un compromis entre un AAV analysant partiellement (et non complètement, car alors le déplacement en temps réel serait pratiquement impossible), et un AAV agissant à l'aveugle (passant à travers les objets).
- ***Distance Factor (D_f)*** – tient compte uniquement des obstacles qui se trouvent à l'intérieur d'un périmètre défini.

La mise à jour de ces deux variables est effectuée par le *Q-Learning* (***Q-values***). Le *Q-Learning* (relation 3.1) apprend à l'AAV à choisir quelle action est à exécuter pour chaque état, qui lui-même dépend de l'état perceptif $\boldsymbol{P_e}$ de l'AAV et des variables endogènes $\boldsymbol{A_f}$ et $\boldsymbol{D_f}$.

A la fin de chaque séquence d'actions, la politique d'action choisie maximise la fonction d'utilité, calculée sur la séquence complète, selon:

$$Q(A_f, D_f, P_e) \leftarrow (1 - \alpha)\, Q(A_f, D_f, P_e) + \alpha [r + \gamma \operatorname{Max} Q(A_f' D_f' P_e')] \qquad (3.1)$$

où: r = renforcement, α = taux d'apprentissage et γ = taux d'amortissement

Puisque l'objectif d'apprentissage de l'AAV est de maintenir ses variables endogènes à l'intérieur de leur zone de validité, le type de mesure de renforcement doit être choisi selon $\alpha = 0.8$, $r = -1$, lorsque la zone de viabilité est dépassée.

3.5 Perception active virtuelle

3.5.1 Introduction

Dans une approche classique, chaque mode de comportement est implémenté avec son propre mécanisme de perception. Nous avons perfectionné la méthode proposée par

Bordeux (1999) en ajoutant les modalités sensorielles de la vision, de l'audition et du toucher des objets graphiques.

L'AAV maintient un ensemble de *pipelines* perceptifs, chacun correspondant à un senseur virtuel. Un *pipeline* est composé de filtres auto-coordonnés ou *AgentFilters*, capables d'extraire l'information significative des données fournies par le senseur correspondant (voir Figure 3-4).

La perception devient active une fois que l'information a été acquise, filtrée et simplifiée par les différents senseurs virtuels. Elle est ensuite intégrée dans notre plate-forme *ALifeE*. Ce système de perception peut être utilisé avec la base de données de chaque senseur virtuel correspondant à une liste d'objets, de sons ou d'événements perçus, comme, par exemple, des collisions.

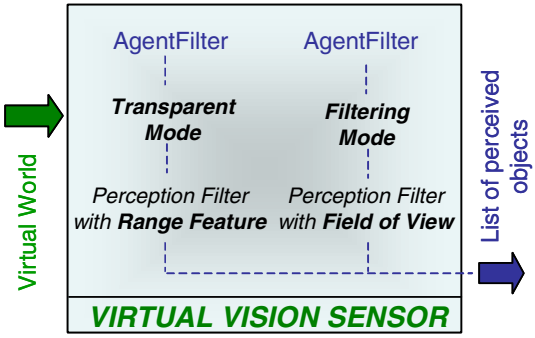


Figure 3-4. *Exemple d'un pipeline de perception active virtuelle d'un senseur virtuel de vision.*

Une perception active virtuelle fournit le cadre à la perception du comportement de l'AAV et prépare la perception unifiée virtuelle. Par exemple, la localisation sonore peut être prédite, à partir de sa localisation visuelle par une transformation des coordonnées centrées sur l'œil en celles centrées sur la tête (Chapitre 2 – Senseurs virtuels).

3.5.2 Méthodologie

Nous avons modélisé, à l'aide de la Figure 3-5, l'ensemble des fonctionnalités nécessaires à notre perception active virtuelle, et qui s'intègre dans notre architecture *ALifeE*.

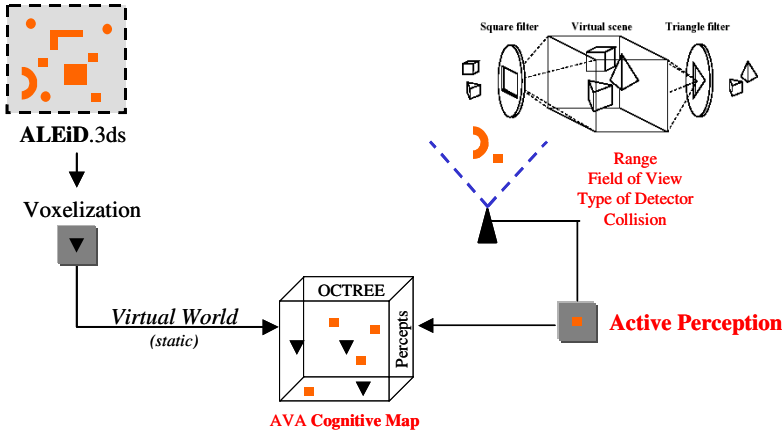


Figure 3-5. *Modélisation de la perception active virtuelle.*

Notre perception active virtuelle explore un monde virtuel inconnu (**ALEiD**.3ds) et construit une "carte cognitive" (octree). Cette dernière est mise à jour à la fois par la voxelization et par l'utilisation de différents pipelines, comme par exemple pour la vision d'objets graphiques que l'AAV a perçus (Percepts).

3.5.3 Implémentation

L'implémentation de la perception active virtuelle est conçue selon le diagramme UML de la Figure 3-6. Le pipeline de perception est divisé en deux branches, contenant l'une un senseur pour la vision – *OpticalSensor*, et l'autre un senseur pour l'audition - *AudioSensor*. De plus, la sous-méthode *AgentFilter* enregistre les objets virtuels sélectionnés, après perception par l'AAV.

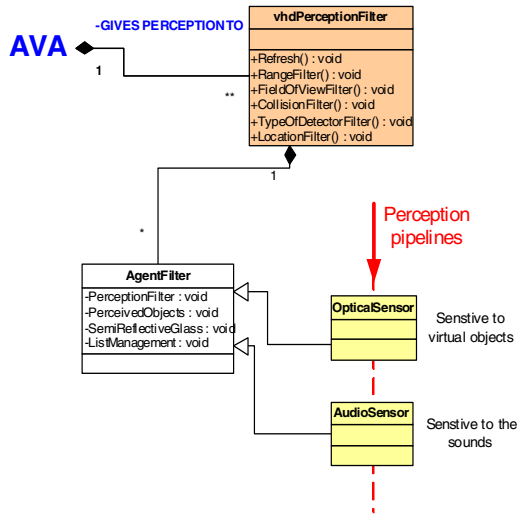


Figure 3-6. *Diagramme UML pour la perception active virtuelle.*

3.6 Perception prédictive virtuelle

3.6.1 Introduction

Une notion importante de la perception virtuelle de l'AAV est la faculté de prédire. Comme elle est l'une des activités principales du cerveau humain, il est naturel qu'elle joue un rôle fondamental dans la perception active. Elle donne, par exemple, la possibilité à l'AAV de diriger son regard et son attention sur un objet sonore.

Notre modèle de perception prédictive virtuelle est basé, en partie, sur la théorie mathématique des observateurs. Les algorithmes sont utilisés pour prédire l'état interne d'un système, généralement non linéaire, à partir des mesures partielles et souvent externes Jacobs (1993). Un observateur (senseur logiciel) comprend typiquement une simulation de système utilisant un modèle interne, si possible approximé, qui est guidé et corrigé par les mesures faites sur ce même système (voir Figure 3-7).

En perception active virtuelle, et sous certaines conditions, l'observateur permet aussi la sélection, ou la combinaison, des mesures. Comme dans un système nerveux, il est le plus utile pour améliorer l'estimation de l'état du système à un moment donné.

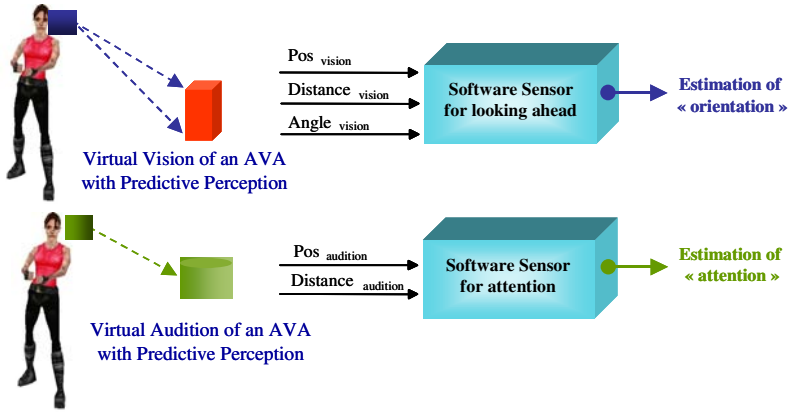


Figure 3-7. *Représentation schématique d'une perception prédictive virtuelle, avec un AAV ayant la possibilité de diriger son regard et son attention ailleurs. Avec son senseur de vision virtuel, l'AAV détermine la position, la distance et l'angle de vision. Le senseur logiciel estime alors l'orientation prédictive de l'objet graphique. Avec son senseur d'audition virtuel, l'AAV détermine la position et la distance de l'objet sonore, et le senseur logiciel estime le degré d'attention prédictive.*

Pour notre perception prédictive virtuelle, nous nous sommes, en plus, inspiré du fonctionnement de l'œil humain. Cette sphère est un organe mobile, contenu dans la cavité oculaire, qui empêche la translation (mouvement avant-arrière) mais qui, grâce aux muscles, permet la rotation dans une infinité de directions. Le champ de vision peut atteindre 200° (gauche/droite, deux fois 100°).

3.6.2 Méthodologie

Nous avons modélisé, à l'aide de la Figure 3-8, l'ensemble des fonctionnalités nécessaires à notre perception prédictive virtuelle, ensemble qui s'intègre dans notre architecture *ALifeE*.

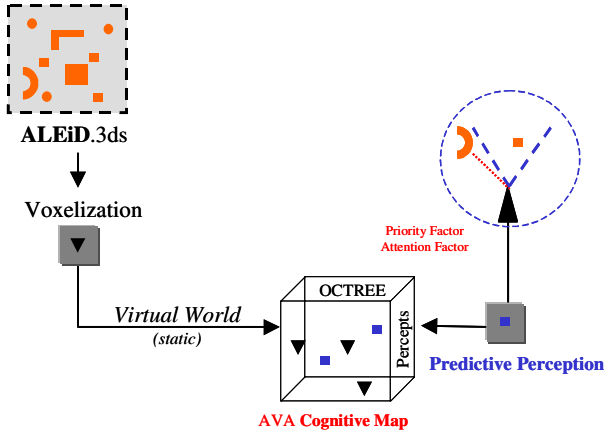


Figure 3-8. *Modélisation de la perception prédictive virtuelle.*

Notre perception prédictive virtuelle explore un monde virtuel inconnu (**ALEiD**.3ds) et construit une "carte cognitive" (octree). Cette dernière est mise à jour à la fois par la voxelization et par le maintien des facteurs d'attention (A_f) et de priorité (P_f) des objets graphiques que l'AAV a perçus (Percepts).

3.6.3 Implémentation

Notre modèle de perception prédictive virtuelle (voir Figure 3-9) repose sur un ensemble d'étapes simultanées:

1. L'identification d'une situation particulière à l'intérieur d'un EV situé.
2. L'identification de l'objet à travers différents mécanismes physiques et contextuels. Le senseur de vision virtuel fournit l'identification et approxime la position de l'objet. La position est représentée par un point entouré d'une silhouette des contours de l'objet.
3. Le calcul du ***Priority Factor*** (P_f) (de 1 à 10) – définit la perception prioritaire du groupe d'objets et est située dans un certain contexte (par exemple, feux de trafic et signaux routiers dans une ville virtuelle).
4. Le calcul de l'***Attention Factor*** (A_f) (de 1 à 10) – détermine la vitesse et la précision de la poursuite de l'objet. Ce facteur est fourni par la proprioception

de l'AAV (voir Figure 3-3). Il est variable dans le temps et revient à 1 lorsque l'objet est identifié. Il est influencé par le niveau d'interpolation de la perception, qui est inversement proportionnel à l'information acquise.

5. L'identification réelle avec extraction de l'information utile.

Notre partie visuelle est basée sur les hypothèses suivantes:

1. L'acte de vision est entrepris par un seul œil. Il correspond à une projection en coordonnées sphériques ($\boldsymbol{\theta}$, $\boldsymbol{\varphi}$ et $\boldsymbol{R}$) de l'$\boldsymbol{AVA}_{x,z}$ et au plan de focalisation par la méthode de projection ($\boldsymbol{\delta}$).
2. Les objets se déplacent lentement et l'$\boldsymbol{AVA}_{x,z}$ se déplace encore plus lentement (en temps réel).
3. Plus l'$\boldsymbol{AVA}_{x,z}$ se rapproche de l'objet, plus la précision est grande.
4. L'$\boldsymbol{AVA}_{x,z}$ est situé par sa position et son sens de déplacement (vecteur de direction) qui dépendent du temps.
5. La mémorisation des mouvements successifs dépend de deux paramètres: l'$\boldsymbol{A}_f$ et le résultat des calculs internes.

Les étapes successives de modélisation pour la perception prédictive virtuelle comprennent:

1. L'**algorithme de visualisation** (ce pseudo-code est illustré à l'aide de la Figure 3-9) fournit un cercle de base, associé au centre de l'objet et affiche aussi, continuellement, sa silhouette. De plus, il implémente un champ de vision approximatif qui situe la distance de l'$\boldsymbol{AVA}_{x,z}$ à l'objet ($\boldsymbol{D}_i$) – portion de surface de couleur orange.
2. L'**algorithme de l'espace de vision** (ce pseudo-code est illustré à l'aide de la Figure 3-9), qui divise l'espace en cercles concentriques centrés sur le vecteur d'orientation de l'observateur, lui-même affiché continuellement sous forme de croix. L'espace est ainsi séparé en segments. Finalement, le champ de vision est divisé en portions de surface, dont la grandeur varie à mesure que la distance à partir de l'AAV s'accroît. L'opération permet une quantification spatiale de l'AAV, avec l' $\boldsymbol{A}_f$ utilisé pour l'augmentation de son niveau. Ceci rapproche le modèle du fonctionnement humain, dans lequel la perception est approximée par rapport à la position réelle. Le modèle, de plus, est simplifié.

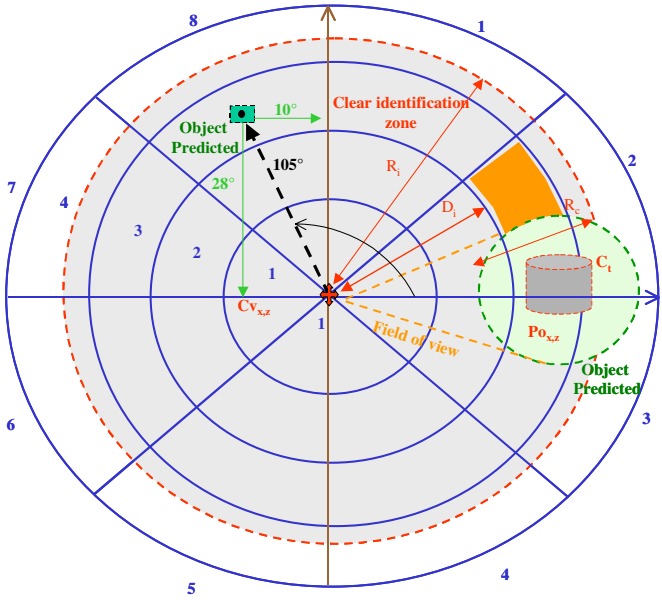


Figure 3-9. *Représentation schématique de la perception prédictive virtuelle. Espace de vision d'un AAV.* ***R_i****:rayon de la zone d'identification claire.* ***D_i****:distance de focalisation entre le centre de vision (****$Cv_{x,z}$****) et la position de l'objet qui doit être prédite.* ***R_c****:facteur de recouvrement entre* ***R_i*** *et* ***D_i****.* ***$Po_{x,z}$****:position de l'objet qui doit être prédite. La portion de surface de contour orange est associée à l'objet devant être prédit.*

3. *La position de l'objet* (***$Po_{x,z}$***) est associée aux trois valeurs nécessaires à l'identification des portions de l'espace visuel contenant l'objet: a) une portion de surface (orange sur la Figure 3-9), b) une partie du plan de focalisation, et c) un champ de vision avec la méthode de projection.

   Deux approches sont utilisées selon que l'objet prédit se déplace ou pas:

4. ***L'approche statique*** – est utilisée si l'objet prédit ne se déplace pas. Un cercle, dont le centre coïncide avec le centre de vision (***$Cv_{x,z}$***) et dont le rayon (***R_i***) est variable, définit la zone où l'objet devrait se trouver. Le rayon du cercle est relié à l'***A_f*** . Afin d'éviter des mouvements inutiles, le cercle associé au centre de l'objet et celui associé au centre de vision doivent partiellement se recouvrir (***R_c***). Ce qui, à nouveau, dépend de l'***A_f*** . On applique le même concept à la notion de champ de vision. Si ces conditions ne sont pas remplies, un mouvement dont la vitesse et la précision sont déterminées par l'***A_f*** devrait se produire. La direction du mouvement est obtenue par la procédure suivante:

projection de la position de la surface sur les axes ***x*** et ***z*** et normalisation au champ de vision défini. Pour ce qui est de la distance de focalisation, le modèle est trivial. L'objet est maintenu au centre du champ de vision jusqu'à ce que l'A_f tombe à ***1***, indication que l'objet prédit a été identifié.

5. ***L'approche dynamique*** – est utilisée si l'objet prédit se déplace. Si l'objet est en mouvement, ses positions successives sont répétées, à des intervalles constants. La vitesse n'intervient pas, puisque l'information perçue par l'œil est approximative et correspond plutôt à la perception d'un mouvement. Ceci est représenté par un vecteur d'orientation affiché continuellement dans une simulation virtuelle.

   Le nombre de points considérés et le degré d'interpolation sont égaux. Ce qui signifie que leur mémorisation part de ***1*** et augmente, selon la complexité et la vitesse du mouvement de l'AAV, d'un facteur qui peut être calculé. L'interpolation permet de déterminer la nouvelle position et d'en déduire le mouvement de l'AAV.

   La vitesse du mouvement de l'AAV dépend de l'A_f. Les coordonnées des points sont utilisées pour induire le mouvement de l'AAV, comme dans l'approche statique. Le mouvement de l'AAV et de l'objet entraîne un changement des positions relatives. La différence entre la position prédite et la position réelle est introduite dans notre modèle d'interpolation. On déduit le mouvement par une interpolation (voir la Figure 3-10) qui utilise le filtre de Kalman (1960) et les travaux de Jacobs (1993).

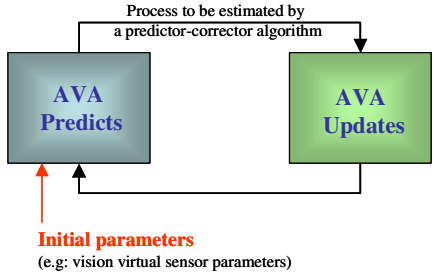


Figure 3-10. *Cycle discret courant du filtre de Kalman. La mise à jour du temps projette l'estimation de l'état courant en avant dans le temps. La mise à jour de la mesure ajuste l'estimation projetée pour une mesure réelle à ce moment-là. Pour les équations correspondantes, la description des paramètres et les algorithmes de Kalman, voir Jacobs (1993) et Kalman (1960).*

L'emploi du filtre de Kalman soulève la question suivante: Etant donné notre connaissance du comportement du système, quelle est l'estimation la meilleure de la position de l'objet? Nous savons comment le système se comporte (équation 3.2) et nous avons les mesures de la position de l'objet. Le filtre de Kalman nous permet alors d'évaluer l'état $x \in R^n$ d'un processus quantifié, réglé dans le temps, qui est régi par l'équation stochastique linéaire aux différences suivante:

$$x_k = \boldsymbol{A}\, x_{k-1} + \boldsymbol{B}\, u_k + w_{k-1} \text{ avec la mesure } z \in R^m \text{ qui est } z_k = \boldsymbol{H}\, x_k + v_k \qquad (3.2)$$

Les variables aléatoires w_k et v_k représentent, respectivement, le processus quantifié et la mesure du bruit. La matrice $\boldsymbol{A}$ relie l'état x à l'instant initial $k\text{-}1$ à l'état à l'instant courant k, en l'absence soit d'une fonction de commande optimale, soit d'un bruit de processus. La matrice $\boldsymbol{B}$ relie l'entrée de la commande $u \in R^l$ à l'état x. La matrice $\boldsymbol{H}$ relie cet état x à la mesure z_k.

3.7 *Artificial Life Environment (ALifeE)* – Perceptions virtuelles

3.7.1 Intégration des perceptions virtuelles

Toutes les fonctions perceptives utilisées dans notre plate-forme *ALifeE* sont représentées sur la Figure 3-11.

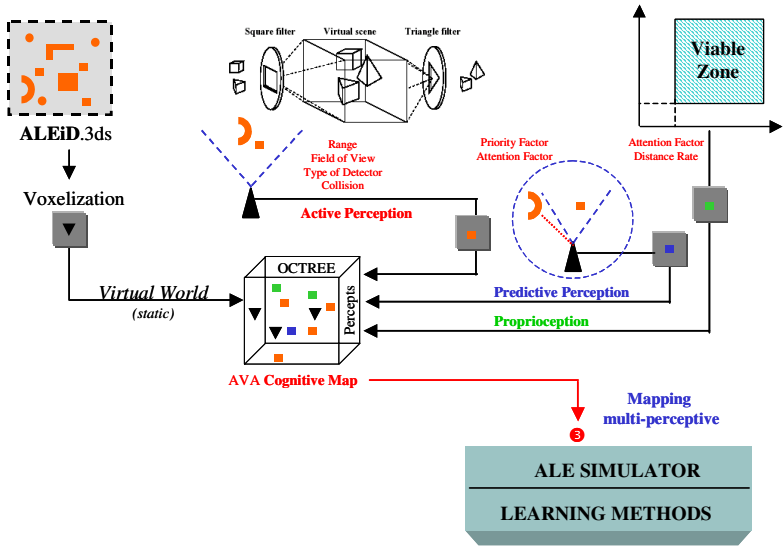


Figure 3-11. *Architecture (ALifeE) ayant conduit à nos résultats expérimentaux. L'information sémantique provenant de la représentation multi-sensorielle et multi-perceptive, et des modèles de mémoire de l'ALifeE, est utilisée par les processus d'apprentissage pour établir des modules de raisonnement de haut niveau ou des règles de mémoire complexes.*

3.7.2 Architecture logicielle et implémentation

L'implémentation de l'ensemble des perceptions virtuelles et de l'architecture de contrôle *ALifeE*, qui joue le rôle d'un "petit système nerveux", est décrite par le diagramme UML de la Figure 3-12.

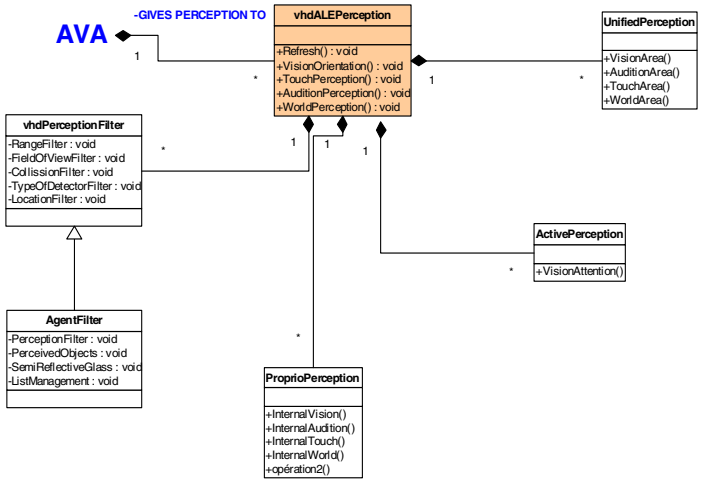


Figure 3-12. *Diagramme UML montrant l'intégration de l'ensemble des perceptions virtuelles et de l'architecture ALifeE.*

3.7.3 Expérimentation selon *ALifeE*, avec ses perceptions virtuelles

A l'aide de scénarios différents, nous avons pu vérifier que notre plate-forme *ALifeE* associe les modalités sensorielles et perceptives de manière cohérente. Nous avons utilisé deux approches pour réaliser une perception unifiée virtuelle: l'une avec une perception active et l'autre avec une perception prédictive d'un AAV.

Modèle simplifié

Au début, nous avons adopté un modèle simplifié pour l'interpolation des différentes positions, réelles et prédites, de l'objet. On utilise l'équation 3.3 pour prédire la succession des positions de l'objet, donc sa trajectoire (voir Figure 3-13).

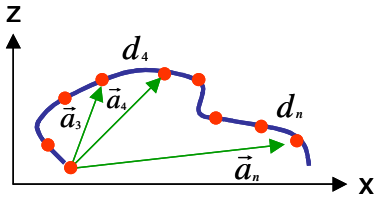


Figure 3-13. *Modèle simplifié de l'interpolation des différentes positions de l'objet.*

$$k = \frac{\sum_{i=1}^{n} (\vec{a}_i \times \vec{a}_n)}{\sum_{i=1}^{n} d_i} \quad \text{avec } k \in [0.0,\ldots, 1.0] \tag{3.3}$$

où: k = coefficient de la tendance du mouvement (trajectoire de l'objet), $\vec{a}_{i...n}$ = positions i de l'objet et d_i = distance entre les positions i.

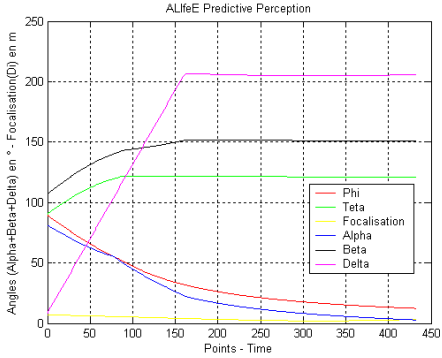
Après plusieurs simulations en temps réel, nous avons déterminé empiriquement que si la somme des produits scalaires par rapport à la somme des distances, se situait entre:

- *1.0< k >0.8*, les trajectoires de l'objet étaient quasi linéaires. Nous prenons alors le dernier vecteur calculé pour prédire la position de l'objet.
- *0.8 < k >0.4*, les trajectoires de l'objet étaient quelquefois linéaires, quelquefois discontinues et saccadées. Nous prenons alors la moyenne des vecteurs calculés pour prédire la position de l'objet.
- *0.4< k >0.2*, les trajectoires de l'objet étaient fortement discontinues et saccadées. La prédiction était alors plus complexe à estimer, et nous utilisions, dans ce cas, le filtre de Kalman pour prédire la position de l'objet.

Modèle complexe

La Figure 3-14 montre comment le filtre de Kalman, lorsque les trajectoires de l'objet étaient discontinues, pouvait interpoler la position prédite, avec ses paramètres: $\boldsymbol{\theta}$ (azimut), $\boldsymbol{\varphi}$ (angle du plan horizontal) et $\boldsymbol{\delta}$ (distance de focalisation), et la position réelle,

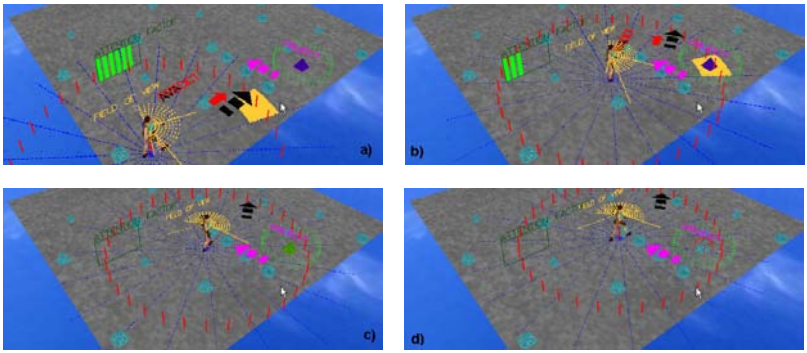
en coordonnées sphériques, des yeux de l'AAV: α (écart, par rapport à θ), β (écart, par rapport à **φ**) et Δ (écart, par rapport à **δ**). Sur cette figure, les 150 premiers points montrent la discontinuité des mouvements de l'AAV avant que celui ne puisse détecter l'objet graphique à prédire. Ensuite, l'AAV continue son chemin vers son prochain objet



graphique à prédire, les 280 points suivants sur la figure.

Figure 3-14. *Filtre de Kalman qui combiné avec notre modèle complexe de perception prédictive virtuelle, était capable d'estimer correctement la position de l'objet, mesuré pendant une "fenêtre" de 430 points-temps du déplacement de l'objet.*

Le modèle d'interpolation pour la prédiction de la trajectoire de l'objet était stabilisé et intégré dans le cadre de notre plate-forme *ALifeE*. Les Figures 3-15a) à d) montre un instantané de l'AAV prédisant la trajectoire d'un objet pyramidal dans son EV, composé de différents objets graphiques tels que cubes, cylindres et pyramides.



Figures 3-15a) à d). *Instantanés ALifeE de la perception prédictive virtuelle. L'objet prédit est une pyramide bleu foncé, située dans l'angle inférieur droit.*

Sur la figure, nous avons représenté les caractéristiques suivantes:

- l'AAV se déplaçant dans un EV (flèche discontinue en noir indiquant le mouvement de marche de l'AAV),
- l'objet graphique (pyramide bleue) se déplaçant dans un EV (flèche discontinue en violet indiquant le mouvement dynamique de l'objet graphique),
- le champ de vision de l'AAV (flèche rouge indiquant le sens du champ de vision), et
- le facteur d'attention ($\boldsymbol{A_f}$) en haut au milieu de la figure (barre-graphe vert foncé).

Nous avons appliqué notre approche à une méthodologie d'apprentissage bayésien non paramétrique, plus précisément la méthode des *k*-plus proches voisins (*k*-ppv), en utilisant un algorithme de recherche à grande vitesse Mitchell (1997).

Le résultat peut se comparer favorablement à ceux d'autres travaux:

- Gilles (2001) comme mentionné plus haut, dans 3.3 vue d'ensemble, a observé que pour qu'une simulation du comportement humain soit efficace "elle doit inclure l'interaction des humains virtuels avec leur environnement et, pour cela, simuler leur perception de ce dernier".
- Pour un AAV, les attentes sont semblables à celles de l'agent pédagogique STEVE Rickel et Johnson (1999). Mais, là où le module de perception fournit l'information aux modules cognitifs et du contrôle du moteur, notre méthode d'apprentissage utilise des "cartes cognitives" multi-sensorielles et multi-perceptives pour mettre à jour l'EV.
- Finalement, comme le remarque Blumberg (1996), "un comportement crédible commence avec une perception fiable".

Nous décrivons, ci-après, un exemple d'intégration multi-perceptive ainsi qu'un essai d'application, avec "Le gardien de but virtuel", qui démontrent la faisabilité du modèle proposé.

3.7.4 Exemple d'intégration multi-perceptive

Nous avons utilisé notre plate-forme *ALifeE* pour l'exploration d'un EV inconnu par un AAV. La mémoire interne visuelle de l'AAV lui permet de se déplacer et de naviguer dans les alentours (voir Figure 3-16).

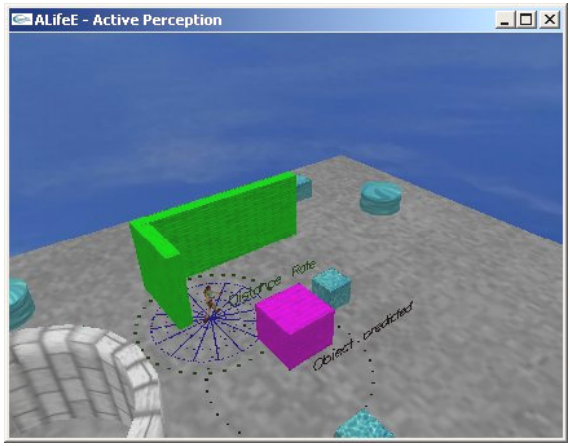


Figure 3-16. *Instantané ALifeE de la perception active virtuelle d'un AAV.*

Sur la Figure 3-16, l'AAV se déplaçant dans un EV et, grâce à sa perception active virtuelle, est à même de percevoir les objets graphiques: le mur (en vert) et le grand cube (en violet). En se basant sur sa perception active virtuelle, l'AAV peut maintenant prédire l'objet grand cube en se servant de sa perception prédictive virtuelle.

3.7.5 Etude de cas, avec "Le gardien de but virtuel"

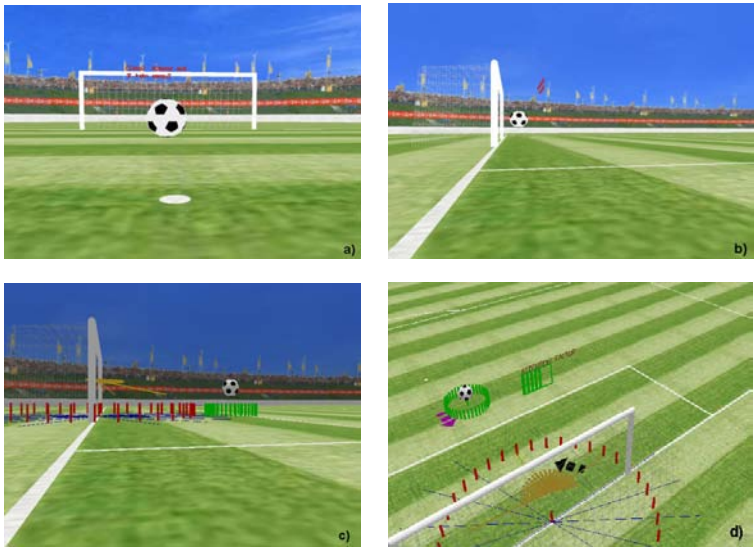
Il s'agit d'un jeu que nous appellerons "Le gardien de but virtuel". Ce jeu consiste à empêcher le ballon de football de pénétrer dans les filets du but en plaçant les mains du gardien sur sa trajectoire. Afin de savoir si le ballon a "rencontré" l'un des gants du gardien de but, il faut effectuer, à chaque instant, une détection de collisions entre le ballon et les gants du gardien. En effet, cette détection permet de savoir s'il y a une interaction entre des objets, c'est-à-dire s'ils occupent le même espace au même instant.

Il existe deux catégories de détection de collisions. La première traite de la collision entre un objet en mouvement et les géométries statiques appartenant à l'environnement. La

seconde étudie la collision entre deux objets qui sont en mouvement. C'est cette dernière qui a été appliquée dans "Le gardien de but virtuel", pour autant que le ballon et les gants du gardien soient constamment en mouvement dans un plan horizontal.

Il y a plusieurs techniques de détection de collisions, notamment pour des objets "englobants". La géométrie des objets virtuels impliqués est cependant assez simple: le ballon correspond à une sphère, alors que les gants sont assimilés à des boîtes aplaties, autrement dit des plans. Par conséquent, si l'on veut savoir s'il y a un impact entre le ballon et l'un des gants, il suffit de tester s'il y a une collision entre une sphère et un plan.

Afin de démontrer la faisabilité de notre approche de "capture intelligente", nous avons utilisé, pour le jeu "Le gardien de but virtuel", les méthodes de perception à la fois active et prédictive, qui ont été décrits dans ce chapitre et qui sont représentées sur les Figures 3-17a) à 3-17d)), plutôt que l'approche classique, mentionnée plus haut, de détection de collisions.



Figures 3-17a) à d). *Instantanés de l'intégration de nos perceptions actives et prédictives représentées sur les quatre figures avec "Le gardien de but virtuel".*

Sur la Figures 3-17a) à 3-17d), le gardien virtuel se déplace (flèche discontinue en noir et champ de vision en jaune) dans un EV – stade de football et, grâce à sa perception

active et prédictive virtuelle, il est à même de percevoir le ballon et sa direction (flèche discontinue en violet). Le facteur d'attention (A_f) est représenté en haut au milieu de la figure (barre-graphe en couleur verte).

***Figure 3-18**. Le taux de succès des penalties (76 %) et le facteur d'attention ($\mathbf{A_f}$) sont représentés avec une vue depuis la cage du gardien de buts.*

3.8 Discussion

Pour obtenir que l'AAV soit pertinent dans l'acquisition, le filtrage et la sélection des stimuli externes et internes du monde virtuel, il est nécessaire de disposer d'une perception active et prédictive avant l'apprentissage. Ces deux perceptions sont fondamentales pour les processus d'apprentissage car elles permettent de filtrer une partie des informations qu'ils ont préalablement traitées Conde et Thalmann (2005c).

L'organisation de toutes les informations sensorielles s'ajoute donc à la sensation primaire, et lui donne un sens. La prise d'information s'avère essentielle. Or, notre environnement étant très riche en informations, ces dernières se trouvent en surabondance. L'AAV doit donc rechercher la plus intéressante. Ce qui distingue un AAV habile d'un autre, c'est sa capacité de bien choisir cette information.

Dans l'approche, présentée que nous présentons dans ce chapitre, et à l'aide des senseurs virtuels décrits au chapitre précédent, nous faisons de l'apprentissage un apprentissage d'abord perceptif. Lequel dépend, à la fois, de la maturation et du processus lui-même. Les objets virtuels, peu différenciés au début, le deviennent davantage avec l'expérience de l'AAV.

Notre architecture *ALifeE,* est capable d'intégrer les senseurs virtuels décrits dans le chapitre précédent et, avec la perception virtuelle, nous sommes maintenant en mesure de fournir aux processus d'apprentissage de bas niveau – modèle comportemental – et de haut niveau – modèle cognitif – une représentation compacte de l'EV. Autrement dit, nous obtenons une dimension *n* – nombre de variables d'entrée - suffisamment petite. Ce qui est important, nous l'avons mentionné dans l'Introduction du chapitre 2, car un espace compact des états aide l'AAV non seulement à s'adapter rapidement, mais aussi à mieux généraliser ce qu'il a appris.

Après avoir proposé de nouvelles approches pour les senseurs virtuels et les perceptions virtuelles dans un EV, et les avoir intégrées de façon cohérente à l'aide de notre architecture *ALifeE*, nous sommes maintenant en mesure de fournir l'ensemble de ces informations Conde et Thalmann (2004). Nous allons les proposer, dans le chapitre suivant, principalement sous la forme d'une "carte cognitive" pour les senseurs virtuels, et d'une autre "carte cognitive" pour les perceptions virtuelles, ainsi que d'une mémoire à court terme pour l'apprentissage d'un modèle comportemental "bas niveau" d'un AAV.

L'homme doit progresser sans cesse vers un but. Voyez un avion : au sol, ce n'est qu'un amas de boulons et de vis, mais en vol il trouve sa raison d'être et devient lui-même. Toutefois il ne peut s'arrêter, et s'il cesse d'avancer, il tombe.

Anonyme

Chapitre 4 – Apprentissage comportemental

4.1 Introduction

Après avoir proposé une approche pour la problématique des variables d'entrée, avec les senseurs virtuels et la perception virtuelle, nous traitons, dans ce chapitre, de l'apprentissage comportemental (*behavioural learning*) d'un AAV, ou apprentissage de bas niveau. En effet, nous avons opté pour une approche qui sépare l'apprentissage d'un AAV en deux parties: l'une, qui étudie celui de bas niveau (*behavioural model*), où la stabilité est faible, et l'autre, celui de haut niveau (*cognitive model*), où la stabilité est forte.

Notre modélisation s'inspire des sciences cognitives et de la théorie typique des neurosciences, plus particulièrement de notre système nerveux, car celui-ci nous montre qu'il y a une séparation entre les boucles rapides, qui constituent les réflexes, et les boucles lentes, lesquelles favorisent une forme particulière de stabilité Sherman et Koch (1998).

Au début du siècle passé, Thorndike (1911) a fait de nombreuses recherches sur l'animal, en s'appuyant également sur le processus d'association stimuli-réponse pour étayer sa théorie du connexionisme. L'apprentissage étant le renforcement d'un lien entre un stimulus et une réponse, Thorndike définit plusieurs lois, telle que celle de l'effet, qui postule que lorsqu'une réponse est satisfaisante, le sujet répétera cette réponse. Nous nous

sommes inspirés, tout au long de ce chapitre, de cette théorie fondamentale, avec quelques variantes cependant, afin de réaliser un apprentissage comportemental efficace et robuste d'un AAV.

4.2 Objectifs

Pour qu'il y ait, véritablement, apprentissage, une stabilité forte du comportement nouveau d'un AAV est nécessaire. L'apprentissage de l'AAV, centré sur l'observation du comportement en tant que réponse aux sollicitations de l'EV dans lequel il se trouve, est basé essentiellement sur les boucles rapides. C'est donc une fonctionnalité d'une action qui repose sur l'automatisme de l'AAV.

Nous avons développé deux nouvelles méthodes pour approximer un modèle comportemental à l'aide des techniques empruntées à l'apprentissage artificiel. Pour une introduction à celles-ci, on peut consulter les excellents ouvrages de Mitchell (1997), et de Cornuéfols et Miclet (2002), si l'on est de langue française. Un modèle comportemental, ou cognitif, utilise la perception virtuelle qu'a un AAV de l'état courant de son EV pour sélectionner la prochaine action à entreprendre. Plus précisément, un modèle comportemental, ou cognitif, réalise l'association: états → actions. En représentant les états et les actions comme des vecteurs de composantes réelles et de dimension fixe n et m ($état \in R^n$ et $action \in R^m$), nous avons une association $R^n \rightarrow R^m$. Ainsi, nos méthodes ont, pour finalité, que $f: \ état \in R^n \rightarrow action' \in R^m$, où $action'$ signifie que celle-ci est obtenue par approximation.

Indépendamment de la méthode d'apprentissage artificiel utilisée, la valeur n (dimension des variables d'entrée) doit être la plus petite possible, comme nous l'avons exposé au Chapitre 2 – Senseurs virtuels. C'est pour cela que le modèle d'apprentissage d'un AAV, comportemental ou cognitif (au chapitre suivant), que nous souhaitons approximer, doit obtenir peu, mais suffisamment, d'informations sur l'état courant de l'EV. Celles-ci seront présentées à l'algorithme d'apprentissage artificiel sous la forme la plus compacte possible.

4.3 Vue d'ensemble

Les techniques d'animation comportementale rendent l'AAV capable de s'engager dans une activité autonome qui lui permettre de prendre ses propres décisions. Elles donnent son nom à l'animation comportementale et constituent le modèle de prise de décision du système.

Un modèle comportemental est un modèle exécutable qui définit comment un AAV doit réagir à l'état courant de son EV. De façon alternative, nous verrons, au chapitre suivant, qu'un modèle cognitif est un modèle exécutable par un AAV, à travers un processus qui permet à ce dernier de choisir entre plusieurs actions possibles, via un arbre de recherche.

L'un des problèmes d'apprentissage les plus fascinants est celui de l'adaptation en ligne par renforcement. Il est, en effet, omniprésent dans les systèmes naturels, y compris chez les organismes les plus simples, et il correspond à une large classe d'applications pour laquelle il n'est pas envisageable de fournir des informations détaillées nécessaires à un apprentissage supervisé. Nous aborderons ce dernier, avec l'apprentissage cognitif, dans le prochain chapitre.

Dans sa forme la plus simple, la situation d'apprentissage par renforcement (AR) implique un système qui agit sur le monde et, par conséquent, soumis à une séquence de signaux correspondant aux états successifs qu'il traverse. De temps en temps, un signal de renforcement, positif ou négatif, sanctionne la séquence de décisions prises par le système. La tâche de ce dernier est de chercher une stratégie de conduite, appelée politique dans ce contexte, qui maximise l'espérance de renforcement dans les situations à venir. Ceci a lieu en passant, en général, soit par une estimation de cette espérance, soit par une fonction des états, soit en fonction des actions du système dans le monde.

L'AR est difficile, essentiellement pour deux raisons principales. Tour d'abord parce que le signal de renforcement, fourni en retour au système, est très pauvre. Ce n'est, généralement, qu'un scalaire, qui n'apporte donc que peu d'informations sur le monde et sur les décisions à prendre. Ensuite, parce que le délai qui sépare le signal de renforcement des décisions qui y ont conduit rend ardue l'attribution de mérite ou de blâme à chacune des décisions prises dans le passé.

Malgré ces difficultés, l'AR a suscité de nombreux travaux, tant en automatique qu'en apprentissage artificiel, depuis plus de quarante ans.

4.3.1 Apprentissage par renforcement (AR)

Parmi les techniques classiques d'informatique cognitive, on trouve l'AR Sutton et Barto (1998). Les algorithmes d'AR permettent, à un ou à plusieurs agents autonomes, d'effectuer une suite d'actions optimales dans un environnement donné, grâce à des techniques de récompense/pénalité ainsi que par la répétition d'essais, pertinents ou pas, par laquelle ces agents apprennent la tâche demandée.

Les processus de décision markoviens

En apprentissage artificiel, le modèle de l'AR est basé sur les processus de décision markoviens. Ces derniers traitent d'un agent autonome, situé dans un environnement qu'il ne connaît pas et où le temps est discrétisé, et avec lequel il doit interagir. A chaque pas de temps t, l'agent est dans un état s^t, et il effectue une action a^t pour se retrouver au pas de temps suivant $t+1$ dans un nouvel état s^{t+1}. Le comportement de l'agent autonome s'identifie à sa politique π, qui est la fonction: S x $A \rightarrow \pi(A)$, laquelle indique, pour tout état $s \in S$, la distribution de probabilité pour que cet agent autonome choisisse les diverses actions $a \in A$ et se trouve dans cet état. Une fois son action choisie, l'agent autonome dispose d'une fonction de transition $\boldsymbol{T}$ qui indique, pour tout couple état $\rightarrow$ action, la distribution de probabilité pour que celui-ci se trouve, au pas suivant, dans chacun des états possibles lorsqu'il exécute cette action.

Dans certains couples état $\rightarrow$ action, l'agent autonome peut recevoir une récompense, ou une pénalité, de l'environnement, sous la forme d'un signal de renforcement. L'objectif global de l'agent autonome est d'adopter un comportement qui lui permette de maximiser la fréquence et la valeur de ses récompenses. Il doit donc procéder à un certain apprentissage pour pouvoir établir un lien de cause à effet entre ses actions et les récompenses, de façon à choisir les meilleures actions.

Pour pouvoir évaluer la politique π de l'agent autonome, on dispose d'une fonction de valeur $\boldsymbol{V^{\pi}(s)}$ qui associe, à chaque état s, une mesure de la récompense cumulée qu'un

agent autonome recevra, s'il suit cette politique à partir de l'état *s*. Cette récompense cumulée est assimilée à la somme de tous les futurs signaux de renforcement et peut s'écrire :

$$\mathbf{R_t} = r_{t+1} + \gamma\, r_{t+2} + \gamma^2\, r_{t+3} + \ldots + r_T = \Sigma^{\infty}_{k=0}\ \gamma^k\, r_{t+k+1} \quad \text{avec } 0 < \gamma < 1 \qquad (4.1)$$

Afin que cette somme n'ait pas une valeur infinie, le terme γ, appelé facteur de d'amortissement, permet de prendre d'autant moins en compte les récompenses reçues que l'agent autonome les recevra plus longtemps dans le temps. On peut alors définir la fonction valeur en *s* pour la politique π récursivement:

$$\mathbf{V}^{\pi}(s) = \Sigma\, \pi(s,a)\, [\, \mathbf{R}(s,a) + \gamma\, \Sigma\, \mathbf{T}(s,a,s')\, \mathbf{V}^{\pi}(s')] \qquad (4.2)$$

L'équation 4.2 est appelée équation de Bellman pour une politique π. Cette équation joue un rôle fondamental. Elle est au cœur de toutes les méthodes d'optimisation qui permettent de définir des algorithmes d'AR.

Différentes classes d'algorithmes

Il existe trois principales classes d'algorithmes qui permettent à un agent autonome, confronté à un processus de décision markovien, de découvrir une politique optimale, autrement dit celle pour laquelle il peut espérer recevoir la récompense cumulée maximale sur le long terme:

- Les algorithmes de programmation dynamique, qui s'appliquent au cas où l'agent autonome dispose d'un modèle d'environnement, c'est à dire lorsque les fonctions de transition ***T*** et de récompense ***R*** sont connues à priori.
- Les méthodes de Monte-Carlo, qui sont simples et ne présupposent pas la connaissance à priori d'un modèle, mais présentent le défaut de ne pas être incrémentales.
- Les méthodes des différences temporelles, qui reposent sur une estimation incrémentale du modèle de l'environnement. Elles combinent par conséquent les idées des méthodes issues de la programmation dynamique et des méthodes de

Monte-Carlo. Comme les premières, elles prennent en compte les corrélations entre les états pour mettre à jour leur évaluation. Et, comme les secondes, elles n'ont pas besoin d'une connaissance a priori de l'environnement. Les méthodes des différences temporelles s'appuient également sur une alternance de phases d'évaluation et de phases d'amélioration.

Nous ne développons pas ici les deux premières classes, qui sont décrites dans Sutton et Barto (1998) et Cornuéjols et Miclet (2002), et nous nous limitons aux méthodes des différences temporelles, qui sont les plus utilisées dans le cadre de l'AR car elles regroupent les points forts des deux autres. Comme les algorithmes de programmation dynamique, elles sont incrémentales. La nouvelle valeur ***V(s)*** est mise à jour en fonction de la valeur estimée précédente ***V(s)***. Comme les méthodes de Monte-Carlo, elles n'ont pas besoin de modèle du monde car elles l'estiment sur la base de l'expérience de l'agent autonome.

La méthode des différences temporelles consiste à comparer deux prédictions (ou estimations) successives de la récompense cumulée que l'agent autonome va recevoir, et ceci à condition d'avoir au moins deux prédictions correctes. Sutton appelle l'erreur dans la satisfaction de cette condition, la *Temporal Difference-Error* (*TD-Error*) Sutton (1988).

L'apprentissage suivi ne consiste alors plus en l'attente du signal de renforcement à long terme, mais en la modification de la fonction valeur $\boldsymbol{V^{\pi}(s)}$ à chaque pas de temps en fonction de la *TD-Error* entre deux prédictions consécutives, modification représentée par la procédure suivante:

$$\mathbf{V}^{\pi}(s) \leftarrow \mathbf{V}^{\pi}(s) + \alpha \, [r^{t} + \gamma \, P^{t} - P^{t-1}] \qquad (4.3)$$

Dans laquelle α joue le rôle de pas d'apprentissage.

En apprentissage artificiel, d'autres méthodes ont dérivé de l'apprentissage par différences temporelles. Il s'agit principalement des méthodes SARSA et *Q-Learning*. Contrairement à la méthode des différences temporelles, l'algorithme SARSA travaille sur la qualité des couples état → action, et non sur la valeur des états, ce qui oblige l'agent autonome à prédire au pas précédent, l'action a^{t+1} qu'il réalisera au pas de temps suivant.

L'algorithme du *Q-Learning*, lui, n'a pas besoin de prédire l'action qui sera effectuée au pas de temps suivant, puisqu'il fait des mises à jour de la politique de l'agent autonome en fonction des actions optimales. Plus simplement, il permet de connaître l'action optimale qui sera effectuée au pas de temps suivant, ce qui lui évite de devoir faire un calcul pour la prédire. Cette propriété fait de l'algorithme *Q-Learning* l'un des plus connus en AR.

Dans la suite de ce chapitre, nous nous limiterons à la méthode des différences temporelles et du *Q-Learning*, afin d'approximer un modèle comportemental d'un AAV.

4.4 Apprentissage d'un modèle comportemental

Nous entendons par apprentissage d'un modèle comportemental, la possibilité donnée à un AAV d'apprendre à réaliser, par lui-même, une tâche de type bas niveau. Ceci permet également d'alléger la charge du concepteur/animateur de l'AAV.

4.4.1 Introduction

Les humains sont toujours situés dans un environnement dans lequel ils interagissent en permanence, à l'aide de leurs senseurs et de leurs effecteurs. Les techniques d'intelligence artificielle (IA) classique ont montré leurs limites très rapidement, car elles sont principalement basées sur des règles d'animation comportementale, programmées en avance par le concepteur.

L'I.A. située peut compenser ces limitations. Son objectif est de concevoir des systèmes artificiels adaptés qui puissent évoluer dans leur environnement, lequel n'est pas entièrement prévisible. Les techniques sont, dans ce domaine, inspirées de la biologie et peuvent être appliquées aussi à nos AAVs, en mesure d'interagir avec leur EV et dans lequel ils peuvent poursuivre plusieurs buts, parfois contradictoires. Pour qu'il y ait évolution, ces AAVs doivent utiliser l'information fournie par leurs senseurs, ainsi que nous l'avions proposé au Chapitre 2 – Senseurs virtuels. Ceux-ci doivent rechercher activement cette information à l'aide de leurs effecteurs, et l'interpréter en fonction de l'environnement qu'ils perçoivent et du but poursuivi Langton (1989).

Dans ce contexte, nous entendons par "animation comportementale" des méthodologies qui rendent chaque AAV "intelligent" et autonome, réagissant à son EV et prenant des décisions qui reposent sur ses systèmes perceptifs, mémoriels et logiques. Par "intelligent", nous entendons qu'un AAV est capable de planifier et d'effectuer des tâches qui partent de l'état actuel de l'EV. Par autonome, nous entendons qu'il est à même de visiter et de mémoriser chaque EV donné, sans intervention d'un avatar.

L'objectif recherché est de permettre à l'AAV d'explorer un EV, jusque là inconnu, de construire des structures sous forme de modèles cognitifs, ou de "cartes cognitives", telles que celles développées aux Chapitres 2 et 3, et basées sur cette exploration. Cette représentation une fois construite, l'AAV pourra aisément communiquer ses cartes, par exemple à d'autres AAVs "naïfs".

4.4.2 Notre nouvelle méthode

L'AR, ainsi que nous l'avons vu dans l'Introduction de ce chapitre, est l'une des techniques d'apprentissage artificiel en sciences cognitives Sutton et Barto (1998) et Kebling et *al.* (1996). Les algorithmes d'AR permettent, à un ou à plusieurs agents, d'effectuer une série d'actions optimales en adéquation avec un environnement donné, grâce aux techniques de récompense/pénalité.

La précision de l'apprentissage est fonction du temps alloué. Un apprentissage rapide donnera une fausse représentation de la tâche demandée, alors qu'un apprentissage lent donnera un bien meilleur résultat de l'action de l'agent. Toutes les techniques de l'AR nécessitent un compromis entre la recherche de nouvelles stratégies et l'utilisation des connaissances déjà acquises.

La "mémoire" d'un AAV est la structure de stockage des actions prédites, avec leurs poids. Lorsque l'agent utilise cette mémoire pour choisir une action, nous appelons cela la phase d'exploitation. En revanche, lorsqu'il recherche d'autres voies, on parle de phase d'exploration.

Notre méthodologie d'approximation d'un modèle comportemental consiste à utiliser les techniques d'AR, tout en développant un nouveau moteur, synchronisé avec notre plate-forme d'animation d'humains virtuels - *Virtual Human Director (VHD)* Ponder

et *al.* (2003), afin d'obtenir une animation comportementale des AAVs dans l'exploration d'un EV.

4.4.3 Intégration

En général, les méthodes d'apprentissage artificiel, telles que les réseaux de neurones artificiels ou les algorithmes génétiques, par exemple, figurent dans des bibliothèques de programmes. Dans notre cas, nous avons dû concevoir un tout nouveau moteur d'AR. Pour ce faire, nous nous sommes inspirés de l'interface en langage C++, proposé par Sutton et Santamaria (1998) et appelée: *Reinforcement Learning Interface (RLI)*.

Bien que l'interface *RLI*, dans sa version en langage C++, propose une architecture complète, aussi bien qu'un ensemble de classes d'objets assez compatible avec les différents problèmes que peut soulever l'AR, le moteur que nous avons développé, *vhdRLService*, n'utilise pas directement cette architecture. En fait, l'interface *RLI* propose une architecture de haut niveau, ce qui aurait rendu notre moteur plus complexe. De plus, cette architecture ne peut pas prendre en charge la simulation de plusieurs agents (voir Figure 4-1).

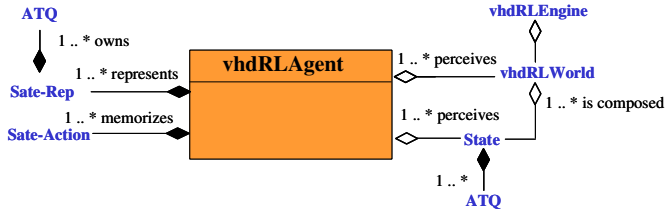


Figure 4-1. *Diagramme UML simplifié de notre moteur vhdRLService.*

En termes généraux, notre moteur d'AR se compose de trois éléments principaux inspirés de l'interface *RLI*: *vhdRLWorld, vhdRLAgent* et *vhdRLEngine*.

4.4.4 Choix des algorithmes d'apprentissage

Deux algorithmes d'AR ont été utilisés pour notre moteur *vhdRLService*. Mais ici, contrairement à l'utilisation classique de l'AR, l'objectif n'est pas de trouver le meilleur algorithme et les meilleurs paramètres pour obtenir un processus rapide d'apprentissage, comme par exemple dans un labyrinthe. Pour cela, deux techniques, celles du *Q-Learning* et celle du *TD-Learning*, ont été implémentées et utilisées dans leur version simplifiée (voir relations 4.4 et 4.5).

$$\text{Pour le } Q\text{-}Learning\text{: } Q^{\pi}(s, a) \leftarrow Q^{\pi}(s, a) + \boldsymbol{\alpha}\ [\ \mathbf{r} + \boldsymbol{\gamma}.Q^{\pi}(s', a') - Q^{\pi}(s, a)\ ] \qquad (4.4)$$

Dans la relation 4.4 ci-dessus, le renforcement ***r*** est le chemin le plus court, qui est égal à *–1*, pour toutes les actions. Ce qui permet d'évaluer la distance nécessaire, en nombre d'actions, pour atteindre l'état terminal le plus proche, représenté par un cercle rouge sur la Figure 4-2. Cette distance, de valeur négative, est indiquée pour chaque état, en noir, à droite, dans chaque cercle vert.

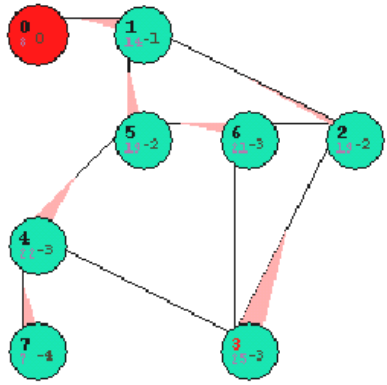


Figure 4-2. *Représentation des états de notre moteur d'AR après apprentissage au moyen de la technique du Q-Learning.*

Typiquement, le taux d'apprentissage **α** décroît avec la longueur de la simulation. Cependant, dans notre moteur, et après de nombreux essais, nous l'avons maintenu

constant ($\alpha=1$) pendant toute la durée de la simulation. Le taux d'amortissement γ a été également maintenu constant ($\gamma=1$).

$$\text{Pour le } \textit{TD-Learning}: \quad V(u) \leftarrow V(u) + \alpha \ [\ r_{t+1} + \gamma.V(s_{t+1}) - V(s_t)\ ] \tag{4.5}$$

Il est à observer que la relation 4.5 ci-dessus est établie de façon que les poids du réseau ne soient mis à jour qu'à la fin de la simulation. En d'autres termes, l'agent met à jour ses connexions seulement lorsqu'il a atteint un état terminal, ceci parce qu'il doit mémoriser l'itinéraire choisi. Par exemple (voir la Figure 4-3), dans le but de construire ce réseau d'états, l'agent est passé par l'état terminal zéro trois fois. Il a donc mis à jour sa fonction de valeur trois fois, comme indiqué en violet, à gauche dans le cercle rouge.

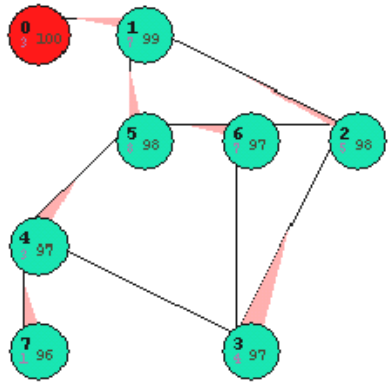


Figure 4-3. *Représentation des états de notre moteur d'AR après apprentissage au moyen de la technique du TD-Learning.*

4.4.5 Résultats expérimentaux

Nous avons synchronisé notre moteur d'AR avec la plate-forme *VHD++* Ponder et *al.* (2003) pour l'animation d'humains virtuels, afin que ceux-ci puissent se déplacer dans un EV, tel qu'une ville virtuelle. L'environnement visé ici correspond à n'importe quel type

d'EV qui impose des contraintes physiques pour la navigation. Elles sont par exemple, dans une ville: les bâtiments publics, les maisons, des rues, etc.

En fait, ce que l'AR permet de trouver, c'est une série d'actions optimales dans un environnement donné, cette approche est cependant inutile dans un environnement ouvert, où tous les états seraient reliés ensemble. Finalement, l'EV doit contenir les états terminaux, définis par l'utilisateur du moteur d'AR.

Nous avons testé notre nouvelle approche pour un EV Conde et *al.* (2003), qui est une ville constituée d'une série de bâtiments et de rues. Les nombres, sur la Figure 4-4, représentent les états du moteur d'AR.

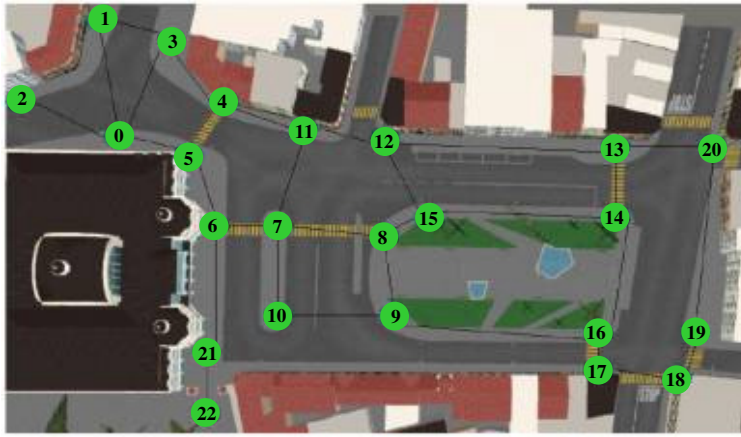


Figure 4-4. *Graphe des états du moteur d'AR de la ville virtuelle sous simulation.*

Q-Learning. Après plusieurs simulations effectuées dans l'environnement défini sur la Figure 4-4, et en mesurant le nombre d'itérations nécessaires pour obtenir le meilleur chemin à parcourir, nous pouvons dire qu'en moyenne un AAV effectue l'apprentissage en 15 itérations, avec une stratégie d'actions aléatoires choisie par le concepteur.

TD-Learning. L'AAV effectue le même apprentissage en 4 itérations, avec une stratégie d'actions choisie par le concepteur.

Nous avons décrit, au Chapitre 6 – Etudes de cas, un peu plus en détail, les résultats obtenus.

4.4.6 Discussion

Bien qu'intéressantes et utiles, les techniques d'AR par différences temporelles et par *Q-Learning* ne fournissent pas à l'AAV l'ensemble des facultés de décision. Parfois même, il est impraticable d'enregistrer une table contenant explicitement tous les facteurs de Q *(Q-factors),* que l'on doit souvent obtenir par approximation.

Nous avons réalisé différentes expérimentations (Chapitre 6 - Etudes de cas) qui utilisent plus particulièrement les deux techniques d'apprentissage comportemental d'un AAV ci-dessus. Elles permettent à l'AAV, d'apprendre de façon automatique, un modèle comportemental nouveau, où la seule information fournie est une fonction de performance (*fitness*) Conde et *al.* (2003). Cette approche peut s'avérer difficile, pour les raisons suivantes:

- Si l'ont veut avoir des résultats stables, on doit approximer la valeur de Q(état, action) avec une grande précision. Nous avons trouvé que, pour cela, il faut un très large réseau de neurones artificiels. C'est pourquoi, apprendre ces valeurs Q(état, action) demande de nombreux jours avec un micro-ordinateur actuel.
- Comme décrit par Sutton et Barto (1998), l'algorithme *Q-Learning* peut devenir difficile à optimiser en pratique, car on doit visiter chaque état → action à de nombreuses reprises.

A notre avis, toutes les difficultés inhérentes au *Q-Learning*, dans notre contexte d'EV situé, et habité par des AAVs, démontrent que cette approche n'est pas adéquate pour automatiser un modèle comportemental nouveau d'un AAV.

Pour cette raison, nous avons voulu développer une méthode alternative pour l'apprentissage, par un AAV, d'un modèle comportemental nouveau, qui intègre un apprentissage par renforcement inverse (ARI) – *Inverse Reinforcement Learning (IRL).* Cette méthode a été conçue pour être stable, simple et rapide. Nous pensons qu'elle est plus intéressante et utile pour la recherche en humains virtuels que celle basée sur le *Q-Learning*, bien qu'elle ne donne pas une garantie absolue d'aboutir à une politique optimale. En effet, même si le *Q-Learning* le permet, il est souvent impraticable à cause de la nécessité de calculer une très grande table de valeurs de Q(état, action).

4.5 Apprentissage automatique de modèles comportementaux inconnus

4.5.1 Introduction

Le comportement, chez les humains, est réalisé par une "carte cognitive" complexe et l'application d'une hiérarchie de stratégies de comportement Larkin (2003). L'ensemble du processus cognitif associatif (*cognitive mapping*) implique l'acquisition, le codage, l'enregistrement, le rappel et le décodage de l'information sur l'environnement Downs et Stea (1973). En fait, une "carte cognitive" individuelle contient souvent de nombreuses inexactitudes ou distorsions Griffin (1973). Beaucoup de ces anomalies résultent du fait que les humains utilisent principalement leur système perceptuel de vision, et ne sont pas capables de traiter chaque chose qu'ils voient, en raison de l'énorme quantité d'informations qu'ils reçoivent Kamwisher et Downing (1998). D'autres anomalies résultent de la voie par laquelle l'information est reçue, avant d'être traitée dans la structure même de la "carte cognitive".

Pour simuler, de la façon la plus semblable possible, le comportement humain, un AAV doit être séparé de son EV et seul l'accès à l'information qui provient de sa perception et de ses systèmes effecteurs doit lui être permis.

Auparavant, au Chapitre 2 – Senseurs virtuels, avec notre plate-forme *ALifeE*, nous avons proposé des méthodologies pour réunir (*mapping*) toutes les informations provenant des senseurs virtuels: de la vision, de l'audition et du toucher, sous forme d'une "carte cognitive". L'approche permet la réassociation partielle (*partial remapping*) de l'information cognitive et sémantique à un niveau comportemental. Par exemple, lorsque l'attention spatiale est amorcée avec la simulation tactile, la localisation de l'attention est seulement partiellement réassociée en coordonnées visuelles. Ce cadre génère l'information multi-sensorielle nécessaire à notre apprentissage comportemental.

Les humains oublient l'information, tandis que si quelque chose est enregistré une fois dans la mémoire mécanique d'un ordinateur, il y demeure pour toujours. Les humains et les animaux traitent sélectivement uniquement l'information qui est la plus importante pour eux, tout en continuant de chercher activement de nouvelles informations. De la

même manière, nous pouvons identifier deux types d'apprentissage dans un système intelligent:

- L'apprentissage actif, où le système sélectionne, filtre et cherche des données pertinentes, et
- L'apprentissage passif, où le système accepte toutes les données entrantes.

Nous proposons donc une nouvelle méthodologie d'apprentissage de bas niveau, combinée avec un apprentissage actif. Ce qui est réalisé par un modèle comportemental qui définit la manière dont un AAV réagit aux stimuli qui proviennent de son EV. Ainsi, cet AAV est en mesure d'apprendre automatiquement des modèles comportementaux inconnus.

4.5.2 Notre nouvelle méthode

Nous avons développé une nouvelle méthode pour réaliser l'apprentissage d'un AAV. Cette méthode utilise une recherche arborescente avec un ARI, étant donné que la technique du *Q-Learning* s'est révélée indésirable, ainsi que nous l'avons montré plus haut.

AR basé sur la planification avec des sous-agents

Dans la technique d'AR, basée sur la planification, un AAV apprend uniquement une politique sous-optimale, la qualité de celle-ci dépendant de la profondeur de la limite recherchée Mitchell (1997) et Pfeiffer et Scheier (1999). Pour garantir que l'intégralité du comportement de l'AAV soit optimal, nous proposons d'utiliser la même approche que la *Q-Decomposition* proposée par Russell et Zimdars (2003), où seule la somme des *Q-Values* des sous-agents détermine l'action optimale. Cette méthode exige que chacun des sous-agents indique, comparativement à sa propre perspective, la valeur pour chaque action. Le sous-agent *j* reporte ces valeurs $Q_j(s, a)$, liées à son action pour l'état courant *s*, vers l'arbitre. Celui-ci choisit ensuite une action qui maximise la somme des *Q-Values* (voir Figure 4-5). Dans notre approche, nous utilisons plutôt des pseudo valeurs, en lieu et place des *Q-Values*, proposées par la *Q-Decomposition*, afin de tenir compte des informations multi-sensorielles pour la vision, l'évitement et la navigation, et pour obtenir un compromis entre ces différentes informations fournies par nos senseurs virtuels.

Apprentissage de l'apprentissage, avec un ARI

Il est parfois impossible d'utiliser directement l'AR, car un AAV peut n'avoir qu'une idée approximative de la valeur de renforcement qui, quand elle est optimisée, doit générer un comportement "désirable". Dans le processus d'apprentissage, une source d'information est constituée par l'observation du comportement d'autres agents "experts", tels que ceux qui imitent ou apprennent à apprendre, et indiquent une approche possible. Dans ce cas, il est communément admis que l'objectif d'observation est d'apprendre une politique représentée par l'association des états → actions.

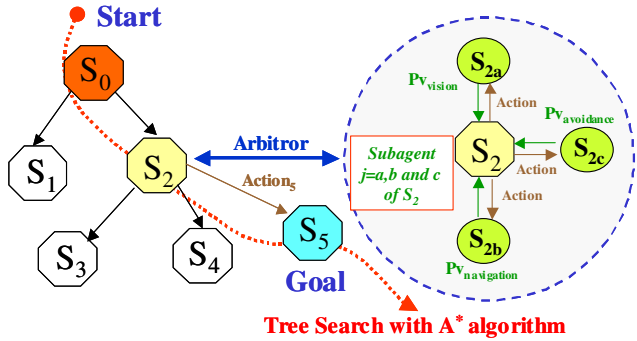


Figure 4-5. *Planification obtenue par une recherche arborescente de toutes les actions proposées pendant un temps limité.*

Nous pensons que la difficulté de spécifier manuellement la valeur de renforcement représente une barrière pour l'application à large échelle de l'AR, ainsi que pour le contrôle optimal des algorithmes. Dès lors, le domaine entier de l'AR est fondé sur la supposition que la valeur de renforcement, aussi bien que la politique, ou la fonction de valeur, constituent la définition la plus succincte, robuste et transférable de la tâche à accomplir. Il nous semble donc naturel de considérer l'approche d'apprendre l'apprentissage avec un ARI Abbeel et Ng (2004) appropriée pour obtenir la valeur de renforcement.

Dans notre méthodologie, l'ARI est utilisé pour apprendre l'apprentissage, afin d'acquérir l'expérience comportementale, et pour assurer l'optimisation de la valeur de renforcement à l'aide d'un système naturel (voir Figure 4-6). Malgré que, en utilisant l'ARI, il soit parfois difficile d'extraire la valeur de renforcement observée Ng et Russel (2000).

Nous n'allons pas tenter de programmer un modèle de comportement explicite en I.A. Avant de commencer à programmer un tel modèle, il est nécessaire d'étudier comment

la tâche doit être accomplie par l'AAV. Ainsi, notre méthodologie pour l'apprentissage de modèles comportementaux d'un AAV nous épargnera ce travail.

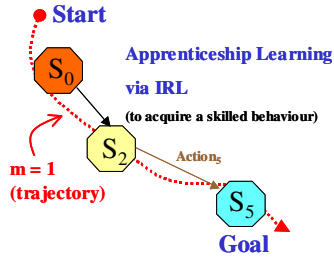


Figure 4-6. *Apprendre l'apprentissage avec un ARI. Il est achevé après m trajectoires générées par l'expert. Sur la figure, m = 1 pour une trajectoire.*

En utilisant notre méthode, un AAV peut automatiquement apprendre des modèles comportementaux inconnus. Cette approche permet d'alléger la charge de programmation imposée au concepteur.

4.5.3 Intégration et implémentation

La réalisation et l'intégration de notre méthode d'apprentissage de bas niveau pour un AAV, que nous avons appelée *AVAlowLEARN*, nous semble plus adéquate pour l'animation comportementale que les techniques basées sur une approche d'apprentissage artificiel. L'implémentation de notre méthode *AVAlowLEARN* pour l'apprentissage automatique de modèles comportementaux inconnus est décrite à l'aide du diagramme UML de la Figure 4-7.

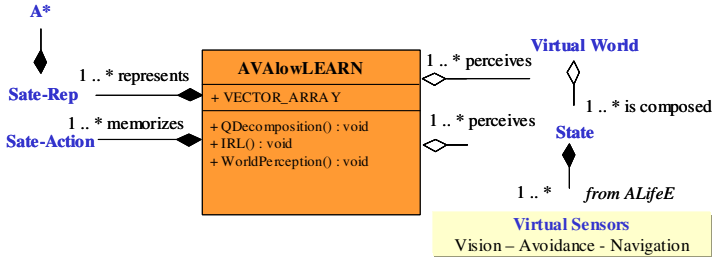


Figure 4-7. *Diagramme UML pour l'intégration de la méthode d'apprentissage automatique de modèles comportementaux inconnus (AVAlowLEARN).*

Trois étapes sont nécessaires pour la conception de notre nouvelle méthode:

- La première utilise l'algorithme A* Russell et Norvig (2003) pour observer les trajectoires (séquence d'états) générées par l'expert, qui entreprend les actions en accord avec sa politique π_E.
- La deuxième intégre la *Q-Decomposition* proposée par Russell et Zimdars (2003) mais qui, dans notre contexte, ajoute nos pseudos valeurs provenant de toutes les composantes multi-sensorielles de la vision, de l'évitement et de la navigation de l'*ALifeE*, afin de choisir les actions. La *Q-Decomposition* tient compte des multiples pseudo valeurs, chacune d'elles étant contenue dans le sous-agent. Il reste l'exigence que chacun des sous-agents doive transmettre l'estimation de sa pseudo valeur à l'arbitre, et recevoir l'action choisie par celui-ci en retour.
- La troisième apprend l'apprentissage avec un ARI et l'algorithme de projection proposé par Abbeel et Ng (2004) et adapté à notre animation comportementale.

Une politique π associe (*mapping*) les états aux distributions de probabilité, lesquelles priment les actions. Etant donné que le renforcement R est exprimable en une combinaison linéaire de caractéristiques ϕ, les valeurs d'espérance de ces caractéristiques, pour une politique donnée π, déterminent complètement la somme des espérances de renforcement escomptées, en accord avec cette politique (voir Figure 4-8).

Ceci étant défini, notre problème est de trouver - pour: a) un *MDP/R - Markov Decision Process*, sans fonction de renforcement R, b) une association de caractéristiques ϕ (*feature mapping*), et c) les valeurs d'espérance des caractéristiques de l'expert μ_E - une politique dont la performance soit aussi proche que celle de l'expert, et où la valeur de renforcement soit inconnue: $R^* = \omega^*.\ \phi$ (ω^* représentant le poids de chacune de ces caractéristiques).

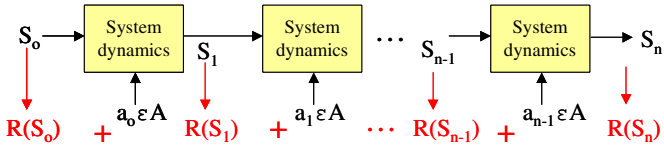


Figure 4-8. *Description de l'AR dans un processus de décision de Markov (MDP) à état fini.*

L'algorithme de projection pour apprendre l'apprentissage, algorithme proposé par Abbeel et Ng (2004) afin de trouver cette politique, est décrit en pseudo-code ci-dessous:

1. De manière aléatoire, nous prenons une politique quelconque $\pi^{(0)}$, $\mu^{(0)} = \mu(\pi^{(0)})$, et nous établissons que i = 1 (nombre d'itérations = 1).

   $\mu(\pi) = E [\sum_{t=0}^{\infty} \gamma^t . \phi(s_t) | \pi] \varepsilon \Re^k$

   (Ceci correspond à la valeur du vecteur des espérances des caractéristiques escomptées $\mu(\pi)$)

2. Nous calculons que $\bar{\mu}^{(i-1)} = \bar{\mu}^{(i-2)} + \frac{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu_E - \bar{\mu}^{(i-2)})}{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu^{(i-1)} - \bar{\mu}^{(i-2)})} (\mu^{(i-1)} - \bar{\mu}^{(i-2)})$

   (Ceci donne la projection orthogonale de μ_E sur la ligne entre $\bar{\mu}^{(i-2)}$ et $\mu^{(i-1)}$).

   Calcul de $\omega^{(i)} = \mu_E - \bar{\mu}^{(i-1)}$

   Calcul de $t^{(i)} = \left\| \mu_E - \bar{\mu}^{(i-1)} \right\|_2$

3. Si la valeur de $t^{(i)} \leq \varepsilon$, alors le calcul est terminé.

4. En nous servant de l'algorithme d'AR, nous calculons la politique optimale $\pi^{(i)}$ pour le *MDP* qui utilise le renforcement $R = (\omega^{(i)})^T \phi$.

   (L'algorithme d'AR nous renvoie la politique optimale)

5. Nous estimons de $\mu^{(i)} = \mu(\pi^{(i)})$.

6. Nous calculons les itérations i = i + 1, et continuons de nouveau à l'étape 2. de l'algorithme.

Après avoir terminé, l'algorithme nous renvoie $\{ \pi^{(i)} : i = 0... n \}$.

L'exécution de cet algorithme ne garantit que la dépendance de l'ensemble des espérances des caractéristiques, mais pas de la véritable valeur de renforcement.

4.5.4 Résultats expérimentaux

Pour tester notre méthode d'apprentissage de modèles comportementaux inconnus et déterminer si elle est capable d'apprendre le comportement montré d'un l'AAV, deux

environnements de simulation ont été conçus: a) un appartement virtuel, et b) une simulation plus complexe, avec la conduite d'une voiture à l'intérieur d'une ville virtuelle.

AAV apprenant à éviter des obstacles et à se diriger dans un appartement virtuel

Nous avons choisi de simuler un AAV à l'intérieur d'un appartement virtuel, dans lequel il peut "vivre" de façon autonome en percevant son environnement, et de lui donner la possibilité de générer des comportements en des endroits spécifiques Figure 4-9. Nous avons pu tester notre méthodologie appliquée à un AAV en implémentant le modèle avec les langages C++ et Python Rossum et Fred (2003) ainsi que notre plate-forme *VHD* pour l'animation d'humains virtuels. Cette dernière a déjà été utilisée lors de l'apprentissage d'un modèle comportemental.

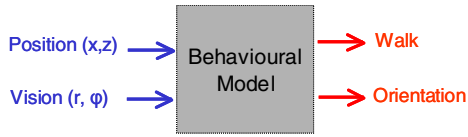


Figure 4-9. *Modèle comportemental (MDP), avec ses entrées et ses sorties.*

Ainsi que nous le constatons sur les Figures 4-10 et 4-11, une interface a été conçue pour surveiller et changer, au besoin, l'évolution des états et des actions de l'AAV, en temps réel. Le module de planification de chemin Kallmann et *al.* (2003a) est utilisé pour l'action d'évitement d'obstacles, lorsque l'AAV se déplace vers un endroit spécifique (voir Figure 4-12). Ce module est également appliqué aux décisions que l'AAV prend à chaque moment. Enfin, la visionneuse en 3D montre, en temps réel, ce que l'AAV décide de faire et peut générer des *keyframes* des actions apprises, à l'aide du moteur de marche Boulic et *al.* (2004).

Figure 4-10. *Vue d'en haut de l'appartement virtuel.*

Figure 4-11. *Vue de l'AAV apprenant en voyant, en évitant et en naviguant, à saisir différents objets virtuels, ici une boîte de CDs.*

Dans notre premier scénario, l'AAV apprend le comportement pour prendre une boîte de CDs posée sur la table de la cuisine (voir Figure 4-11). Ensuite, il la porte dans le bureau. Après cela, l'AAV marche dans les différentes pièces, tout en faisant attention d'observer et d'éviter les objets virtuels, et se dirige entre les portes à l'aide de la planification de chemin.

Dans notre deuxième scénario, l'AAV apprend à prendre un verre qui est posé sur la table du salon (voir Figure 4-13), ou dans la chambre à coucher. Il le dépose ensuite à la cuisine.

Figure 4-12. *Vue d'en haut de l'appartement virtuel avec, en médaillon, la planification de chemin, après apprentissage.*

Dans chacun de nos scénarios, notre méthode était qualitativement capable d'imiter le comportement montré. Nous avons obtenu les meilleurs résultats (voir Figure 4-14) en l'appliquant à cinquante itérations en moyenne, et en choisissant une politique par inspection.

Figure 4-13. *Vue de l'AAV apprenant en voyant, en évitant et navigant, à saisir différents objets virtuels, ici un verre*

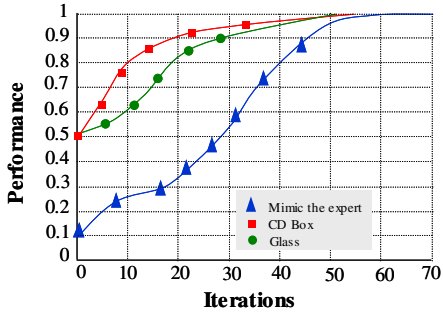


Figure 4-14. *Résultats obtenus avec notre méthode AVAlowLEARN. Toutes les animations ont été générées en temps réel avec OpenGL® (3.0 GHz avec une carte vidéo nVIDIA GeForce FX Go5350).*

AAV apprenant à conduire une voiture à l'intérieur d'une ville virtuelle

Nous avons implémenté notre approche pour l'apprentissage de modèles comportementaux inconnus, appliqués à la simulation de conduite d'une voiture à l'intérieur d'une ville virtuelle. L'AAV est un pilote conduisant une voiture. Le pilote et le copilote, qui lui apprend, ont le contrôle de la conduite par la pédale de gaz et la direction du volant (voir Figure 4-15). Les commandes sont des valeurs réelles, car l'espace des actions est continu, et la voiture peut se déplacer à n'importe quel endroit, ou prendre une nouvelle direction.

L'espace des actions étant continu, il est nécessaire de le quantifier pour réaliser l'exécution en temps réel. Par exemple, 2 minutes de simulation avec 15

séquences/seconde générant 1'800 échantillons. En conséquence, les actions possibles du pilote et du copilote deviennent limitées.

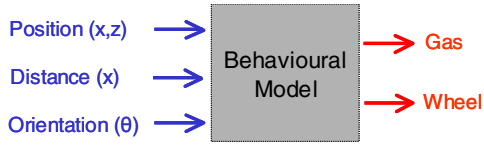


Figure 4-15. *Modèle comportemental (MDP) avec ses entrées et ses sorties.*

Notre approche communique l'information sémantique, exigée pour un comportement correct de notre méthode *AVAlowLEARN*, de deux manières : a) en ajoutant de l'information sémantique aux objets dynamiques, et b) en ajoutant des outils pour le son et les piétons, par exemple, et des objets sémantiques comme des signaux routiers. Nous détaillons au Chapitre 6 – Etudes de cas, la méthodologie que nous avons utilisée pour ajouter des informations sémantiques à l'aide de modules sémantiques enfichables (*plug in*).

Nous avons pu créer l'information sémantique pour notre simulation, mais elle comportait un humain virtuel statique. Nous souhaitions cependant obtenir de l'information sémantique dynamique et avons, pour cela, employé la pseudo-perception (voir la Figure 4-16) ainsi que la perception virtuelle, que nous avons développées au Chapitre 3 – Perception virtuelle, afin d'atteindre notre objectif.

Dans la simulation de conduite d'une voiture, ϕ est le vecteur des caractéristiques indiquant les différents choix de conduite, lesquels doivent être classifiés, tels qu'une collision avec une autre voiture, un piéton ou un objet virtuel à l'intérieur de la ville virtuelle. Le vecteur inconnu ω^* indique le poids relatif de ces différents choix.

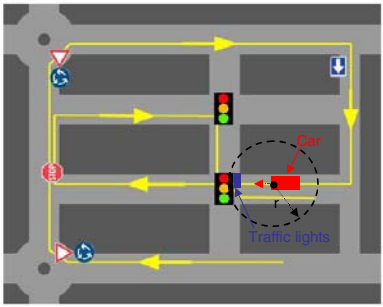


Figure 4-16. *AAV apprenant à conduire une voiture à l'intérieur d'une ville virtuelle, avec la pseudo-perception visuelle.*

Suivre aveuglément la trajectoire de l'expert ne fonctionnerait pas car le modèle du trafic est chaque fois différent.

Nous appliquons notre méthode à l'apprentissage des comportements pour: a) éviter des piétons, et b) tenir compte des feux de circulation, ce qui nécessite d'arrêter la voiture à un feu rouge, de ralentir à un feu orange et, finalement, de continuer de rouler à un feu vert.

Le résultat final est un comportement de conduite correct, puisque le pilote a pu planifier assez en avance, afin d'éviter les piétons et de réagir correctement aux feux de circulation (voir les Figures 4-17 et 4-18).

Figure 4-17. *Pilote "voyant" les feux de circulation et le piéton à l'intérieur d'une zone circulaire. L'AAV, qui a appris le comportement, l'informe, au moen de ses indicateurs, qu'il doit freiner (B pour "brake" est inscrit en rouge, en bas à droite de la figure).*

Figure 4-18. *Piéton entamant la traversée de la route selon les règles définies. Le pilote et le copilote virtuel est indiqué par deux cercles verts.*

Nous avons obtenu les meilleurs résultats (voir Tableau 4-19) en appliquant notre méthode à quarante itérations en moyenne, et en choisissant une politique par inspection.

Tableau 4-19. *Résultats obtenus avec notre méthode AVAlowLEARN. Toutes les animations ont été générées en temps réel avec OpenGL® (3.0 GHz avec une carte vidéo nVIDIA GeForce FX Go5350).*

Features	Gas	Wheel
μ_e	0.060	0.290
$\mu(\pi)$	0.059	0.266
ω	0.107	0.059

4.5.5 Discussion

Notre méthode *AVAlowLEARN* s'est révélée rapide, simple, et robuste. Elle a aussi permis un apprentissage automatique de modèles comportementaux nouveaux, pour des tâches complexes. Nous pensons donc qu'elle sera plus utile à la communauté des chercheurs en animation comportementale qu'une technique basée sur l'approche classique du *Q-Learning*.

A la différence du *Q-Learning*, notre méthode ne donne pas la garantie de trouver une politique optimale mais, comme nous l'avons mentionné auparavant, le *Q-Learning* est habituellement impraticable lorsque l'on doit traiter une grande quantité de *Q-factors*. Ce qui a été démontré par notre première méthode d'apprentissage d'un modèle comportemental Conde et *al.* (2003).

Notre méthode d'apprentissage apprend à un AAV une politique sub-optimale, dont la qualité dépend de la limite choisie de la profondeur de recherche. Elle démontre aussi qu'une optimalité n'est sans doute pas nécessaire pour un comportement intelligent. Car, un comportement intelligent est plus juste, et plus réaliste, qu'un comportement parfait.

Avec notre méthode *AVAlowLEARN*, un AAV peut automatiquement apprendre un modèle comportemental nouveau Conde et Thalmann (2005b). Cette approche décharge le concepteur du travail d'imaginer un modèle explicite. Par ailleurs, elle permet de modéliser des tâches pour lesquelles il serait difficile, ou pratiquement impossible, de développer un tel modèle.

Il y a néanmoins quelques faiblesses dans notre méthode. En exécutant l'apprentissage d'un l'AAV, il est quelquefois difficile de trouver une politique qui corresponde exactement au modèle comportemental désiré. En outre, la valeur de renforcement doit être exprimable en combinaison linéaire des caractéristiques connues.

La méthodologie qui conduit à l'apprentissage automatique d'un modèle comportemental est difficile, mais intéressante. Elle peut donner à l'infographie interactive, en particulier à celle du marché du divertissement, une dimension entièrement nouvelle. Il peut également être intéressant, pour un animateur, d'entraîner intéractivement un AAV à l'apprentissage comportemental. Tout en évitant d'utiliser une valeur de renforcement, souvent difficile à définir.

Nous avons présenté deux nouvelles méthodologies d'apprentissage comportemental. Au chapitre suivant, nous verrons que ces comportements de bas niveau, appris par l'AAV, peuvent être généralisés à l'aide d'un modèle cognitif approximé. Et aussi qu'un AAV doit avoir la capacité "d'oublier", ce qui est très important dans un apprentissage, tel que celui d'un comportement humain non-markovien, ou non-stationnaire.

Ce sont les pensées d'un homme qui déterminent sa vie.

Marc-Aurèle

Chapitre 5 – Apprentissage cognitif

5.1 Introduction

Ce chapitre traite de l'apprentissage cognitif, ou de haut niveau, d'un AAV. Cet apprentissage s'appuie essentiellement sur des techniques d'apprentissage artificiel, et plus particulièrement sur les agents intelligents, ainsi que sur les principales idées tirées de travaux en sciences cognitives.

Nous cherchons à obtenir une stabilité forte du comportement nouveau d'un AAV, soit en distinguant les domaines cognitifs, émotifs et conatifs (c'est à dire conduisant à l'action et à l'effort), soit en partant de la fonctionnalité d'une action basée sur le comportement à court et à long terme de l'AAV - lequel dispose néanmoins d'une certaine faculté d'oubli - et en lui attribuant des capacités minimales d'évolution.

Ainsi que nous l'avons montré au chapitre précédent, l'aptitude d'un AAV constitue un élément important de réalisation de sa propre performance, laquelle dépend également de sa motivation. En effet, cette motivation influence son apprentissage de plusieurs façons. D'abord, et de manière spécifique, elle dirige son action vers certains stimuli, certains buts, ou au contraire l'éloigne de ceux-ci. Ensuite, elle oriente sa perception vers des aspects pertinents, ainsi que nous l'avons montré au Chapitre 3 – Perception virtuelle.

La combinaison des actions et des automatismes de l'AAV, basés sur des "cartes cognitives" performantes, telles que décrites au chapitre précédent - Chapitre 4 - Apprentissage comportemental - permet d'interagir par des boucles lentes et rapides sur cet AAV.

Par ailleurs, les méthodes existantes d'apprentissage des modèles cognitifs présentent un certain nombre de faiblesses. En particulier, elles sont généralement peu économes en temps de calcul, ce qui limite leur utilisation. D'autre part, les modèles cognitifs se comportent parfois comme les modèles comportementaux de manière imprévue, étant donné qu'il est pratiquement impossible de les tester de façon exhaustive pour l'entier de leurs espaces d'état, en particulier si ces derniers ont des entrées continues.

Dans ce chapitre, nous proposons deux nouvelles méthodes d'approximation d'un modèle cognitif qui utilisent, entre autres, différentes variantes de l'algorithme des plus proches voisins. L'objectif de ces méthodes est de proposer une solution pour résoudre la plupart des problèmes que nous venons de décrire.

Il existe bien des méthodes intéressantes et puissantes d'apprentissage, chacune d'elles ayant ses avantages et ses inconvénients. Notre contribution étant d'adapter celles-ci à l'animation comportementale d'un AAV, en y introduisant différents mécanismes qui permettent une continuité (*smooth*) et un mixage (*blending*), lesquels correspondent à l'espace des états.

Nous traiterons aussi, dans ce chapitre, de l'évolution de l'apprentissage d'un AAV par adaptation à son EV. L'AAV est, en effet, capable d'acquérir, sans les différencier, les phases d'apprentissage et d'utilisation des connaissances.

5.2 Objectifs

Pour présenter un comportement réaliste, un AAV, qui peut être un piéton dans une ville virtuelle ou un animal "vivant" dans un EV, doit être autonome et "intelligent". Nous devons notamment traiter les données à l'aide d'une technique d'apprentissage artificiel, afin de simuler le comportement de l'AAV et de le rendre autonome Thalmann (2004). De nombreuses techniques comportementales existent et peuvent être divisées en deux principales catégories Russel et Norvig (2003): a) celles qui disposent d'un moteur

"complexe" (voir Figure 5-1), ou de type Top → Down, et b) celles qui disposent d'un moteur "simple" (voir Figure 5-2) avec système réparti, ou de type Bottom → Up Nolfi et Floreano (2001).

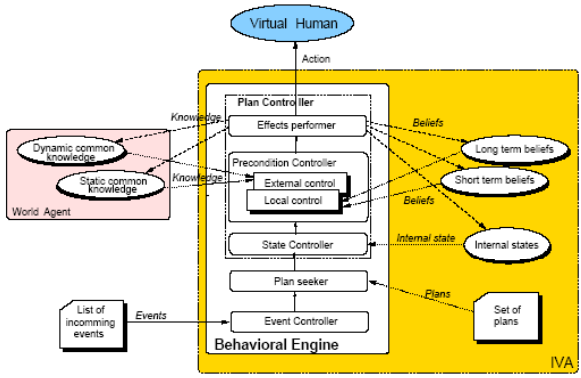


Figure 5-1. *Représentation schématique d'un agent virtuel intelligent, conçu avec un moteur comportemental "complexe" (modèle de type BDI) Caicedo et al. (2001).*

Les AAVs sont traditionnellement contrôlés par l'une des trois approches principales suivantes :

- Les scripts (*story telling*), qui permettent d'obtenir un niveau très détaillé du contrôle. Cette approche a cependant le désavantage d'être très peu flexible.
- Les agents réactifs (*reactive agents*), qui réagissent à un environnement en cours d'évolution selon un ensemble de règles.
- La logique *BDI* (*Beliefs, Desires, Intentions*), semblable aux agents réactifs, dont l'implémentation la plus populaire est probablement le jeu "Les Sims".

Un agent autonome ayant des mécanismes de sélection de l'action qui utilisent une approche à base de scripts, a tendance à paraître trop "robotisé". Plusieurs chercheurs Kaelbing et *al.* (1998), Ramachandran et Bree (2001), Schmitzberger et *al.* (2002) et Arleo et *al.* (2004) ont essayé de résoudre ce problème en utilisant des approches probabilistes, mais sans succès pour les cas complexes.

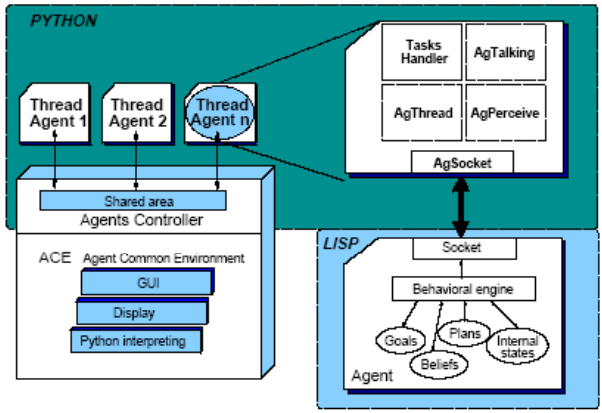


Figure 5-2. *Représentation schématique d'un agent virtuel intelligent, conçu avec un moteur comportemental "simple" Monzani et al. (2001).*

Aujourd'hui, beaucoup de moteurs comportementaux utilisent l'approche à base de scripts, la plupart du temps avec leur logiciel de simulation. Il s'agit de cas orientés très spécifiques. Ils peuvent, cependant, s'avérer très complexes à implémenter dans un cas général, notamment dans la simulation de conduite d'une voiture dans une ville virtuelle. Ils sont plus adéquats, en revanche, pour des modèles comportementaux de marche ou de natation par exemple, ainsi que nous l'avons montré au Chapitre 4 – Apprentissage comportemental.

De plus, il semble difficile d'améliorer, dans un EV, la manifestation de la présence (*Sense of Presence*) Slater and Usoh (1993) d'un AVA avec une approche *Top* → *Down*. Nos AAVs ont une perception virtuelle et un mécanisme d'apprentissage propres, et cette approche, trop rigide, produit un comportement fortement "robotisé" Pettre et *al.* (2003). Nous proposons, en lieu et place, d'utiliser l'information sémantique, tant statique que dynamique, avec notre perception virtuelle, dans une approche de type *Bottom* → *Up*, afin d'établir un comportement autonome, cohérent et robuste de l'AAV.

5.3 Vue d'ensemble

L'évolution a offert au cerveau humain un nombre de caractéristiques qui sont absentes dans l'architecture de Von Neumann, voire dans celle des machines parallèles. Notre cerveau possède, entre autres : une architecture massivement parallèle, un mode de traitement, une mémoire distribuée, une capacité d'apprentissage, des capacités de généralisation et d'adaptation, une résistance aux pannes, et une consommation énergétique faible. Dans ce chapitre, nous allons approximer, à l'aide d'un modèle d'apprentissage cognitif, une partie des propriétés mentionnées, à savoir l'apprentissage par la classification, par la généralisation et par l'adaptation.

5.3.1 Définition d'un modèle d'apprentissage cognitif

Le modèle d'apprentissage cognitif définit ce que l'AAV connaît, comment il peut acquérir cette connaissance et comment elle peut être utilisée pour planifier ses actions. L'approche traditionnelle, pour un modèle d'apprentissage cognitif, est l'approche symbolique. Celle-ci utilise une logique de premier ordre, souvent appelée *situation calculus*, où l'EV est considéré comme une séquence de situations, chacune étant une "photographie" de l'état de cet EV.

La partie la plus importante d'un modèle d'apprentissage cognitif est son planning. Ce dernier a la tâche de formuler une séquence d'actions qui sont espérées pour atteindre un objectif. Ce planning est réalisé au moyen d'un arbre de recherche de toutes les actions possibles au cours du temps (voir Figure 5-3).

L'approche symbolique traditionnelle, pour un modèle d'apprentissage cognitif, possède les points forts suivants : elle est explicite, elle a une sémantique formelle, et elle rend possible une lecture et une exécution maîtrisables. Elle a aussi des fondements mathématiques solides et elle est bien établie dans la théorie de l'apprentissage artificiel. Cependant, elle a quelques faiblesses dans le domaine de l'animation comportementale.

Comme le planning est réalisé à l'aide d'un arbre de recherche (voir Figure 5-3) et que le facteur de complexité est le nombre d'actions à considérer, l'ensemble possible de ces dernières doit être supposé de petite taille afin que les contraintes en temps réel soient satisfaites. Pour maintenir ces contraintes, nous devons nous limiter à des plans courts

sous-optimaux. Un autre problème spécifique à l'animation comportementale d'un AAV est que l'application finale doit activer plusieurs AAVs simultanément pour qu'ils soient capables d'interagir en temps réel. De plus, il peut arriver que, dans une animation comportementale, il n'y ait pas véritablement d'état terminal.

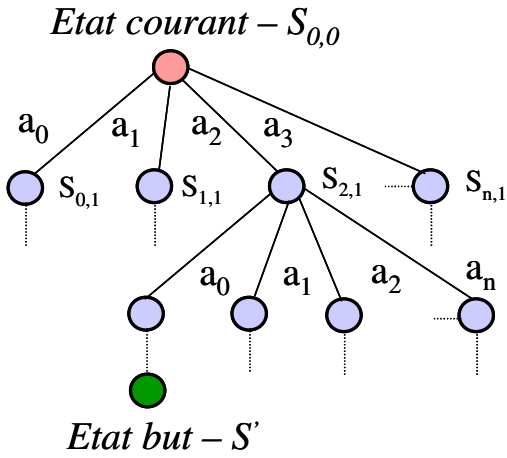


Figure 5-3. *Planning réalisé au moyen d'un arbre de recherche de toutes les actions possibles au cours du temps.*

Dans beaucoup de simulations faites avec un micro-ordinateur actuel, il est pratiquement impossible de terminer le planning de l'animation comportementale par l'approche symbolique traditionnelle. Une autre limitation est qu'il n'est pas possible pour un AAV d'apprendre, par lui-même, un modèle cognitif.

Il y a de nombreuses méthodes d'apprentissage artificiel qui pourraient être utilisées pour approximer un modèle cognitif. Nous avons choisi d'utiliser la technique de classification par les plus proches voisins *k-ppv* (*k-nearest neighbour algorithm*) car, comparativement à une technique qui fait appel aux réseaux de neurones artificiels (RNAs), la *k-ppv* fournit une approximation locale de la fonction cible. De plus, elle peut être utilisée automatiquement sans que le programmeur doive sélectionner avec précaution les informations de l'espace des entrées.

Une description plus détaillée des modèles d'apprentissage cognitif est donnée dans le livre de Funge (1999) et dans les articles de Terzopoulos (1999), puis du même Funge (2000).

5.3.2 Classification des données

Le problème de la classification des données est d'associer à un prototype de l'information de nature très variable, fournie en entrée. Cette information et décrite par un vecteur des caractéristiques, lequel représente une classe parmi les nombreuses classes pré-spécifiées. Citons, parmi les applications de cette approche, la reconnaissance de formes, la reconnaissance vocale et la classification des signaux d'un électrocardiogramme, par Conde et Karrakchou (1994).

Comme nous l'avons décrit au Chapitre 2 – Senseurs virtuels, la méthode de groupage (*clustering*) a été longtemps utilisée pour la construction de caractéristiques (*feature construction*), l'idée étant de remplacer un groupe de variables "similaires" par un groupe centré (*cluster centroïd*), lequel devient alors une caractéristique. Les algorithmes les plus populaires incluent le *k-means* et le *hierarchical clustering* Duda et *al.* (2001), qui permettent la construction, entre autres, de prototypes destinés à la classification de données.

5.3.3 Apprentissage bayésien non paramétrique

Dans l'apprentissage bayésien, on part d'hypothèses, en fonction des données d'apprentissage, à priori pour les réviser. Cette approche est optimale, au sens probabiliste du terme : les hypothèses obtenues à posteriori sont ainsi les plus vraisemblables.

D'un point de vue opérationnel, l'adaptation de l'apprentissage bayésien nécessite, en général, d'une part une connaissance, à priori, de la vraisemblance des hypothèses en concurrence et, d'autre part, celle de la probabilité des données d'apprentissage, conformément à ces hypothèses. Si ces données sont connues, l'hypothèse la plus vraisemblable, compte tenu de ces dernières, pourra être retenue. En pratique, ces données n'étant pas connues exactement, elles doivent être estimées. Dans ce chapitre, nous

utilisons essentiellement les techniques non paramétriques, avec la méthode des plus proches voisins *k-ppv*.

Les techniques non paramétriques reposent sur l'estimation d'une densité de probabilité, pour laquelle aucune régularité fonctionnelle n'est supposée à priori. Ces méthodes sont basées, cependant, sur l'hypothèse fondamentale que les distributions de probabilité, ou fonctions recherchées, sont localement régulières.

Soit une densité de probabilité inconnue $p(x)$, la probabilité Q pour qu'une forme x, issue de cette distribution, soit observée dans la région $R \in X$, est donnée par l'équation 5.1:

$$Q = \int_R p(\boldsymbol{u})\, d(\boldsymbol{u}) \approx p(\boldsymbol{x}).V \qquad (5.1)$$

où V est le volume de la région R.

5.3.4 Apprentissage d'une règle de classification

Dans le cas où l'on cherche à apprendre une règle de classification, la méthode bayésienne consiste à estimer, en un point x donné, la densité de probabilité de chaque classe, afin de choisir celle qui possède la valeur la plus grande.

On suppose donc que l'on connaît m points de R^d (d: dimension, ou nombre de variables d'entrée) obtenus par tirages indépendants, selon la densité qui caractérise la classe ω. Comment peut on estimer $p(x|\omega)$, au point x, à partir d'un ensemble d'apprentissage ?

Le principe, comme décrit auparavant, consiste à définir, autour de x, une certaine région R_m (en pratique, une hypersphère ou un hypercube) et à compter le nombre k_m de points de l'échantillon d'apprentissage inclus dans ce volume (voir Figure 5-4).

L'un des problèmes, avec les méthodes par fonction noyau (*kernel*), est que leur taille est fixe. Si elle est trop grande, l'approximation est trop "lisse", par rapport à la réalité. Si elle est trop petite, l'estimation, dans les régions de faible densité, peut être nulle ou très approximative. Il faudrait donc que cette taille soit établie en fonction de la position dans l'espace X . C'est ce que réalise la méthode des plus proches voisins *k-ppv*.

Dans cette dernière, le nombre k de points, dans la région autour de x, est fixe et on fait, en revanche, varier le volume V. On considère donc une hypersphère (on utilise, en général, une distance euclidienne) centrée sur x et on fait varier le noyau jusqu'à ce que cette sphère contienne k points. L'estimation de la densité est alors donnée par le rapport k / mV, où m est le nombre total de points dans l'échantillon de données. Cela revient à choisir une fonction noyau simple, constante sur une hypersphère, et qui contient les k points. On peut donc passer directement à une règle de décision en classant une forme inconnue x *et* en prenant la classe qui est majoritaire dans les k points d'apprentissage les plus proches. Cette règle est appelée règle des k-plus proches voisins où k est le nombre de voisins considérés.

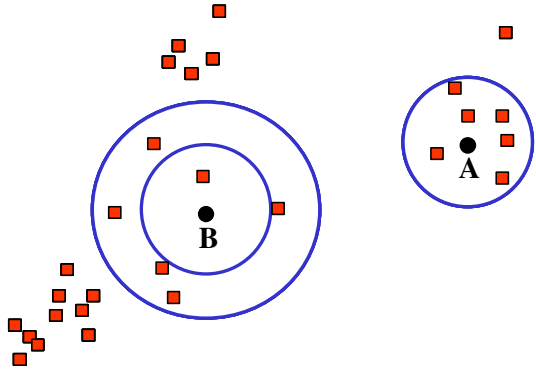


Figure 5-4. *Tirages indépendants, représentés par des carrés rouges, selon une certaine distribution dans le plan R^2, dont la densité est plus forte autour du point A qu'autour du point B. En effet, pour le même volume autour des points A et B, k_m vaut, respectivement, 6 et 1. Pour avoir k_m = 6 autour du point B, il faut augmenter le volume.*

Dans le cas où $k = 1$, la règle s'appelle la règle de classification du plus proche voisin. Elle assigne simplement à x la même étiquette que le point d'apprentissage le plus proche. Il est remarquable que cette règle (voir relation 5.2), soit extrêmement simple et possède un comportement asymptotique, et également excellent vis-à-vis du risque minimal de Bayes.

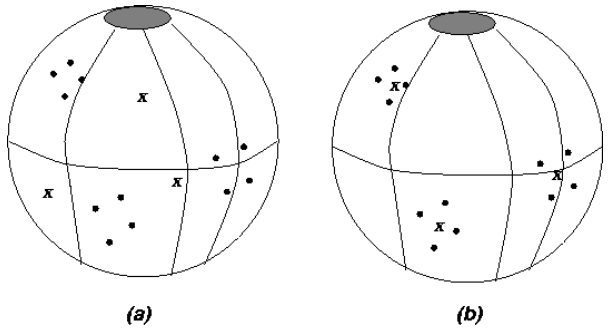
$$R_B \quad R_{k\text{-}ppv} \quad R_{(k\text{-}1)\text{-}ppv} \ldots \quad R_{1\text{-}ppv} \quad 2R_B \tag{5.2}$$

$R_{k\text{-}ppv}$ = probabilité d'erreur de la règle des *k-ppv*. Elle converge vers le risque bayésien R_B lorsque *m*, le nombre total d'échantillons, tend vers l'infini pour tout *k*.

On résume souvent la relation 5.2 par: "la moitié de l'information sur la classification optimale d'un point inconnu est disponible dans son seul plus proche voisin".

5.3.5 Apprentissage compétitif

L'exemple le plus connu d'apprentissage par compétition est la technique *LVQ* (*Learning Vector Quantization*) pour la compression de données, très largement utilisée dans le cadre du traitement de la parole, du stockage d'images et de la transmission de données. Il s'agit de représenter un ensemble, ou une distribution, de vecteurs à l'aide d'un nombre restreint de vecteurs prototypes, ou d'un livre de codes. Une fois ce livre de codes construit et agréé par l'émetteur et le récepteur, il ne reste qu'à transmettre, ou stocker, l'index du vecteur prototype correspondant au vecteur de données. Etant donné ce vecteur de données, son vecteur prototype peut être trouvé en cherchant le vecteur prototype le plus voisin dans le livre de codes (voir Figures 5-5a) et b)). Dans nos méthodes pour l'apprentissage d'un modèle cognitif, nous allons surtout utiliser deux types d'apprentissage compétitif, à savoir : *LVQ* et *DSM* (*Decision Surface Mapping*) Geva et Sitte (1991).



Figures 5-5a) et b). *Interprétation géométrique d'un apprentissage compétitif. Nous supposons que les vecteurs à apprendre sont normalisés. Ils sont représentés par des points sur les sphères. On admet que le réseau possède trois cellules, dont les poids sont initialisés aléatoirement. Leurs positions initiales et finales sont représentées, respectivement en (a) et (b), par des* ***x****. Les trois catégories naturelles des prototypes ont été établies, et les poids associés aux cellules correspondent aux barycentres des catégories.*

Nous avons explicité ci-dessous, à l'aide des relations 5.3 et 5.4, le mécanisme d'apprentissage compétitif :

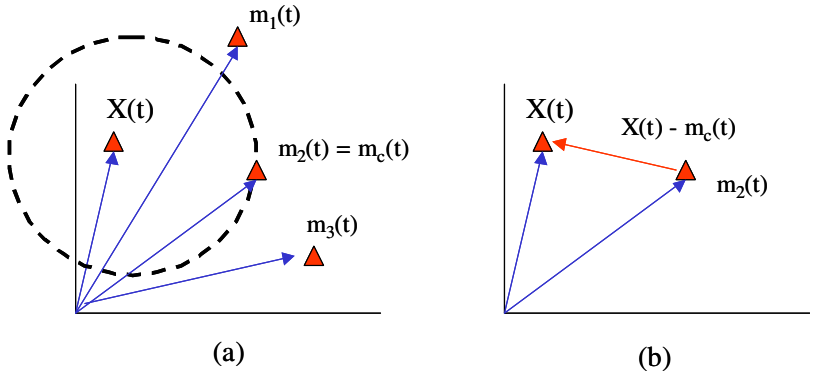
$$m_c(t+1) \leftarrow m_c(t) + \alpha(t)\,[\,X(t) - m_c(t)] \qquad (5.3)$$

$m_c(t)$ est récompensé, car la classe $m_c(t)$ coïncide avec $X(t)$

$$m_c(t+1) \leftarrow m_c(t) - \alpha(t)\,[\,X(t) - m_c(t)] \qquad (5.4)$$

$m_c(t)$ est pénalisé car la classe s'éloigne de $X(t)$

où $0 < \alpha(t) < 1$



Figures 5-6a) et b). *Illustration de la relation 5.4 pour la quantification vectorielle effectuée par la méthode LVQ.*

5.4 Apprentissage d'un modèle cognitif

5.4.1 Introduction

Des recherches étendues sur les moteurs comportementaux, pour contrôler des animaux ou des humains virtuels autonomes animés, ont été effectuées par Reynolds (1987), Tu et Terzopoulos (1994) et Blumberg et Galyean (1996). Toutes ces techniques

ont produit des résultats impressionnants mais sont néanmoins limitées de deux façons. Premièrement, elles n'ont aucune capacité d'apprentissage propre et sont donc confinées à des comportements pré-spécifiés explicites. Deuxièmement, elles n'exécutent qu'un contrôle comportemental et non cognitif de l'AAV. Le modèle comportemental, comme nous l'avons montré au chapitre précédent, signifie surtout une prise de décision réactive, tandis que le modèle cognitif implique un raisonnement et une planification à long terme de la tâche à accomplir.

L'apprentissage comportemental en ligne a été exploré pour l'animation comportementale par Burke et *al.* (2001) et Tomlinson et Blumberg (2003). Un exemple significatif, décrit par Blumberg et *al.* (2002), implique un chien virtuel auquel un usager enseigne, de façon interactive, à montrer un comportement désiré. Cette dernière approche est basée, en partie, sur l'apprentissage par renforcement, lequel a montré, au chapitre précédent, qu'il peut fonctionner très correctement. Il manque cependant un support pour un raisonnement à long terme, afin de pouvoir accomplir des tâches complexes. En outre, puisque toutes ces techniques sont conçues pour être employées en ligne, leurs capacités d'apprentissage sont limitées en raison de la nécessité d'une grande vitesse interactive.

L'apprentissage cognitif pour l'animation comportementale a été initialement proposé par Funge et *al.* (1999) pour doter les humains virtuels d'un raisonnement à long terme. Celui-ci fournit à un humain virtuel assez d'intelligence pour accomplir automatiquement et d'une façon crédible des tâches complexes.

5.4.2 Notre nouvelle méthode

La méthodologie que nous avons conçue pour l'approximation d'un modèle cognitif est basée sur le cadre interactif à quatre étapes, décrit par Duric et *al.* (2002), en sciences cognitives, et que nous avons déjà utilisé, en partie, dans le Chapitre 3 – Perception virtuelle. Ce cadre passe de l'analyse des entrées sensorielles moteur à l'interprétation des mouvements, puis aux émotions de l'utilisateur et, enfin, à la construction d'une compréhension de l'état cognitif. En raison de l'intégration du comportement perceptuel et de la modélisation cognitive, le composant interactif du système améliore le traitement de chaque étape. Nous avons ajouté de l'information sémantique afin d'avoir des entrées significatives au niveau comportemental du moteur d'I.A.

En utilisant notre Etude de cas – un simulateur de conduite automobile dans une ville virtuelle – nous avons vérifié que notre plate-forme *ALifeE* intégrait les modalités sensorielles et perceptuelles d'une manière logique. Nous avons appliqué également une approche non paramétrique d'apprentissage, la méthode des plus proches voisins *k-ppv*, comme nous l'avons décrite auparavant.

Notre méthodologie est établie sur la modélisation cognitive traditionnelle, avec l'objectif de réduire au maximum deux faiblesses importantes : a) les faibles performances des modèles cognitifs, et b) le temps relativement long de leur construction. Un modèle cognitif définit ce qu'un AAV sait, comment cette connaissance est acquise, et comment elle peut être employée pour planifier ses actions. Nous avons décrit auparavant la façon d'apprendre, avec un tel modèle cognitif.

La plupart des algorithmes d'apprentissage artificiel font des suppositions générales, mais faibles, au sujet de la nature des données d'apprentissage. En conséquence, ils exigent typiquement de grandes quantités de données pour apprendre des classificateurs précis. Normalement, la performance s'améliore pendant que l'algorithme exploite plus d'informations, algorithme généralement meilleur à l'identification qu'à la généralisation. Ce problème peut être résolu en tirant profit de la connaissance antérieure, afin d'éliminer les classificateurs contradictoires. Par conséquent, les algorithmes d'apprentissage résultants pourront apprendre avec très peu d'exemples nécessaires à cet apprentissage.

La technique *k-ppv* fait implicitement une évaluation comparative de toutes les densités de probabilité de classe, proches du point à reconnaître. Elle choisit simplement la plus probable et fait une approximation de la décision bayésienne, ainsi que nous l'avons décrit auparavant. Puis, un vecteur de quantification est déterminé. La technique se charge alors de remplacer la combinaison finie des points par un nombre limité de prototypes, ainsi que nous l'avons exposé auparavant.

Il y a un risque, cependant, d'incorporer la connaissance antérieure, puisque cela peut ajouter un biais à l'apprentissage. Si la connaissance est incorrecte, elle éliminera alors tous les classificateurs précis. En conséquence, les algorithmes d'apprentissage tendent à performer assez bien sur de petits ensembles d'entraînement mais, à mesure que la quantité de données augmente, comme dans la simulation de conduite d'une voiture dans une ville virtuelle, leur performance diminue parce qu'ils s'adaptent en dessous des valeurs données.

Dans la plupart des problèmes réels, toutes ces techniques sont limitées par l'espace très grand des états possibles. Ces algorithmes ont généralement besoin d'un temps de calcul mesuré à la puissance trois du nombre d'états. Par conséquent, Mitchell (1997) et d'autres chercheurs se sont concentrés sur la construction des approximations gérables, du point de vue informatique, à la fois pour la politique, la fonction de valeur et le modèle cognitif.

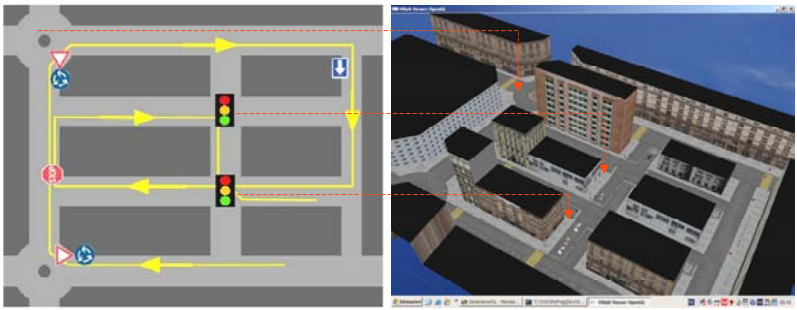
Le choix de l'algorithme *k-ppv* fournit une approximation locale de la fonction cible et peut être employé automatiquement, sans que le programmeur doive sélectionner les entrées. Elle garantit l'apprentissage de la fonction cible basé sur la qualité des exemples fournis et la mémorisation des décisions prises par la planification, à l'aide d'un modèle cognitif. La prise de décision par le modèle cognitif est un élément très important de l'information, l'association (*mapping*) sera probablement plus lisse (*smoother*) si cette information se présente en entrée séparée de l'algorithme *k-ppv*.

D'une façon générale, avec l'approche *k-ppv*, la diminution du nombre de points réduit l'espace de recherche et le problème de stockage. Ceci mène également à une réduction du temps de calcul. Nous avons donc, chaque fois, utilisé une phase pré-calculée, avant la phase de recherche, afin de réorganiser l'espace d'apprentissage.

La méthode *k-ppv* n'exige pas une séparation des diverses classes d'apprentissage. Par conséquent, nous avons choisi un sous-domaine de points d'apprentissage. Cependant, cette méthode rend nécessaire un stockage explicite de nombreux exemples de la fonction cible. Cette méthode permet de découvrir automatiquement les entrées nécessaires pour se rapprocher de la fonction cible, comme dans notre modèle cognitif pour la simulation de conduite d'une voiture. En outre, le choix des métriques pour la *k-ppv* influence le taux d'erreur et de rejet.

5.4.3 Intégration et implémentation

Nous avons implémenté notre approche pour améliorer la sensation de présence (*Sense of Presence*) dans trois Etudes de cas appliquées à la simulation de conduite d'une voiture dans une ville virtuelle (voir Figures 5-7a) et 5-7b)). La première utilise un modèle cognitif explicite, la seconde un modèle cognitif approximatif, sans *ALifeE*, et la troisième un modèle cognitif approximatif, avec *ALifeE*.



Figures 5-7a) et 5-7b). *Simulation de conduite d'une voiture dans une ville virtuelle. L'information sémantique, telle que les signaux routiers et les feux de circulation, est incluse.*

1ère Etude de cas - Modèle cognitif explicite

Dans cette première Etude, nous avons utilisé un modèle cognitif explicite, donc programmé. L'AAV est un pilote conduisant une voiture à l'intérieur de la ville virtuelle. Le pilote et le co-pilote ont une double commande par l'intermédiaire de la pédale de gaz et du volant (voir Figure 5-8). Ces commandes correspondent à la réalité, l'espace des actions étant continu. La voiture peut se déplacer à n'importe quel endroit, ou prendre n'importe quelle direction.

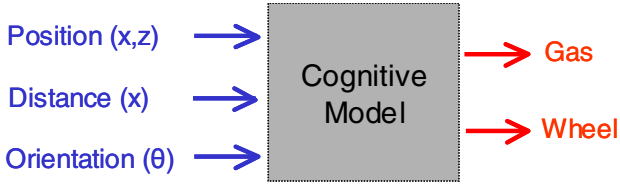


Figure 5-8. *Modèle cognitif explicite, avec ses entrées et ses sorties.*

2ème Etude de cas – Modèle cognitif approximatif avec pseudo-perception, sans ALifeE

La deuxième Etude est réalisée avec un modèle cognitif approximatif, sans utiliser l'*ALifeE*, mais avec des caractéristiques de pseudo-perception. Ces dernières sont appliquées afin de comparer les performances obtenues avec l'Etude de cas qui intègre notre plate-forme *ALifeE*. En effet, dans la plupart des environnements de simulation

d'AAVs, les modalités sensorielles et la perception ne sont pas intégrées d'une manière correspondant à la réalité. Dans cette étude, la pseudo-perception visuelle est fournie par le champ de vision de l'AAV, qu'il soit pilote ou co-pilote, sous forme d'une zone circulaire (voir Figure 5-9). Les objets dynamiques (comme les voitures) et statiques (comme les signaux routiers et les feux de circulation) sont représentés par des symboles graphiques rectangulaires (voir à l'intérieur du cercle en pointillés sur la Figure 5-9).

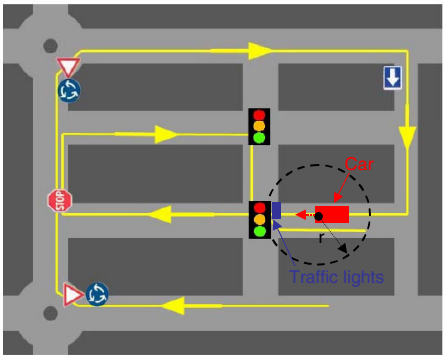
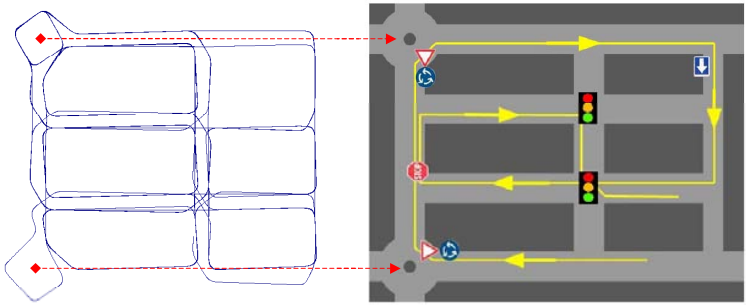


Figure 5-9. *Pseudo-perception visuelle: la voiture "voit" les feux de circulation à l'intérieur d'une zone circulaire.*

Pour tester l'identification des signaux routiers et des feux de circulation, nous avons intégré cette méthode de pseudo-perception visuelle de façon qu'elle puisse déterminer l'objet le plus proche d'un rayon donné "r" de la zone circulaire (voir Figure 5-9).

L'algorithme *k-ppv* a permis d'approximer une politique simple. Il n'est utile que pour un modèle cognitif et un but à la fois. Les Figures 5-10a) et 5-10b)) illustrent la formation du modèle cognitif approximatif de conduite et du chemin de la voiture à l'intérieur de la ville virtuelle. Après cela, l'information permet de simuler le comportement cognitif de l'AAV co-pilote, qui devient progressivement l'instructeur de l'AAV pilote.

Il peut exister plus d'un modèle cognitif pour un but donné, de sorte qu'une plus grande variété, et/ou robustesse, puissent être atteintes. Il est également possible d'employer l'algorithme *k-ppv* avec plusieurs modèles cognitifs explicites, placés dans le même "cerveau" de l'AAV.



***Figures 5-10a) et 5-10b)**. Instantanés du chemin de la voiture de notre modèle cognitif approximatif avec pseudo-perception.*

3ème Etude de cas – Modèle cognitif approximatif avec notre ALifeE

Ainsi que nous l'avons mentionné dans l'introduction à notre nouvelle méthode (voir 5.4.2), nous avons besoin d'un environnement animé, dans lequel les AAVs manifestent un comportement autonome. En raison de la complexité du moteur comportemental à contrôler, nous proposons une méthode plus simple, qui utilise l'information sémantique avec une perception virtuelle. Cette approche s'intègre dans notre plate-forme *ALifeE*.

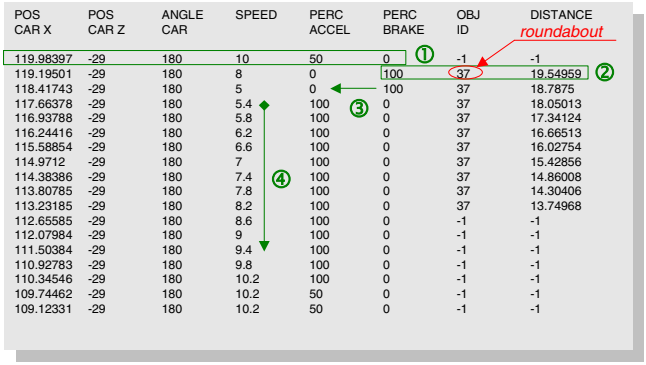
Nous avons, au Chapitre 6 – Etudes de cas, décrit plus en détail l'intégration et l'implémentation de notre simulateur de conduite automobile dans une ville virtuelle, et plus particulièrement la partie qui comprend les objets statiques et dynamiques, ainsi que l'information sémantique.

5.4.4 Résultats expérimentaux

Dans notre 1ère Etude de cas, le résultat final était un comportement de conduite pauvre, puisque le pilote ne pouvait pas planifier assez en avance les manoeuvres adéquates de conduite de la voiture dans la ville virtuelle. En outre, puisque la quantification de l'espace des actions est trop grossière, la conduite n'était pas très "lisse" (*smooth*).

Dans la 2ème Etude de cas, nous avons amélioré la planification de notre modèle cognitif approximatif, en tirant profit des caractéristiques de la pseudo-perception. Nous avons donc obtenu de meilleurs résultats, et une quantification plus fine de l'espace des actions.

Dans la 3ème Etude de cas, nous avons détaillé, ci-après, le processus complet du comportement du simulateur.



POS CAR X	POS CAR Z	ANGLE CAR	SPEED	PERC ACCEL	PERC BRAKE	OBJ ID	DISTANCE
119.98397	-29	180	10	50	0	-1	-1
119.19501	-29	180	8	0	100	37	19.54959
118.41743	-29	180	5	0	100	37	18.7875
117.66378	-29	180	5.4	100	0	37	18.05013
116.93788	-29	180	5.8	100	0	37	17.34124
116.24416	-29	180	6.2	100	0	37	16.66513
115.58854	-29	180	6.6	100	0	37	16.02754
114.9712	-29	180	7	100	0	37	15.42856
114.38386	-29	180	7.4	100	0	37	14.86008
113.80785	-29	180	7.8	100	0	37	14.30406
113.23185	-29	180	8.2	100	0	37	13.74968
112.65585	-29	180	8.6	100	0	-1	-1
112.07984	-29	180	9	100	0	-1	-1
111.50384	-29	180	9.4	100	0	-1	-1
110.92783	-29	180	9.8	100	0	-1	-1
110.34546	-29	180	10.2	100	0	-1	-1
109.74462	-29	180	10.2	50	0	-1	-1
109.12331	-29	180	10.2	50	0	-1	-1

Figure 5-11. *Modèle cognitif approximatif avec ALifeE et un comportement de conduite "correct".*

Les comportements "correct" et "faux" de conduite d'une voiture virtuelle, s'approchant d'un giratoire, sont représentés, respectivement sur les Figures 5-11 et 5-12.

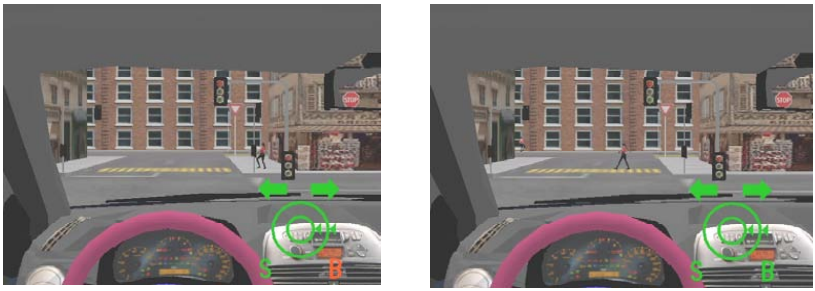
Quand un signal routier de giratoire est vu, le comportement "correct" est de ralentir, si la vitesse est trop élevée (voir Figure 5-11). A la position {119,98397, -29}, la voiture virtuelle se déplace le long de l'axe positif de la direction *x* (ANGLE CAR = 180), à une vitesse de 10 [m/s]. Lorsque la vitesse est constante, la pédale de gaz (PERC ACCEL) et la pédale de freins (PERC BRAKE) ont les valeurs, respectivement, de 50 et 0 % (étape 1). Un instant plus tard, la voiture "voit" un signal routier de giratoire, identifié par le numéro 37 (OBJ ID). La distance à ce panneau est 19,54959 [m] (étape 2). Un script en langage Python est utilisé pour ralentir la voiture virtuelle pendant qu'elle entre dans le giratoire. En effet, la pédale de freins est maintenant 100 % active, alors que l'accélération a diminué à 0 % (étape 3). La vitesse diminue jusqu'au seuil déterminé par le script en Python, puis la voiture accélère à nouveau (PERC ACCEL = 100 %) pour atteindre une vitesse constante, approximativement 10 [m/s] (étape 4).

Dans le cas d'un comportement de conduite "faux", la vitesse de 10 [m/s] ne diminue pas avant l'entrée dans le giratoire (étape 4 sur le Figure 5-12). Une pénalité doit être appliquée, afin que l'apprentissage puisse se faire avec ces données.

POS CAR X	POS CAR Z	ANGLE CAR	SPEED	PERC ACCEL	PERC BRAKE	OBJ ID	DISTANCE
119.98397	-29	180	10	50	0	-1	-1
119.19172	-29	180	10	50	0	37	19.54636
118.39947	-29	180	10	50	0	37	18.76992
117.60722	-29	180	10	50	0	37	17.99484
116.81496	-29	180	10	50	0	37	17.22134
116.02271	-29	180	10	50	0	37	16.44961
115.23046	-29	180	10	50	0	37	15.67994
114.43821	-29	180	10	50	0	37	14.91263
113.64596	-29	180	10	50	0	37	14.14807
112.85371	-29	180	10	50	0	37	13.38673
112.06145	-29	180	10	50	0	37	12.62919
111.2692	-29	180	10	50	0	-1	-1
110.47695	-29	180	10	50	0	-1	-1
109.6847	-29	180	10	50	0	-1	-1
108.89245	-29	180	10	50	0	-1	-1
108.1002	-29	180	10	50	0	-1	-1

Figure 5-12. *Modèle cognitif approximatif, avec ALifeE et un comportement de conduite "faux".*

Les Figures 5-13a) et 5-13b) montrent qu'un conducteur identifie un piéton qui souhaite traverser la rue, et qu'il doit donc freiner. En bas à droite de la Figure 5-13a) la lettre B de "brake" s'affiche en rouge.

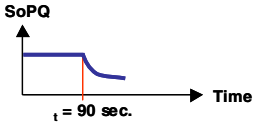
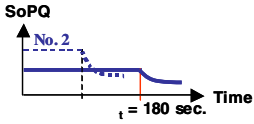


Figures 5-13a) et 5-13b). *Un piéton souhaite traverser la rue selon les règles définies. Le panneau informe le conducteur qu'il doit freiner. La lettre B s'affiche en rouge, dans le coin, en bas à droite de la Figure 5-13a). Le pilote et le co-pilote sont indiqués au moyen de cercles concentriques verts.*

Nous avons obtenu de bons résultats (voir Tableau 5-14), avec notre méthode d'approximation d'un modèle cognitif, en employant notre plate-forme *ALifeE*. La conduite de la voiture à l'intérieur de la ville virtuelle est plus rapide et "lisse" qu'avec le modèle cognitif explicite de la 1ère Etude de cas – ou avec les caractéristiques de pseudo-perception

dans la 2ème Etude de cas. De plus, la sensation de présence (*Sense of Presence*) s'est améliorée Slater and Usoh (1993).

Tableau 5-14. *Résultats obtenus pour la qualité de la sensation de présence avec notre méthode AVAhighLEARN. Toutes les animations ont été générées en temps réel avec OpenGL® (3.0 GHz avec une carte vidéo nVIDIA GeForce FX Go5350).*

Case Study	Time	Average Speed [m/sec.]	Nb. of collisions	Accelerations braking	Sense of Presence Quality [SoPQ]
No. 1	++	20	-	+	without **SoPQ**
No. 2	-	10	↗	++	
No. 3	+	15	→	-	

Explications pour le Tableau 5-14 :

- Les courbes pour la qualité de la sensation de présence - *Sense of Presence Quality* (*SoPQ*) pour les études de cas No. 2 et No. 3 n'atteignent pas l'axe du temps. En effet, l'immersion décroît graduellement, mais jamais complètement. Le conducteur de la voiture à la fin de la journée peut-être fatigué et perdre sa vigilance, mais seulement après quelques heures.
- *L'étude de cas No. 1* – Le conducteur conduit la voiture rapidement dans la ville virtuelle. Il n'y a pas d'évaluation de la *SoPQ* car le conducteur ne ralentit pas aux intersections et traverse les ronds points. De plus, la ville virtuelle semble déserte et pas vraiment réaliste.
- *L'étude de cas No. 2* – Le conducteur conduit la voiture moins rapidement dans la ville virtuelle, parce qu'il doit tenir compte des obstacles et des piétons qui ne

sont pas vraiment "intelligents". Le conducteur est très attentif et la *SoPQ* semble meilleure. Néanmoins, celle-ci décroît quand $_t$= *90* secondes. Ensuite le conducteur commence à ressentir de la fatigue.

- *L'étude de cas No. 3* – Le conducteur conduit la voiture moins ou plus rapidement dans la ville virtuelle, parce qu'il doit tenir compte des obstacles, des autres voitures et des piétons qui sont autonomes, ainsi que de la sémantique des signaux de la route. La *SoPQ* décroît quand $_t$= *180* secondes. Ensuite le conducteur commence à ressentir de la fatigue.

5.4.5 Discussion

Un AAV possède un ensemble de compétences (*skills*) élémentaires et des senseurs. Nous avons montré, dans notre simulation de conduite d'une voiture, que le but de l'adaptation de l'AAV est d'acquérir de nouvelles compétences. L'AAV apprend l'organisation d'un système perceptuel, et développe l'association: états → actions (*mapping*). L'organisation corrige la perception apprise et les actions acquises.

Beaucoup d'observations suggèrent que l'apprentissage perceptuel commence chez les enfants et se poursuit à l'âge adulte. Il est très souvent placé dans un contexte qui ne tient pas compte de la rétroaction qui influence le mécanisme de contrôle. Les compétences sont en relation avec la perception, par exemple celle d'un conducteur de voiture qui reçoit, toujours plus, la rétroaction de son EV. Les piétons, à l'intérieur de notre simulateur, sont très importants puisqu'ils améliorent, parce qu'ils la perturbent, la vigilance du conducteur. Et ils augmentent également le réalisme de notre ville virtuelle Conde et *al.* (2005).

Une méthode approximative est suffisante, à notre avis, pour réaliser une amélioration de la sensation de présence d'un AAV et pour obtenir une animation crédible, laquelle n'exige pas d'exactitude. Et qui ne soit pas particulièrement "robotisée". Nous avons pu, en outre, observer que de meilleurs résultats sont obtenus avec la méthode *k-ppv*, plutôt qu'avec un modèle cognitif explicite, donc programmé. Dans notre approche, l'AAV a appris, à l'aide d'une *k-ppv*, en utilisant comme exemples d'apprentissage des plans

optimaux élevés, afin d'obtenir de meilleurs résultats en temps réel. Pour construire une représentation continue de l'espace, un ensemble suffisant d'exemples d'apprentissage est donc nécessaire, tout comme dans le contexte de conduite d'une voiture.

Notre méthodologie est basée sur les théories d'apprentissage artificiel, et sur l'état actuel de la recherche dans ce domaine. En raison de l'utilisation de la méthode *k-ppv*, l'exactitude des résultats ne peut être absolument garantie. Une *k-ppv*, apprise en l'appliquant à un exemple avec assez de variantes, serait plus performante. On doit tenir compte de la connaissance apprise, qui est en général perturbée par du "bruit", donc incorrecte. Les algorithmes d'apprentissage artificiel fonctionnent en partant d'un ensemble de données, lesquelles ne constituent qu'un échantillon statistique des cas possibles.

Pour modéliser l'aspect évolutif de l'apprentissage cognitif d'un AAV, nous proposons de créer un EV incertain, régi par des règles floues, et de donner au concepteur la possibilité de prendre le contrôle de chaque entité. Pour simuler l'apprentissage de la conduite d'une voiture, nous avons utilisé un co-pilote qui validera cette approche par la suite.

5.5 Apprentissage automatique de modèles cognitifs

5.5.1 Introduction

Il y a plusieurs techniques d'apprentissage artificiel possibles pour automatiser l'apprentissage d'un modèle cognitif. Nous avons estimé important de tester celles qui nous semblaient potentiellement intéressantes pour l'animation comportementale d'un AAV, ainsi que de présenter les avantages et les inconvénients de chacune d'entre elles, tout en les comparant à d'autres techniques dans ce même domaine.

Réseaux de neurones artificiels (RNAs)

Les RNAs (*multi-layer perceptron)* sont une technique de régression globale car l'ensemble du réseau contribue au calcul d'une réponse. Cette technique fonctionne bien parce qu'elle est compacte et globale, mais tout cela seulement si l'apprentissage de l'association: états → actions est continu. Par conséquent, il est souvent nécessaire de

prendre plusieurs jours pour construire une représentation efficace et compacte de l'espace des états, afin de pouvoir l'utiliser en entrée du RNA. Pour en sortir, on doit souvent disposer de plusieurs RNAs pour un simple modèle cognitif, chacun couvrant un sous-ensemble distinct de l'espace des états. Un RNA demande peu d'exemples, de 5.000 à 15.000 habituellement pour pouvoir bien généraliser ce qu'il apprend Dinerstein et *al.* (2004). De plus, en raison de leur puissance de généralisation, les RNAs tendent à ne pas trop mélanger (*blind out*) les exemples de prise de décision avec le bruit, les erreurs et l'effet d'escalier (*aliasing*).

Séparateurs à vastes marges(SVMs)

Les *SVM (Support Vector Machine)* Vapnik (1995) et Joachmins (1999), constituent une autre technique de régression globale, en relation avec les RNAs. La seule différence, par rapport aux besoins en animation comportementale, entre les deux méthodes est que l'apprentissage des SVMs garantit d'atteindre un minimum global de l'erreur quadratique, alors que l'apprentissage (*backpropagation*) des RNAs converge vers un minimum local.

Les SVMs ont montré qu'ils ont les mêmes avantages et inconvénients que l'approche par les RNAs. En particulier, il a été prouvé, par ces deux techniques, qu'il est nécessaire d'utiliser une formulation efficace et compacte de l'espace des états. Les SVMs nécessitent deux à trois fois plus d'exemples d'apprentissage que les RNAs. Le seul avantage significatif que nous ayons trouvé en utilisant les SVMs, en lieu et place des RNAs, est que l'erreur d'approximation est en général plus petite.

Méthode des plus proches voisins (k-pvv)

La méthode des plus proches voisins *k-ppv* est probablement la méthode d'apprentissage artificiel local la plus connue. C'est un exemple de raisonnement par cas (*case-based reasoning*) que nous avons déjà pu utiliser, dans notre méthode d'apprentissage, pour l'approximation d'un modèle cognitif. Contrairement aux RNAs et aux SVMs, qui sont compacts, la *k-ppv* donne tout un ensemble d'exemples d'associations cibles (*target mapping*) qui lui ont été fournis.

La *k-ppv* a montré, dans nos expérimentations, qu'elle est simple à utiliser et qu'elle fonctionne remarquablement bien pour l'approximation des modèles cognitifs Conde et Thalmann (2005a). Ceci est dû, d'abord, au fait que le programmeur ne doit pas concevoir soigneusement la représentation de l'espace des états pour obtenir une association "lisse" et simple, états → actions. Le résultat c'est que les techniques de régression compactes, comme les RNAs et les SVMs, ne fonctionnent pas très bien pour apprendre une telle association. La *k-ppv*, par contre, n'a pas ce genre de difficulté. Bien que la *k-ppv* utilise des exemples explicites, une représentation sous-optimale de l'espace des états peut souvent nécessiter une grande dimension, laquelle demande une croissance exponentielle du nombre d'exemples, afin de peupler chaque dimension supplémentaire.

Nous avons trouvé qu'avec une valeur de $k = 3$, cela fonctionnait bien, la régression se maintenant localement, mais suffisamment en général pour fournir une animation "lisse" (*smooth*). Nous avons utilisé un *kd-tree* pour effectuer la recherche de cas (*lookup of cases*) rapidement. Et nous avons également échelonné les axes de l'espace d'entrée, afin de minimiser l'erreur quadratique moyenne (*mean-squared-error*).

Nos simulations en temps réel, avec l'approche *k-ppv*, ont montré comment obtenir une régression adéquate d'associations difficiles. Ce type de régression peut, en effet, aboutir à une animation saccadée (*jittery*), en raison d'une généralisation incorrecte. Cependant, il faut tenir compte du fait que la *k-ppv* ne généralise pas aussi bien que les RNAs car elle est plus sujette aux saccades, par suite de bruit ou d'erreurs, dans les exemples de comportement de l'AAV.

Autres techniques d'apprentissage artificiel

Nous avons essayé d'autres techniques d'apprentissage artificiel, mais les résultats obtenus ne sont pas suffisamment intéressants pour que nous en tenions compte. Elles ont produit, à la fois, des résultats mauvais ou pas assez performants. Ci-dessous, nous avons résumé les quelques enseignements tirés.

Ayant obtenu de bons résultats avec la technique des *k-ppv*, nous avons aussi essayé une recherche de table (*lookup table*) par résolution adaptative (*adaptive resolution*). Les résultats ne sont pas meilleurs qu'avec l'approche classique qui utilise la *k-ppv*, mais le code est clairement plus complexe. Nous avons aussi essayé de remplacer la simple

pondération métrique, dans la *k-ppv*, par une fonction radiale de base (*radial basis function*), mais la précision n'a pas été sensiblement supérieure.

Nous avons trouvé que les techniques de régression locale sont souvent plus utiles, pour notre problème d'animation comportementale, que les techniques de régression globale car le comportement d'un AAV n'est souvent pas simple, et l'association pas "lisse". Ceci est particulièrement vrai lorsque des entrées sous-optimales ont été sélectionnées.

En nous basant sur les constatations précédentes, nous avons opté pour une approche pondérée, celle-ci combinant des régressions locales et une régression globale, pour réaliser automatiquement un modèle d'apprentissage cognitif, avec son aspect évolutif, que nous décrivons ci-après.

5.5.2 Notre nouvelle méthode

Notre méthode d'apprentissage pour l'approximation d'un modèle cognitif, présentée ci-dessus, est actuellement tout à fait adaptable (*scalable*). Nous proposons, dans cette continuité, qu'au lieu d'utiliser UN moteur d'apprentissage (*machine learner*), nous en utilisons PLUSIEURS, séparés, afin d'automatiser l'apprentissage d'un modèle cognitif, chacun de ces moteurs apprenant un sous-ensemble distinct de l'association: états → actions. Par exemple, la prise de décision, dans différentes régions de l'espace d'états, peut être relayée par une information d'état différente. Et donc ces moteurs peuvent utiliser différentes formulations des états, ce qui réduit ainsi le nombre de dimensions. De même, si un AAV a plusieurs objectifs potentiels distincts d'apprentissage, ceux-ci peuvent être appris séparément. Pour permettre une communication (*switching*) continue entre les moteurs pendant l'animation, les actions proposées par chacun d'eux sont combinées pendant un certain temps (voir Figure 5-14).

La technique *k-ppv* est simple et séduisante pour résoudre des problèmes de classification, ou de "régression", comme nous l'avons montré. Cette technique non paramétrique traite de l'estimation d'une densité de probabilité, pour laquelle aucune régularité fonctionnelle n'est supposée à priori. Cette même technique repose cependant sur l'hypothèse fondamentale que les distributions, ou fonctions recherchées, sont localement régulières.

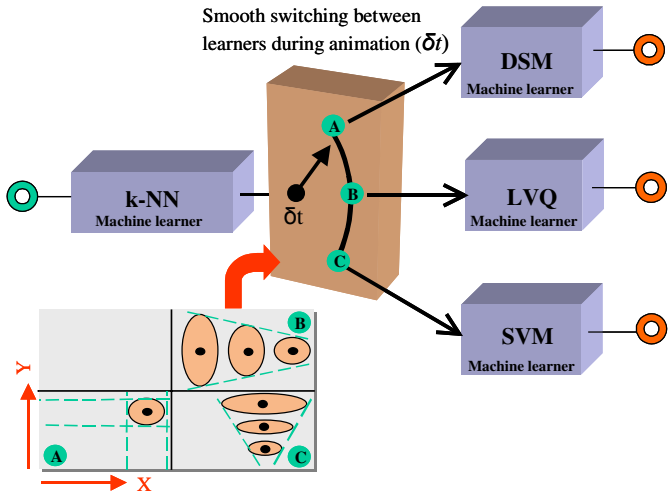


Figure 5-14. *Modèle cognitif avec une combinaison continue de moteurs d'apprentissage. Pour la zone A, la dimension X est plus appropriée parce qu'un léger mouvement le long de cet axe X peut changer l'étiquette de classe. Pour la zone B, c'est la dimension Y qui est plus appropriée. Pour la zone C, les deux dimensions sont également applicables.*

Une telle hypothèse devient non valable pour de grandes dimensions et un nombre infini d'exemples, à cause de la "malédiction de la dimension", ainsi que nous l'avons vu au début du Chapitre 2 – Senseurs virtuels. Un grand biais peut être induit dans ces conditions lorsque l'on utilise la technique *k-ppv*. L'emploi d'une approche adaptative locale de la métrique devient indispensable si l'on veut maintenir les probabilités de classes conditionnelles proches de l'uniformité, et donc réduire l'estimation du biais. Nous avons utilisé, pour améliorer notre automatisation d'apprentissage d'un modèle cognitif, une technique qui calcule une métrique locale flexible, à l'aide des SVMs.

La frontière marginale maximale trouvée par le SVM est utilisée pour déterminer la direction la plus discriminante, au voisinage du point cherché (*query's neighborhood*). Cette direction fournit un schéma du poids local pour les caractéristiques d'entrée (voir Figure 5-15). Cette approche, en fait, accélère le processus de classification en calculant hors ligne (*off-line*) l'information relevante qui définit la pondération locale. Nous avons choisi cette approche, qui est utilisée dans l'exploitation de données (*data mining*) Domeniconi et Gunopulos (2001), car elle améliore l'efficacité de la recherche du plus

proche voisin pour de grandes dimensions, en particulier dans les espaces des états → actions continus, en raison de la réduction de la dimension locale, qui résulte de la pondération des caractéristiques.

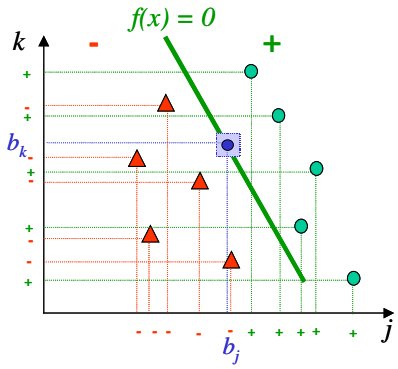


Figure 5-15. *Schéma de la pondération locale des caractéristiques d'entrée. Tout au long de la caractéristique j, b_j sépare les vecteurs supports positifs des négatifs. Par contre, le long de la caractéristique k, les supports vecteurs tardent à ordonner les positifs et les négatifs.*

5.5.3 Apprentissage avec évolution

Le modèle d'apprentissage et la connaissance de chaque AAV, pour la tâche à accomplir, sont représentés par les modèles prédéfinis, tels que nous les avons vus au chapitre précédent.

Prenons l'exemple du saut en hauteur. Un AAV est supposé avoir précédemment acquis la connaissance de la manière de sauter en faisant le mouvement correspondant du corps. Cependant, il doit améliorer son exécution afin d'atteindre un objectif prédéfini. Un spécialiste du saut en hauteur peut devoir faire plusieurs tentatives, avant de réussir à sauter par-dessus la barre horizontale. Ici, la simulation d'une telle tâche exige la modélisation d'un apprentissage évolutif, pendant lequel les capacités de l'athlète virtuel s'améliorent.

Notre nouvelle méthode, qui intègre le processus d'évolution, implique, à la fois la "capture du comportement" et une architecture de l'unité d'apprentissage (*Learning Unit Architecture (LUA)*). Cette dernière, qui fournit une "unité cognitive" différente pour chaque contexte environnemental, permet ainsi, par notre méthode, d'apprendre des

politiques sensibles à ce contexte. Ces politiques se trouvent alors dans le "cerveau" de l'AAV, et le choix de la *k-ppv* la plus appropriée est déterminé par l'état interne correspondant de l'AAV (voir Figure 5-16).

On définit certains critères de sélection, qui guident l'évolution du "cerveau" de l'AAV. Ce processus évolutionniste, appliqué à l'AAV, a pour objet de ne pas créer une dichotomie marquée entre la phase d'apprentissage et la phase d'utilisation. N'oublions pas que nous, les humains, "si nous sommes intelligents, c'est en partie parce que nous vivons en société", et que nous évoluons grâce au contact de nos semblables.

Nous traitons ici de l'évolution après apprentissage, au moyen de notre modèle cognitif. Il s'insère dans le prolongement des capacités d'autonomie et d'intelligence d'un AAV, nécessaires à une animation comportementale réaliste, dans un EV donné.

Pour le processus d'évolution, nous utilisons des caractéristiques, telles que la possibilité d'oubli après un certain temps, ou lorsque les cas sont peu importants. Si une association: états → actions est enregistrée depuis longtemps, et/ou est très semblable à une autre récemment ajoutée, elle est susceptible d'être enlevée.

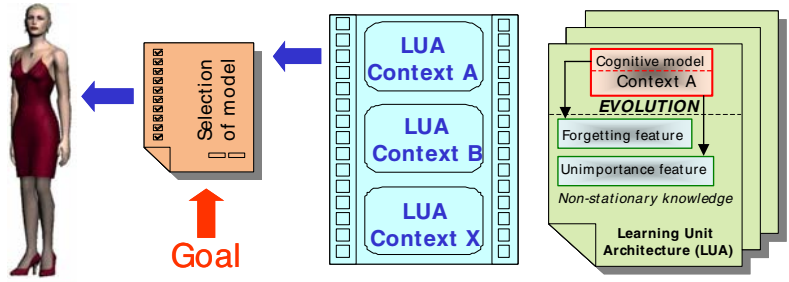


Figure 5-16. *Apprentissage de haut niveau de l'AAV (AVAhighLEARN), avec notre architecture de l'unité d'apprentissage - Learning Unit Architecture (LUA).*

5.5.4 Comportements de haut niveau avec évolution

L'AAV a la capacité d'oublier. Cette faculté est très importante pour progresser dans l'apprentissage car elle simule la dynamique propre au comportement humain, qui est de type non stationnaire, comme nous l'avons mentionné au chapitre précédent. En d'autres

termes, si des associations : états → actions ont été perçues il y a un certain temps déjà, ou sont similaires à celles qui se présentent, il est nécessaire de les supprimer. Du fait que notre méthode est en mesure d'oublier les "vieilles" associations, elle peut apprendre les comportements non stationnaires (voir Figure 5-16). Néanmoins, la faculté d'oublier de l'AAV peut l'amener à la faute, si l'animateur attend un temps suffisamment long avant de répéter le comportement.

Un bénéfice intéressant de cette technique est que l'AAV peut s'adapter "en ligne", car il peut remplir les "trous" grâce à son expérience. En d'autres termes, le programmeur n'a pas besoin de construire avec précaution l'AAV pour qu'il puisse gérer chaque situation de façon appropriée. Ce qui rend aussi cet AAV plus robuste.

5.5.5 Intégration et implémentation

L'implémentation de l'apprentissage automatique de modèles cognitifs est décrite à l'aide du diagramme UML à la Figure 5-17.

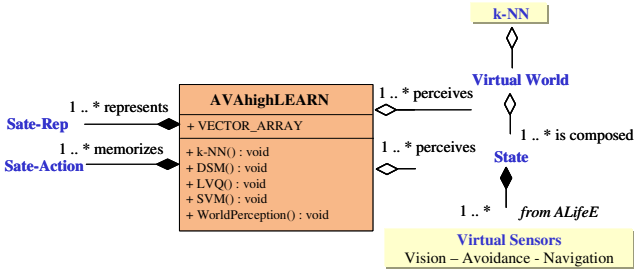
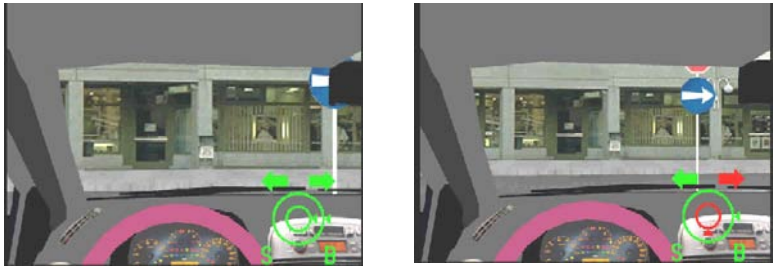


Figure 5-17. *Diagramme UML pour l'implémentation de la méthodologie de l'apprentissage automatique de modèles cognitifs.*

5.5.6 Résultats expérimentaux

Nous avons testé notre méthodologie avec le concept *LUA*, pour pousser les comportements des piétons, qui sont des AAVs, à évoluer et pour montrer la faculté du pilote de la voiture "à oublier", deux choses qui sont essentielles pour apprendre les comportements humains.



Figures 5-18a) et b). Le pilote souhaite tourner à gauche. Le co-pilote, se basant sur la connaissance apprise des signaux routiers, l'informe qu'il doit tourner à droite. La flèche s'affiche en rouge, au bas à droite de la Figure 5-18b). L'indicateur de direction du volant du co-pilote est indiqué par un cercle rouge.

Le résultat final est un comportement de conduite "correct", puisque le pilote peut planifier, assez en avance et de manière appropriée les manœuvres, à l'intérieur de la ville virtuelle. Ainsi que percevoir l'information sémantique: les piétons, les signaux routiers et les feux de signalisation. Nous avons obtenu les meilleurs résultants en appliquant notre méthode à quarante itérations en moyenne (voir Tableau 5-19).

Tableau 5-19. *Résultats obtenus avec notre méthode AVAhighLEARN. Toutes les animations ont été générées en temps réel avec OpenGL® (3.0 GHz avec une carte vidéo nVIDIA GeForce FX Go5350).*

	k-NN	DSM	LVQ	SVM
Temps d'exécution	15 µs	9 µs	8 µs	6 µs
Mémorisation	1.4 MB	1.3MB	1.0 MB	24 KB

5.5.7 Discussion

Nous avons mentionné auparavant que, dans notre domaine d'intérêt, les techniques de régression locale sont souvent plus puissantes que celles de régression globale. En effet, dans l'animation comportementale d'humains virtuels, le comportement d'un AAV est souvent une simple association: états → actions continue (*smooth mapping*). Ce qui est particulièrement vrai lorsque les entrées sous-optimales sont sélectionnées.

La technique des *k-ppv*, probablement la méthode d'apprentissage artificiel local la mieux connue, est un exemple de raisonnement par cas (*case-base reasoning*). Contrairement aux RNAs et aux SVMs, qui sont compacts, les *k-ppv* constituent une bibliothèque de tous les exemples de l'association: états → actions de la cible (*target mapping*) qui a été fournie à l'AAV.

La technique des *k-ppv* nous a montré, par nos expérimentations, qu'elle est simple à utiliser et qu'elle fonctionne remarquablement bien pour l'automatisation des modèles cognitifs Conde et Thalmann (2005a). Car le concepteur ne doit pas établir avec soin la représentation de l'espace des états (*space state*) afin de fournir une association: états → actions "lisse" (*smooth*) et simple. Car autrement cette association pourrait devenir particulièrement difficile (*rough*), et on serait alors conduit à un échec d'apprentissage, alors que les *k-ppv* n'ont pas un tel problème.

Cependant, depuis que les *k-ppv* utilisent des exemples explicites, une représentation de l'espace des états sous-optimale pourrait nécessiter une plus grande dimension. Ce qui conduirait à une croissance exponentielle du nombre d'exemples, pour pouvoir peupler chaque dimension additionnelle. On peut néanmoins l'accepter, les exigences de stockage restant habituellement très raisonnables car notre méthode bénéficie des "cartes cognitives" des senseurs virtuels, et de la perception virtuelle. Qui ont été décrites, respectivement, aux Chapitres 2 et 3.

Les *k-ppv* ont démontré leur capacité de réaliser une approximation adéquate d'associations difficiles. Cette approximation peut cependant conduire à une animation saccadée (*jittery animation*), à cause d'une généralisation incorrecte. De plus, la méthode *k-ppv* ne fait pas une généralisation aussi puissante que les RNAs, ainsi l'animation peut être sujette à plus de saccades, en raison du bruit, ou des erreurs de comportement, indiqués à l'AAV.

Comme nous l'avons montré pour notre plate-forme *ALifeE*, notre espace des états peut être incomplet. Néanmoins, nous l'avons vu, une approximation de cet espace des états est acceptable. Au chapitre suivant, nous montrons, par différentes Etudes de cas, où la précision de nos méthodes est suffisante pour obtenir un comportement correct de l'AAV. Ce qui peut se produire dans des EVs très différents, tels que: a) un appartement virtuel, b) une ville virtuelle, et c) la simulation de conduite d'une voiture dans une ville virtuelle.

La plupart des gens regardent les choses comme elles sont et se demandent « pourquoi ? ». Moi je regarde les choses comme elles pourraient être et je demande « pourquoi pas ! ».

J. F. Kennedy

Chapitre 6 – Etudes de cas

6.1 Introduction

Ce chapitre traite des différentes Etudes de cas que nous avons mentionnées tout au long du Chapitre 4 – Apprentissage comportemental, et du Chapitre 5 – Apprentissage cognitif. L'objectif recherché est, ici, de décrire plus en détail les hypothèses retenues pour la modélisation de chaque Etude de cas, ainsi que les concepts implémentés avec nos nouvelles méthodologies.

Nous avons voulu également, avec chaque Etude de cas, couvrir différents aspects de l'apprentissage perceptif, en faisant appel à nos nouvelles méthodologies pour les senseurs virtuels, décrites au Chapitre 2 et pour la perception virtuelle, décrites au Chapitre 3.

6.2 Objectifs

Pour la 1ère et la 2ème Etude de cas, que nous allons voir, nous utiliserons notre plate-forme *Virtual Human Director (VHD++)* Ponder et *al.* (2003), laquelle est du type

middleware. Elle a déjà été utilisée par différents projets de réalité virtuelle dans notre laboratoire et permet de simuler l'animation d'humains virtuels. Elle est développée conjointement au MIRALab (Université de Genève) et au VRLab (Ecole Polytechnique Fédérale de Lausanne). Son architecture, illustrée par la Figure 6-1, impose au minimum deux éléments principaux: le *vhdRuntimeEngine*, moteur d'animation de l'humain virtuel, et les *vhdServices,* enfichables sous forme de services faits à la demande.

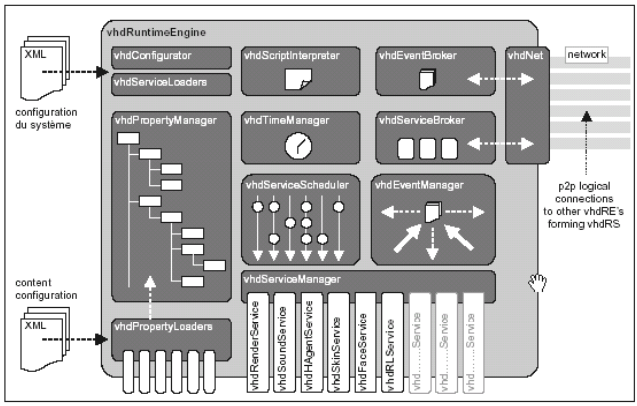


Figure 6-1. *Structure interne de VHD++ composée de services fondamentaux comme le vhdRuntimeEngine moteur d'animation de l'humain virtuel et les vhdServices enfichables: vhdSoundService, vhdFaceService et vhdRLService, ce dernier implémentant, ici, notre 1ère Etude de cas.*

La plate-forme *VHD++* comprend un environnement de développement pour l'animation comportementale, mais ne dispose actuellement d'aucun service d'IA située, par exemple des modalités sensorielles pour la vision, l'audition et le toucher, la perception virtuelle, ou des capacités cognitives, telles que la "mémoire" et l'apprentissage de bas et de haut niveau, que nous avons décrits aux chapitres précédents.

Pour la 3ème Etude de cas, nous avons adapté le simulateur de conduite automobile, développé en partie par le VRLab dans le cadre du projet européen VIRTUAL Kallmann et *al.* (2003b), à un environnement graphique plus simple et qui fait appel à la bibliothèque OpenGL® Angel (2004), mis au point par la société Silicon Graphics Inc.

Dans les trois Etudes de cas que nous traitons plus spécifiquement dans ce chapitre, nous avons cherché, chaque fois, à utiliser des enfichables (*plug-ins*) sous forme de modules qui implémentent nos nouvelles méthodes. De cette façon, ces enfichables sont à

même d'être sollicités, de façon interactive, par des scripts-interpréteurs en langage Python Rossum et Fred (2003). Ce qui nous permet de disposer d'un environnement interactif de test dans le cadre de notre animation comportementale.

6.3 Etude de cas No 1 – Agents Autonomes Virtuels dans une ville virtuelle

6.3.1 Description

En partant des nouvelles méthodes que nous avons développées au Chapitre 4 – Apprentissage comportemental, nous avons conçu un enfichable *vhdRLService* pour le déplacement des piétons dans une ville virtuelle.

6.3.2 Scénario

Dans Guye-Vuillième et Thalmann (2001) et Kallmann et *al.* (2001), les AAVs évoluent à l'intérieur d'EVs ouverts. L'EV considéré dans notre scénario, correspond à n'importe quel type d'environnement qui impose des contraintes physiques pour la navigation, par exemple une ville avec des bâtiments publics, des maisons, des rues, etc.

En fait, comme nous l'avons proposé au Chapitre 4 – Apprentissage comportemental, nous allons rechercher une série d'actions optimales dans un environnement donné. Cette approche serait inutile dans un environnement ouvert où tous les états seraient reliés ensemble. Par conséquent, l'EV doit contenir des états terminaux, définis par l'utilisateur de l'enfichable, *vhdRLService* dans notre Etude de cas Conde et *al.* (2003).

6.3.3 Expérimentation

Nous avons testé notre nouvelle approche avec un EV représentant une ville constituée d'une douzaine de bâtiments et de quelques rues.

Les AAVs peuvent être considérés comme des visiteurs (sans but précis, avec exploration de l'information). Grâce aux paramètres d'apprentissage, nous pouvons produire de bonnes ou de mauvaises représentations de l'EV, ce qui nous permet de simuler le comportement d'AAVs visiteurs (voir Figure 6-2), ou d'AAVs experts (avec un but précis, avec exploitation de l'information), en fonction de la durée de l'apprentissage.

Figure 6-2. *AAVs "visiteurs" à l'intérieur d'une ville virtuelle*

L'apprentissage par renforcement (AR) semble être bien adapté pour simuler la visite d'un EV, selon un comportement précis:

- Visite au hasard, avec une stratégie de type exploratoire, car le but de tels AAVs est la découverte de l'EV, sans objectif précis (voir Figure 6-2).
- Visite précise, avec une stratégie de type exploitation, car de tels AAVs ont seulement un objectif à atteindre, aussi rapidement que possible (voir Figure 6-3). Cet objectif utilise l'exploration seulement pendant les premières étapes dans un EV.

L'AR n'est pas utilisé ici dans son approche classique, mais en l'introduisant dans un EV comme moteur comportemental pour l'exploration, lequel permet d'apprendre et de visiter cet EV. Par conséquent, et contrairement aux algorithmes d'AR, notre intérêt se concentre, ici, plus sur l'apprentissage que sur l'exploitation de ce dernier. C'est, en effet, l'apprentissage plutôt qu'une optimisation qui nous permet de simuler l'animation comportementale des AAVs. Ceci constitue véritablement une nouvelle utilisation de la technique d'AR.

Figure 6-3. *AAVs regroupés dans un lieu précis et un AVA perdu découvrant une ville virtuelle.*

Dans notre moteur actuel, le réseau contenant des valeurs d'AR indique la politique qui permet d'atteindre le but le plus proche. De cette manière, l'utilisateur n'a pas besoin de demander à l'AAV d'atteindre un but précis, car l'AAV se dirige de lui-même vers son état terminal le plus proche.

Par contre, si l'AAV n'a pas un état terminal bien défini, un tel inconvénient peut être résolu en utilisant un AR avec des buts multiples Whitehead et *al.* (1993). Avec cette technique, le moteur a autant de réseaux qu'il a de buts à disposition. Elle emploie, cependant, plus de mémoire car elle exige un réseau pour chaque but. En lieu et place, nous proposons d'utiliser la méthode alternative avec un apprentissage par renforcement inverse (ARI), développée au Chapitre 4 – Apprentissage comportemental, et l'avons implémentée dans notre Etude de cas No 2.

Figure 6-4. *Deux AAVs communiquant dans une ville virtuelle.*

En conclusion, notre objectif a été de permettre à l'AAV d'explorer une ville virtuelle, jusque là inconnue, et de construire des structures, sous forme de modèles

cognitifs, ou de "cartes cognitives", comme celles décrites au Chapitre 2 – Senseurs virtuels et au Chapitre 3 – Perception virtuelle, et qui sont basées sur cette exploration. Dès que cette représentation construite, l'AAV peut aisément communiquer sa connaissance à d'autres AAVs "naïfs" (voir Figure 6-4).

6.4 Etude de cas No 2 – Extension de l'Artificial Life Environment (*ALifeE*) - *ALE Simulator*

6.4.1 Description

En partant des nouvelles méthodes que nous avons développées au Chapitre 4 – Apprentissage comportemental et au Chapitre 5 – Apprentissage cognitif, nous avons conçu les deux enfichables correspondants: *vhdAVAlowLEARNService* Conde et Thalamann (2005b) et *vhdAVAhighLEARNService* Conde et Thalamann (2005a) pour le déplacement d'un AAV dans un appartement virtuel.

6.4.2 Scénario

Nous avons choisi de simuler l'AAV dans un EV suivant: un appartement où il peut "vivre" de façon autonome et générer plusieurs comportements à des endroits spécifiques. Nous avons combiné l'ensemble des fonctionnalités nécessaires à notre simulation en étendant notre plate-forme *ALifeE*, laquelle a été décrite au Chapitre 2 – Senseurs virtuels et au Chapitre 3 – Perception virtuelle. Nous avons appelé *ALE Simulator*, cette extension.

Nous avons pu tester notre modèle comportemental, qui implémente la méthode en langage C++, grâce au module Python Rossum et Fred (2003) de la plate-forme en temps réel *VHD++* Ponder et *al.* (2003) pour la simulation avancée d'humains virtuels.

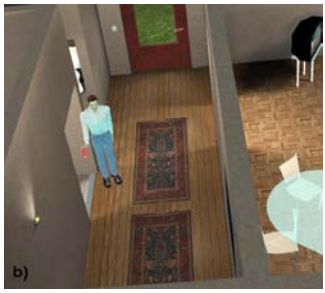
Ainsi que nous le voyons sur la Figure 6-5, une interface a été conçue pour surveiller, et changer au besoin, l'évolution des états et des actions de l'AAV en temps réel. Le module de planification de chemin Kallmann et *al.* (2003a) est appliqué à l'évitement d'obstacles lorsque l'humain virtuel se dirige vers un endroit spécifique (voir Figure 6-7).

Ce module est également appliqué aux décisions que l'AAV prend à chaque instant, et il joue les *keyframes* pour des actions apprises à l'aide du moteur de marche Boulic et *al.* (2004). Enfin, la visionneuse en 3D montre, en temps réel, ce que l'AAV décide de faire à chaque instant.

Figure 6-5. *Vue d'en haut de l'appartement virtuel utilisé par notre ALE Simulator.*

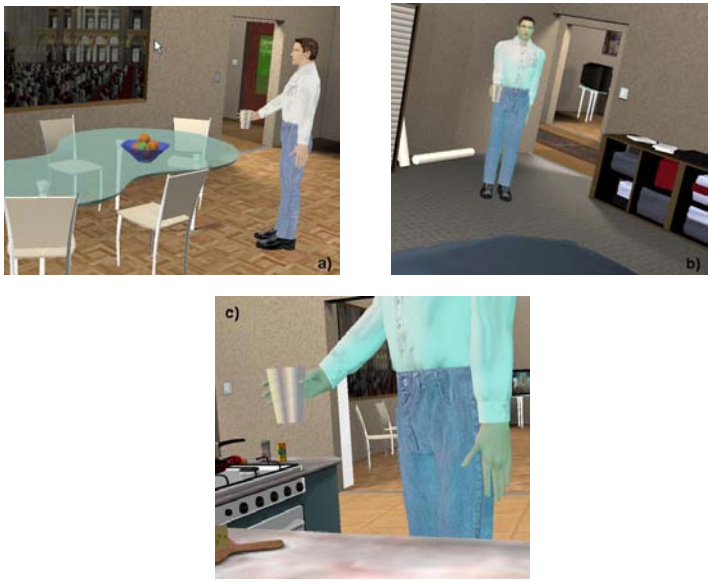
6.4.3 Expérimentation

Dans le premier scénario, l'AAV apprend comment voir, éviter et naviguer pour saisir une boîte de CDs sur la table de la cuisine, puis marche jusqu'à la pièce qui fait office de bureau et, finalement, la pose sur le bureau (voir Figures 6-6a) à c)).



Figures 6-6a) à c). *Instantanés de l'AAV qui apprend à saisir une boîte de CDs. Elle est maintenant posée sur le bureau.*

Dans le deuxième scénario, l'AAV apprend comment voir et saisir un verre qui est posé sur la table du salon, ou dans la chambre à coucher. Il l'apporte ensuite à la cuisine (voir Figures 6-7a) à c)). L'AAV doit apprendre à optimiser la marche dans les différentes pièces de l'appartement, tout en étant attentif à observer et à éviter les objets, ainsi qu'à se diriger en s'aidant de la planification de chemin entre les différentes portes (voir Figure 6-8).



Figures 6-7a) à c). *Instantanés de l'AAV qui apprend à saisir un verre dans le salon ou la chambre à coucher.*

Figure 6-8. *Vue d'en haut de l'appartement virtuel, avec la planification du chemin après apprentissage.*

6.5 Etude de cas No 3 – Simulation de conduite automobile dans une ville virtuelle - *VIRTUAL Simulator*

6.5.1 Description

Dans un premier temps, la compréhension du fonctionnement de la conduite, dans VIRTUAL Kallmann et *al.* (2003b), a été nécessaire pour l'intégration de ce dernier à l'environnement des bibliothèques OpenGL® (voir Figure 6-9, partie ①). La première étape a été d'implémenter la conduite de la voiture virtuelle au moyen d'une console de jeu, avec dosage de l'accélérateur et changement de vitesses par les touches de fonction. La gestion de la position de la caméra a aussi fait partie de cette étape. Par la suite, il nous a paru utile d'intégrer les rétroviseurs de la voiture. Puis, il a fallu être attentif à ce que les scripts de configuration et de comportement Python fonctionnent, car les modules sensoriels se servent de ces mêmes scripts pour déclencher certaines réactions. La gestion des collisions avec la librairie *V-Collide* Hudson et *al.* (1997), ainsi que le concept des voitures autonomes sur rail virtuel, terminent cette partie d'adaptation et d'intégration de VIRTUAL à l'environnement OpenGL®.

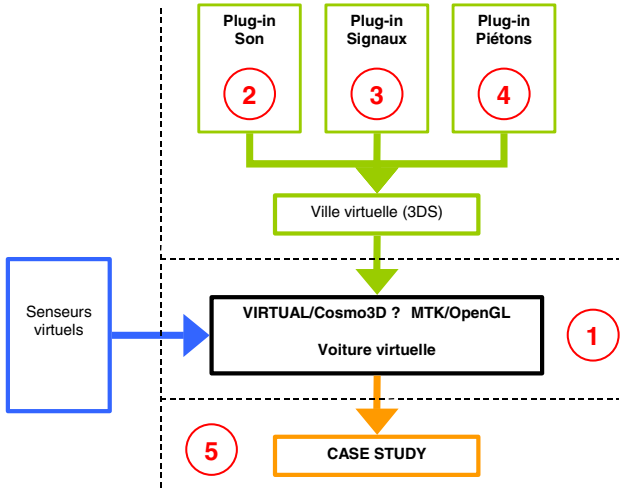


Figure 6-9. *VIRTUAL Simulator.*

Ensuite, nous avons développé trois enfichables:

- L'enfichable pour le son (partie ②), qui permet, de définir les caractéristiques d'un environnement sonore en spécifiant les sources sonores, les obstacles, les paramètres d'absorption et de réverbération, etc.
- L'enfichable pour les signaux routiers (partie ③), qui s'inspire du concept des *Smart Objects* Kallmann et Thalmann (2002), et qui attribue une signification aux panneaux et aux feux de signalisation routiers, afin que le module sensoriel de vision puisse les reconnaître et agir en conséquence.
- L'enfichable pour les piétons (partie ④), qui gère deux sortes de personnes, celles qui marchent dans la ville avec un but précis, et celles qui déambulent au hasard.

Ces enfichables ont été intégrés dans un environnement 3Ds max® Kulagin (2003), qui et un logiciel de visualisation.

Finalement, c'est par notre étude de cas (partie ⑤), que nous pouvons vérifier si les modalités sensorielles développées au Chapitre 2 – Senseurs virtuels et au Chapitre 3 – Perception virtuelle permettent d'apprendre la conduite d'une voiture dans une ville virtuelle, par des méthodes d'apprentissage de bas niveau comme au Chapitre 4 Apprentissage comportemental et de haut niveau comme celles au Chapitre 5 – Apprentissage cognitif Conde et *al.* (2005).

Interface pour le co-pilote

Lorsque la voiture est conduite dans la ville virtuelle, il est utile d'avoir un retour visuel en cas d'erreur de comportement du pilote. Par exemple, si le conducteur s'engage dans un carrefour alors que le feu de circulation est au rouge, un avertissement doit lui faire comprendre qu'il faut freiner et s'arrêter. Pour cela, nous avons conçu une "interface co-pilote", en surimpression (voir Figure 6-10). Les flèches gauche/droite indiquent que le conducteur doit tourner à gauche, ou à droite. Le cercle extérieur représente le volant du conducteur, tandis que le cercle intérieur représente celui du co-pilote. La lettre "S" indique un changement de rapport (*shift*) et la lettre "B" indique le freinage (*brake*).

Figure 6-10. *Interface du co-pilote signalant qu'il faut tourner à droite. Le cercle en trait discontinu jaune indique le champ de vision du pilote.*

Enfichable pour le son

L'enfichable avec un script 3Ds max® permet à l'utilisateur d'exporter des primitives sonores dans des fichiers au format *xml*, lisibles à partir d'une application qui implémente la librairie du son dans *VHD++*, plate-forme utilisée pour les Etudes de cas No 1 et No 2. Ces primitives sonores sont de quatre types: environnement, source sonore, auditeur, et obstacle.

Une scène est constituée de plusieurs environnements, qui ne peuvent toutefois pas se chevaucher. Un environnement est un espace dans lequel le son se propage selon certaines caractéristiques, sous forme de paramètres. Une cave représente, par exemple, un environnement où il y a beaucoup de réverbération, comparativement à une petite chambre meublée. L'environnement est donc caractérisé par un contour, ou une géométrie, et des paramètres, lesquels affectent la propagation du son. Il n'est cependant par nécessaire de connaître le détail de tels paramètres, puisque le module sonore de *VHD++* fournit un certain nombre d'environnements prédéfinis.

Par le biais de la librairie du son, qui provient de *VHD++*, la voiture virtuelle dispose d'un bruit de moteur dont la fréquence varie en fonction du régime. Il est également possible de klaxonner. Ces sources sonores, ainsi que l'environnement, contiennent des paramètres par défaut et ne tiennent donc pas compte des fichiers générés par l'enfichable.

L'auditeur de l'EV est, quant à lui, situé, par défaut, au niveau de la caméra. Sa position et son orientation sont mises à jour automatiquement, lors des déplacements.

Les propriétés des obstacles sont générées par l'enfichable du son, dans un fichier au format *xml*. Ainsi, chaque bâtiment de la ville virtuelle voit sa géométrie transformée en une liste de points qui vont délimiter l'obstacle sonore.

Enfichable pour les informations sémantiques

Afin de coupler notre Etude de cas avec les senseurs virtuels développés au Chapitre 2 – Senseurs virtuels, et la perception virtuelle développée au Chapitre 3 – Perception virtuelle, il est nécessaire que certains objets de l'EV, par exemple les panneaux et les feux de signalisation, aient une sémantique qui leur soit attachée.

Admettons, en effet, que le senseur virtuel de la vision, par une suite d'opérations, incluant essentiellement des tris, sache reconnaître l'objet le plus proche dans son champ de vision. Puisque ce senseur virtuel n'implémente pas de reconnaissance de formes, comme en vision artificielle, il faut lui donner un moyen de déterminer de quel type d'objet il s'agit. Par exemple, si le senseur virtuel voit un panneau de Stop, il doit pouvoir accéder à l'information "panneau de Stop", puis agir en conséquence, c'est à dire stopper la voiture.

Nous nous sommes inspirés du comportement, réadapté, des *Smart Objects* Kallmann et Thalmann (2002) pour l'implémentation proprement dite. Un utilitaire 3Ds max® exporte, dans un fichier, la position, l'orientation et la sémantique d'un objet sélectionné dans un EV ouvert. Ce fichier est ensuite lu par l'application, afin de créer une liste d'objets sémantiques consultable par l'AAV, à l'instant où il voit un objet dans son champ de vision.

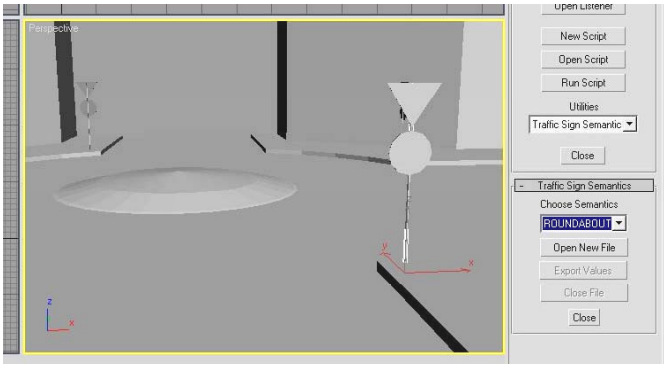


Figure 6-11. *Sélection d'un panneau aux environs d'un giratoire.*

Comme évoqué précédemment, le script exporte une liste de triplets *{position, orientation, sémantique}* qui représente les objets dans l'EV à sémantique particulière. Dans cette Etude de cas, il s'agit uniquement de panneaux routiers et de feux de signalisation (voir Figure 6-11).

Simulation de la vision

Afin de pouvoir tester la reconnaissance des panneaux routiers et des feux de signalisation, la classe *Semantic* intègre une méthode de perception visuelle (voir Figure 6-12) qui détermine l'objet le plus proche, dans une zone circulaire de rayon donné, ici, rayon = 10. Cette perception est couplée à la perception virtuelle que nous avons proposée au Chapitre 3 – Perception virtuelle.

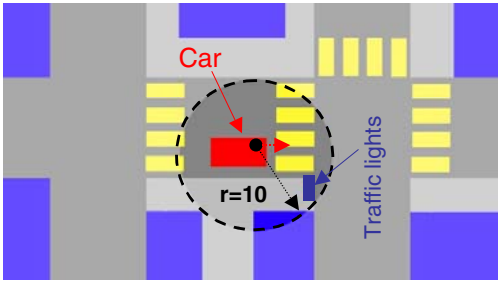


Figure 6-12. *Perception visuelle: le pilote de la voiture voit les panneaux routiers et les feux de circulation dans une zone circulaire autour de celle-ci*

6.5.2 Enfichable pour les piétons

Il s'agit de permettre l'intégration des piétons dans l'EV du conducteur, afin d'ajouter des difficultés supplémentaires à la conduite autonome. En effet, des piétons prioritaires, qui traversent la rue, représentent plus d'obstacles auxquels le conducteur doit être attentif, donc une occasion supplémentaire de commettre des erreurs d'apprentissage.

Cette approche se résume à la création de deux sortes de piétons: ceux qui se baladent aléatoirement dans la ville, et ceux qui ont un but précis vers lequel ils se rendent par un chemin idéal. Ces piétons se déplacent sur les trottoirs et ne peuvent traverser les rues que sur les passages pour piétons, où ils ont la priorité. L'idée suggérée au départ, et qui a été retenue, est de créer des couloirs, dans lesquels les piétons se déplacent.

Afin de déterminer ces couloirs, et de calculer le chemin idéal pour un piéton qui a un but, le concepteur doit d'abord s'atteler à une autre tâche. En effet, nous ne disposons que du modèle 3Ds max® d'une ville, dont il faut extraire l'information, d'abord sur l'emplacement des trottoirs, et ensuite sur celui des passages pour piétons. La méthode retenue, pour cela est de se servir d'un script 3Ds max®, assez similaire à celui des *Smart Objects* Kallmann et Thalmann (2002), lequel exporte deux fichiers: une liste de piétons, dont il sera question par la suite, et une liste des points d'intersection qui représentent, en réalité, les extrémités des passages pour ces piétons. Pour chacune des extrémités de ces derniers, l'utilisateur doit donner un point d'intersection et spécifier les directions que le piéton peut prendre, en partant de ce point. La position du point d'intersection est déterminée par celle de l'objet sélectionné (voir Figure 6-13).

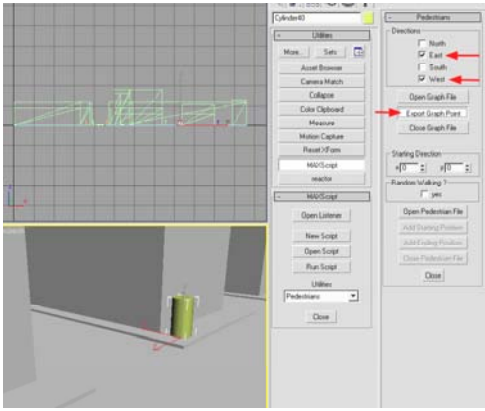


Figure 6-13. *Détermination de l'extrémité d'un passage pour piétons.*

Les directions sont indiquées par Nord, Est, Sud et Ouest. La Figure 6-13 illustre la correspondance entre les repères de notre EV en OpenGL®, et ceux de 3Ds max®. Ainsi, un point d'intersection ayant les directions Nord et Est (point [-20, 0, 10] de référence du cylindre sur la Figure 6-13) permet à un piéton, qui y passe, de se diriger soit dans le sens positif de l'axe X, soit dans le sens négatif de l'axe Z, dans notre EV OpenGL®.

Dans un deuxième temps, une fois le graphe d'états établi (voir Figure 6-14), il faut calculer les chemins les plus courts de chacun des points du graphe vers tous les autres. Ainsi, un piéton ayant un but va pouvoir emprunter un chemin idéal entre son point de départ et son point d'arrivée.

Le problème de la détermination des chemins les plus courts entre un point de départ (*starting point*) et tous les autres d'un graphe est un sujet connu et peut être résolu par l'algorithme de Dijkstra Papadimitriou (1982):

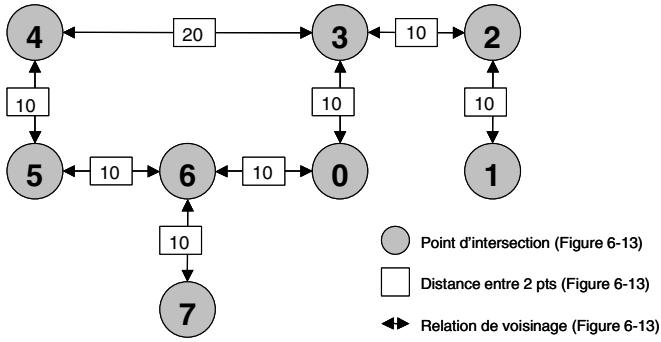


Figure 6-14. *Graphe d'états, généré à partir des points de la Figure 6-13.*

Donnée :	un réseau $A = (V, E, c)$ connexe, $\|V\| = n$, $\|E\| = m$, où $c:E \rightarrow R_+$ est une pondération non négative des arcs du graphe $G = (V, E)$. Un sommet particulier $s \in V$.
Résultat :	pour tout sommet $i \in V$, la longueur λ_i d'un plus court chemin de s à i ($\lambda_i = \infty$ s'il n'existe pas de chemin de s à i dans G) ainsi que le prédécesseur immédiat $p(i)$ du sommet i dans un tel chemin.
Algorithme :	
(1)	$\lambda_s = 0$, $\lambda_i = \infty$ pour tout i ! = s ; $p(i)$ = NULL pour tout i ; $T = V$
(2)	tant que T ! = vide alors faire :
(3)	soit i le sommet de T de plus petite étiquette λ_i (départager arbitrairement en cas d'égalité)
(4)	si un tel sommet n'existe pas ($\lambda_j = \infty$ pour tout $j \in T$) : STOP, les sommets encore dans T ne sont pas atteignables depuis s.
(5)	sinon, retirer i de T et pour tout successeur j de i encore dans T tester si $\lambda_j > \lambda_i + c_{ij}$ auquel cas poser $\lambda_j = \lambda_i + c_{ij}$ et $p(j) = i$.

Le résultat, produit par cet algorithme, avec les points de la Figure 6-13, est donc obtenu au moyen du graphe de la Figure 6-14. Ce graphe se lit de la manière suivante lorsque, par exemple, le piéton veut se diriger du point 2 au point 7. On lit tour d'abord la $7^{ème}$ paire {40, 6} qui nous informe que la distance cumulée du point 2 au point 7 est de 40, et que le prédécesseur du point 7 dans le chemin le plus court est le point 6. Il faut donc lire la $6^{ème}$ paire {30, 0}, qui indique le point 0 comme prédécesseur. Et il est ainsi possible, au fur et à mesure, de reconstruire le chemin à reculons, pour finalement arriver au parcours idéal: $2 \rightarrow 3 \rightarrow 0 \rightarrow 6 \rightarrow 7$.

Fonctionnement d'un piéton aléatoire

Rappelons qu'un piéton aléatoire n'a pas de but vers lequel il se déplace. Un tel piéton va simplement aller d'un état à un autre, dans le graphe d'états (voir Figure 6-15). Le choix de la direction à prendre, une fois parvenu à une intersection, est totalement aléatoire, si bien qu'il peut arriver que le piéton fasse un même parcours plusieurs fois de suite.

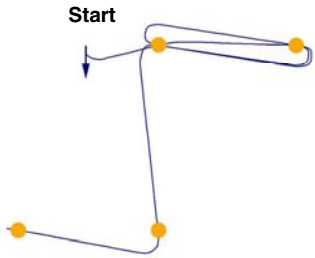


Figure 6-15. *Exemple de parcours d'un piéton aléatoire.*

Un piéton aléatoire est caractérisé par une position et une direction de départ, mais pas par une position d'arrivée. La position de départ, de même que pour un piéton avec but, peut être un état du graphe d'états. Si ce n'est pas le cas (voir Figure 6-14), le piéton va d'abord se diriger vers le point d'entrée du graphe d'états.

Fonctionnement d'un piéton avec but

A la différence du piéton aléatoire, le piéton avec but suit un chemin bien précis, déterminé par l'algorithme de Dijkstra. La Figure 6-16 illustre un résultat typique d'un piéton avec but.

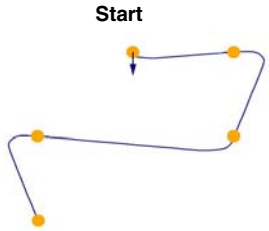


Figure 6-16. *Exemple de parcours d'un piéton avec but.*

Collisions entre piétons et évitement

Afin d'éviter de gérer des collisions entre piétons, on a mis en place des couloirs unidirectionnels. Un piéton ne se déplace donc plus que dans un couloir dans lequel il ne peut pas rencontrer d'autres piétons venant d'une direction opposée.

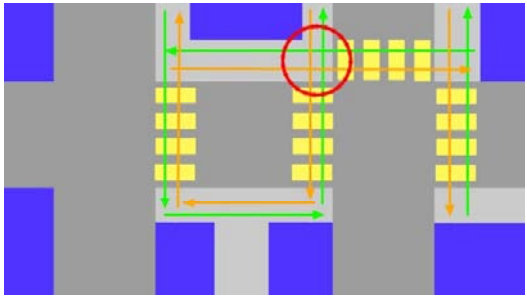


Figure 6-17. *Concept des couloirs unidirectionnels.*

Comme l'illustre la Figure 6-17, les couloirs posent, malgré tout, des problèmes de collisions au niveau des croisements (voir cercle rouge), si bien que d'autres mesures doivent être implémentées pour éviter les collisions entre piétons.

La première mesure consiste à utiliser une librairie de gestion des collisions et d'associer une géométrie à chacun des piétons. Le choix de la librairie s'est porté sur *V-*

Collide Hudson et *al.* (1997), parce qu'elle a déjà été utilisée pour gérer les collisions entre des voitures et des bâtiments. Ceci permet de traiter également les collisions éventuelles, entre les voitures et les piétons, voire entre les piétons et les immeubles.

Une seconde mesure doit tenir compte que lorsqu'une collision a lieu entre deux piétons, ceux-ci quittent leurs parcours pré-calculés, c'est à dire qu'ils se retrouvent dans un état qui n'est plus représenté dans le graphe d'états des parcours dans la ville. La solution consiste alors à insérer un nouvel état temporaire dans le graphe, par lequel le piéton est obligé de passer pour éviter la collision, avant de revenir sur son parcours normal.

Enfin, une troisième mesure consiste à utiliser des règles de priorités entre les piétons. Ce qui permet d'éviter les collisions parfaitement symétriques, ou les collisions-trappes, comme celles illustrées respectivement par les Figures 6-18 et 6-19.

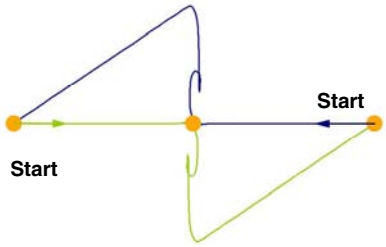


Figure 6-18. *Exemple d'une collision parfaitement symétrique.*

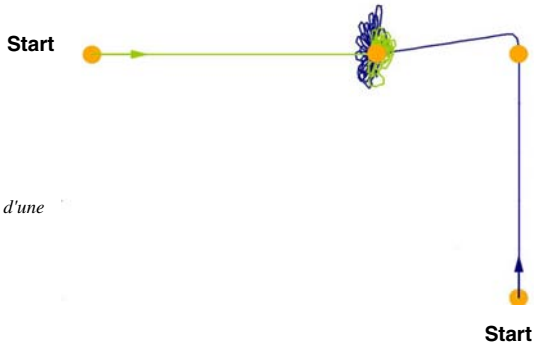


Figure 6-19. *Exemple d'une collision-trappe.*

6.5.3 Scénario pour l'étude de cas

L'objectif de ce scénario est de montrer que les méthodes nouvelles d'apprentissage, développées au Chapitre 2 – Senseurs virtuels, et au Chapitre 3 – Perception virtuelle, permettent d'apprendre un comportement de conduite "correct", en approximant un modèle comportemental (Chapitre 4) et un modèle cognitif (Chapitre 5).

Dans un premier temps, la voiture autonome se déplace sur un chemin prédéterminé, en alternant les comportements "corrects" et "incorrects". Dans un second temps, cette voiture roule de manière totalement aléatoire dans la ville virtuelle. Ces deux cas aident à connaître le comportement de conduite, pour lequel les senseurs virtuels vont extraire l'information nécessaire à l'apprentissage. Les résultats de cet apprentissage peuvent être réinjectés dans l'application, par le biais d'un scénario de validation (voir Figure 6-20).

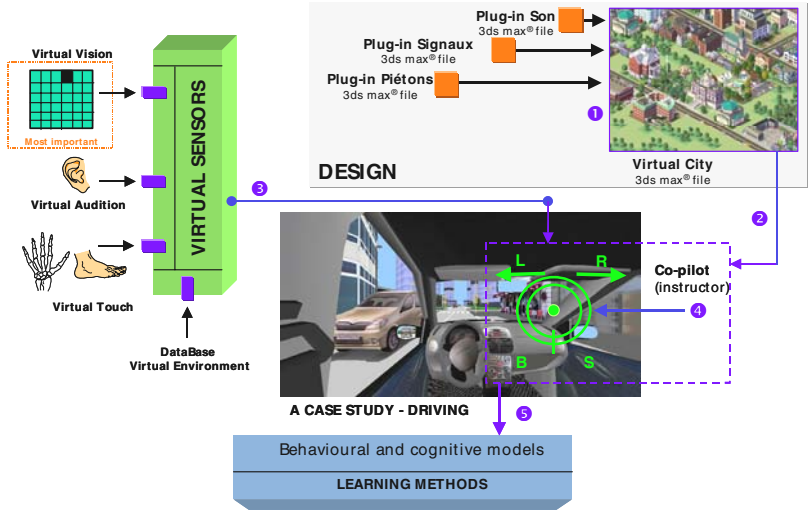


Figure 6-20. *Illustration du scénario de notre étude de cas.*

6.5.4 Expérimentation

Nous avons implémenté notre approche, afin d'apprendre les modèles comportementaux et cognitifs appliqués à la simulation de conduite d'une voiture, à l'intérieur d'une ville virtuelle. L'AAV est un pilote conduisant une voiture à l'intérieur de cette ville. Le pilote et le co-pilote ont le double contrôle: de l'accélération et de la direction (voir Figure 6-21). Les commandes sont des valeurs réelles (par exemple, l'espace d'actions est continu), la voiture peut se déplacer à n'importe quel endroit, et prendre n'importe quelle orientation.

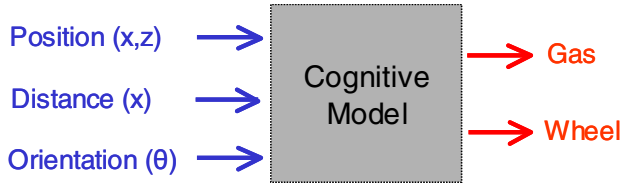


***Figure 6-21**: Modèle cognitif avec les entrées et les sorties.*

En utilisant nos enfichables pour les informations sémantiques, pour le son et pour les piétons, avec 3Ds max®, les concepteurs disposent de l'information nécessaire pour construire tous les types d'EV. Nous avons créé l'information sémantique pour notre simulation, mais elle avait un caractère statique. Cependant, nous souhaitions obtenir une information sémantique dynamique et nous avons donc employé la pseudo-perception (voir Figure 6-22), ou la perception virtuelle Conde et Thalmann (2004), afin d'atteindre notre objectif.

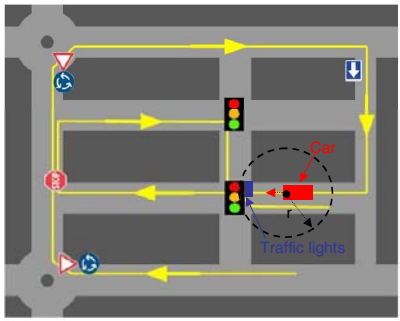


***Figure 6-22**: Un AAV apprenant à conduire une voiture à l'intérieur d'une ville virtuelle, avec une pseudo-perception visuelle.*

Nous avons appliqué notre nouvelle méthode d'apprentissage pour apprendre les comportements: éviter des piétons et tenir compte des feux de circulation (arrêter la voiture à un feu rouge, ralentir à un feu orange et, enfin, continuer de conduire à un feu vert).

Le résultat final est un bon comportement de conduite, puisque le pilote peut planifier assez en avance les manœuvres de la voiture en parcourant la ville virtuelle, pour éviter les piétons et réagir correctement aux feux de circulation (voir Figues 6-23a) à 6-23d)).

Figures 6-23a) à 6-23d). *Les senseurs virtuels et le modèle comportemental de conduite de la voiture. Un piéton souhaite traverser la rue selon les règles définies. L'interface informe le conducteur qu'il doit freiner (B – en rouge dans l'angle inférieur droit de la Figure 6-23a) et 6-23c)). Le pilote et le co-pilote sont indiqués au moyen des deux grands cercles verts.*

6.6 Discussion

Il est vrai que beaucoup de chercheurs de la communauté des sciences cognitives supposent que le comportement humain est non Markovien Nadel (2003). Cependant, les

modèles Markoviens ont souvent prouvé leur efficacité dans la littérature qui traite des agents autonomes Carberry (2001) et Kerkez et Cox (2003).

Nous avons présenté trois études de cas, traitées par nos nouvelles méthodes d'apprentissage des AAVs, qui donnent la possibilité à ces derniers de s'adapter rapidement et de mieux interagir avec l'utilisateur, ou l'animateur. Nos techniques étant très générales, il faut fournir moins d'informations externes et internes à l'AAV, lesquelles informations peuvent, de plus, être utilisées par d'autres AAVs.

Nous pensons que notre contribution est importante, car nous proposons une solution pour un apprentissage perceptif rapide, en ligne (*on-line*), pour des AAVs, comme nous l'avions suggéré dans notre introduction générale (Chapitre 1).

Pourquoi nos nouvelles méthodes sont-elles suffisantes pour un apprentissage en ligne des AAVs ? Parce qu'elles acquièrent toute la connaissance non-stationnaire dont ces derniers ont besoin pour sélectionner leurs actions optimales. Afin de choisir de façon optimale quelles actions sélectionner, l'AAV nécessite simplement de choisir, parmi toutes les actions potentielles, la meilleure pour une situation donnée. En apprenant avec précision le comportement humain, l'AAV peut prédire les résultats de ses actions, et donc sélectionner, de façon optimale, ce qu'il doit faire.

Nous allons, dans le chapitre suivant, conclure notre travail par une synthèse, par un exposé de nos contributions, et par nos propositions d'améliorations.

L'idéal est comme une étoile. Vous ne réussirez pas à la tenir entre vos doigts. Mais comme le marin sur l'océan sans fin, choisissez-la comme votre guide, et en la suivant, vous atteindrez votre but.

C. Schurz

Chapitre 7 – Conclusions

7.1 Synthèse

La perception est fortement reliée à la connaissance de l'environnement, elle ne doit pas être étudiée et comprise de façon isolée, mais être liée aux processus cognitifs: l'apprentissage, la mémorisation, et la résolution de problèmes. L'une des contributions majeures de cette thèse, est justement de proposer une nouvelle approche pour le sens de la présence – *Sense of Presence* – en utilisant des méthodologies qui permettent à un Agent Autonome Virtuel d'effectuer un apprentissage perceptuel. Ce sens de la présence est essentiel, entre autres, pour les applications en réalité virtuelle immersive, plus particulièrement pour les simulateurs, notamment de conduite automobile et de pilotage aéronautique.

Nous avons tenté, à l'aide de nos méthodes, d'augmenter la crédibilité d'un EV pour la simulation de conduite d'une voiture. Ayant appliqué le concept de la vie artificielle, nous avons peuplé cette simulation d'humains autonomes virtuels, dotés de perception, également virtuelle, et de capacités d'analyse et d'apprentissage. Les sensations et la perception connectent le cerveau humain au monde. Nos méthodes imitent ainsi partiellement certaines fonctions du cerveau humain, tandis que la perception humaine

permet la sensation de présence en termes de couleurs, de bruits, de mouvements, de textures et d'émotions.

Nos méthodologies ont le grand avantage d'être entièrement contrôlables par des paramètres fondamentaux, de façon que l'environnement virtuel puisse être facilement modifié – comme par exemple les feux de circulation et les signaux routiers. Les techniques que nous avons proposées sont appliquées à l'apprentissage, mais pourraient, bien évidemment, l'être également au divertissement, à l'ergonomie, et aux simulations militaires. Nos concepts, qui reposent sur cet apprentissage perceptuel, représentent une alternative novatrice, laquelle renforce l'immersion, et la "vie", à l'intérieur d'un simulateur.

L'autonomie est l'un des objectifs les plus importants pour l'animation d'un Agent Autonome Virtuel. Nous considérons, en général, cette autonomie comme une qualité, et la faculté, pour cet Agent, d'être géré par lui-même. Sa perception des éléments de son environnement est essentielle car elle le rend conscient de ce qui a changé autour de lui. C'est là véritablement le plus important à devoir être simulé, avant de pousser plus avant. La plupart des perceptions incluent, mais sans se limiter à elles, la vision et l'audition. L'adaptation et l'intelligence définissent alors comment cet Agent Autonome Virtuel peut raisonner sur ses perceptions, tout particulièrement lorsque des événements imprévus surviennent. De la même manière, si des événements prédictibles se présentent à nouveau, il est nécessaire qu'il ait une bonne capacité de mémoire car des comportements similaires peuvent toujours surgir. Enfin, l'émotion (ou perception instantanée) ajoute du réalisme, par exemple dans les relations "affectives" entre Agents Autonomes Virtuels.

Ont voit donc que l'apprentissage est un résultat du fonctionnement du système qui vient d'être décrit. Ce système, en fonctionnant, modifie l'environnement par les actions que le sujet exerce sur lui. Mais il modifie aussi le sujet lui-même. Les comportements antérieurs se trouvent développés et améliorés par l'exercice. Les comportements nouveaux sont élaborés et des réactions correspondantes s'ensuivent, qui s'intègrent au registre des comportements précédents du sujet considéré. Les définitions traditionnelles de l'apprentissage reposent sur la nécessaire stabilité des comportements récents. Pour qu'il y ait apprentissage, il faut donc que le comportement appris par l'Agent Autonome Virtuel, face à une situation donnée, soit reproductible.

Mais cet apprentissage n'est pas lié à un type précis de situation, il ne suppose pas nécessairement, pour être réalisé, l'existence d'un environnement particulier. L'expérience

montre que la vie quotidienne (activité professionnelle, moments de détente, vie associative ou familiale, etc.) contribue à l'assimilation de modes nouveaux de comportement et constitue, par conséquent, une situation d'apprentissage. Cet apprentissage se présente comme une imitation des situations que propose la vie. Toutes les informations reçues contribuent à modeler notre manière d'être. D'une manière générale, la formation de la personnalité de l'Agent Autonome Virtuel, et de tout le système des ses habitudes, est une conséquence de son comportement.

Ainsi, l'apprentissage apparaît comme un processus permanent et uniforme, aux conséquences non nécessairement prévues à l'avance. Une même situation peut modifier simultanément le savoir, le savoir-faire et le savoir être.

L'animation continue (comportement de bas niveau) est le résultat de la proprioception (qui inclut les motivations, les états "mentaux" internes ou émotionnels, etc.) et des réactions aux stimuli externes. L'animation comportementale de bas niveau est une grande voie ouverte de la recherche. Créer des humains virtuels autonomes, capables de sélectionner automatiquement des actions, n'est pas nouveau. L'état de l'art dans le domaine de ces humains virtuels nous montre qu'ils sont à même d'exécuter une variété de tâches, avec un certain niveau d'autonomie dans leur prise de décisions. Mais il reste qu'ils ne ressemblent pas à des humains réels, l'un des problèmes étant que ces humains virtuels ne présentent pas la subtilité de gestes et de manières qui caractérise un humain réel: expression faciales, langage du corps, etc.

7.2 Nos contributions

Nos principales contributions sont les suivantes:

- Une nouvelle approche pour l'intégration de senseurs virtuels
- Une nouvelle approche pour l'intégration de perceptions virtuelles
- Une nouvelle méthode pour la perception prédictive
- Une nouvelle architecture *Artificial Life Environment (ALifeE)* pour un Agent Autonome Virtuel
- Une nouvelle méthode pour l'apprentissage d'un modèle comportemental

- Une nouvelle méthode pour l'apprentissage d'un modèle cognitif
- Une nouvelle méthode pour l'apprentissage cognitif (adaptatif) avec évolution

3.2 Limitations et développements futurs

Beaucoup de travaux doivent encore être entrepris pour comprendre les différentes particularités des sens tels que la vision, l'audition et le toucher, objets de nos expériences. Nos résultats montrent que les modèles et les théories ont encore des insuffisances. Dans un certain nombre de cas, nos méthodes reposent sur des hypothèses approximatives, par manque de compréhension, en sciences cognitives, du fonctionnement humain. Pour les sens mentionnés et leur fonctionnement, de nouveaux développements conduiront à des technologies novatrices et à des systèmes qui augmenteront la "sensation de présence".

L'adaptation – apprentissage en ligne par l'Agent Autonome Virtuel – due à l'interaction avec un utilisateur humain dans un environnement virtuel, est un problème difficile, mais important, dans l'animation par ordinateur. Nous suggérons donc de développer une méthode à plusieurs niveaux, pour une adaptation rapide de l'humain virtuel. En visant spécifiquement des environnements où il y a un rapport coopératif, ou concurrentiel, entre l'humain virtuel et l'humain réel, qui agissent l'un sur l'autre.

Dans cette approche, une méthode distincte d'apprentissage est appliquée à chaque niveau du modèle comportemental, ou cognitif, de l'humain virtuel. Ce qui permet une adaptation efficace aux observations et aux expériences effectuées par cet humain virtuel à chaque niveau. Et qui amène aussi une distinction, dans le temps, entre les observations et les expériences qui fournissent les leçons appropriées pour chacun de ces niveaux. Ainsi, l'humain virtuel peut rapidement, de manière robuste, apprendre comment améliorer, de façon interactive, son comportement avec un utilisateur humain donné. Cette approche devrait, dans le futur, être conçue comme générale et facilement intégrable dans la plupart des systèmes d'animation comportementale existants.

Un autre usage possible de nos nouvelles méthodes consisterait en la création complète, à façon et en ligne, de modules d'aide à la décision pour un nouvel Agent Autonome Virtuel. Ce qui est similaire au *Behaviour Capture* ou *Cloning* de la littérature. Enfin, cette approche peut aussi être utilisée pour communiquer les comportements appris à d'autres Agents Autonomes Virtuels "naïfs", ou "primitifs".

Références

Par ordre alphabétique

Abbeel et Ng (2004) — P. Abbeel, and A.-Y. Ng. Apprenticeship Learning via Inverse Reinforcement Learning. In *Proceedings of Conference on Machine Learning ICML*, 2004.

Angel (2004) — E. Angel. *OpenGL : A primer*, Addison-Wesley, Published, 2nd edition, 2004. See also http://www.opengl.org.

Arleo et *al.* (2004) — A. Arleo, F. Smeraldi, and W. Gerstner. Cognitive navigation based on non uniform Gabor space sampling, unsupervised growing networks, and reinforcement learning. In *IEEE Transactions on Neural Networks,* **15**(3):639-52, 2004.

Bersini (1994) — H. Bersini. Reinforcement Learning for Homeostatic Endogenous Variables. In *Proceedings of 3rd Int. Conference On Simulated and Adaptive Behaviour*, pp. 325-333, MIT Press, 1994.

Berthoz (1997) — A. Berthoz. *The Brain's Sense of Movement*. Ed. Odile Jacob, 1997.

Blumberg (1996) — B. Blumberg. *Old Tricks, New Dogs: Ethology and Interactive Creatures*. Ph.D. Dissertation, MIT Media Lab, 1996.

Blumberg et Galyean (1996) — B. Blumberg, and T. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of ACM SIGGRAPH*, pp. 47-54, 1996.

Blumberg et *al.* (2002) — B. Blumberg, M. Downie, Y. Ivanov, B. Berlin, M. Johnson, and B. Tomlinson. Integrated learning for interactive synthetic

characters. In *Proceedings of ACM SIGGRAPH,* pp. 417-426, 2002.

Booch et *al.* (1998) G. Booch, J. Rumbaugh, and I. Jacobson. The Unified Modeling Language User Guide, Addison Wesley Professional.

Bordeux et *al.* (1999) C. Bordeux, R. Boulic, and D. Thalmann. An Effcient Perception Pipeline for Autonomous Agents. In *Proceedings of Eurographics*, pp. 123-130, 1999.

Boulic et *al.* (2004) R. Boulic, B. Ulicny, and D. Thalmann. Versatile Walk Engine, Journal of Game Development, **1**(1):29-52, Michael van Lent Editor, Charles River Media, 2004.

Braitenberg (1984) V. Braintenberg. *Vehicles: experiments in synthetic psychology*, MIT Press, 1984.

Burke et *al.* (2001) R. Burke, D. Isla, M. Downie, Y. Ivanov, and B. Blumberg. Creature smarts: The art and architecture of a virtual brain. In *Proceedings of the Computer Game Developers Conference*, 2001.

Caicedo et *al.* (2001) A. Caicedo, J.-S. Monzani, and D. Thalmann. Toward Life-Like Agents: Integrating Tasks, Verbal Communication And Behavioural Engines. In *The Virtual Reality Journal*, 2001.

Carberry (2001) S. Carberry. Techniques for plan recognition. In *User Modeling and User-Adapted Interaction*, **11**:31-48, 2001.

Carollo (2002) C. Carollo. *Sound Propagation in 3D Environment*. Ed. Ion Storm, 2002.

Chopra et Badler (2001) S. Chopra, and N. Badler. Where to look? Automating attending behaviours of virtual human characters. In *Proceedings of Autonomous Agents and Multi-Agent Systems,* **4**(1-2):9-23, 2001.

Conde et Karrakchou (1994) T. Conde, and M. Karrakchou. Automatic neural detection of anomalies in electrocardiogram signals. In *Proceedings of International Conference on Artificial Neural Networks in Engineering*, pp. 701-706, 1994.

Conde et *al.* (2003) T. Conde, W. Tambellini, and D. Thalmann. Behavioral Animation of Autonomous Virtual Agents helped by

Reinforcement Learning. Th. Rist et al. (Eds.): Intelligent Virtual Agents 2003. In *Lecture Notes in Computer Science,* Vol. 2792, pages 175-180, Springer-Verlag, Berlin Heidelberg, 2003.

Conde et Thalmann (2004) T. Conde, and D. Thalmann. An Artificial Life Environment for Autonomous Virtual Agents with multi-sensory and multi-perceptive features. In *Journal of Computer Animation & Virtual Worlds*, **15**(3-4):311-318, 2004.

Conde et Thalmann (2005a) T. Conde, and D. Thalmann. Autonomous Virtual Agents Learning a Cognitive Model and Evolving. T. Panayiotopoulos et al. (Eds.): Intelligent Virtual Agents 2005. In *Lecture Notes in Computer Science,* Vol. 3631, pp. 88-98, Springer-Verlag Berlin Heidelberg, 2005.

Conde et Thalmann (2005b) T. Conde, and D. Thalmann. Learnable Behavioral Model for Autonomous Virtual Agents: Low-Level Learning. In *Technical Report, Virtual Reality Lab, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2005.*

(submitted for publication)

Conde et Thalmann (2005c) T. Conde, and D. Thalmann. Goal Keeper Coach Based on Virtual Active and Predictive Perception. In *Technical Report, Virtual Reality Lab, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2005.*

(submitted for publication)

Conde et *al.* (2005) T. Conde, F. Vexo, and D. Thalmann. Virtual Perception and Learning : Tools to Improve the Sense of Presence Quality. In *Technical Report, Virtual Reality Lab, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2005.*

(submitted for publication)

Cornuéjols et Miclet (2002) A. Cornuéjols, and L. Miclet, *Apprentissage artificiel*, Ed. Eyrolles, Paris, 2002.

Courty et Marchand (2003) N. Courty, and E. Marchand. Visual perception based on salient features. In *IEEE International Conference on Intelligent Robots and Systems, IROS*, pages 1024-1029, 2003.

Dinerstein et *al.* (2004) J. Dinerstein, P.-K. Egbert, H. de Garis, and N. Dinerstein. Fast and learnable behavioral and cognitive modeling for virtual character animation. In *Journal of Computer Animation and Virtual Worlds*, **15**:395-408, 2004.

Domeniconi et Gunopulos (2001) C. Domeniconi, and D. Gunopulos, Adaptive Nearest Neighbor Classification using SupportVector Machines. In *Advances in Neural Information Processing Systems14*, MIT Press, pp. 223-229, 2001.

Downs et Stea (1973) R. Downs, and D. Stea. *Cognitive maps and spatial behavior. Image and Environment.* R. Downs and D. Stea, Eds. Chicago: Adline Publishing, pp. 8-26, 1973.

Duda et *al.* (2001) O. Duda, P.-E. Hart, and D.-G. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2001.

Duric et *al.* (2002) Z. Duric, Z. Gray, W. Heishman, F.-Li, A. Rosenfeld, M.-J. Schoelles, C.-D. Schunn, and H. Wechsler. Integrating perceptual and cognitive modeling for adaptive and intelligent human-computer interaction. In *Proceedings of the IEEE*, **90**(7):1272-89, 2002.

Edelman et Tonomi (2000) G. Edelman, and G. Tonomi. *A Universe of Consciousness: How Matter Becomes Imagination.* Basic Books, 2000.

Elfes (1991) G. Elfes. Occupancy Grid: A Stochastic Spatial Representation for Active Robot Perception. In *6th Conference on Uncertainly in AI*, pp. 60-70, 1991.

Funge (1999) J. Funge. In *AI for Games and Animation: A Cognitive Modeling Approach*. A.K. Peters: Natick, MA, 1999.

Funge et *al.* (1999) J. Funge, X. Tu, and D. Terzopoulos. Cognitive modeling: Knowledge, reasoning, and planning for intelligent characters. In *Proceedings of ACM SIGGRAPH*, pp. 29-38, 1999.

Funge (2000) J. Funge, J. Cognitive modelling for games and animation. In *Communications of the ACM'00*, **42**(7):39-48, 2000.

Geva et Sitte (1991) S. Geva, and J. Sitte. Adaptive Neatest Neighbor Pattern Classification. In *IEEE Transactions on Neural Networks*, **2**(2):318-322, 1991.

Gilles (2001) M. Gilles. Practical Behavioural Animation Based On Vision and Attention. *University of Cambridge Computer Laboratory, Technical Report TR522*, 2001.

Griffin (1973) D. Griffin. Topographical Orientation. *Image and Environment*. R. Downs and D. Stea, Eds. Chicago: Adline Publishing, pp. 296-299, 1973.

Guye-Vuillième et Thalmann (2001) A. Guye-Vuillème, and D. Thalmann. A High-level Architecture for Believable Social Agents, In *VR Journal*, Springer, **5**:95-106, 2001.

Guyon et Elisseeff (2003) I. Guyon, and A. Elisseeff. An inroduction to variable and feature selection. In *Journal of Machine Learning Research*, **3**:1157-1182, 2003.

Hill (2000) R.-W. Hill. Perceptual Attention in Virtual Humans: Towards Realistic and Believable Gaze Behaviours. In *Simulating Human Agents*, Fall Symposium, 2000.

Huang et *al.* (1995) Z. Huang, R. Boulic, N. Magnenat-Thalmann, and D. Thalmann. A Multi-sensor Approach for Grasping and 3D Interaction. In *Proceedings of Computer Graphics International*, pp. 235-254 Academic Press, 1995.

Hudson et *al.* (1997) T. Hudson, M. Lin, J. Cohen, S. Gottschalk, and D. Manocha. V-COLLIDE: Accelerated Collision Detection for VRML. In *Proceedings of VRML*, pp. 119-125, 1997.

Itti (2003) L. Itti. Visual Attention. In *The Handbook of Brain Theory and Neural Networks*, 2nd Ed., (M.-A. Arbib Eds.), pp. 1196-1201, MIT Press, 2003.

Jacobs (1993) O.-L.-R Jacobs. *Introduction to Control Theory*, 2nd Edition. Oxford University Press, 1993.

Joachmins (1999) T. Joachmins. "Making large-scale SVM learning praticable". In Advances in Kernel Methods – Support Vector Learning, B. Schölkopf, C. Burger, and A. Smola (Eds.), MIT-Press, 1999.

Joel et *al.* (2002) D. Joel, Y. Niv, and E. Ruppin. Actor-Critic Models of the Basal Ganglia: New Anatomical and Computational Prespectives. In *Neural Networks*, **15**:535-547, 2002.

Kaelbing et *al.* (1998) L.-P Kaelbing, M.-L. Littman, and A.-R. Cassandra. Planning and acting in partially observable stochastic domains. In *Artificial Intelligence,* **101**(1-2):99-134, 1998.

Kalman (1960) R.-E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *In Transaction of the ASME – Journal of Basic Engineering*, pp. 35-45, 1960.

Kallmann et *al.* (2001) M. Kallmann, E. de Sevin, and D. Thalmann, Constructing Virtual Human Life Simulations. In *Deformable Avatars*, Kluwer Publ., pp.240-247, 2001.

Kallmann et Thalmann (2002) M. Kallmann, and D. Thalmann. Modelling Behaviors of Interactive Objects for Virtual Reality Applications. In *Journal of Visual Languages and Computing*, **13**:177-195, 2002.

Kallmann et *al.* (2003a) M. Kallmann, H. Bieri, and D. Thalmann. Fully Dynamic Constrained Delaunay Triangulations. In *Geometric Modelling for Scientific Visualization*, **4**:74-123. Heidelberg, Germany, 2003.

Kallmann et *al.* (2003b) M. Kallmann, P. Lemoine, D. Thalmann, F. Cordier, N. Magnenat-Thalmann, C. Ruspa, and S. Quattrocolo. Immersive vehicle Simulators for Prototyping, Training and Ergonomics. In *Proceedings on Computer Graphics International*, pp. 90-95, 2003.

Kamwisher et Downing (1998) N. Kamwisher, and P. Downing . Separating the wheat from the chaff. In *Science*, **282**:57-58, 1998.

Karabassi (1999) E.-A. Karabassi. A Fast Depth-Buffer-Based Voxelization Algorithm. In *Journal of graphic tools,* **4**(4):5-10, 1999.

Kebling et *al.* (1996) L.-P. Kaebling, M. Littman, and A. Moore, Reinforcement Learning: a Survey, In *Journal of Artificial Intelligence Reserach*, **4**:237-285, 1996.

Kerkez et Cox (2003) B. Kerkez, and M. Cox. Incremental case-based plan recognition with local predictions. In *International Journal of Artificial Intelligence Tools*, **12**(4):413-463.

Kuffner et Latombe (1999) J.-J. Kuffner, and J.-C. Latombe. Fast Synthetic Vision, Memory, and Learning Models for Virtual Humans. In *Proceedings of Computer Animation, IEEE*, pp. 118-127, 1999.

Kulagin (2003) Boris Kulagin. Advanced 3Ds Max 5: Modeling and Animating, Independent Pub Group Ed., 2003

Langton (1989) C. Langton, *Artificial Life*, Addison-Wesley, 1989.

Larkin (2003) P. Larkin. Achieving human style navigation for synthetic characters. A survey. In *Proceedings of Neural Networks and Computational Intelligence*, pp. 30-37, 2003.

Ledoux (1996) J. Ledoux. *The Emotional Brain.* Ed. Suster, 1996.

Mitchell (1997) T. Mitchell. *Machine Learning*. Ed. McGraw Hill, 1997.

Monzani et *al.* (2001) J.-S. Monzani, A. Caicedo, and D. Thalmann. Integrating Behavioural Animation Techniques. In *Proceedings of Eurographics,* **20**(3), 2001.

Nadel (2003) N. Nadel. Encyclopedia of Cognitive Science, Nature Pub. Group, London, 2003

Ng et Russel (2000) A.-Y. Ng, and S. Russell. Algorithms for Inverse Reinforcement Learning. In *Proceedings of Conference on Machine Learning*, pp. 663-670, 2000.

Nolfi et Floreano (2001) S. Nolfi, and D. Floreano (2001). In *Evolutionary Robotics*. The Biology, Intelligence, and Technology of Self-organizing Machines, MIT Press.

Noser et *al.* (1995) H. Noser, O. Renault, D. Thalmann, and N. Magnenat-Thalmann. Navigation for Digital Actors based on Synthetic Vision, Memory and Learning. In *Computers and Graphics*, **1**:7-19, 1995.

Noser et Thalmann (1995) H. Noser, and D. Thalmann. Synthetic Vision and Audition for Digital Actors. In *Proceedings of Eurographics*, pp. 325-336, 1995.

Papadimitriou (1982) C.-H. Papadimitriou, and K. Steiglitz. *Combinatorial Optimization - Algorithms and Complexity*, Prentice-Hall, 1982.

Peters et O'Sullivan (2002) C. Peters, and C. O'Sullivan. Synthetic Vision and Memory for Autonomous Virtual Humans. In *Computer Graphics Forum*, **21**(4):743-753, 2002.

Pettre et *al.* (2003) J. Pettre, J.-P Laumond, and T. Simeon. A 2-stages locomotion planner for digital actors. In *Proceedings of ACM SIGGRAPH /*

Eurographics Symposium on Computer Animation, pp. 258-264, 2003.

Pfeiffer et Scheier (1999) R. Pfeiffer, and C. Scheier. *Understanding Intelligence*. MIT Press, 1999.

Ponder et *al.* (2003) M. Ponder, G. Papagiannakis, T. Molet, N. Magnenat-Thalmann, and D. Thalmann. VHD++ Development Framework: Towards Extendible, Component Based VR/AR Simulation Engine Featuring Advanced Virtual Character Technologies. In *Computer Graphics International (CGI)*, pp. 96-104, 2003.

Pouget (2002) A. Pouget. A computational perspective on the neural basis of multi-sensory spatial representations. In *Nature Reviews/Neuroscience*, **3**:741-747, 2002.

Ramachandran et Bree (2001) S. Ramachandran, and D.-S. Bree. Learning action selection in autonomous agents. In *IEEE International Conference on Systems, Man, and Cybernetics,* **5**(7-10), 3391-3396, 2001.

Renault et *al.* (1990) O. Renault, N. Magnenat-Thalmann, and D. Thalmann. A Vision-based Approach to Behavioural Animation. In *Journal of Visualization and Computer Animation*, **1**:18-21, 1990.

Reynolds (1987) C.-W. Reynolds. Flocks, herds and schools: A distributed behavioural model. In *Computer and Graphics* **21**(4):25-34, 1987.

Reynolds (1993) C.-W. Reynolds. An evolved, vision-based behavioral model coordinated group motion. In *From Animals to Animats, 2nd International Conference on Simulation of Adaptive Behavior,* pp. 384-392, MIT Press, 1993.

Rickel et Johnson (1999) J. Rickel, and W.-L. Johnson. Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control. In *Applied Artificial Intelligence* **13**:343-382, (Special issue on Animated Interface Agents), 1999.

Rossum et Fred (2003) G. van Rossum, and L. Fred. *The Python Language Reference Manual*, Jr. Drake (Eds.), 2003.

Russell et Norvig (2003) S. Russell, and P. Norvig. *Solution Manual for Artificial Intelligence: A Modern Approach*." 2nd Ed., Prentice Hall, 2003

Russell et Zimdars (2003) S. Russel, and A.-L. Zimdars. Q-Decomposition for Reinforcement Learning Agents. In *Proceedings of International Conference on Machine Learning*, pp. 656-663, 2003.

Schmitzberger et *al.* (2002) E. Schmitzberger, J.-L. Bouchet, M. Dufaut, D. Wolf, and R. Husson. Capture of homotopy classes with probabilistic road map. In *IEEE/RSJ International Conference on Intelligent Robots and System,* **3**:2317-2322, 2002.

Sherman et Koch (1998) S.-M. Sherman, and C. Koch. "Thalamus" in Synaptic Organization of the Brain, 4th Ed., G.-M. Shepherd (éd.), Oxford University Press, pp. 289-328, 1998.

Slater et Usoh (1993) Slater, M., Usoh, M. (1993). Presence in immersive Virtual Ennvironments. In *Proceedings IEEE Virtual Reality*, pp. 33-40.

Sowa (1964) J.-F. Sowa. *Conceptual Structures*, Ed. Addison Wesley Company, 1964.

Strösslin et *al.* (2002) Th. Strösslin, Ch. Krebser, A. Arleo, and W. Gerstner. Combining Multimodal Sensory Input for Spatial Learning. In *Proceedings of ICANN*, pp. 87-92. LNCS 2415, Springer-Verlag, 2002.

Sutton (1988) R.-S. Sutton. Learning to Predict by the Method of Temporal Differences. In *Machine Learning*, **3**:9-44, 1988.

Sutton et Barto (1998) R.-S. Sutton, and G. Barto. *Reinforcement Learning : An Introduction*. MIT Press, 1998.

Sutton et Santamaria (1998) R.-S. Sutton, and J.-C. Santamaria, "A Standard Interface for Reinforcement Learning Software in C++", version 1.1, 1998.

Takeuchi et *al.* (1992) T. Takeuchi, M. Unuma, and K. Amakawa. Path planning and its application to human animation systems. In *Proceedings of Computer Animation*, pp. 163-174, 1992.

Terzopoulos (1999) D. Terzopoulos. Artificial life for computer graphics. In *Communications of the ACM,* **42**(8):33-42, 1999.

Thalmann (2004) D. Thalmann. Control and Autonomy for Intelligent Virtual Agent Behaviour. *In Proceedings of the 3rd Hellenic Conference*

on Artificial Intelligence SETN 2004, Lecture Notes in Artificial Intelligence, **3025**:515-524, Springer Verlag, 2004.

Thornton (2003) C. Thornton. Indirect sensing through abstraction learning. In *Intelligent Data Analysis*, **7**(3):1-16, 2003.

Thorndike (1911) L. Thorndike. *Animal Intelligence : Experimental Studies*, 1911.

Tolman (1948) E.-C Tolman. Cognitive maps in rats and men. *Psychological Review*, **55**:189-208, 1948.

Tomlinson et Blumberg (2003) B. Tomlinson, and B. Blumberg. Alphawolf: Social learning, emotion and development in autonomous virtual agents. In *Proceedings of Innovative Concepts for Agent-Based Systems, First International Workshop on Radical Agent Concepts, WRAC 2002,* Lecture Notes in Computer Science, **2564**:35-45, Springer 2003.

Tu et Terzopoulos (1994) X. Tu, and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behaviour. In *Proceedings of ACM SIGGRAPH*, pp. 43-50, 1994.

Vapnik (1995) V. Vapnik. *The nature of statistical learning theory*. Springer-Verlag, 1995.

Wenzel (1992) E.-M. Wenzel. Localization in Virtual Acoustics Displays. In *PRESENCE*, **1**(1): 80-107, 1992.

Whitehead et *al.* (1993) S. Whitehead, J. Karlsson, and J. Tenenberg. Learning Multiple Goal Behavior via Task Decomposition and Dynamic Policy Merging. In Connell and Mahadevan (Eds.), *Robot Learning*, Kluwer Academic Publishers, 1993.

Wilhelms (1990) J. Wilhelms. Behavioural animation using an interactive network. In *Proceedings of the Computer Animation*, pp. 95-106, 1990.

Glossaire

Sciences cognitives et neurosciences

Carte cognitive (cognitive map)

Concept attribué à E. Tolman (1948), qui signifie la représentation mentale qu'un humain se fait de l'organisation de l'espace dans lequel il se trouve. Cette représentation interne mentale implique que l'humain est capable d'inférer les relations, les distances ainsi que les directions qui lient différents points de l'espace sans avoir eu une expérience directe. Ainsi, l'humain peut atteindre un but en passant par des chemins, sans jamais les avoir emprunter auparavant. Les expériences en psychologie cognitive montrent que cette carte cognitive est construite grâce à la perception visuelle

Par analogie, l'expression « carte cognitive » est souvent employée dans des domaines non spatiaux mais qui sont facilement descriptibles en termes spatiaux, comme la résolution de problème, relations parentales, analyses statistiques, perception des couleurs, organisation de bases de données, etc.

Modèle mental

Est une représentation permettant de simuler mentalement le déroulement d'un phénomène pour anticiper les résultats d'une action. La notion de modèle mental est également largement employée en ergonomie cognitive et interaction homme-machine.

Modèle cognitif

Etudie l'ensemble des phénomènes du codage, du stockage de la manipulation de l'information par le système nerveux central.

Perception

Ensemble des mécanismes et des processus par lesquels l'organisme prend connaissance du monde et de son environnement sur la base des informations élaborées par ses sens.

Perception unifiée

Suppose que les mécanismes centraux actifs vont permettre la levée des ambiguïtés, le rattrapage ou l'anticipation des retards différentiels entre senseurs, ainsi que l'unification des espaces par des mécanismes biologiques astucieux qui ne sont pas seulement des changements de coordonnées, etc. La perception est une interprétation, sa cohérence est une construction dont les règles dépendent de facteurs endogènes et des actions que nous projetons Berthoz (1997).

Proprioception

Le terme proprioceptif désigne la sensibilité du système nerveux à divers stimuli comme la pression ou la tension qui affecte les muscles, les os, les tendons et les articulations. Les propriocepteurs sont des récepteurs microscopiques qui permettent la sensibilité proprioceptive.

Sensori-motrice

Désigne la coordination qui s'effectue, lorsqu'on souhaite atteindre un objet que l'on a localisé, entre l'information sur la position de cet objet dans l'espace et les commandes des muscles pour l'atteindre. L'œil, en son centre, dispose d'un plus grand nombre de cellules qui lui permettent d'analyser finement un objet. Les yeux fixent d'abord la cible de manière à l'amener dans cette zone centrale (la fovéa). Puis le cerveau tient compte de la position des yeux et de la position de la tête et détermine ainsi la localisation de l'objet. L'écartement des yeux lui indique la distance à laquelle se trouve l'objet. La courbure du cristallin (qui assure la netteté de l'image) fournit aussi une information sur la distance (vision en relief).*Système nerveux*

Un système en réseau formé des organes des sens, des nerfs, du cerveau, de la moelle épinière, etc. Il coordonne les mouvements musculaires, contrôle le fonctionnement des organes, véhicule les informations sensorielles et motrices vers les effecteurs, et, chez les animaux dotés d'un cerveau, régule la pensée et les émotions.

Chez les vertébrés on distingue traditionnellement le système nerveux central (encéphale et moelle épinière) du système nerveux périphérique (nerfs crâniens sensori-moteurs, nerfs rachidiens, système entérique).

Par analogie, l'expression « petit système nerveux » a été employé pour véhiculer les informations sensorielles provenant de nos senseurs pour la vision, l'audition et le toucher vers les effecteurs de l'Agent Autonome Virtuel (AAV).

Apprentissage artificiel

Apprentissage perceptif

Apprentissage qui vise à accroître l'habileté d'un Agent Autonome Virtuel (AAV) à recueillir de l'information de son environnement.

Vie artificielle

Exploite les concepts issus des courants cognitiviste et connexionniste qui partagent les sciences cognitives. Elle propose donc des solutions qui s'appuient tantôt sur l'approche symbolique, tantôt sur l'approche auto-organisationnelle ou construit ses modèles à partir d'une combinaison des deux méthodologies.

Réalité virtuelle

Sensation de présence

Est nécessaire pour l'interaction sur les objets du monde (réel ou virtuel). Les perceptions (visuelle, auditive et tactile) sont à l'origine de cette sensation. Les perceptions incohérentes diminuent le réalisme des scènes (images et multimédia) qui supportent l'interaction. Le monde perçu n'est plus conforme aux représentations mentales des utilisateurs.

Effecteur: dispositif d'entrée/sortie en réalité virtuelle permettant l'échange d'informations entre l'utilisateur et l'ordinateur.

Animation comportementale

Le but des modèles comportementaux est de simuler le comportement de toutes sortes d'individus vivants (plantes, animaux et êtres humains). Les modèles comportementaux peuvent être classifiés en deux catégories : a) les modèles de transformation interne, provoquant des changements externes de l'objet (croissance de plantes, muscles, etc.) et b) les modèles d'animation externe, définissant le comportement extérieur d'un être, ses actions et ses réactions, de manière individuelle (animal et humain) ou collective (nuée d'oiseaux, banc de poissons et troupeau de mammifères).

Curriculum vitae

Toni Conde

Ch. de Vuillonnex 46
CH – Confignon/GE

E-mail: toni.conde@bluewin.ch
Born July 6, 1961, in Morges, Switzerland.
First languages: French and Spanish. Second languages: English and German.

BIOGRAPHY

Professional career : Ecole Polytechnique Fédéarale de Lausanne (EPFL); EM Microelectronic-Marin Ltd. (the Swatch Group Ltd.); Conde + Partners Consultants Ltd. and Neuron Research Ltd. (two start-ups sold to a European major company in software engineering and consulting); Advisor for the foundation of Marvel Communications Ltd. - Web Intelligence company (the Swissquote Group Ltd.); Advisor for the foundation of Qualimatest Ltd. (measurement and computer science); Ingenia Systems Ltd.; Various executive positions held with Swisscom Ltd including: Senior Consultant, Manager of Consulting and Design, Head of Account Specialized Sales, Head of department, Principal Expert and Director of Consulting Services; Consultant for various multinational companies, banks and international organizations; Speaker on ICT-related topics; Member of numerous committees. Today at NetExpert Ltd. with the executive position of Principal Expert.

➔ **Since October 2002 at partial time (70 %)**

Few success stories : FINEXPRO - Intelligent systems for financial analysis; ETHICS - ETH Information Control System (network libraries for Swiss Federal Institutes); WELCOM - World ELectronic COMmunity of World Economic Forum (showed at Davos'97 Annual Meeting - Building the Network Society); Technology eXchange Program (TXP) - 1st ATM network in Switzerland (Geneva MAN) with ADSL Technology (impulsor of ADSL technology); GDCNet - Geneva Diplomatic Community Network; ATAC - Advanced Technology & Applications Center at Geneva; ITU TELECOM WORLD 1999 - Head of Business development and Integrated Solutions of Swisscom Group (all content of the Event); INTEGRAL satellite-ESA's INTErnational Gamma-Ray Astrophysics Laboratory, design of network between the Mission Operations Center in Darmstadt (European Space Agency) and Geneva (host the ISDC-Integral Science Data Center). International Committee of Red-Cross (ICRC), first most important network designed in Europe using IP-Telephony technology with GigaEthernet - Cisco Systems IP-Telephony Award 2002. ITU TELECOM WORLD 2003 – Chief Operating Officer "IT and Telecommunications Services"; Design of LAN/MAN and IP-Telephony for the new Worldwide Headquarter and Research Centre of Serono Int. (more than 10'000 data and voice ports) at Geneva.

Academic experience : During 1988-2001, Senior lecturer at The University of Applied Sciences Western Switzerland - HES-SO in Computer and Communications Sciences (Teaching: C/C++, Advanced Computer Science and Communication Systems).

Reviewer : Journal of Computer Animation and Virtual Worlds, Intelligent Virtual Agents 2003, Eurographics 2004, Autonomous Agents and Multi-Agent Systems 2004, ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2004, Computer Animation and Social Agents 2004/2005.

Program Committee Member : Intelligent Virtual Agents 2005.

Miscellaneous : Nominated by IEEE Senior member in 1998, Computer Society.

Other skills : Private Pilot licence single engine PPL(A)-JAR and multi-engine PPL(A)-FAA. Instrument rating (IFR).

Awards

Journal of Computer Animation & Virtual Worlds, **15**(3-4):311-318, 2004. Special Issue on the very best papers of Computer Animation and Social Agents 2004.

Fully refereed Publications

Conde et Thalmann (2005a) T. Conde, and D. Thalmann. Autonomous Virtual Agents Learning a Cognitive Model and Evolving. T. Panayiotopoulos et al. (Eds.): Intelligent Virtual Agents 2005. In *Lecture Notes in Computer Science,* Vol. 3631, pp. 88-98, Springer-Verlag Berlin Heidelberg, 2005.

Conde et Thalmann (2004) T. Conde, and D. Thalmann. An Artificial Life Environment for Autonomous Virtual Agents with multi-sensory and multi-perceptive features. In *Journal of Computer Animation & Virtual Worlds*, **15**(3-4):311-318, 2004.

Conde et *al.* (2003) T. Conde, W. Tambellini, and D. Thalmann. Behavioral Animation of Autonomous Virtual Agents helped by Reinforcement Learning. Th. Rist et al. (Eds.): Intelligent Virtual Agents 2003. In *Lecture Notes in Computer Science,* Vol. 2792, pages 175-180, Springer-Verlag Berlin Heidelberg, 2005.

Conde et Karrakchou (1994) T. Conde, and M. Karrakchou. Automatic neural detection of anomalies in electrocardiogram signals. In *Proceedings of International Conference on Artificial Neural Networks in Engineering*, pages 701-706, 1994.

Not refereed Publications

Conde (1989) T. Conde. ENERGEXPERT: to optimize the electric consumption of power. In *Proceeding of International Conference on Expert Systems and Their Applications*, Vol. 2, pages 45-52, Avignon, 1989.

Conde (1987) T. Conde. FINEXPRO: an expert system for analyzing companies. In *Proceeding of International Conference on Expert Systems in Banking*, Vol. 1, pages 23-29, Lugano, 1987.

Technical Reports

Conde et *al.* (2005) T. Conde, F. Vexo, and D. Thalmann. Virtual Perception and Learning : Tools to Improve the Sense of Presence Quality. In *Technical Report, Virtual Reality Lab, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2005.*

(submitted for publication)

Conde et Thalmann (2005b) T. Conde, and D. Thalmann. Learnable Behavioral Model for Autonomous Virtual Agents: Low-Level Learning. In *Technical Report, Virtual Reality Lab, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2005.*

(submitted for publication)

Conde et Thalmann (2005c) T. Conde, and D. Thalmann. Goal Keeper Coach Based on Virtual Active and Predictive Perception. In *Technical Report, Virtual Reality Lab, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2005.*

(submitted for publication)