République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider - Biskra

Faculté des Sciences et de la Technologie

Département de Génie Electrique

N° d'ordre :…….

Série :……..

THÈSE

Présentée pour obtenir le diplôme de **Doctorat en Sciences**

Spécialité: **AUTOMATIQUE**

Par:

CHERROUN Lakhmissi

Thème :

Navigation Autonome d'un Robot Mobile par des Techniques Neuro-Floues

Soutenu publiquement le : 22 / 05 / 2014 devant les membres de Jury :

KAZAR Okba	Président de jury	*Professeur*	Université de Biskra
BOUMEHRAZ Mohamed	Rapporteur	*Professeur*	Université de Biskra
BENOUDJIT Nabil	Examinateur	*Professeur*	Université de Batna
GUESMI Kamel	Examinateur	*Maître de Conférences*	Université de Djelfa
GUESBAYA Tahar	Examinateur	*Maître de Conférences*	Université de Biskra
HAFAIFA Ahmed	Examinateur	*Maître de Conférences*	Université de Djelfa

Remerciements

Avant tous, Je remercie le **Dieu** le tout puissant de m'avoir donné le courage, la patience et la force durant toutes ces années d'étude.

Je tiens à remercier vivement mon encadreur Monsieur **Mohamed BOUMEHRAZ,** Professeur à l'université de Biskra pour m'avoir dirigé et soutenu au cour de la réalisation de ce travail. Son esprit scientifique et compréhensif qui m'a beaucoup aidé. Sa disponibilité et ses conseils ont été indispensable à la concaténation de ce travail.

J'ai été profondément honoré que Monsieur **O. KAZAR**, Professeur à l'université de Biskra, ait accepté de présider ce jury.

Je mesure tout l'honneur qu'à bien voulu me faire Monsieur **N. BENOUDJIT**, Professeur à l'université de Batna en acceptant d'examiner cette thèse. Qu'il trouve ici l'expression de mon profond respect.

Je tien à remercier vivement Monsieur **K. GUESMI,** Maître de conférences à l'université de Djelfa, qui m'ont fait l'honneur d'examiner mon travail.

Que Monsieur **T. GUESBAYA,** Maître de conférences à l'université de Biskra, trouve ici l'expression de mes sincères remerciements pour l'intérêt qu'il a porté à ce travail en acceptant de l'examiner.

Mes sincères remerciements vont aussi à Monsieur **A. HAFAIFA,** Maître de conférences à l'université de Djelfa, d'avoir accepté de faire partie de ce jury qu'il trouve ici l'expression de mon respect.

Que tous mes amis et collègues trouvent ici le témoignage de mon amitié et de ma reconnaissance.

Enfin, une reconnaissance et des remerciements particuliers à toute ma famille pour leur soutien permanant le long de mes études.

Dédicace

Je dédie ce travail:

A mes Parents, que le Dieu me les gardes,

A ma Femme et mon fils Imad-Eddine,

A mes frères et mes sœurs,

A la Mémoire de ma Grande Mère,

A tout les membres de ma Grande Famille

….

Table des matières

Chapitre I: Navigation Autonome d'un Robot Mobile

Chapitre II: Les Systèmes d'Inférence Flous

Chapitre III: Les Réseaux de Neurones Artificiels et l'Apprentissage par Renforcement

Chapitre IV: Navigation Autonome par des Contrôleurs Flous

Chapitre V: Navigation Autonome par des Contrôleurs Neuro-Flous

Chapitre VI: Navigation Autonome par des Contrôleurs Flous Entrainés par Apprentissage par Renforcement

Liste des figures

Liste des Tableaux

Liste des Algorithmes

Symboles et Abréviations:

SIF	Système d'Inférence Flou
RNA	Réseaux de Neurones Artificiels
AR	Apprentissage par Renforcement
ANFIS	Adaptive Network based Fuzzy Inference System
QLF	Q-Learning Flou
SARSA	State-Action-Reward-State-Action
TS(0)	Takagi-Sugeno d'ordre zéro
TD	Temporal Difference
(x_r, y_r)	Coordonnées du robot mobile
θ_r	Angle d'orientation du robot mobile
d_{rg}	Distance entre le robot et le but
θ_{rg}	Angle entre l'orientation actuelle du robot et celle de la cible
l	Longueur du châssis
D	Largeur du robot
α	Angle de braquage
V_r	Vitesse linéaire de déplacement
$\mu_{A_x}(x)$	Degré d'appartenance de la variable x
β_t	Cœfficient d'apprentissage
s_t	Situation perçue
a_t	Action choisie
r_t	Valeur de renforcement
V^{π}	Fonction valeur
$Q(s,a)$	Fonction qualité
w_{ij}	Valeur du poids synaptique
d_i	Distance mesurée à un obstacle
D_{min}	Distance minimale à un obstacle
D_{max}	Distance maximale mesurée

Liste des Travaux:

Publications:

L. Cherroun, R. Mechgoug and M. Boumehraz, "Path Following Behavior for an Autonomous Mobile Robot using Fuzzy Logic and Neural Networks", *Revue de Courrier du Savoir Scientifique et Technique, Université de Biskra*, vol. 12, pp. 63-70, 2011.

L. Cherroun and M. Boumehraz, "Designing of Goal Seeking and Obstacle Avoidance Behaviors for a Mobile Robot Using Fuzzy Techniques", *Journal of Automation and Systems Engineering (JASE)*, vol. 6, no. 4, pp. 164-171, 2012.

L. Cherroun and M. Boumehraz, "Path Following Behavior for an Autonomous Mobile Robot using Neuro-Fuzzy Controller", *Accepted in International Journal of Systems Assurance Engineering and Management, (IJSA),* Springer.

L. Cherroun and M. Boumehraz, "Fuzzy Logic and Reinforcement Learning based Approaches for Mobile Robot Navigation in Unknown Environment", *The Mediterranean Journal of Measurement and Control (MEDJMC),* vol. 9, no. 3, pp. 109-117, 2013.

L. Cherroun and M. Boumehraz, "Fuzzy Behavior Based Navigation Approach for Mobile Robot in Unknown Environment*", Journal of Electrical Engineering (JEE),* vol. 13, no. 4, pp. 284-291, 2013.

Conférences Internationales:

L. Cherroun and M. Boumehraz, "Tuning Fuzzy Controllers by Q-learning for Mobile Robot Navigation", *The International Conference on Electrical Engineering, Electronics and Automatics (ICEEA'10),* November 2010, Bejaia University, Algeria.

L. Cherroun, R. Mechgoug and M. Boumehraz, "Path Following Behavior for an Autonomous Mobile Robot using Fuzzy Logic and Neural Networks", *The Second International Conference on Image and Signal Processing and their Applications (ISPA'10),* December 2010, Biskra University, Algeria.

L. Cherroun, M. Boumehraz and R. Mechgoug, "Neuro-Fuzzy Controller for the Path Following Behavior and Moving Target Pursuing by a Mobile Robot", *The International Conference on Automation and Mechatronics (CIAM'11),* November 2011, Oran University, Algeria.

L. Cherroun and M. Boumehraz, "Intelligent Systems based on Reinforcement Learning and Fuzzy Logic Approaches, Application to Mobile Robotic", *The IEEE International Conference on Information Technology and e-Services (ICITeS'2012),* pp. 431-436, March 2012, Tunisia.

L. Cherroun and M. Boumehraz, "Path Following Behavior for an Autonomous Mobile Robot using Neuro-Fuzzy Controller", *The IEEE International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT'2012),* 2012, Sousse, Tunisia.

L. Cherroun and M. Boumehraz, "Fuzzy-Based Reactive Controllers for Simple Goal Seeking and Obstacle Avoidance by a Mobile Robot in unknown Environment", *The International Conference on Electrical Engineering (ICEE'2012),* pp. 119-124, May 2012, Algiers, Algeria.

Introduction Générale

Depuis quelques années, un intérêt croissant est porté, au sein de la communauté robotique, au développement des systèmes intelligents autonomes dans le cadre de la robotique mobile [CUE05]. Un tel intérêt peut être perçu comme une conséquence logique à l'apparition des applications potentiels des machines intelligentes (industriels, services, manutention ou encore de l'aide à la mobilité des personnes âgées ou handicapées,…). L'objectif est de mettre les robots dans des tâches variées telles que les interventions sur des sites accidentés, la manipulation sur sites sensibles ou nucléaires, ainsi pour l'exploration des sites maritimes et planétaires…etc.

L'enjeu principal de la robotique mobile actuelle consiste à développer des systèmes de navigation intelligents, où la navigation autonome est un axe de recherche qui vise à donner à une machine la capacité de se mouvoir dans un environnement sans assistance, ni intervention humaine pour accomplir un but désiré [LAT91, LAU01]. La tâche de navigation consiste à donner au robot la possibilité d'obtenir les informations dont il a besoin pour raisonner et le doter de capacité de locomotion adaptée à son environnement [BOR96]. Cependant, celle-ci implique des systèmes complexes dans la réalisation, où leurs maîtrises posent d'importants problèmes non seulement technologiques mais aussi scientifiques. Un robot mobile autonome est un système mécanique qui doit être en mesure de prendre des décisions pour effectuer des mouvements en fonction des informations sur sa position et sur l'environnement qu'il contourne. L'intérêt est d'atteindre un but désiré, tout en s'adaptant à certaines variations des conditions de fonctionnement sans intervention humaine [SHU06].

De manière générale, on regroupe sous l'appellation "*robots mobiles*" l'ensemble des robots à base mobile [FIL04]. Les autres robots sont distingués par le type de locomotion qu'ils soient: marcheurs, sous-marins ou aériens [BAY07].

La recherche en robotique mobile s'intéresse à la conception des systèmes intelligents dotés par des techniques de commande efficaces pour le déplacement d'un robot mobile, où la sécurité soit prioritaire par rapport à l'optimalité [CUE05, SHU06]. Au début, le système de navigation est disposé d'un modèle de l'environnement dans lequel sont représentés les principaux éléments fixes: obstacles, murs, portes, meubles…etc. Ces systèmes de navigation traditionnels [KHA86, BOR91] présentent certaines difficultés dans les applications temps réel puisqu'ils nécessitent une grande capacité de calcul et de mémoire, exigent une planification complète de l'environnement, et restent incompatibles avec les exigences des robots en temps réel en termes de rapidité et de réactivité. Cependant, lorsque l'environnement devient plus complexe (*i.e.* partiellement connu ou dynamique), il apparait indispensable que le robot mobile soit doté de capacités décisionnelles apte à le faire réagir automatiquement sans collision avec les objets imprévus, percevoir, analyser et modéliser son environnement. Ensuite, à partir de l'information disponible, le robot doit planifier sa trajectoire de mouvement. Finalement, le système de contrôle doit exécuter la séquence des actions élaborées.

D’autre part, les stratégies de commande réactives offrent des solutions intéressantes; qui utilisent directement l’information issue des capteurs du robot pour atteindre le but avec évitement des obstacles si l'environnement est inconnu ou dynamique, en se basant sur des techniques intelligentes [REI94, BEO95, SAF97, MAA00, ABD04, FAT06, LEF06].

Plusieurs approches ont été proposées pour résoudre le problème de navigation autonome d'un robot mobile, les plus utilisées sont celles de l'intelligence artificielle: la logique floue, les réseaux de neurones artificiels, l'apprentissage automatique et les systèmes hybrides. Dans l'approche floue, le comportement du robot est décrit par des règles linguistiques de type *Si-Alors* [SAF97, SER02, ZAV03, SIM04, YAN05, FAT06, WAN08]. Ces règles sont obtenues généralement à partir des connaissances d'un expert humain. Cependant, lorsque l'environnement est complexe et dynamique, il est difficile de construire une base de règles convenables parce que le nombre de situations à considérer est très élevé. D’autre part, le thème de l’apprentissage et l’adaptation est devenu très important dans les dernières années. Les techniques les plus utilisées sont celles qui font appel à des techniques d’auto-apprentissage [SUT98] et les systèmes neuronaux [AND95]. Les capacités d'apprentissage des réseaux de neurones peuvent être utilisées pour résoudre ce problème. L'idée de base est d'entraîner le robot mobile pour un ensemble de situations et la capacité de généralisation des réseaux de neurones lui permet de prendre les décisions adéquates dans le cas de nouvelles situations [THR95, FIE98, GAU99, JAN04]. L'apprentissage supervisé nécessite la disponibilité d'un nombre convenable de paires situation-action. Malgré sa convergence rapide, il est difficile d'obtenir un nombre suffisant de données pour l'apprentissage. D'autre part, l'apprentissage par renforcement semble être une solution prometteuse pour résoudre ce problème puisqu'il ne requiert pas la disponibilité de paires situation-action pour l'apprentissage [SUT98]. Au lieu de programmer un robot pour qu'il effectue une mission, on peut le laisser seul apprendre sa propre stratégie. La technique consiste à déterminer quelles sont les meilleures actions à réaliser. Si ces actions permettent au robot d’atteindre son objectif, leurs liens d’activation sont renforcés. L’idée fondamentale de l'apprentissage par renforcement est d’améliorer un comportement après chaque interaction avec l’environnement, de manière à maximiser une récompense numérique reçue. Il faut au début que le robot explore son environnement pour associer les situations et les actions à des récompenses positives (bonus) ou négatives (malus), et en déduire au fur et à mesure l’attitude à adopter pour maximiser sa récompense moyenne [WAT92, GLO99].

La commande floue a montré son efficacité pour la navigation des robots mobiles [SER02, YAN05, FAT06, WAN08], mais la construction d'un contrôleur flou performant pour un mouvement souhaité n'est pas toujours facile [MEN95, PAS98]. L'inconvénient majeur est le manque d'une méthodologie systématique pour la conception, due au nombre important de paramètres à régler (les paramètres des fonctions d'appartenances, les paramètres de la partie conclusion et les règles d'inférences). On trouve dans la littérature plusieurs méthodes de réglage des contrôleurs flous par l'intégration des propriétés des systèmes flous avec d'autres approches de l'intelligence artificielle, telles que: les réseaux de neurones [JAN93, GOD99], l'apprentissage par renforcement [BER92, GLO94, JOU98], les algorithmes génétiques [HER95, ABD04], colonie de fourmis [BOU09], ou bien les faires combinées [TOU97, CAN03, ZHO07]…etc. Ces méthodes dites hybrides combinent les propriétés de chaque approche afin d'optimiser les paramètres des systèmes d'inférence flous. Elles sont capables de générer une solution optimale ou quasi-optimale.

L'objectif de ce travail est d'étudier et d'appliquer des techniques neuro-floues pour la navigation autonome d'un robot mobile dans un environnement inconnu, afin de permettre au robot de se mouvoir d'une position initiale à une autre finale en évitant les obstacles. On utilise d'une part l'approche comportementale à base de la logique floue, et d'autre part le paradigme de l'apprentissage. Les techniques employées pour aborder ce problème sont basées sur les systèmes d'inférence flous, les réseaux de neurones artificiels et l'apprentissage par renforcement.

Cette thèse est organisée de la manière suivante:

Le premier chapitre est consacré à la navigation autonome d'un robot mobile. Un aperçu général sur le domaine de la robotique mobile sera abordé pour examiner en bref les types des robots mobiles, les différentes parties constitutives et les architectures de contrôle existantes pour un robot mobile.

Dans le deuxième chapitre, on présente les systèmes d'inférence flous (SIF). Nous commençons par énoncer les fondements théoriques des sous-ensembles flous et de la logique floue, suivi par la description de la structure générale d'un contrôleur flou, en montrant leurs types.

Dans le troisième chapitre, nous présentons une étude sur les réseaux de neurones artificiels (RNA) et l'apprentissage par renforcement (AR). La première partie est consacrée à une brève présentation des RNA, son principe de fonctionnement, les types existants et son apprentissage. Puis dans la deuxième partie de ce chapitre, nous présentons les fondements mathématiques de l'apprentissage par renforcement, les concepts de base, ensuite nous décrivons les principaux algorithmes.

Les trois derniers chapitres présentent nos contributions:

Dans le quatrième chapitre, nous décrivons des planificateurs locaux à base de la logique floue pour la navigation autonome d'un robot mobile. On a proposé des systèmes de commande réactifs basés sur les comportements, en décomposant la tâche globale en un ensemble des sous-tâches secondaires: convergence vers un but et évitement d'obstacles.

Le cinquième chapitre aborde l'application des contrôleurs neuro-flous pour la navigation autonome d'un robot mobile. Nous proposons des contrôleurs: flou, neuronal et neuro-flou de type ANFIS pour accomplir une tâche spécifique par le robot qui est la poursuite d'une trajectoire de référence.

Dans le dernier chapitre, on applique des contrôleurs flous entrainés par apprentissage par renforcement pour la navigation d'un robot. Tout d'abord, on utilise l'algorithme Q-learning standard avec la discrétisation des espaces d'état et d'action. Puis, une extension de cet algorithme pour le cas continu en utilisant les systèmes d'inférence floue sera présentée. C'est une méthode d'optimisation des systèmes flous de manière automatique. Ces systèmes de renforcement-flous seront utilisés pour les différents comportements d'un robot mobile: convergence vers un but, évitement d'obstacles, suivi de murs.

Le long des trois derniers chapitres, des exemples de simulations sont fournis afin de montrer les performances des méthodes proposées pour la navigation autonome d'un robot mobile.

Nous terminerons ce manuscrit par une conclusion générale récapitulant ce qui a été fait et expose les perspectives de ce travail.

Chapitre I

Navigation Autonome d'un Robot Mobile

I.1 Introduction

La robotique est un domaine de recherche important qui fait appel aux connaissances croisées de plusieurs disciplines. L'objectif est l'automatisation des systèmes mécaniques, en les dotant par des capacités de perception, de décision et d'action permettant au robot d'interagir rationnellement avec son environnement sans intervention humaine [LAU01]. La robotique mobile est un domaine dans lequel l'expérience pratique est primordiale. Au début, les robots font leur apparition dans l'industrie grâce aux capacités des manipulateurs: des bras imitant le bras humain et capables d'utiliser différents outils pour accomplir divers tâches. Au fur et à mesure, ces bras gagnent de nouveaux degrés de liberté à l'aide de plates formes mobiles; c'est l'apparition des robots mobiles à roues [FIL04]. Les premiers robots autonomes utilisaient la roue mais pour certaines applications où la géométrie de l'environnement se diffèrent (terrain accidenté, endroits difficiles à atteindre), les recherches ont orientés vers d'autres moyens de locomotion, inspirés du monde des animaux. Ainsi, les robots à pattes sont apparus et on connaît déjà des robots hexapodes ou des robots bipèdes. Des problèmes tels que la conception mécanique, la perception de l'environnement, la modélisation de l'espace de travail, la planification de trajectoires sans collision et la synthèse des lois de contrôle non linéaires sont des exemples de différents sujets de recherche dans la robotique mobile [SHU06]. L'intérêt indéniable de cette descipline est d'avoir permis d'augmenter considérablement nos connaissances sur la localisation et la navigation des systèmes autonomes. Cet axe de recherche étend le domaine d'application de la navigation autonome à toutes les sortes d'environnement non structurés, dù principalement à la nécessité d'aider ou de remplacer l'intervention humaine.

L'objectif de ce chapitre est de donner un bref exposé sur le domaine de la robotique mobile, en particulier la navigation autonome des robots mobiles à roues. Nous présentons l'autonomie du robot mobile, les grandes classes et les types des capteurs utilisés. On s'intéresse principalement aux méthodes de contrôle développées pour la commande de ces machines autonomes.

I.2 Autonomie d'un robot mobile

Le robot mobile est un agent physique réalisant des tâches dans son environnement, doté de capacités de perception, de décision et d'action. L'objectif est de permettre au robot d'interagir rationnellement avec son environnement automatiquement (sans intervention humaine) [LAT91]. Cette nouvelle machine est caractérisée par sa capacité à être programmée pour réaliser des tâches très diverses. Mettant en œuvre en particulier un ensemble des capteurs et un ensemble des actionneurs [BOR96]. Ses capacités en matière de manipulation d'objets lui ont permis de s'intégrer dans des lignes de production industrielle, où elle se substitue à l'homme dans les tâches difficiles, répétitives ou à risque pour l'être humain.

Le problème posé par l'étude de l'autonomie d'une machine peut être résumé en la détermination à chaque pas de temps la commande qui doit être envoyée aux actionneurs, connaissant d'une part la mission à accomplir et d'autre part l'acquisition des valeurs retournées par les différents capteurs. Il s'agit de déterminer un lien entre la perception et l'action connaissant les buts à atteindre. Nous considérons qu'un système est autonome si [CUE05]:

- Il est capable d'accomplir sans intervention humaine les objectifs pour lesquels il a été conçu,
- Il est capable de choisir ses actions afin d'accomplir ces missions.

Une machine autonome peut être définie comme étant l'association d'un système d'intelligence artificielle avec des capacités de perception, de modélisation de son environnement et de son propre état. Elle doit être aussi dotée de capacités d'actions sur son propre état et sur son environnement. Pour cela, le robot doit suivre le schéma correspondant au paradigme (*Percevoir-Décider-Agir*) [REI94]. La figure I.1(a) présente l'interaction du robot avec son environnement. La manière dont le robot mobile gère ces différents éléments est définie par son architecture de contrôle, qui peut éventuellement faire appel à un modèle interne de l'environnement ou une stratégie intelligente pour lui permettre de planifier ses actions à long terme.

Bien que, comme nous le verrons par la suite, plusieurs architectures existant au niveau de la satisfaction de ce paradigme, l'activité d'un tel robot se ramène aux tâches suivantes comme illustré sur la figure I.1(b):

- **Percevoir:** le robot doit acquérir des informations sur l'environnement dans lequel il évolue par l'intermédiaire de ses capteurs. Ces informations permettent de mettre à jour un modèle de l'environnement (architectures hiérarchiques ou délibératives) ou peuvent être directement utilisées comme entrées de comportement de bas niveau (architecture purement réactive);
- **Décider:** le robot doit définir des séquences d'actions résultant d'un raisonnement appliqué sur un modèle de l'environnement ou répondant de manière réflexe à des stimuli étroitement liés aux capteurs;
- **Agir:** il doit exécuter les séquences d'actions élaborées en envoyant des consignes aux actionneurs par l'intermédiaire des boucles d'asservissements.

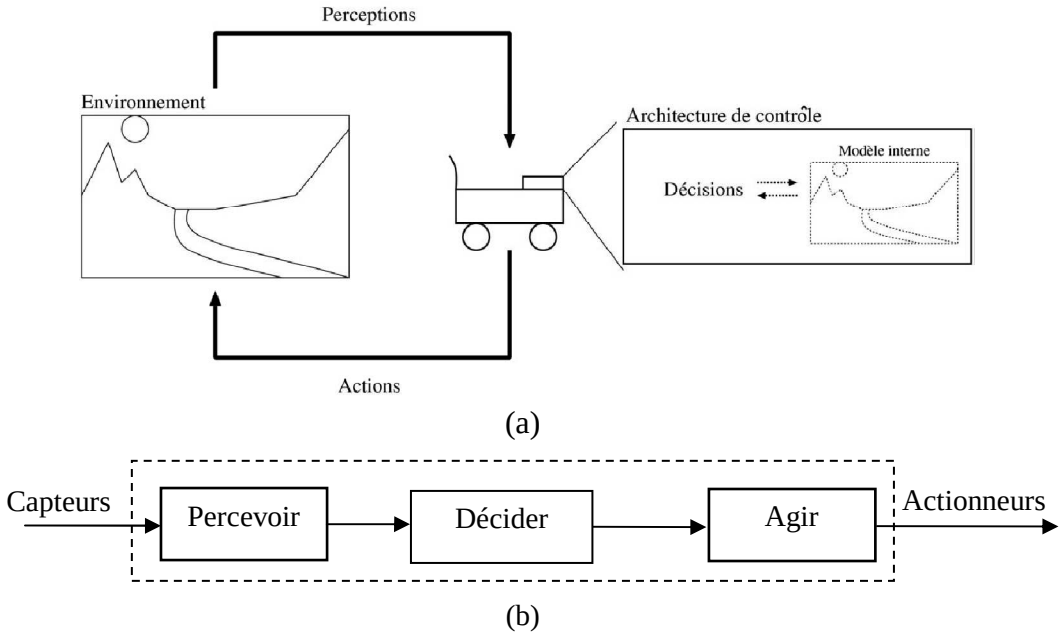


Figure I.1 (a) Interaction entre le robot et l'environnement, (b) Etapes de traitement automatique

I.3 Classification des robots mobiles

La caractéristique la plus remarquable d'un robot mobile est évidement son moyen de locomotion. Celui-ci dépend directement du type d'application visé ainsi que de type de terrain dans lequel le robot mobile doit évoluer (environnement d'interieur, extérieur, libre ou encombré d'obstacles,…). Les robots mobiles sont classés généralement selon le type de locomotion utilisé en quatre groupes distincts; qu'ils soit: à roues, à chenilles, à pattes ou avec d'autres moyens de locomotions [BOR96, CUE05]. En effet, le type de locomotion définit deux types de contraintes :

- *Les contraintes cinématiques*, qui portent sur la géométrie des déplacements possibles du robot dans l'environnement de navigation.
- *Les contraintes dynamiques*, liées aux effets du mouvement (accélérations, vitesses bornées, présence de forces d'inertie ou de frottement). Ces facteurs influent sur le mouvement exécuté.

Selon la cinématique, un robot est dit holonome, s'il peut se déplacer instantanément dans toutes les directions possibles. Il est dit non holonome, si ses déplacements autorisés sont des courbes dont la courbure est bornée.

Dans ce qui suit, on décrit brièvement les grandes classes des robots mobiles:

I.3.1 Robot mobile à roues

Compte tenu de la simplicité du mécanisme de locomotion utilisé, ce type de robot est le plus répondu actuellement. La plupart des robots mobiles à roues opèrent dans des sites aménagés, des sites industriels ou des environnements intérieurs; mais il existe également des applications en environnements extérieurs, comme l'exploration spatiale [FIL04]. La grande majorité des robots de ce type présente des contraintes de non holonomie qui limitent le mouvement instantané que le robot peut réaliser, car il existe pour toutes ces roues un point

unique (*centre instantané de rotation* (CIR)) de vitesse nulle autour duquel le robot tourne de façon instantanée. La commande se fait par la motorisation des roues installées. Ces contraintes augmentent la complexité du problème de planification de trajectoire et son contrôle [LAU01, LEF06]. Les robots mobiles à roues peuvent être classés en plusieurs types avec des propriétés intéressantes: unicycle, différentielle, tricycle, de type voiture, omnidirectionnelle et à traction synchrone [BAY07]. Les modèles des robots mobiles à roues existant sont illustrés sur les figures suivantes (figure I.2 au figure I.7).

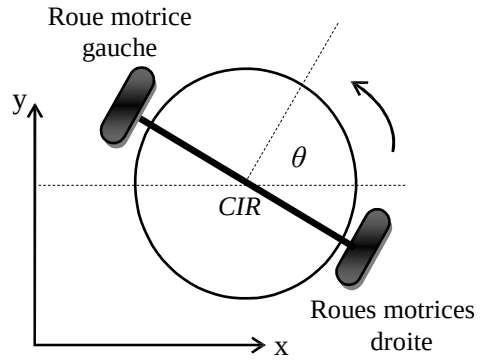


Figure I.2 Robot mobile de type unicycle

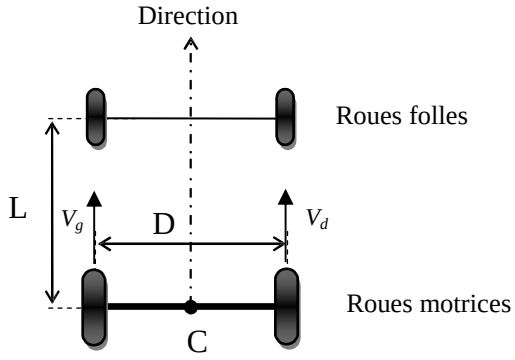


Figure I.3 Robot mobile différentielle

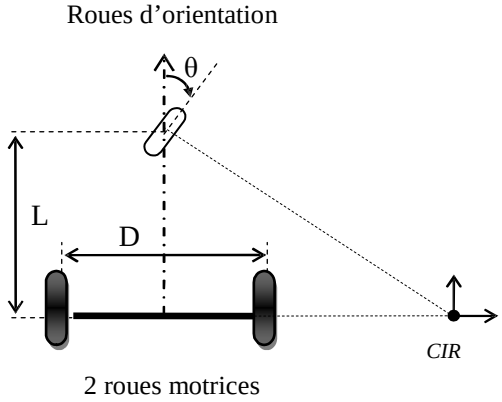


Figure I.4 Robot mobile de type tricycle

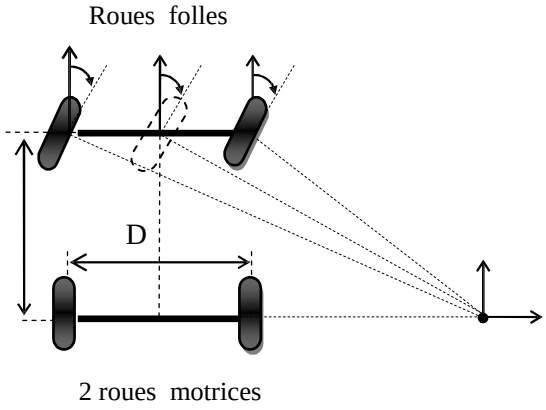


Figure I.5 Robot mobile de type voiture

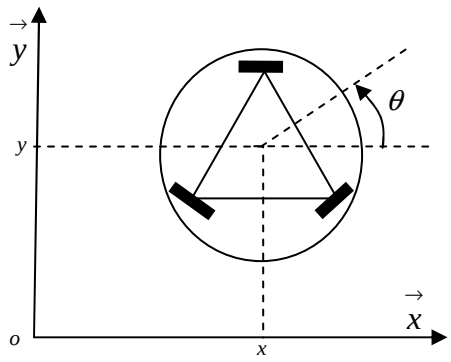


Figure I.6 Robot omnidirectionnel

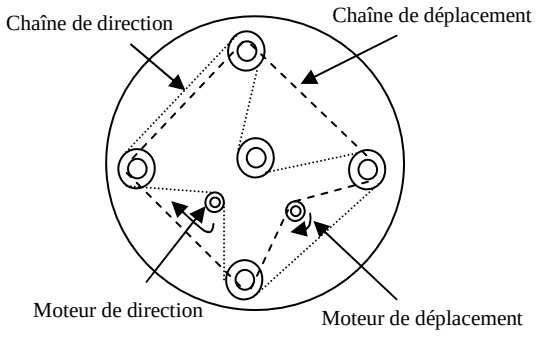


Figure I.7 Robot mobile à traction synchrone

I.3.2 Robot mobile utilisant la chenille

Lorsque le terrain est accidenté, les roues perdent leur efficacité de locomotion. Ceci limite la capacité de mouvement du robot mobile équipé de ce type de système de locomotion. Dans ces conditions, les chenilles sont plus intéressantes car elles permettent d'augmenter l'adhérence au sol et de franchir des obstacles plus importants [FIL04].

I.3.3 Robot mobile à pattes

Dans la situation ou le terrain est encore plus incertain, avec de grandes différences de hauteur comme par exemple un escalier ou un terrain très accidenté, les deux types précédents ne sont plus efficaces, et on fait recours aux robots mobiles à pattes. Ils ont des points d'appui discrets sur le terrain et sont donc la solution à ce problème de mouvement. Par contre, la conception et le contrôle d'un engin à pattes sont très complexes. En plus, la vitesse d'évolution est généralement très réduite. La commande est très difficile, dépond de la multiplicité des actionneurs utilisés. *Aibo* de Sony est un exemple d'un robot mobile à pattes [FIL04, BAY07].

I.3.4 Autres moyens de locomotion

Cette catégorie englobe les robots mobiles qui utilisent un moyen de locomotion différent des trois précédents. Par exemple, les robots mobiles qui se déplacent par reptation, les robots sous-marins, les robots d'exploration spatiale et les robots volants,…etc. Les applications et la commande de ces robots sont très spécialisées, l'architecture est en général spécifique à l'application visée [BAY07].

Pour utiliser et gérer ces machines d'une manière efficace, elles doivent être équipées par un ensemble de capteurs et d'actionneurs de réaction pour un mouvement souhaité.

I.4 Les capteurs comme sources d'informations

La commande des robots mobiles est basée sur deux types d'informations importantes; les informations proprioceptives et les informations extéroceptives [BOR96, SHU06]. Le système de perception est très important pour la sécurité du robot si l'environnement est encombré d'obstacles fixes ou bien mobiles (autres robot). Pour se focaliser sur le problème de navigation, nous allons nous restreindre dans ce chapitre aux capteurs utiles pour la tâche de navigation.

I.4.1 Capteurs intéroceptifs: fournissent des données sur l'état interne du robot (vitesse, position, orientation,…). Ces informations renseignent le robot en cas de mouvement, sur son déplacement dans l'espace (la localisation). Ce sont des capteurs que l'on peut utiliser directement, mais ils souffrent d'une dérive au cours du temps qui rend leur utilisation seul inefficace ou avec limitation. Nous citons par exemple: l'odomètre, radar doppler, systèmes inertiels,…[BOR96, FIL04].

I.4.2 Capteurs extéroceptifs: ont pour objectif d'acquérir des informations sur l'environnement proche du véhicule. Ils fournissent des mesures caractéristiques de la position que le robot peut acquérir dans son environnement par la détection des objets qui contourne. Ces informations peuvent être de natures très variée. Nous citons comme exemple les télémètres à ultrason, infrarouge, laser, les caméras,…etc.

Pour la navigation autonome d'un robot mobile et selon la mission visée (à accomplir), on peut utiliser aussi les capteurs suivants [BOR96, LAU00, FIL04] :

- *Les capteurs tactiles*: qui sont le plus souvent utilisés pour des arrêts d'urgence lorsqu'ils rencontrent un obstacle qui n'avait pas été détecté par le reste du système de perception.

- *Les boussoles:* permettant par la mesure du champ magnétique terrestre, de déduire la direction du nord. Ces capteurs peuvent utiliser différentes technologies et ont l'avantage de fournir une direction de référence stable au cours du temps.

- *Les balises:* dans certaines applications, il est également possible d'utiliser des balises dont on connaît la position, et qui pourront être facilement détectées par le robot, afin de faciliter sa localisation. Le robot sera alors équipé d'une antenne directionnelle qui lui permettra de détecter la direction des différentes balises afin de déduire sa position.

- Le GPS (*Global Positioning System*): est un système de balises universel dont les balises sont placées sur des satellites en orbite terrestre. Ce système permet donc d'avoir une mesure de la position dans un repère global couvrant la terre avec une précision variant de quelques dizaines de mètres à quelques centimètres suivant les équipements utilisés.

I.5 Navigation Autonome d'un robot mobile

La navigation d'un robot mobile est une tâche qui consiste, généralement, à trouver un mouvement libre dans l'espace de configuration (environnement de travail) sans collisions avec les obstacles proche du robot, qui est noté C_{libre} . L'espace de configuration est l'ensemble des paramètres caractérisant la position du robot dans son environnement (position et orientation). Ce mouvement amène le robot d'une configuration initiale $q_0 = q(t_0) = (x_0, y_0, \theta_0)$ à une autre finale désirée $q_f = q(t_f) = (x_f, y_f, \theta_f)$ [CAN03, CUE05, LEF06].

La figure I.8 présente une description de la tâche de navigation d'un robot mobile. Comme montré, le robot démarre d'une situation initiale *s*, il doit exécuter les actions de mouvement, qui sont généralement la vitesse et l'angle de braquage ($v(t)$ et $\alpha(t)$) lui permettant de se mouvoir vers une nouvelle situation *s(t+1)*. La navigation est obtenue à travers un processus itératif comme suit:

1. À chaque instant *t*, avec $t = 0,1,..., k,...$, le robot doit mesurer les distances aux obstacles de l'environnement d_i et les positions: courante et finale: $(x_r(t), y_r(t), \theta_r(t))^T$, $P_g(x_g, y_g)$;
2. Le système de contrôle détermine les variables de commande adéquate $v_r(t+1)$ et $\alpha(t+1)$,
3. Le robot exécute ces actions en déplaçant vers les nouvelles coordonnées,
4. Répéter le même processus (les étapes: 1, 2 et 3) de détection de la situation et génération des actions jusqu'à la destination finale appelée but.

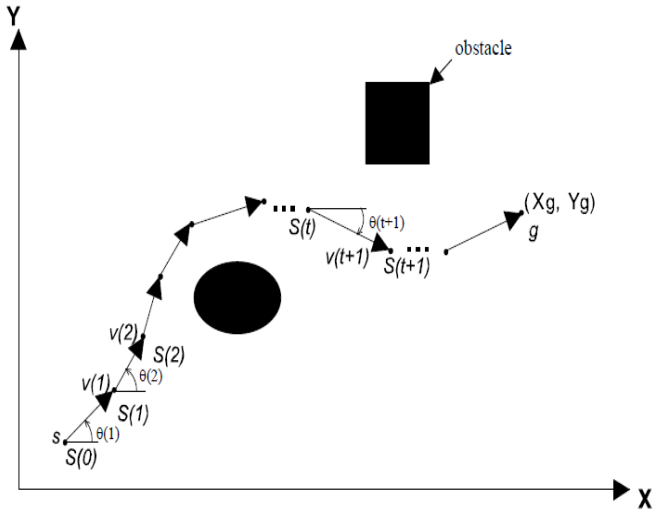


Figure I.8 Processus de navigation autonome

On peut donner des définitions différentes de la tâche de navigation selon le but recherché. Il existe néanmoins une très grande variété de travaux et des méthodes permettant d'aborder ce problème difficile. Les stratégies de navigation permettant à un robot mobile de se déplacer pour rejoindre un but sont extrêmement diverses. Pour exécuter une tâche de navigation autonome, un robot mobile doit mettre en œuvre un certain nombre de fonctionnalités, parmi les quelles [SHU05, BAY07]:

I.5.1 Planification de mouvement

L'un des principaux objectifs du robot mobile est de pouvoir évoluer dans un environnement complexe encombré d'obstacles pour atteindre son but final [HAC08]. Il a besoin de construire une trajectoire définie comme une séquence de déplacement sans collision avec ces obstacles entre la position initiale (point de démarrage) et le point but ou cible (figure I.8). La planification de trajectoire dans sa formulation classique est le problème du calcul de ce chemin, dans un modèle géométrique de l'environnement cela est fait en introduisant le concept d'espace des configurations qui permet de transformer le problème de la recherche d'un chemin pour un système à *n* degrés de liberté dans l'espace euclidien en celui du mouvement d'un point dans un espace à *n* dimensions où le robot est représenté par un point. Plusieurs approches sont proposées pour la planification de trajectoire. Cependant, les plus utilisées sont la planification globale et locale [LAT91].

- ***Planification globale de trajectoire:*** c'est la modélisation de l'espace de l'environnement par un graphe, ou la recherche de la trajectoire est basée sur l'utilisation des algorithmes des graphes, on peut citer: le graphe de visibilité, la décomposition cellulaire…etc [FIL04, SHU05].
- ***Planification locale de trajectoire:*** Généralement, l'environnement du robot mobile est inconnu, et le robot ne dispose pas, à priori, aucune information sur l'environnement. Il est nécessaire donc de réaliser une planification locale de type réflexe. Pendant le déplacement, le robot mobile doit analyser son environnement et prendre la décision en fonction de cette analyse [KHA89, BOR91, FIL04]. Les méthodes réactives de l'intelligence artificielle sont considérées comme des approches de planification locale.

I.5.2 Localisation

Afin d'exécuter le mouvement planifié, le robot doit se localiser dans son environnement en estimant la position et l'orientation par rapport à un repère fixe. L'estimation peut s'effectuer soit par une mesure des déplacements du robot, soit par une mesure de sa position absolue dans l'environnement. Plusieurs approches de localisation sont utilisées en robotique mobile. Elles peuvent être classées en trois catégories: localisation basée sur une carte, localisation par rapport à des balises et localisation par rapport à d'autres robots [BOR96].

Du fait des incertitudes sur les mesures utiles pour la localisation, elle est généralement modélisée dans un cadre probabiliste. C'est ainsi que les méthodes de localisation se regroupent en deux catégories, soit :

- ***Localisation à l'estime*** ou relative: obtenue par des informations issues des capteurs proprioceptifs, et consiste à déterminer la variation des coordonnées de position lors d'un déplacement en mesurant tout simplement les distances parcourues et les directions empruntées depuis sa position initiale.
- ***Localisation absolue:*** obtenue par des informations issues de capteurs extéroceptifs; le robot doit toujours connaître sa situation pour se déplacer d'un point à un autre en identifiant des repères artificiels, la méthode des balises est la plus employée.

I.5.3 Suivi de trajectoire

Le suivi de trajectoire est une mission importante pour un robot mobile. Il consiste à calculer les commandes envoyées aux actionneurs permettant de réaliser le mouvement planifié. Un robot est un système dynamique commandé par bouclage pour la poursuite de sa trajectoire de référence [FIL04, CHE11]. On s'intéresse dans ce travail à l'étude de cette fonctionnalité.

I.5.4 Évitement d'obstacles

Le suivi de la trajectoire planifiée ne permet pas de garantir l'absence des collisions avec les objets statiques ou dynamique existants. L'évitement d'obstacles est un comportement de base présent dans quasiment tous les mouvements des robots mobiles [BOR89, CAN03, WAN03]. Ces collisions peuvent se produire lors de l'exécution de la trajectoire, dues à une localisation imparfaite, un plan imprécis ou des obstacles qui n'étaient pas dans le modèle de l'environnement utilisé pour la planification de trajectoire. Le robot mobile autonome doit avoir une capacité d'évitement d'obstacles efficace.

I.5.5 Parking

Le parking est la phase finale de la navigation autonome, car l'objectif d'une mission de navigation est souvent d'atteindre une configuration finale spécifiée avec une grande précision, Le succès de navigation dépend de la réalisation de cet objectif de stationnement.

Tous ces éléments ou tâches font que le mouvement initialement planifié doit être adapté lors de son exécution et que des stratégies d'évitement réactives d'obstacles doivent être mises en œuvre. On adoptera des stratégies différentes en fonction du type de système, de sa vitesse, et de champ d'application. Après cet aperçu sur les éléments constitutifs et nécessaires pour faire naviguer un robot mobile dans son environnement. Il s'agit maintenant d'utiliser au mieux la motricité du robot et sa localisation pour accomplir la tâche de navigation autonome; le robot

doit être doté d'un système de contrôle adéquat. Pour cela, dans ce qui suit, on présente quelques types des systèmes de contrôle existant.

I.6 Architectures de contrôle des robots mobiles

Un robot est un système complexe qui doit satisfaire des exigences variées en utilisant un ensemble logiciel appelé architecture de contrôle du robot. Cette architecture permet donc d'organiser les relations entre les trois grandes fonctions: la perception, la décision (planification) et l'action. Ces architectures peuvent néanmoins être classées en trois groupes (approches) [REI96, FIL04] comme montré sur la figure I.9.

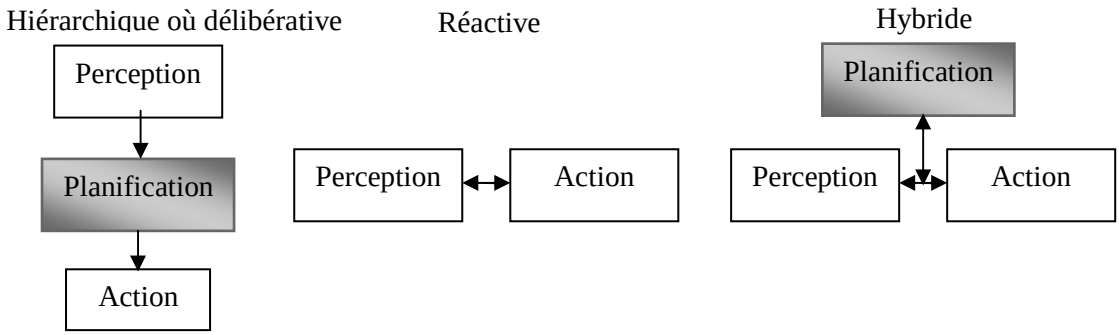


Figure I.9 Architectures de contrôle pour les robots mobiles

I.6.1 Approche délibérative (penser puis agir)

Les premiers robots mobiles dérivés des recherches en intelligence artificielle utilisaient des contrôleurs hiérarchiques, dont le fonctionnement repose essentiellement sur la disponibilité d'un modèle de l'environnement permettant de représenter toutes les informations pertinentes pour le déplacement du robot [AZO06]. Elles ont l'avantage de prouver l'existence d'une solution optimale permettant au robot d'atteindre le but assigné. Mais un tel modèle peut être insuffisant dans un environnement inconnu ou dynamique car au moment de la réalisation de l'action, l'environnement peut avoir changé et la décision n'est plus valide. Le traitement est décomposé en une série d'opérations successives comme décrit sur la figure I.10.

1. Traiter les données sensorielles qui fournissent au robot des informations sur son environnement,
2. Construire ou à mettre à jour, à partir des données acquises, un modèle du monde dans lequel le robot évolue. Ce modèle peut être par exemple une identification du type et de la position des obstacles que le robot doit éviter,
3. Ce modèle est utilisé par la suite pour planifier une suite d'états permettant au robot mobile d'effectuer la tâche visée,
4. Calculer puis d'exécuter les actions afin de suivre le plan généré par l'étape précédente.

Dans ce type d'architecture, le robot utilise toutes les informations sensorielles disponibles et toutes les connaissances internes sauvegardées et raisonne sur les prochaines actions à réaliser. Le robot doit construire et ensuite évaluer tous les plans possibles pour trouver celui qui atteint le but. La planification requiert l'existence d'une représentation interne du monde qui permet au robot d'avoir une idée de futur et de prédire les résultats d'actions possibles dans les différents

états perçus. Le modèle interne doit donc être précis et récent. Quand le temps est suffisant afin de générer un modèle du monde précis, cette approche permet au robot d'agir stratégiquement en sélectionnant la meilleure action pour une situation donnée.

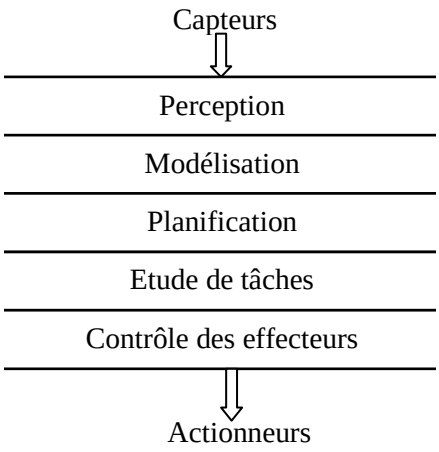


Figure I.10 Architecture hiérarchique

La partie la plus importante est celle qui concerne la *modélisation* et la *planification* qui peuvent prendre un temps assez long, et ceci notamment en environnement peu structuré. En cas d'insuffisance du modèle ou capacité de traitement, cette approche ne peut être appliquée. Les écarts entre le modèle et l'environnement ne peuvent être pris en compte que via un nouveau cycle "*perception-modélisation-planification*", ce qui est peut réactif et nécessite l'utilisation de nouvelles méthodologies assez puissantes. Ces méthodes doivent établir un couplage entre la perception et l'action et le traitement local des différentes tâches [REI94].

I.6.2 Approche réactive (ne pas penser mais agir)

En 1986, Brooks [BRO86] a proposé une approche réactive qui se distingue par l'abandon des phases de *modélisation* et de *planification*. Le contrôle du robot se fait alors sans utiliser le modèle de l'environnement (figure I.11). Les stratégies de navigation réactives n'utilisent que les valeurs courantes des capteurs et non des données provenant d'un modèle interne, pour décider de l'action à effectuer. Le principe de l'approche réactive est basé sur la décomposition de la tâche de navigation en un ensemble de comportements actifs de base (explorer, aller au but, éviter les obstacles, suivi des murs,…). Pour guider le robot, il faut donc choisir à chaque instant lequel de ces comportements est à activer?. Ce problème est connu dans la littérature scientifique sous le nom de *sélection de l'action*. La solution proposée par Brooks est de diviser la tâche globale en un ensemble des sous tâches secondaires. Cette architecture est appelée "*subsumption*", utilise une hiérarchie des comportements qui se déclenchent donc selon un ordre de priorité en fonction des données de perception (fusion des comportements).

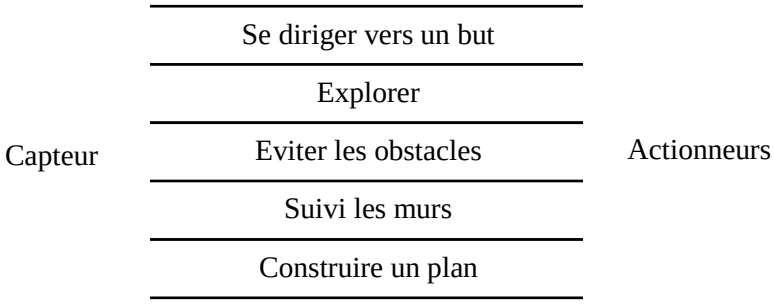


Figure I.11 Architecture réactive

La figure I.12 décrit les différentes couches comportementales d'un robot en utilisant la structure subsumption. Cette architecture peut être définie comme une hiérarchie de comportements plutôt que comme une hiérarchie fondée sur une abstraction des données. Le concept de base est le suivant: si complexe soit-il, peut être décomposé en comportements élémentaires représentant chacun un niveau de compétence. Ces différents niveaux sont placés en couches parallèles et chaque compétence couple directement la perception avec l'action. Une telle architecture a, au moins, trois avantages:

- D'abord, elle peut réagir aux éventualités en temps réel dû au parallélisme,
- Chaque comportement définit une tâche indiquée ce qui facilitera leur contrôle et modification, par exemple: ajouter ou enlever un comportement,
- Elle présente une bonne robustesse; le système peut fonctionner même si un ou plusieurs comportements échouent. Elle apporte donc l'avantage d'avoir une trajectoire qui s'adapte aux changements de l'environnement.

Cette architecture est une approche purement réactive vient de son manque de capacités de raisonnement de haut niveau et de modularité [BRO86, HOF00, FAT06].

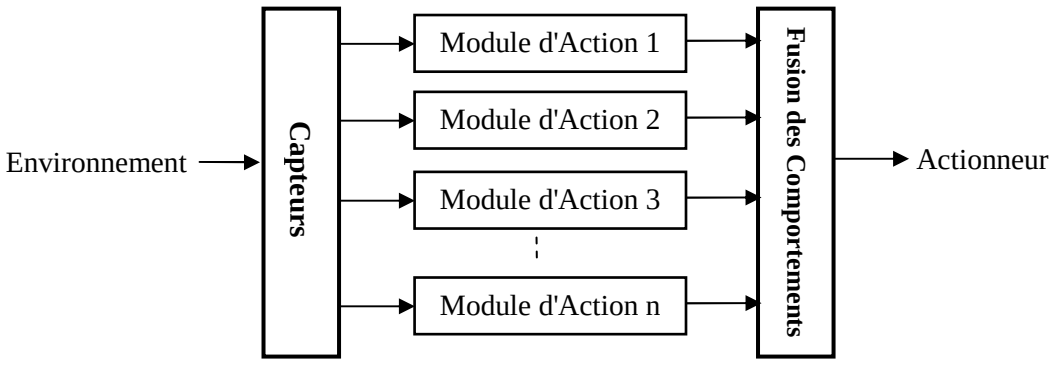


Figure I.12 Architecture modulaire de Subsumption

En effet, sans représentation interne de l'état de l'environnement, il est très difficile de planifier une suite d'actions en fonction d'un but à atteindre. Les robots utilisant cette architecture sont donc en général efficaces pour la tâche précise pour laquelle ils ont été conçus et dans l'environnement pour lequel ils ont été prévus, mais sont souvent difficiles à s'adapter à une tâche différente. Puisque, le robot n'a qu'une vue très réduite de son environnement, ces architectures ne peuvent garantir le succès de la mission en toute circonstance. Les réussites de ces

architectures sont liées au couplage direct entre la perception et l'action qui permet une prise en compte très rapide des phénomènes dynamiques de l'environnement et donc une robustesse et fiabilité dans les situations complexes.

I.6.3 Approche hybride (penser et agir indépendamment en parallèle)

Cette architecture combine les meilleurs aspects des architectures réactives et délibératives: elle combine la réponse temps-réel de la réactivité avec la rationalité et l'efficacité de la délibération [FIL04]. En effet, la composante réactive s'occupe des besoins immédiats du robot comme l'évitement d'obstacles et fonctionne ainsi en un temps court et utilise des données et signaux externes directs, alors que la composante délibérative utilise des représentations internes, abstraites et symboliques du monde et fonctionne en un temps très long. Tant que les sorties des deux composantes ne sont pas en conflit, le système n'a pas besoin de plus de coordination. Cependant, ces deux parties interagissent pour tirer les avantages de chacune. La composante réactive doit outrepasser la composante délibérative si le monde présente un certains défit immédiat et imprévisible et la partie délibérative doit informer la partie réactive afin de guider le robot vers des trajectoires et des buts plus efficaces. L'interaction des deux parties de l'architecture requiert une composante intermédiaire dont sa construction est particulièrement le plus grand défi dans la conception hybride comme illustré sur la figure I.9. Ainsi, les architectures hybrides sont souvent appelées "architectures à trois couches", composées des couches réactive, intermédiaire et délibérative. Un grand effort de recherche a été conduit dans la manière de concevoir ces composantes et leurs interactions.

I.6.4 Approche collective

En cas d'application qui nécessite l'utilisation de plusieurs robots mobiles (la robotique collective), l'architecture de contrôle comportementale collective tire son inspiration de la biologie et essaie de modéliser le comportement des animaux dans leurs environnements complexes d'évolution [FER95, AZO06]. Les composantes de base de ces architectures sont des comportements observés d'activité émergeant des interactions entre le robot et son environnement. De tels systèmes sont construits d'une façon ascendante commençant par un ensemble de comportements de survie tel que l'évitement de collisions qui couple les entrées perçues aux actions du robot. Des comportements sont ajoutés pour fournir au robot plus de capacités lui permettant de réaliser des tâches de plus en plus complexes telle que le suivi de mur, la poursuite d'une cible, l'exploration ou le "homing" (rentrez chez-soi). Ces nouveaux comportements sont introduits dans le système incrémentale des plus simples aux plus compliqués jusqu'à ce que leurs interactions aboutissent aux capacités désirées du robot. Comme les architectures hybrides, les architectures comportementales peuvent être organisées en couches mais à l'inverse de ces architectures, les couches ne diffèrent pas beaucoup les unes des autres en termes de temps et représentation utilisées. Toutes les couches sont représentées par des comportements, des processus qui récoltent des entrées et s'envoient des sorties.

I.7 Conclusion

La robotique est la branche de l'intelligence artificielle concernée par l'étude des systèmes automatiques capables d'interagir directement avec le monde physique. C'est une automatisation de ses machines, ou l'objectif est d'augmenter les capacités de localisation et de navigation dans son espace de travail. Ce chapitre nous a permis de présenter un exposé sur le domaine de la robotique mobile, en définissant la navigation autonome des robots mobiles, les types des robots et les capteurs utilisés. On a présenté aussi les principales architectures utilisées pour le contrôle des robots mobiles en citant les avantages et les inconvénients de chacune.

Les robots mobiles à roues sont les robots mobiles les plus répandus, à cause de ses structures mécanique simple et ses commandes relativement plus facile que les autres robots qui diffèrent par leur moyen de locomotion. La commande d'un robot mobile se divise généralement en trois étapes principales: perception, décision et action. La dernière étape concerne l'exécution des mouvements planifiés, c'est une étape qu'il doit maîtriser efficacement pour accomplir ses missions avec succès. Dans notre travail, nous utilisons un modèle de ce type des robots pour tester les structures de commande étudiées.

Après cet aperçu, dans le chapitre suivant nous présenterons une étude générale sur les systèmes d'inférences flous et les fondements théoriques des ensembles flous. Nous décrivons la structure générale d'un contrôleur flou avec ces composants pour l'utiliser par la suite au problème de navigation autonome d'un robot mobile.

Chapitre II

Les Systèmes d'Inférence Flous

II.1 Introduction

Le nombre de travaux de recherche sur le développement des contrôleurs par la logique floue ainsi que le nombre d'applications industrielles de la commande floue a augmenté exponentiellement dans les trois dernières décennies. Les bases théoriques de la logique floue ont été réellement introduites en 1965 par le professeur Lotfi A. Zadeh de l'université de Berkeley en Californie [ZAD65]. Il a introduit le concept des sous-ensembles flous pour approcher le raisonnement humain à l'aide d'une représentation adéquate des connaissances imprécises, incertaines ou vagues. En 1974, Mamdani [MAM74] était le premier à étudier la possibilité de commander un système dynamique par des règles d'inférence floues. Puis un an après, Mamdani et Assilian [MAM75] développaient le premier contrôleur par logique floue pour la commande d'une turbine à vapeur. Les premières applications industrielles et commerciales ont vu le jour au Japon presque dix ans après [SUG84, KIS85]. Depuis, plusieurs entreprises ont fabriqué des produits destinés au grand public utilisant la technologie de la commande floue. Les premiers travaux de Mamdani introduisaient la méthode de raisonnement flou, nommée le raisonnement min-max de Zadeh-Mamdani. En 1985, Takagi et Sugeno [TAK85] introduisaient une description différente des conclusions des règles floues. Ces travaux ont donné naissance à deux grandes classes de contrôleurs flous: les contrôleurs flous de Mamdani et les contrôleurs flous de Takagi-Sugeno (TS). Ils diffèrent essentiellement dans les conclusions des règles floues: le premier utilise des sous-ensembles flous et le second utilise des fonctions scalaires.

Dans ce chapitre, nous exposons en bref un aperçu général sur la logique floue, les fondements mathématiques de la théorie des sous-ensembles flous et les systèmes d'inférence flous. Ensuite, nous présenterons la structure des contrôleurs flous.

II.2 La logique floue et l'ensemble flou

La logique floue peut être vue comme une extension de la logique booléenne; de plus, elle permet de traiter des variables linguistiques dont les valeurs sont des mots ou expressions du langage naturel. Dans cette section, on présente les propriétés de base des ensembles classiques et des ensembles flous [BUH94, PAS98, GOD99].

II.2.1 Concept de base d'un ensemble flou

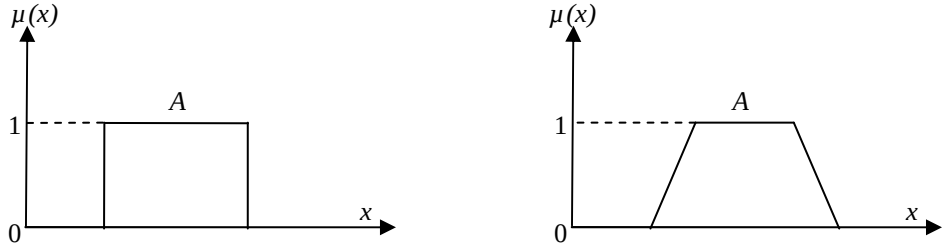
Un ensemble classique (figure II.1 (a)) *A* de *U* est défini par une fonction caractéristique notée $\mu_A(x)$ telle que:

$$\mu_A(x) = \begin{cases} 1, x \in A \\ 0, x \notin A \end{cases} \quad \text{(II.1)}$$

Le concept de base de la théorie des ensembles flous est la notion de sous-ensemble flou. Cette notion provient du constat que «très souvent, les classes d'objets rencontrés dans le monde physique ne possèdent pas de critères d'appartenance bien définis».

Soit *U* une collection d'objets (par exemple $U = R^n$) appelée univers de discours. Un ensemble flou *A* dans *U* est caractérisé par une fonction d'appartenance, notée: $\mu_A(x)$, $(\mu_A : U \to [0,1])$, qui appliquée à un élément *x* de *U*, retourne un degré d'appartenance $\mu_A(x)$ de *x* à *A* (figure II.1 (b)). Un ensemble flou peut être considéré comme une généralisation de l'ensemble classique. La fonction d'appartenance d'un ensemble classique peut prendre seulement deux valeurs $\{0,1\}$. Un ensemble flou peut être représenté comme un ensemble de paires ordonnées:

$$A = \{x, \mu_A(x) / x \in U\} \quad \text{(II.2)}$$



(a) (b)

Figure II.1 Exemples des fonctions d'appartenance
(a) Logique classique (b) Logique floue

Si *U* est discret, *A* est représenté par :

$$A = \sum_{x_i} \mu_A(x_i) / x_i \quad \text{(II.3)}$$

Si *U* est continu, A est représenté par :

$$A = \int_x \mu_A(x_i) / x_i \quad \text{(II.4)}$$

Les ensembles flous ont le grand avantage de constituer une représentation mathématique des termes linguistiques largement utilisés dans le langage naturel des experts.

II.2.2 Fonctions d'appartenance

Afin de permettre un traitement numérique des variables linguistiques dans la prise de décision pour le calcul, une définition des variables linguistiques à l'aide des fonctions d'appartenance s'impose. Dans ce contexte, on associe à chaque valeur de la variable linguistique une fonction d'appartenance désignée par $\mu_A(x)$ où x est la variable linguistique, tandis que *A* indique l'ensemble concerné. Une valeur précise de $\mu_A(x)$ sera désignée par le degré ou le facteur d'appartenance. Le plus souvent, on utilise pour les fonctions d'appartenance les fonctions suivantes (figure II.2) [BUH94, WAN94, PAS98]:

II.2.2.1 ***Fonction triangulaire:*** elle est définie par trois paramètres $\{a,b,c\}$ qui déterminent les coordonnées des trois sommets (figure II.2 (a)).

$$\mu_A(x) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \tag{II.5}$$

II.2.2.2 ***Fonction trapézoïdale:*** elle est définie par quatre paramètres $\{a,b,c,d\}$ (figure II.2 (b)):

$$\mu_A(x) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-b}\right), 0\right) \tag{II.6}$$

II.2.2.3 ***Fonction gaussienne:*** elle est définie par deux paramètres $\{\sigma, m\}$ (figure II.2 (c)):

$$\mu_A(x) = \exp\left\{-\frac{(x-m)^2}{2\sigma^2}\right\} \tag{II.7}$$

II.2.2.4 ***Fonction sigmoïde:*** une fonction sigmoïde est définie par deux paramètres $\{a,c\}$ (figure II.3 (d)):

$$\mu_A(x) = \frac{1}{1+\exp\{-a(x-c)\}} \tag{II.8}$$

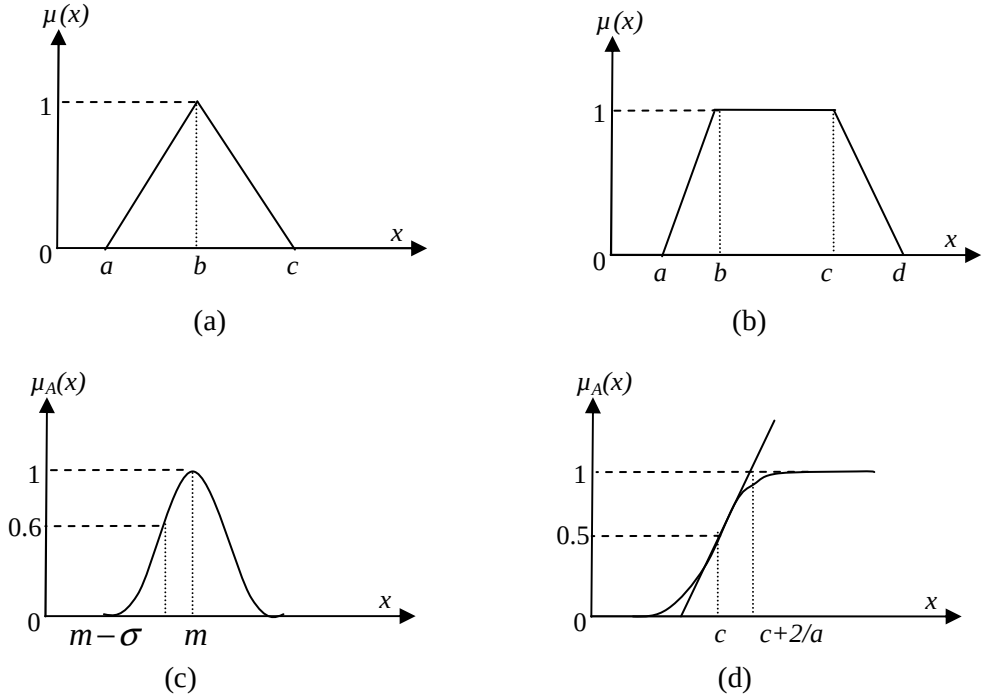


Figure II.2 Formes usuelles des fonctions d'appartenance

II.2.3 Variable linguistique

C'est une variable dont les valeurs ne sont pas des nombres, mais des mots ou phrases exprimés en langage naturel. La raison pour laquelle on utilise cette représentation, est que le caractère linguistique est moins spécifique que le caractère numérique. Une variable linguistique est généralement représentée par un triplet $(x, T(x), U)$ dans lequel:

- x est le nom de la variable linguistique (distance, angle, erreur,...),
- $T(x)$ est l'ensemble des valeurs linguistiques qui sont utilisées pour caractériser x,
- U est l'univers de discours de la variable linguistique x.

Par exemple, si la distance est considérée comme une variable linguistique définie sur l'univers de discours $U = [0, D_{max}]$, ses valeurs linguistiques peuvent être définies comme suit:

T(Distance)= {Zéros(Z), Petite (P), Moyenne (M), Grande (G), Très Grande (TG)}.

Les symboles linguistiques peuvent être considérés comme des ensembles flous dont les fonctions d'appartenance sont représentées sur la figure II.3.

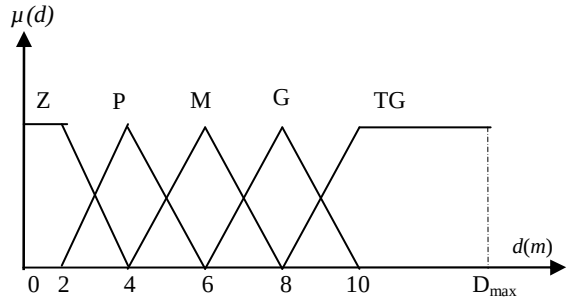


Figure II.3 Représentation de la variable linguistique (distance)

II.2.4 Opérations sur les ensembles flous

Les principaux opérateurs dans la théorie des ensembles classiques sont: le complément, l'intersection et l'union. Pour deux ensembles classiques *A* et *B* d'un univers *U*, nous avons [BUH94, MEN95, PAS98]:

- Le complément de *A*, noté $\overline{A}$:

$$\overline{A} = \{x / x \notin A\} \tag{II.9}$$

- L'intersection de *A* et *B*, noté $A \cap B$:

$$A \cap B = \{x / x \in A \wedge x \in B\} \tag{II.10}$$

- L'union de *A* et *B*, noté $A \cup B$:

$$A \cup B = \{x / x \in A \vee x \in B\} \tag{II.11}$$

La représentation formelle des ensembles flous par des fonctions d'appartenance a permis de généraliser les opérateurs des ensembles classiques au cas flou.

Soit *A* et *B* deux ensembles flous définis dans l'univers de discours *U* par les fonctions d'appartenance $\mu_A(x)$ et $\mu_B(x)$ respectivement.

Les opérateurs de la logique floue sont définis comme suit :

1. ***Egalité floue:*** Deux ensembles flous *A* et *B* sont égaux $(A = B)$ si et seulement si:

$$\forall x \in U, \mu_A(x) = \mu_B(x) \tag{II.12}$$

2. ***Sous-ensemble flou:*** *A* est un sous ensemble de *B* $(A \subseteq B)$ si et seulement si:

$$\forall x \in U, \mu_A(x) \leq \mu_B(x) \tag{II.13}$$

3. ***Complémentation floue:*** Le complément de *A* est un ensemble flou dans *U*, noté $\overline{A}$, donné par :

$$\forall x \in U, \mu_{\overline{A}}(x) = 1 - \mu_A(x) \tag{II.14}$$

4. ***Union floue:*** L'union de deux ensembles flous *A* et *B* est un ensemble flou $(A \cup B)$ de fonction d'appartenance:

$$\forall x \in U : \mu_{A \cup B} = \mu_A(x) + \mu_B(x) \quad \text{(II.15)}$$

Le symbole «+» représente la *co-norme* triangulaire. La co-norme triangulaire la plus utilisée dans le domaine de la commande est :

$$\mu_A(x) + \mu_B(x) = \max(\mu_A(x), \mu_B(x)) \quad \text{(II.16)}$$

5. ***Intersection floue:*** l'intersection de deux ensembles flous *A* et *B* est un ensemble flou ($A \cap B$) de fonction d'appartenance:

$$\forall x \in U : \mu_{A \cap B} = \mu_A(x) * \mu_B(x) \quad \text{(II.17)}$$

Le symbole «*» représente la norme triangulaire [PAS98]. Les normes triangulaires les plus utilisées dans le domaine de la commande sont :

$$\mu_A(x) * \mu_B(x) = \min(\mu_A(x), \mu_B(x)) \quad \text{(II.18)}$$

$$\mu_A(x) * \mu_B(x) = \mu_A(x) \times \mu_B(x)) \quad \text{(II.19)}$$

6. ***Produit cartésien:*** Si $A_1,..., A_n$ sont des ensembles flous respectivement définis sur $U_1,..., U_n$, leur produit cartésien est un ensemble flou, dénoté par $A_1 \times ... \times A_n$, défini sur le produit $U_1 \times ... \times U_n$ avec la fonction d'appartenance:

$$\mu_{A_1 \times .. \times A_n}(x_1,..., x_n) = \mu_{A_1}(x_1) * \cdots * \mu_{A_n}(x_n) \quad \text{(II.20)}$$

7. ***Relation floue:*** Une relation floue représente le degré de présence, ou d'absence d'une association entre les éléments de deux ou de plusieurs ensembles flous. Une relation floue d'ordre *n* est un ensemble flou défini sur $U_1,..., U_n$ par l'expression suivante:

$$R_{X_1 \times .. \times X_n} = \left\{ ((U_1,...,U_n), \mu_R(x_1,...,x_n)) \middle| (x_1,...,x_n) \in U_1 \times ... \times U_n \right\} \quad \text{(II.21)}$$

8. ***Composition des relations floues:*** Soit *R* et *S* deux relations floues définies respectivement dans $X \times Y$ et $Y \times Z$. La composition de *R* et *S* est un ensemble flou, symbolisé par $R \circ S$, de fonction d'appartenance:

$$\mu_{R \circ S}(x, z) = \left\{ (x, z), \sup_{y \in Y} (\mu_R(x, y) * \mu(y, z)) \right\} \quad \text{(II.22)}$$

9. ***Implication floue:*** L'implication floue est un opérateur qui permet d'évaluer le degré de vérité d'une règle de la forme**:**

Si *x est A* Alors *y est B*

À partir des valeurs de la prémisse d'une part, et de celle de la conclusion d'autre part. Ce degré de vérité est évalué à partir des degrés d'appartenance de *x* à *A* et de *y* à *B* comme suit :

$$\mu_R(x, y) = imp(\mu_A(x), \mu_B(x)) \quad \text{(II.23)}$$

Les opérateurs les plus utilisés en commande floue sont les implications de Mamdani et de Larsen :

- Implication de Mamdani: $\mu_R(x, y) = \min(\mu_A(x), \mu_B(x))$ (II.24)
- Implication de Larsen: $\mu_R(x, y) = \mu_A(x) \times \mu_B(x)$ (II.25)

II.2.5 Raisonnement flou

En logique classique, la méthode d'inférence la plus utilisée est le modus ponens. Elle permet à partir de la règle de type « *Si x est A Alors y est B*» et du fait "*x est A*" de conclure 1e fait " *y est B*", qui sera ajouté à la base des faits. Zadeh a étendu cette méthode au cas flou, sous le nom modus ponens généralisé [JAN97, MEN95, PAS98] pour permettre de raisonner lorsque les règles ou les faits sont connus de façon imparfaite. Le *modus ponens* et le *modus ponens généralisé* se résument comme suit :

	Modus Ponens	**Modus Ponens Généralisé**
Fait (prémisse)	*x est A*	*x est A′*
Règle	*Si x est A Alors y est B*	*Si x est A Alors y est B*
Déduction (Conclusion)	*y est B*	*y est B*'

A partir de la règle « *Si x est A Alors y est B*» et du fait «*A*», on déduit un nouveau fait «*B′*» qui est caractérisé par un sous-ensemble flou dont la fonction d'appartenance est donnée par:

$$\mu_{B'}(y) = Sup_x(\mu_{A'}(x) * \mu_R(x,y)) \qquad \text{(II.26)}$$

Les fonctions d'appartenance $\mu_{A'}(x)$ et $\mu_R(x,y)$ caractérisent respectivement le fait «*A′*» et la règle d'inférence.

II.3 Les systèmes d'inférence flous (SIF)

Un système flou (contrôleur flou) peut être interprété selon deux points de vue: mathématique ou logique. D'un point de vue mathématique, un système flou est une fonction non linéaire reliant un vecteur de données d'entrée à un vecteur de sortie et, de point de vue logique, un système flou est un système à base de connaissance particulière (système expert) composé de quatre modules principaux, à savoir: la base de règles, la fuzzification, le moteur d'inférence et la défuzzification. La figure II.4 montre le schéma synoptique d'un tel contrôleur flou [BUH94, YAG94, MEN95, GOD99]:

Nous allons rappeler dans ce qui suit une description sommaire de chaque module composant le contrôleur flou [JAN97]:

- ***Interface de fuzzification:*** ce module traduit les données numériques caractérisant l'état du système pour fournir une caractérisation floue des variables du système flou sous forme symbolique.
- ***La base de règles floues:*** ou base de connaissance du processus, elle est composée de l'ensemble des renseignements que nous possédons sur le processus. Elle permet de définir les fonctions d'appartenance et les règles floues qui décrivent le comportement du système. C'est le cœur du système entier dans le sens où tous les autres composants sont utilisés pour interpréter et combiner ces règles pour former le système final.
- ***La logique de prise de décision*****:** ou moteur d'inférence floue: une action, sous forme symbolique, est décidée à l'aide des techniques de raisonnement flou en fonction des variables floues précédemment calculées.
- ***Interface de défuzzification:*** ce module traduit l'action floue issue de l'inférence en une grandeur physique directement applicable au processus à commander.

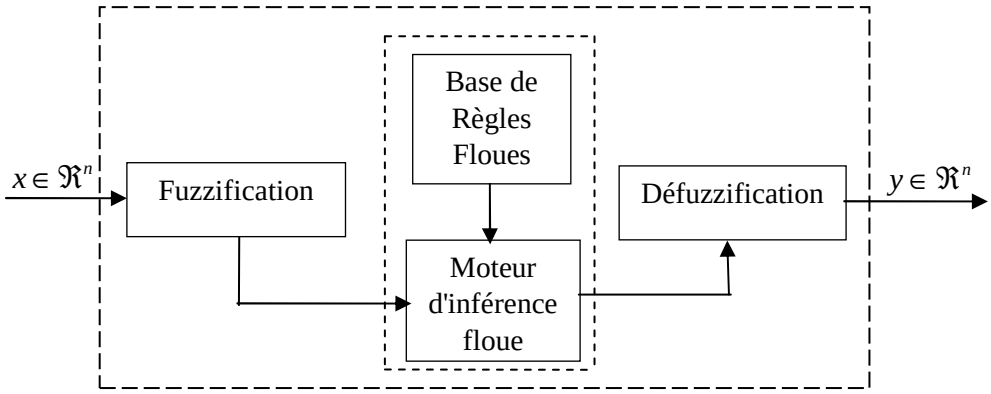


Figure II.4 Configuration de base d'un système flou

Il faut noter qu'il existe un grand nombre de possibilités de réalisation de systèmes flous. En effet, pour chaque module constitutif d'un système flou (Figure II.4), il existe une multitude de choix différents, et chaque combinaison de ces choix engendre une classe de systèmes flous.

II.4 Représentation mathématique des systèmes flous

II.4.1 Fuzzification

La fuzzification consiste à relier le point numérique $x_0 = [x_{01}, x_{02}, ..., x_{0n}]^T$ de U de A à l'ensemble flou $A_x = [A_{x1}, A_{x2}, ..., A_{xn}]^T$ dans $U = U_1 \times ... \times U_n$ où A_{x_i} est un ensemble flou dans U_i. Il existe deux méthodes de fuzzification suivant la définition de A_x.

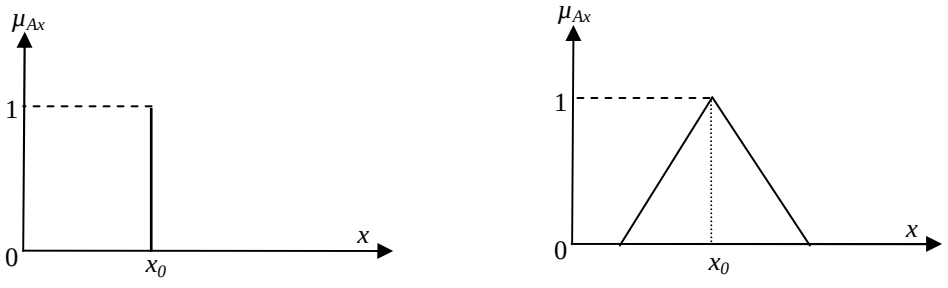
- A_x est un singleton flou défini par :

$$\mu_{A_x}(x) = \begin{cases} 1, x = x_0 \\ 0, x \neq x_0 \end{cases} \quad \text{(II.27)}$$

Dans ce cas, on considère que la valeur de *x* est précise et certaine (figure II.5 (a)).

- A_x est un ensemble flou de fonction d'appartenance $\mu_{A_x}(x_0) = 1$ et $\mu_{A_x}(x)$ décroît lorsque *x* s'éloigne de x_0.

Dans ce cas, est pris en compte le comportement de la variable autour de la valeur x_0. Par exemple, une variable est modélisée par une fonction d'appartenance triangulaire présentée sur la figure II.5 (b).



(a) (b)

Figure II.5 Méthodes de fuzzification: (a) singleton, (b) ensemble flou

II.4.2 Base de règles floues

Une base de règles floues *R* est une collection de règles floues de la forme «*SI-ALORS* », $R = [R_1, R_2, ..., R_m]$. Une règle floue R_i est donnée sous le modèle de « Mamdani » comme suit [MAM75]:

$$R_i : Si\ x_1\ est\ A_{i1}\ et\ x_2\ est\ A_{i2} et \ldots et\ x_n\ est\ A_{in}\ \ ALORS\ \ y\ est\ B_i \quad \text{(II.28)}$$

Ou sous le modèle de « Takagi-Sugeno (TS) » sous la forme:

$$R_i : Si\ x_1\ est\ A_{i1}\ et\ x_2\ est\ A_{i2} et \ldots et\ x_n\ est\ A_{in}\ \ ALORS\ \ y\ est\ f_i(x) \quad \text{(II.29)}$$

$f_i(x)$ est un polynôme (hyperplan). Si le polynôme est d'ordre zéro (sous forme d'une constante), on dit que le modèle est de TS d'ordre zéro (TS0), et si le polynôme est du premier ordre, on dit que le modèle est de TS d'ordre un [TAK85].

II.4.3 Moteur d'inférence floue

Le moteur d'inférence floue utilise la base des règles floues pour effecteur une transformation à partir des ensembles flous dans l'espace d'entrée vers les ensembles flous dans l'espace de sortie en se basant sur les opérations de la logique floue. L'antécédent de la règle *R*, définit un produit cartésien de $A_{i1}, A_{i2}, ..., A_{1n}$, et la règle elle-même R_i, est vue comme une implication. Soit $B_i = A_x \circ R_i$ dans *V*. La fonction d'appartenance de B_i est définie par la règle compositionnelle (II.30):

$$\mu_{B'}(y) = Sup_{x \in A_x}(\mu_A(x) * \mu_{R_i}(x, y)) \quad \text{(II.30)}$$

Dans le jeu de règles du système flou interviennent les opérateurs flous "ET(AND)" et "OU(OR)". L'opérateur "ET" s'applique aux variables à l'intérieur d'une règle, tandis que l'opérateur "OU" lie les différentes règles. Plusieurs types de raisonnement flou ont été proposés dans la littérature suivant la réalisation des opérateurs flous "ET" et "OU" et le type des règles floues utilisées. Les trois moteurs d'inférence floue les plus utilisés sont: le moteur de Mamdani, de Sugeno et celui de Tsukumoto.

II.4.3.1 *Méthode de Mamdani*

Mamdani fut le premier à utiliser la logique floue pour la synthèse des commandes [MAM74]. Il a utilisé le minimum comme opérateur de conjonction et d'implication. Les règles correspondant à l'équation (II.28) où B^i est un sous ensemble flou. Généralement, les B^i forment une partition de l'espace de sortie. L'inférence floue correspond aux étapes suivantes, pour un vecteur d'entrée $x = (x_1, ..., x_n)^t$ [GLO99] :

1. Calcul du degré d'appartenance de chaque entrée aux différents sous ensembles flous:

$$\mu^i_{A_j}(x_j), \text{ pour } j = 1 \text{ à } n \text{ et } i = 1 \text{ à } N \quad \text{(II.31)}$$

où : *n* est la dimension d'espace d'entrée et N : Nombre des règles floues.

2. Calcul de valeur de vérité de chaque règles, pour $i = 1$ à N :

$$\alpha_i(x) = \min_j(\mu^i_{A_j}(x_j)) \quad j = 1 \text{ à } n \quad \text{(II.32)}$$

3. Calcul de la contribution de chaque règle selon l'équation (eq. II.33):

$$\mu_i(y) = \min(\alpha_i(x), \mu_{B_i}(y)) \tag{II.33}$$

4. Agrégation des règles en utilisant l'opérateur max:

$$\mu(y) = \max_i(\mu_i(y)) \tag{II.34}$$

Le résultat est donc un sous ensemble flou caractérisée par sa fonction d'appartenance. Pour obtenir une conclusion exacte, il faut défuzzifier. Plusieurs méthodes peuvent être utilisées : la méthode de centre de gravité donne :

$$y = \frac{\int u\mu(u)du}{\int \mu(u)du} \tag{II.35}$$

Ou dans le cas discret:

$$y = \frac{\sum_k u_k \mu(u_k)}{\sum_k \mu(u_k)} \tag{II.36}$$

Cette implémentation est appelée (*min, max, barycentre*). Ils existent plusieurs variantes de la méthode de Mamdani comme (min, produit, barycentre) et (produit, somme, barycentre)…etc [MAM85, PAS98].

II.4.3.2 *Méthode de Takagi-Sugeno*

Les SIFs de type Takagi-Sugeno constituent un cas particulier important; ou la conclusion n'est pas symbolique mais une fonction des entrées: $B^i = f(x_1,...,x_n)$, la forme la plus utilisée est la suivante:

$$B^i = \sum_{j=0}^{n} b_j^i x_j \tag{II.37}$$

Un SIF de type Takagi-Sugeno réalise une fusion de modèles locaux par interpolation. Un SIF est dit d'ordre zéro ou de méthode de Takagi-Sugeno simplifiée si la conclusion est une constante, et SIF d'ordre 1 quand la conclusion est un hyperplan (l'équation II.37). Le modèle de Takagi-Sugeno d'ordre 0 est le plus utilisé. Dont les conclusions sont des nombres réels, ou singletons, (pouvant être considérés comme la valeur modale d'un sous ensemble flou) [TAK85].

Pour le vecteur d'entrée $x = (x_1,...,x_n)^t$, la sortie d'un système d'inférence flou de type Takagi-Sugeno d'ordre 0 est calculée selon les étapes suivantes [GLO99] :

1. Calcul du degré d'appartenance de chaque entrée aux différents sous ensembles flous :

$$\mu_{A_j}^i(x_j), \text{ pour } j = 1 \text{ à } n \text{ et } i = 1 \text{ à } N \tag{II.38}$$

2. Calcul de la valeur de vérité de chaque règle pour un vecteur d'entrée x, pour $i = 1$ à N

$$\alpha_i(x) = ET(\mu_{A_1^i}(x_1),...,\mu_{A_n^i}(x_n)) \tag{II.39}$$

3. Calcul de la sortie :

$$y = \frac{\sum_{i=1}^{N} \alpha_i(x).b^i}{\sum_{i=1}^{N} \alpha_j(x)} \tag{II.40}$$

Les SIFs de type Takagi-Sugeno permettent un passage aisé d'une expression symbolique définie par la base de règle à une valeur numérique exploitable.

II.4.3.3 *Méthode de Tsukumoto*

Dans ce cas, des fonctions monotoniques sont associées aux variables de sortie. La sortie totale est une moyenne pondérée des degrés de confiance des règles floues et des valeurs des fonctions des variables de sortie [PAS98]. La figure II.6 illustre les types du raisonnement flou pour un système flou à deux entrées et une base de connaissances de deux règles floues [BOU09]. On constate que les différences viennent de la spécification de la partie conclusion d'une part, et de la méthode de défuzzification d'autre part. Dans cette thèse on limite à l'utilisation des systèmes flous les plus utilisés dans la commande des processus, à savoir les systèmes flous de Takagi-Sugeno à conclusion constante (d'ordre zéro).

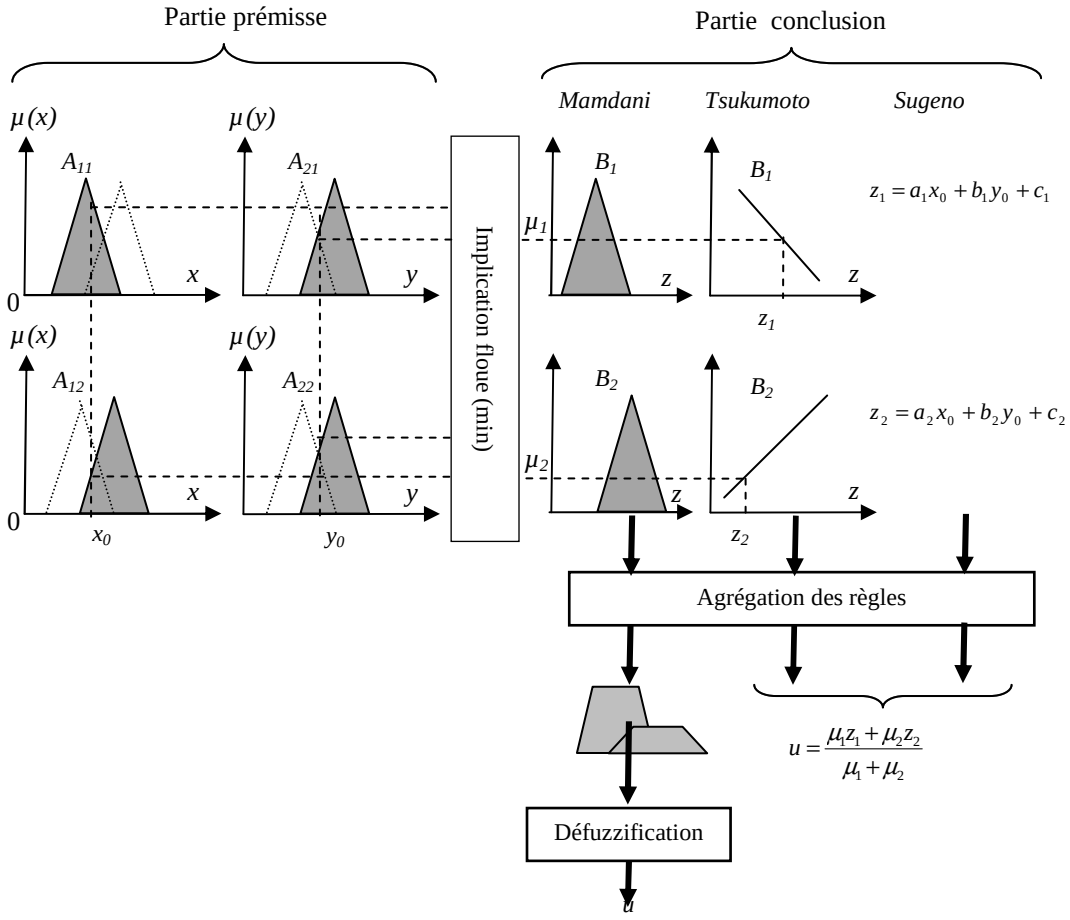


Figure II.6 Différents modèles d'inférence flous

II.5 Approximation des fonctions par les systèmes flous

Deux raisons principales amènent à utiliser les systèmes flous comme contrôleurs des systèmes dynamiques. Premièrement, ce type de systèmes flous a la propriété d'approximateur universel de fonctions continues avec un degré de précision quelconque à condition d'utiliser un nombre suffisant de règles floues. Deuxièmement, les systèmes flous sont construits à partir de règles floues de la forme *Si-Alors*, de ce fait, les informations linguistiques ou mathématiques

disponibles d'un expert peuvent éventuellement être incorporées dans le contrôleur. Dans la littérature consacrée aux systèmes flous, on dispose d'un nombre important de publications montrant que les systèmes flous sont des approximateurs universels (voir par exemple [LOS93, WAN94, GLO99]), c'est-à-dire, pour toute fonction réelle continue *f* définie sur un compact C de R^n, et pour tout $\varepsilon > 0$, il existe un SIF tel que :

$$(\forall\, x \in C),\ (\|f\,(x) - SIF(x)\| < \varepsilon) \qquad \text{(II.39)}$$

Notons, cependant, que la propriété d'approximation universelle ne donne pas une méthode de construction du système flou *SIF(x)*, mais elle garantisse seulement son existence. De plus, pour un degré de précision quelconque, il faut utiliser un nombre important de règles floues.

II.6 Conclusion

Dans ce chapitre, nous avons introduit les principes de base des ensembles flous qui sont une généralisation du concept d'ensembles classiques. Contrairement à la fonction caractéristique d'un ensemble net qui prend la valeur 0 ou 1, la fonction d'appartenance d'un ensemble flou prend ses valeurs dans l'intervalle [0,1]. A partir des ensembles flous, nous pouvons construire des systèmes flous. Un système d'inférence flou est une unité de prise de décision composée de quatre parties essentielles: la fuzzification, la base de règles, le moteur d'inférence et la défuzzification. L'architecture d'un système flou est déterminée par une meilleure compréhension des ensembles flous et des opérateurs flous. Nous avons constaté qu'il n'existe pas un seul type de système flou, mais il y en a plusieurs. Un utilisateur des systèmes flous doit décider sur la méthode de fuzzification, le type des fonctions d'appartenance, le type des règles floues, la méthode du raisonnement flou et la stratégie de défuzzification. Les systèmes flous sont des approximateurs universels. En fait, ils peuvent approcher n'importe quelle fonction à partir de données numériques. Le fonctionnement d'un contrôleur flou dépend d'un nombre important de paramètres (méthode de fuzzification, le type des fonctions d'appartenance, le type des règles floues, la méthode du raisonnement flou et la stratégie de défuzzification) qu'il faut déterminer lors de la conception. Comme ces paramètres s'influencent mutuellement, leur réglage n'est donc pas aisé. Par contre, les contrôleurs flous présentent la possibilité d'incorporer des connaissances expertes dans leurs structures, ce qui peut aider à la recherche des paramètres optimaux des contrôleurs flous en utilisant d'autres approches.

Dans le chapitre 3, nous exposerons les deux méthodes de l'intelligence artificielle utilisées dans ce travail; les réseaux de neurones artificiels et l'apprentissage par renforcement. Ces méthodes permettent l'ajustement des paramètres des règles floues et l'optimisation des contrôleurs flous d'une façon très efficace.

Chapitre III

Les Réseaux de Neurones Artificiels et l'Apprentissage par Renforcement

III.1 Introduction

L'apprentissage est considéré comme une tâche de construction de nouvelles connaissances ou amélioration des connaissances existantes. Le but est d'améliorer les performances du système en tenant compte des ressources et des compétences dont il dispose. Il existe plusieurs formes d'apprentissage et de modification de comportement, selon le type d'informations disponibles; on peut citer: l'apprentissage supervisé, non supervisé et l'apprentissage par renforcement. Chacun avec ses propriétés intéressantes.

Dans ce chapitre, nous allons introduire deux techniques importantes de l'intelligence artificielle pour l'adaptation des comportements; les réseaux de neurones artificiels avec ces capacités d'apprentissages, et la deuxième est un apprentissage automatique basé sur un signal critique d'évaluation (apprentissage par renforcement). Grâces aux plusieurs caractéristiques, l'utilisation de ces deux approches pour l'amélioration des comportements et la commande des processus devient un domaine très important dans les dernières décennies [GUE88, FIE98, SUT98, GLO00, JAN04]. Les définitions et les notions de base seront présentées dans ce chapitre.

III.2 Définitions

Avant de développer les notions des approches d'apprentissage étudiées, nous allons revenir sur quelques notions utiles comme:

III.2.1 Intelligence Artificielle (IA)

L'intelligence artificielle est la reproduction, par des moyens artificiels de toutes les formes de l'intelligence humaine pour un objectif final, qui s'intéresse à la conception des systèmes intelligents. L'IA est la capacité à raisonner, à apprendre, et à s'adapter face à de nouveaux changements qu'ils sont des éléments principaux que nous recherchons [FER95].

III.2.2 Agent

D'après Ferber [FER95]; un agent est une entité autonome réelle ou abstraite qui est capable d'agir sur elle-même et sur son environnement, et peut communiquer avec d'autre agents, dont le comportement est la conséquence de ses observations, ses connaissances et de ces interactions avec son environnement. Il doit être (autonome, interactif, adaptatif, rationnel, coopératif et intelligent…). La notion d'agent se diffère selon l'utilisation.

III.2.3 Apprentissage

L'apprentissage automatique fait référence au développement, à l'analyse et à l'implémentation de méthodes qui permettent à une machine d'évoluer grâce à un processus d'apprentissage, et ainsi de remplir des tâches qu'il est difficile ou impossible de remplir par des moyens algorithmiques plus classiques. L'objectif est d'extraire et exploiter automatiquement l'information présentée dans un jeu de données. On peut schématiser le processus d'apprentissage comme montré sur la figure III.1, l'environnement agit sur le module d'apprentissage de l'agent. Cela à pour effet de consulter et de modifier la base de connaissances pour arriver à une exécution adéquate. De cette exécution, un retour est attendu pour évaluer le résultat obtenu.

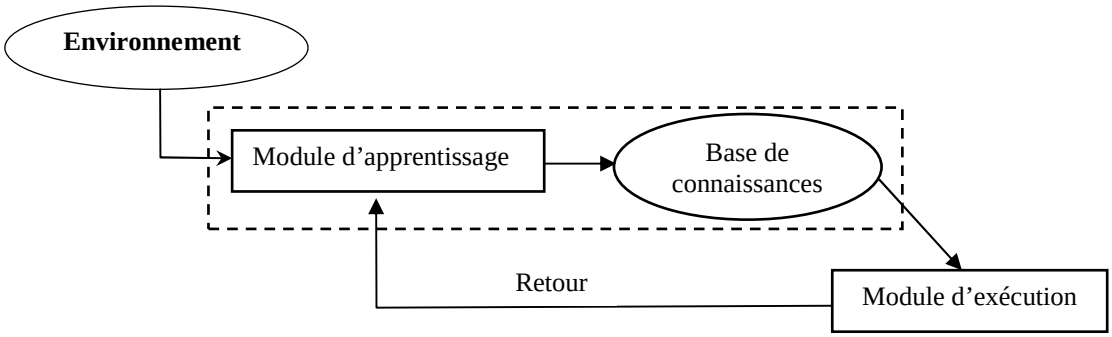


Figure III.1 Représentation de l'apprentissage automatique

Les algorithmes d'apprentissage peuvent se catégoriser selon le type d'apprentissage, qu'ils emploient [AND95]:

- ***Apprentissage supervisé:*** le contrôleur (ou le maître) fournit l'action qui devrait être exécutée. L'utilisation nécessite un expert capable de fournir un ensemble d'exemples formés de situations et d'actions correctes associées (figure III.2) [AND95]. Ces exemples doivent être représentatifs de la tâche à accomplir.
- ***Apprentissage non supervisé:*** dans lequel l'apprenant doit identifier par lui-même la meilleure réponse possible, il n'y a pas de réponse désirée. La tâche peut être par exemple dans ce cas de créer des regroupements de données selon des propriétés communes (catégorisation) [GAU99].
- ***Apprentissage par renforcement (AR):*** le contrôleur a un rôle d'évaluateur et non pas d'instructeur, l'information disponible est un signal de renforcement; généralement appelé critique (figure III.3). Son rôle est de fournir une mesure indiquant si l'action générée est appropriée ou non. Le contrôleur doit déterminer et de modifier ses actions de manière à obtenir une meilleure évaluation dans le futur [SUT98].

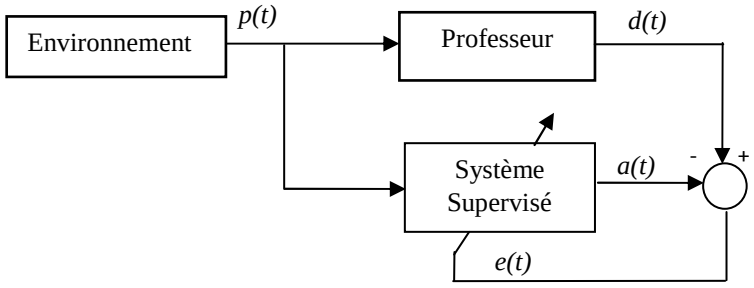


Figure III.2 Représentation de l'apprentissage supervisé

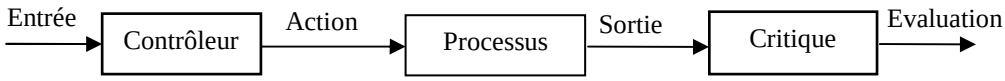


Figure III.3 Représentation de l'apprentissage par renforcement

Dans ce qui suit, on va donner un bref exposé sur les réseaux de neurones artificiels. En présentant leurs principes et les différentes architectures existantes.

III.3 Réseaux de neurones artificiels (RNA)

Les réseaux de neurones artificiels sont des systèmes de traitement de l'information dont la structure s'inspire de celle du système nerveux [PAR04]. Ils sont destinés à effectuer des tâches auxquelles les approximateurs traditionnels semblent moins adaptés. Dans le domaine de traitement du signal et de l'automatique, les dernières années ont été marquées par des avancées significatives tant du point de vue fondamental que du point de vue des applications industrielles [PER03]. Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau [AND88, CLA92].

L'origine de l'inspiration des réseaux de neurones artificiels remonte à l'année 1890 où W. James a introduit le concept de mémoire associative. Il propose ce qui deviendra une loi de fonctionnement pour l'apprentissage des RN, connue plus tard sous le nom de loi de Hebb. Quelques années plus tard, en 1943, J. Mc Culloch et W. Pitts ont donnés leurs noms à une modélisation du neurone biologique (un neurone au comportement binaire). Ils sont montrés que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes. En 1949, D. Hebb présente dans son ouvrage "*the organization of behavior*" une règle d'apprentissage. C'est dans les années soixante, que les premiers modèles du perceptron ont fait leur apparition par F. Rosenblatt. C'est dans ce cadre que se développent les algorithmes d'apprentissage supervisé, fondés sur le terme d'erreur entre sortie produite et sortie souhaitée, qui est à l'origine de la règle de rétro-propagation (apprentissage supervisé). C'est alors qu'en 1960, le modèle Adaline (Adaptive Linear Element) est développé par B. Widrow et Hoff. Dans sa structure, le modèle ressemble au perceptron, cependant la loi d'apprentissage est différente. Celle-ci est à l'origine de l'algorithme de rétro-propagation de gradient très utilisé aujourd'hui avec les Perceptrons Multi Couches (PMC). M. Minsky et S. Papert publient ensuite en 1969 un ouvrage qui met en évidence les limitations théoriques des perceptrons [PAR04]. Les recherches dans ce domaine sont fortement diminuées jusqu'en 1972, où T. Kohonen présente ses travaux sur les mémoires associatives et propose des applications à la reconnaissance de formes.

En 1982, J. Hopfield a introduit un modèle de réseau de neurones complètement récurrent. Les cartes auto-organisatrices de Kohonen mettant en place une règle d'apprentissage non-supervisée fondée sur la compétition. Enfin, en 1986, D. Rumelhart, G. Hinton et R. Williams ont publiés l'algorithme de rétro-propagation du gradient de l'erreur pour les perceptrons multicouches, ouvrant la voie à de nombreuses applications. A partir de ce moment, la recherche sur les réseaux de neurones a connue plusieurs applications aux cours des années.

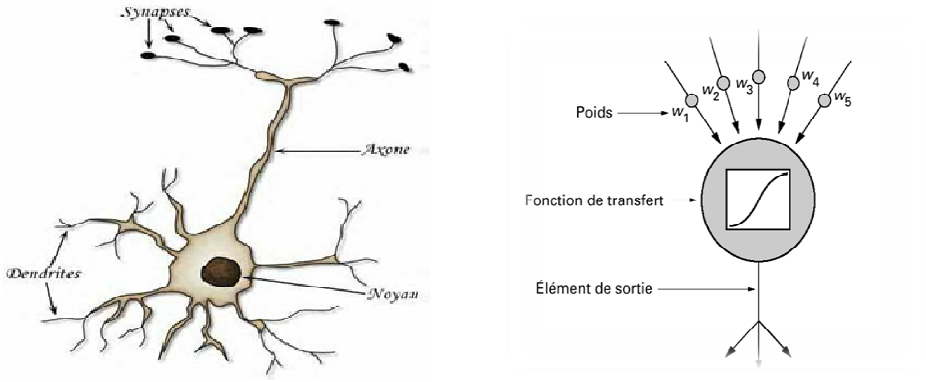
Aujourd'hui il existe plusieurs applications dans pratiquement tous les domaines: reconnaissance de formes, modélisation, commande, classification, engineering, financier, économiques, aéronautique, …etc.

III.3.1 Structure d'un neurone

Le neurone est une cellule composée d'un corps cellulaire et d'un noyau (figure III.4 (a)) [AND95, GAU99]. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites. C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma, corps du neurone. L'information traitée par le neurone s'oriente ensuite le long de l'axone pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire entre l'axone du neurone afférent et les dendrites du neurone efférent. La jonction entre deux neurones est appelée la synapse.

Par analogie avec le neurone biologique, le comportement du neurone artificiel est modélisé par deux phases comme présenté sur la figure III.4 (b):

- Un opérateur de sommation, qui élabore le potentiel *a*. Cet opérateur effectue la somme pondérée des entrées. On soustrait parfois à cette somme la valeur de seuil d'activation.
- Un opérateur non linéaire qui calcule la valeur de l'activation en utilisant une fonction de transfert (fonction d'activation).



(a) (b)

Figure III.4 Représentation d'un neurone biologique et un neurone artificiel

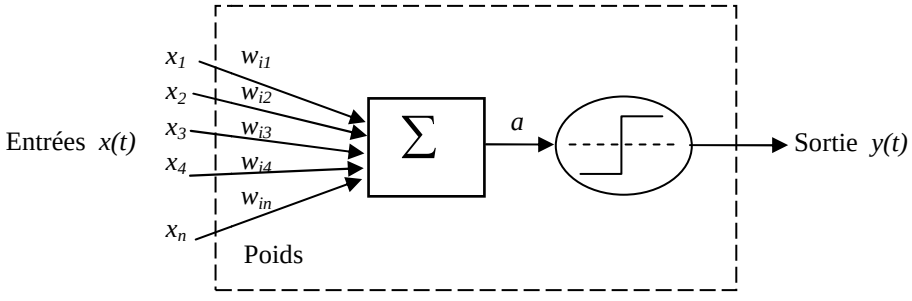


Figure III.5 Modèle d'un neurone artificiel

Chaque neurone artificiel est un processeur élémentaire de traitement. Il reçoit un nombre variable d'entrées en provenance de neurones amont. A chacune de ces entrées est associé un

poids w_{ij} représentant la force de la connexion [GOD96, GAU99]. Le neurone artificiel modélisé par Mc Culloch et Pitts est représenté par la figure III.5.

x_i*:* entrée du neurone *i* (ou sortie de neurone amont *j*).

w_{ij}*:* la valeur du poids synaptique de la connexion dirigée du neurone *j* vers le neurone *i*,

- Un poids positif indique un effet excitateur du neurone émetteur *j* vers le neurone récepteur *i*,
- Un poids négatif indique un effet inhibiteur.

La fonction de combinaison renvoie le produit scalaire entre le vecteur des entrées et le vecteur des poids synaptiques. Autrement dit, elle calcule la somme pondérée des entrées selon l'expression suivante:

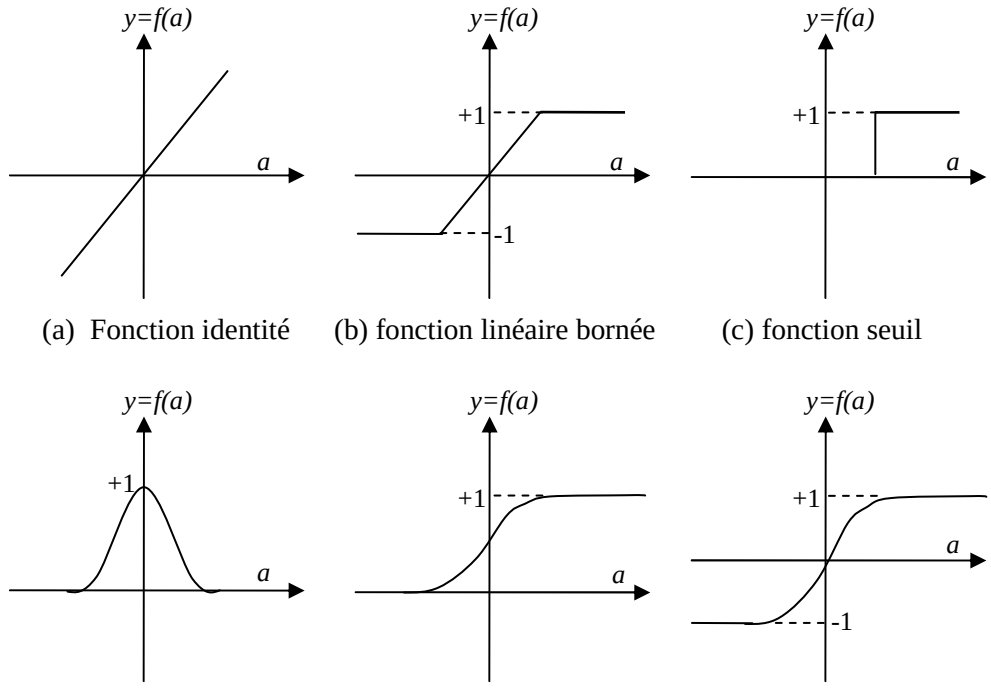
$$a_i = \sum_{j=1,n} w_{ij}.x_j \qquad \text{(III.1)}$$

À partir de cette valeur, une fonction d'activation *f*, calcule la valeur de l'état du neurone. Cette valeur sera transmise aux neurones aval. Les neurones les plus fréquemment utilisés sont ceux pour lesquels la fonction *f* est une fonction non linéaire d'une combinaison linéaire des entrées.

$$y_i = f(a_i - \theta_i) \qquad \text{(III.2)}$$

III.3.2 Fonctions d'activation

La fonction de transfert ou d'activation définit la valeur de sortie d'un neurone en termes des niveaux d'activité de ses entrées. Cette fonction peut prendre différentes formes possibles telles que: fonction linéaire à seuil, fonction seuil, fonction gaussienne,…etc (figure III.6).



(a) Fonction identité (b) fonction linéaire bornée (c) fonction seuil

(d) Fonction gaussienne (e) fonction sigmoïde (f) fonction tangente hyperbolique

Figure III.6 Formes usuelles de la fonction d'activation

III.3.3 Structures d'interconnexion

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Elle peut être quelconque, mais le plus souvent il est possible de distinguer une certaine régularité qui différencie entre ces réseaux [JAM92, AND95, PAR04]. On peut citer:

III.3.3.1 *Réseau multicouche (Multi Layer Perceptron)*

Les neurones sont arrangés par couche. Il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones des couches en aval (Figure III.7 (a)). Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Ceci nous permet d'introduire la notion de sens de parcours de l'information (de l'activation) au sein d'un réseau et donc définir les concepts de neurone d'entrée, neurone de sortie. Par extension, on appelle *couche d'entrée* l'ensemble des neurones d'entrée, *couche de sortie* l'ensemble des neurones de sortie. Les *couches intermédiaires* n'ayant aucun contact avec l'extérieur sont appelées *couches cachées.*

III.3.3.2 *Réseau à connexions récurrentes*

Les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche. Ces connexions sont le plus souvent locales (Figure III. 7 (b)).

III.3.3.3 *Réseau à connexions locales*

Il s'agit d'une structure multicouche, mais qui à l'image de la rétine, conserve une certaine topologie. Chaque neurone entretien des relations avec un nombre réduit et localisé de neurones de la couche avale (Figure III. 7 (c)). Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique.

III.3.3.4 *Réseaux à fonction radiale*

Ce sont les réseaux que l'on nomme aussi RBF (Radial Basic Fonctions). L'architecture est la même que pour les PMC (perceptron multicouches) mais avec une seule couche cachée. Cependant, les fonctions de base utilisées ici sont des fonctions gaussiennes (Figure III.7 (d)).

III.3.3.5 *Réseau à connexion complète*

C’est la structure d'interconnexion la plus générale (Figure III.7 (e)). Chaque neurone est connecté à tous les neurones du réseau et à lui-même.

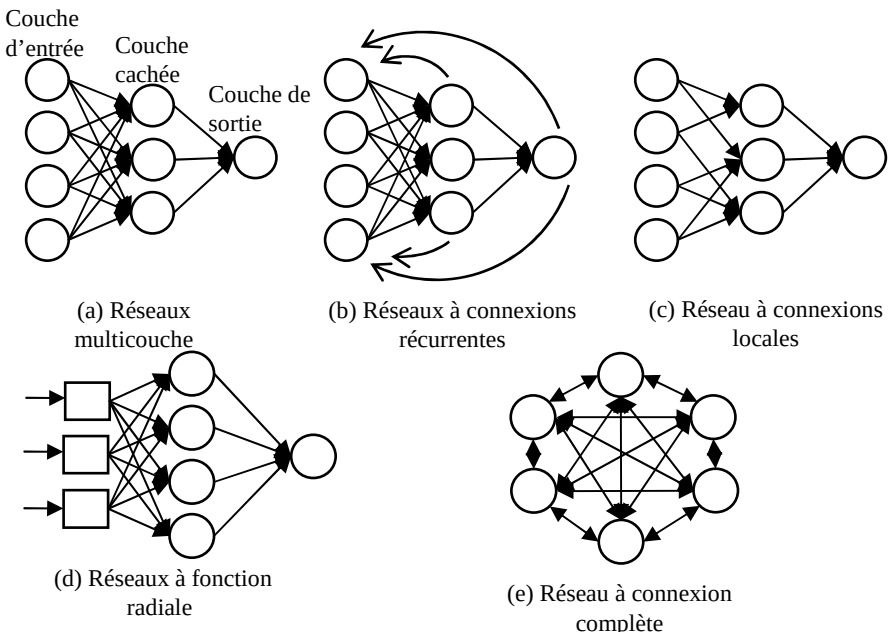


Figure III.7 Les différentes structures d'interconnexion des RNA

III.3.4 Apprentissage des RNA

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux neuronaux. L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré en modifiant les poids d'interconnexion [TOU92, AND95]. Celui-ci peut s'effectuer par présentation répétée d'une série de "patrons" ou "modèle" et peut être supervisé ou non. Dans le cas des RN multicouches, les approches les plus utilisées sont celles inspirées de la méthode de rétro-propagation du gradient. Comme on a cité précédemment, les techniques d'apprentissage sont regroupées en deux classes; supervisé et non supervisé. Cette distinction repose sur la forme des exemples d'apprentissage [GAU99, PAR04]:

1. ***Apprentissage supervisé:*** ces algorithmes utilisent une base de données composée d'un ensemble de paires entrée-sortie. L'apprentissage dans ce cas, consiste à calculer les poids d'interconnexion de telle manière que les sorties du réseau de neurones soient, pour les exemples utilisés lors de l'apprentissage, aussi proches que possibles des sorties désirées, qui peuvent être la valeur de la fonction approchée ou de la sortie du processus à modéliser, ou encore la sortie souhaitée du processus à commander.
2. ***Apprentissage non supervisé:*** le calcul des coefficients dans ce cas, se fait sur la base de la suite des vecteurs d'entrés en optimisant une fonction de cout. Les cartes auto-organisatrices de Kohonen est un modèle de référence pour l'apprentissage non supervisé.

Pour l'apprentissage des RNAs multicouches, l'algorithme de la rétro-propagation du gradient de l'erreur "*backpropagation*" est le plus utilisé. Le principe consiste à la modification des poids des connexions en minimisant une fonction de cout (erreur quadratique) [AND88, TOU92, JAM92]. Il existe plusieurs règles de modification des poids dont les principales sont: la règle de Hebb, la règle de Widrow-Hoff.

III.3.6 Les RNA pour la commande des processus

Les réseaux de neurones artificiels constituent des outils particulièrement efficaces pour la modélisation et la commande des systèmes complexes ou mal définis. Commander un processus, c'est déterminer l'action à appliquer, pour lui assurer un comportement donné, en dépit de perturbations. Cette action est délivrée par un organe ou une loi de commande réalisée par un RNA. Nous séparons ces approches en deux classes principales suivant qu'elles nécessitent ou non l'identification préalable d'un modèle du processus commandé. Nous parlerons de méthode de commande *indirecte* lorsqu'un tel modèle est nécessaire, par opposition aux méthodes *directes* qui n'en nécessitent pas [GUE88, BOU93, PER03].

III.3.6.1 *Identification par RNA*

L'idée de base est de substituer aux modèles paramétriques classiques des modèles neuroneaux dont les paramètres sont adaptaés par une procédure d'apprentissage appropriée en utilisant les données entrées-sorties du système.

1. Modélisation directe: consiste à entrainer le réseau pour représenter la dynamique directe du système où l'erreur entre les deux sorties est utilisée pour l'adaptation des paramètres du modèle. Les réseaux de neurones récurrents sont les plus adaptés pour ce type de modélisation:

- ***Modélisation série-parallèle:*** le but est d'entrainer le réseau pour que la transformation non linéaire *f* soit une approximation de la fonction du système. Dans cette structure, l'entrée du modèle comporte la sortie actuelle et les valeurs précédentes de la réponse du système réel.
- ***Modélisation parallèle:*** l'entrée du réseau comporte les valeurs décalées de la réponse du modèle. L'avantage est de présenter des performances meilleures dans le cas de perturbations.

2. Modélisation inverse: consiste à entrainer un modèle neuronal pour qu'il génère les commandes nécessaires de déplacer le système de son état actuel $x(t)$ à un état désiré $x_d(t+1)$. Le but de la procédure d'entrainement est l'adaptation des paramètres du modèle pour qu'il approche la fonction inverse f^{-1} (évaluer l'erreur sur la commande qui sera ensuite utilisée pour l'apprentissage):

- ***Apprentissage généralisé:*** en utilisant la connaissance qualitative que l'on a sur le comportement du système, des signaux compatibles avec son fonctionnement lui sont appliqués. La sortie du système sert alors d'entrée au modèle neuronal dont la sortie est comparée avec le signal d'entrainement. En utilisant cette erreur pour l'apprentissage du réseau, cette structure force le réseau à apprendre le comportement inverse du système.

- ***Apprentissage spécialisé:*** dans cette approche, le modèle inverse neuronal est placé en série avec le système, et reçoit à son entrée un signal d'entrainement appartenant à l'espace des sorties opérationnelles désirées du système. Le système est considéré comme une couche supplémentaire dont les poids sont fixes. Le but de la procédure d'apprentissage est de minimiser l'erreur sur les sorties du modèle inverse.

III.3.6.2 *Commande par RNA*

Deux architectures de contrôle contenant des réseaux de neurones sont possibles. Le premier type est une configuration auto-réglable, dans laquelle un réseau ajuste les paramètres d'un controleur conventionnel. Le second type est une configuration plus simple, dans laquelle le réseau commande directement le système. Les structures de contrôle peuvent ètre divisées en cinq structures de base [BOU93]:

- ***Contrôle supervisé:*** dans ce cas, les entrées du réseau sont les informations que reçoit l'opérateur humain sur l'état du système, les sorties désirées sont les actions délivrées par l'opérateur.
- ***Contrôle direct par modèle inverse:*** ce type de contrôle utilise un modèle inverse placé en série avec le système à commander. Le modèle inverse est capable de générer les commandes nécessaires pour une tâche désirée.
- ***Contrôle adaptatif neuronal:*** le réseau est utilisé pour remplacer les transformations linéaires adoptées par les techniques standard de commande adaptative (à modèle de référence, controleurs auto-réglables,…).
- ***Rétropropagation d'utilité:*** le principe est de calculer les variations par rétropropagation à travers le temps. Le comportement désiré du système sur un horizon déterminé est spécifié par un critère de performance (fonction d'utilité) qui doit etre défini en fonction des sorties du système et des sorties désirées. Le controleur neuronal est entrainé de façon à minimiser ce critère.
- ***Contrôle à apprentissage par renforcement:*** cette structure de commande comporte un réseau d'évaluation de performance du système (sous forme des récompenses et des punitions), et un réseau d'action pour générer les commandes qui optimisent le critère de performance.

Les RNA possèdent plusieurs caractéristiques intéressantes pour la réalisation des systèmes de commande parmi lesquelles [TOU92, AND95, PER03]:

- la parallélisation du traitement leur confère une grande rapidité de calcul et les rend très adaptés aux applications temps réel. Ceci est d'autant plus vrai lorsque l'on s'intéresse à des réalisations réellement parallèles sur des machines multiprocesseurs,
- le caractère distribué et fortement redondant du traitement réalisé leur donne une bonne résistance aux pannes internes,
- leur capacité de généralisation garantisse une bonne résistance aux bruits. Ceci est particulièrement important lorsque les capteurs permettant de mesurer l'état ou la sortie du processus commandé sont soumis à des perturbations. C'est notamment le cas en robotique, où l'une des grandes difficultés concerne l'obtention de données sensorielles de bonne qualité.
- Faculté d'apprentissage à partir d'exemples représentatifs, par "rétro-propagation de l'erreur". L'apprentissage ou la construction du modèle est automatique,
- La simplicité d'utilisation.

Mais entre outre, les réseaux de neurones présentent quelques inconvénients:

- L'absence de méthode systématique permettant de définir la meilleure topologie du réseau et le nombre de neurones à placer dans les couches,
- Le choix des valeurs initiales des poids du réseau et le réglage des paramètres jouent un rôle important pour la convergence,
- Le problème du sur-apprentissage ou les minimums locaux qui limité l'utilisation,
- La difficulté d'obtenir une base de données pour l'apprentissage.

Dans la partie suivante de ce chapitre, on va présenter le principe de l'apprentissage par renforcement, la formalisation mathématique des Processus de Décision Markoviens *(PDM)*. Nous présenterons aussi les différents algorithmes de cette méthode d'apprentissage.

III.4 Apprentissage par Renforcement (AR)

III.4.1 Définition et principe

L'apprentissage par renforcement est une approche de l'intelligence artificielle qui permet l'apprentissage d'un agent par l'interaction avec son environnement; afin de trouver, par un processus essais-erreurs, l'action optimale à effectuer pour chacune des situations que l'agent va percevoir pour maximiser ses récompenses [KAE96, SUT98]. C'est une méthode de programmation ne nécessitant que de spécifier les moments auxquels punir ou récompenser l'agent. Il n'est nul besoin de lui indiquer quoi faire dans telle ou telle situation, l'agent se charge d'apprendre par lui même en renforçant les actions qui s'avèrent les meilleures [TOU99]. On considère un agent situé dans un environnement, avec lequel il peut interagir. L'agent à l'instant *t* perçoit d'une part cet environnement par la perception s_t, et peut agir en exécutant l'action a_t, et d'autre part; reçoit une récompense r_t. L'interaction de l'agent avec son environnement est représenté sur la figure III.8, où s_{t+1} *et* r_{t+1} sont la situation et la récompense suivante à l'instant (*t+1*). L'agent ne connaît pas la situation dans laquelle il se trouve que par l'intermédiaire de l'historique de ses perceptions. Son but dans le cadre de l'apprentissage par renforcement est de trouver le comportement le plus efficace dans une application donnée, c'est à dire savoir, dans chaque situation perçue, quelle action à accomplir pour maximiser l'espérance des récompenses? [SUT98, GLO00, PRE07].

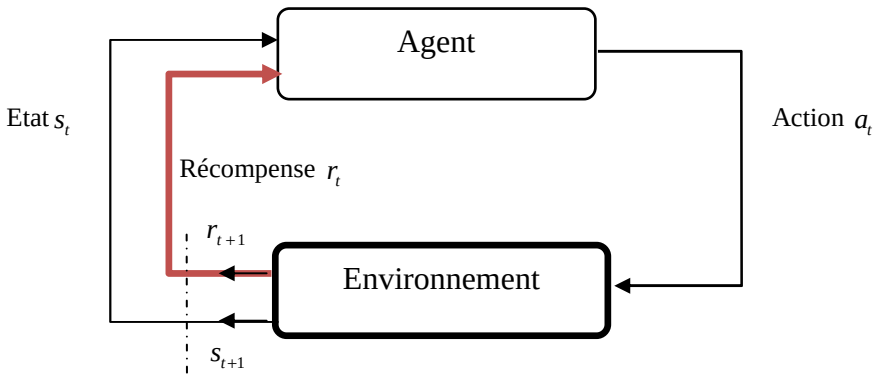


Figure III.8 Système d'apprentissage par renforcement

III.4.2 Processus de décision markoviens (*PDM*)

La formalisation mathématique des algorithmes d'apprentissage par renforcement telle que nous la connaissons aujourd'hui s'est développée à partir de 1988 lorsque Sutton [SUT88] puis Watkins en 1989 [WAT89] ont fait le lien entre leurs travaux et le cadre théorique de la commande optimale proposée par Bellman en 1957 [BEL57] avec la notion de *PDM*. L'apprentissage par renforcement est basé sur l'apprentissage à temps discret des paramètres d'une chaîne de Markov. Les chaînes de Markov sont représentées par des états et de transitions entre ces états. Elles sont définies par un quadruplet (S, A, P, R) [SUT98] où:

S: est un ensemble d'états fini,

A: est un ensemble d'actions fini. On note $A(s)$ l'ensemble des actions possibles dans l'état s, avec: $A = \bigcup_{s \in S} A(s)$,

P: est la fonction de transition décrivant la dynamique de l'environnement. Elle définit une distribution de probabilité de choisir une action à un état donné:

$$P : S \times A \times S \to [0,1]$$

$$(s, a, s') \mapsto P(s, a, s') = \Pr\left[s_{t+1} = s' \middle| s_t = s, a_t = a\right]$$

R: est la fonction de retour (récompense ou punition). On associe pour chaque transition une valeur de probabilité sur R :

$$R : S \times A \times S \to \Re \qquad (s, a, s') \mapsto R(s, a, s') = E\left[r_t \middle| s_{s+1} = s', s_t = s, a_t = a\right]$$

Il peut exister un ensemble d'états finaux $F \subset S$; quand l'agent atteint l'un de ces états, sa tâche est terminée. C'est une tâche épisodique ou à horizon fini. Généralement les *4* ensembles sont fixes au cours du temps sinon, on dirait que le problème est non stationnaire.

III.4.3 Propriété de Markov

Supposant que l'état courant et le retour courant ne dépendent que de l'état précédent et de l'action qui vient d'être émise. C'est une propriété fondamentale, doit être respecté par tout les *PDM*. C'est la propriété de Markov (on dit aussi que l'environnement est markovien). Il n'est alors nul besoin de mémoire pour prendre des décisions au mieux: seule la connaissance de l'état courant est utile. La seule trace d'une mémoire réside dans l'apprentissage de comportement effectué par l'agent [PRE07].

III.4.4 Politique

Le comportement de l'agent est défini par une *politique* $\pi : \{S, A\} \to [0,1]$, qui guide l'agent de manière probabiliste en spécifiant, pour chaque état s la probabilité de réaliser l'action a (donc $\pi(s) = a$). Le but est de trouver la politique optimale π^* maximisant la récompense à long terme, on la note $(s, a) \mapsto \pi(s, a) = \Pr\left[a_t = a \middle| s_t = s\right]$. On cherche donc à résoudre un problème de contrôle optimal, où l'apprentissage par renforcement ne se définit pas par une certaine classe d'algorithmes; mais par le problème qu'il cherche à résoudre. Celui de la commande optimale [SUT91]. On définit le retour reçu par l'agent à partir de l'instant *t* par la somme suivante:

$$R_t = r_t + r_{t+1} + \cdots + r_{t+k-1} \qquad \text{(III.5)}$$

III.4.5 Fonction valeur

Pour mesurer la performance du comportement d'un agent qui est associé naturellement à un problème de la forme d'un *PDM*; on utilise l'appellation de la fonction valeur qui est l'espérance de gain si l'on suit la politique π à partir de l'état courant *s* [SUT98]. Néanmoins, on préfère souvent une espérance de gain pondéré qui fait apparaître dans le calcul du gain un coefficient de décroissance γ où : $\gamma \in]0,1[$. La fonction de retour devient alors:

$$R_t = \sum_{k=1}^{\infty} \gamma^k r_{t+k} \tag{III.6}$$

$$V_\pi(s) = E_\pi(R_t | s_t = s) = E_\pi\left(\sum_{k=1}^{\infty} \gamma^k r_{t+k} | s_t = s \right) \tag{III.7}$$

γ est le facteur de dépréciation (*discount factor*). Il permet de régler l'importance que l'on donne aux retours futurs par rapport aux retours immédiats.

Tel que si : - γ près de 1 : les retours futurs sont prisent en compte,

- γ près de 0 : les retours futurs sont négligés.

La formule (III.7) calcule une espérance de gain si l'on part de l'état initial $s_t = s$, et l'on suit la politique π, et en notant r_t la récompense obtenue au pas de temps t. Il faut toutefois noter que ce critère n'est pas la meilleure évaluation de la politique, mais il reste le plus pratique à mettre en œuvre. En plus d'état, d'action, fonction de récompense, et de fonction de probabilité de transition, un autre concept important de *PDM* est la fonction qualité $Q^\pi(s,a)$, qui inclut la fonction valeur d'état et d'action [SUT98]. Elle est définie pour une politique fixée π par:

$$Q^\pi(s,a) = E_\pi(R_t | s_t = s, a_t = a) \tag{III.8}$$

Avec une forte relation entre ces deux fonctions; où la fonction valeur pour un état s étant la moyenne des $Q^\pi(s,a)$; pondérées par la probabilité de chaque action :

$$V^\pi(s) = \sum_{s \in S} \pi(s,a) Q^\pi(s,a) \tag{III.9}$$

$$Q^\pi(s,a) = \sum_{s' \in S} P(s,a,s') \left[R(s,a,s') + \gamma V^\pi(s') \right] \tag{III.10}$$

La propriété essentielle de ces fonctions valeurs qui permet de construire les différents algorithmes d'apprentissage est la récurrence; où cette fonction peut se définir en fonction de la valeur de tous les états de l'ensemble S. Cette relation est connue par l'équation de Bellman.

III.4.6 Equation d'optimalité de Bellman

La fonction valeur permet de caractériser la qualité d'une politique, elle donne, pour chaque état, le retour futur si l'on suit cette politique. Elle permet également de comparer les politiques en définissant un ordre partiel définit sous la forme suivante [KAE96, SUT98, FIL04]:

$$\pi \geq \pi' \Leftrightarrow \forall s, V^\pi(s) \geq V^{\pi'}(s) \tag{III.11}$$

Avec si: $\pi = \pi'$, alors $\pi = \pi^*$

Cet ordre permet de définir la fonction valeur de la politique optimale que l'apprentissage par renforcement va chercher à estimer:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \tag{III.12}$$

C'est une fonction qui peut aussi s'exprimer pour un pair (état-action) :

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{III.13}$$

Avec la relation suivante:

$$Q^*(s,a) = E(r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a) \tag{III.14}$$

Il est également possible de décrire une relation de récurrence pour la fonction valeur optimale qui sera légèrement différente de l'équation de Bellman. On parle alors d'équation d'optimalité de Bellman, qui peut s'écrire:

$$\begin{aligned} V^*(s) &= \max_a E(r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a) \\ &= \max_a \sum_{s'} P(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right] \\ &= \sum_{s'} P(s,a,s') \left[R(s,a,s') + \gamma \max_{a' \in A's)} V^*(s',a') \right] \end{aligned} \tag{III.15}$$

Et de même pour la fonction qualité:

$$\begin{aligned} Q^*(s,a) &= E\left(r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1},a') | s_t = s, a_t = a \right) \\ &= \sum_{s'} P(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q^*(s',a') \right] \end{aligned} \tag{III.16}$$

Cette équation traduit par l'opérateur $\max_a$ le fait que la politique optimale choisit l'action qui va maximiser la récompense [KAE96, SUT98].

Si l'environnement est un *PDM* fini, ces équations de Bellman ont une solution unique [SUT98], qui va donner pour chaque état la récompense maximale que pourra recueillir l'agent à partir de cet instant. A partir de la connaissance de V^*, il est très simple de construire une politique optimale, en associant à chaque état les actions *a* qui permettent de réaliser le maximum de l'équation d'optimalité de Bellman (eq. III.17):

$$a' = \arg\max_a E(r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a) \tag{III.17}$$

Ou, si on utilise la fonction qualité :

$$a' = \arg\max_a Q^*(s,a) \tag{III.18}$$

Il faut cependant noter que la construction de la politique à partir de V^* nécessite la connaissance du modèle du monde afin d'estimer r_{t+1} et s_{t+1}. Cette connaissance est inutile si on utilise Q^*. Tout le problème pour l'apprentissage est d'estimer V^* ou Q^* pour en déduire une politique optimale. On peut remarquer que si l'on connaît complètement le problème (si on connaît $P(s,a,s')$ et $R(s,a,s')$), il est possible de calculer directement V^* ou Q^* en résolvant le système des équations d'optimalité de Bellman pour chaque état. Cette solution est peu applicable car l'environnement est en général inconnu et de plus elle ne correspond pas à des caractéristiques souhaitables de l'apprentissage tel que la capacité à apprendre par essais et erreurs.

III.4.7 Programmation dynamique (PD)

La programmation dynamique [BEL57] est un ensemble de méthodes permettant de calculer une politique optimale dans un *PDM* connu, en utilisant les propriétés des équations de Bellman, où le modèle de l'environnement est connu. L'objectif est d'estimer la fonction valeur optimale V^* afin d'en déduire une politique optimale π^* [FIL04]. On trouve deux méthodes: l'itération de la valeur et l'itération de la politique. Cette dernière a deux fonctions, qui sont l'évaluation et l'amélioration de la politique. Les valeurs de chaque état sont les entrées du processus d'amélioration de politique. Le but de l'amélioration de la politique est d'ajuster la politique selon les nouvelles valeurs d'état, et le résultat de cette amélioration est une nouvelle politique optimale [PRE07].

Si l'agent ne dispose pas d'un modèle de la dynamique de son environnement, il doit effectuer son apprentissage de la politique optimale en interagissant avec son environnement. A partir des conséquences de ses interactions, il déduira une estimation de la fonction valeur (ou de la fonction qualité) qu'il raffinera petit à petit, d'où il déduira une politique. C'est un apprentissage (en-ligne) alors que les algorithmes de la programmation dynamique que l'on a vu précédemment font un apprentissage (hors-ligne) sans interagir avec leur environnement.

III.4.8 Méthode de Monte-Carlo

L'estimation de la fonction valeur est réalisée à partir d'un ensemble de séquences (*état-action-récompense-état-action*) réalisées par l'agent. Pour les états de ces séquences, il est alors possible d'estimer V simplement par la moyenne des retours [FIL06]:

$$V^{\pi}(s) = \frac{1}{N(s)} \sum_{i=1}^{\infty} R(s) \qquad \text{(III.19)}$$

Où $N(s)$ est le nombre de séquences ou apparaît s , $R(s)$ est le retour après la première visite de l'état s , c'est à dire la somme pondérée des récompenses après cette visite. La méthode de Monte-Carlo fait l'estimation de la valeur de chaque état indépendamment; contrairement à la programmation dynamique qui doit estimer simultanément tout les états, ce qui permet par exemple de se concentrer sur des zones plus importantes de l'espace d'états. Cette méthode s'applique de la même façon pour une fonction $Q(s, a)$ [TOU99].

L'utilisation de Q et de la méthode de Monte-Carlo permet donc de découvrir la politique optimale sans aucun modèle de l'environnement, en utilisant uniquement des expériences réalisées dans cet environnement. Ceci suppose que tout ces couples (*s*,*a*) soient visités une infinité de fois ou ajouter un comportement d'exploration qui va assurer que toutes les actions seront réalisées avec un certaine probabilité [SUT91].

Les deux méthodes précédentes ont chacune un avantage important. La méthode de Monte-Carlo permet d'apprendre à partir d'expériences et la programmation dynamique pour sa part possède la propriété intéressante d'utiliser les estimations des états successeurs pour estimer la valeur d'un état, on parle de *"bootstrap"*. Cette caractéristique permet une convergence beaucoup plus rapide que la méthode de Monte-Carlo [FIL04]. Les méthodes d'apprentissage par différences temporelles vont réunir ces deux propriétés et constituent les méthodes les plus utilisées pratiquement [SUT88].

III.4.9 Méthodes de différence temporelle

L'algorithme élémentaire d'apprentissage par renforcement, dit algorithme de «différence temporelle» s'appelle aussi *TD(0)* [SUT88, SUT98, PRE07]. Cet algorithme repose sur une comparaison entre la récompense que l'on reçoit effectivement et la récompense que l'on s'attend à recevoir en fonction des estimations construites précédemment.

Si les estimations des fonctions valeur aux états s_t et s_{t+1}, notées $V^\pi(s_t)$ et $V^\pi(s_{t+1})$, étaient exactes, on aurait :

$$V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots \quad \text{(III.20)}$$

$$V^\pi(s_{t+1}) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \quad \text{(III.21)}$$

On aura donc : $V^\pi(s_t) = r_t + \gamma V^\pi(s_{t+1}) \quad \Rightarrow \quad \underbrace{r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)}_{TD} = 0$ (III.22)

On voit que l'erreur de différence temporelle mesure l'erreur entre les valeurs effectives des estimations $V^\pi(s_t)$ et les valeurs qu'elles devraient avoir $V^\pi(s_{t+1})$. En fait, pendant l'apprentissage de la fonction V^π, *TD* n'est pas nul. Si l'estimation de V^π est trop optimiste en s ; *TD* est négative. Et si elle est trop pessimiste; le *TD* est positive.

Aussi, *TD* peut être utilisé comme une correction à apporter à $V^\pi(s_t)$. En notant V_t^π l'estimation de la fonction valeur à l'instant t, on peut corriger cette estimation comme suit (eq. III.23):

$$V_{t+1}^\pi \leftarrow \begin{cases} V_t^\pi(s) + \beta\left[r_t + \gamma V_t^\pi(s_{t+1}) - V_t^\pi(s)\right] & \text{pour} \quad s = s_t \\ V_t^\pi(s) & \text{sinon} \end{cases} \quad \text{(III.23)}$$

Où $\beta \in [0,1]$ est un cœfficient d'apprentissage.

III.4.9.1 *Algorithme TD(0)*

C'est un algorithme d'évaluation de la politique; dans lequel le taux d'apprentissage β doit varier. En effet, lors des premiers épisodes les corrections sont importantes car l'estimation de V^π est mauvaise; cependant, au fur et à mesure des épisodes, la précision de cette estimation augmente et les corrections deviennent moins significatives. Par contre, du fait de non déterminisme de l'environnement, les corrections peuvent être importantes alors que l'estimation de V^π est bonne. Pour cela β doit être initialement proche de 1, et diminue ensuite pour tendre vers 0. Pour un *PDM* fixe et une politique fixe, l'algorithme *TD(0)* converge asymptotiquement vers V^π à condition que [WAT89, SUT98]:

- chaque état soit visité une infinité de fois théoriquement,
- Le taux d'apprentissage respecte les conditions suivantes:

$$\sum_t \beta_t(s) = +\infty \quad \text{et} \quad \sum \beta_t^2(s) < +\infty, \forall s \in S$$

et : $$\beta_t(s) = \frac{1}{1 + \textit{nombre de visites de l'état } s \textit{ depuis le début de l'épisode}} \quad \text{(III.24)}$$

III.4.9.2 *Algorithme SARSA*

SARSA est l'une des trois méthodes principales de l'apprentissage par renforcement. Les deux autres méthodes sont le Q-Learning et l'Acteur Critique [SUT98]. Toutes ces méthodes sont basées sur les Différences Temporelles (*TD*); calculées en termes de changements observés sur l'environnement de l'agent, et employées pour mettre à jour la fonction valeur d'état et la fonction valeur d'action. L'algorithme SARSA est similaire à l'algorithme TD(0), il travaille sur les valeurs des couples *(s,a)* plutôt que sur la valeur des états [PRE07]. Son équation de mise à jour est la suivante:

$$Q(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \beta\left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \quad \text{(III.25)}$$

L'information nécessaire pour réaliser une telle mise à jour est la réalisation d'une transition (s_t, a_t, r_t, s_{t+1}, a_{t+1}), d'où découle le nom de l'algorithme. Effectuer ces mises à jour implique que l'agent détermine avec un pas de regard en avant quelle est l'action a_{t+1} qu'il réalisera lors du pas de temps suivant, lorsque l'action a_t dans l'état s_t l'aura conduit dans l'état s_{t+1}. L'algorithme SARSA converge vers la politique optimale si toutes les paires *(état-action)* sont visitées une infinité de fois théoriquement (mais en pratique c'est pour un grand nombre), et la stratégie du choix d'action tend vers une stratégie gloutonne avec $\varepsilon = \frac{1}{t}$.

$Q(s,a) \leftarrow 0, \ \forall (s,a) \in (S,A)$
Pour un grand nombre **faire**
  Initialiser l'état initial s_0
  $t \leftarrow 0$
  Choisir l'action à émettre a_t en fonction de la politique de Q (ε gloutonne)
  Emettre l'action choisie a_t
  Répéter
    Emettre a_t
    Observer r_t et s_{t+1}
    Choisir l'action à émettre a_t en fonction de la politique de Q
    $Q(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \beta\left[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$
    $t \leftarrow t+1$
  Jusqu'à $s_t \in F$
Fin pour

Algorithme III.1 Algorithme SARSA

III.4.9.3 *Algorithme Q-Learning*

L'algorithme Q-learning se présente comme une simplification de l'algorithme SARSA, Il s'agit sans doute de l'algorithme d'apprentissage par renforcement le plus connu et le plus utilisé en raison des preuves formelles de convergence et des applications existantes [WAT92, SUT98]. Le principe du Q-learning consiste à apprendre la fonction qualité en interagissant avec l'environnement. Trois fonctions principales participent au Q-learning (figure III.9): une fonction d'évaluation, une fonction de renforcement et une fonction de mise à jour [TOU99]. A partir de la situation actuelle telle qu'elle perçue par l'agent, la fonction d'évaluation propose

une action en se basant sur la connaissance disponible au sein de la mémoire interne. Cette connaissance est stockée sous forme de valeur d'utilité (qualité) associée à une paire (situation-action). L'action sélectionnée est celle qui présente la meilleure probabilité de renforcement. Cette proposition d'action est cependant altérée pour permettre l'exploration de l'espace des paires état-action. Après l'exécution de l'action dans le monde réel, la fonction de renforcement est utilisée par la fonction de mise à jour pour ajuster la valeur Q associée à la paire situation-action qui vient d'être exécutée. L'apprentissage est incrémental car l'acquisition des exemples est réalisée séquentiellement dans le monde réel [KAE96, SUT98].

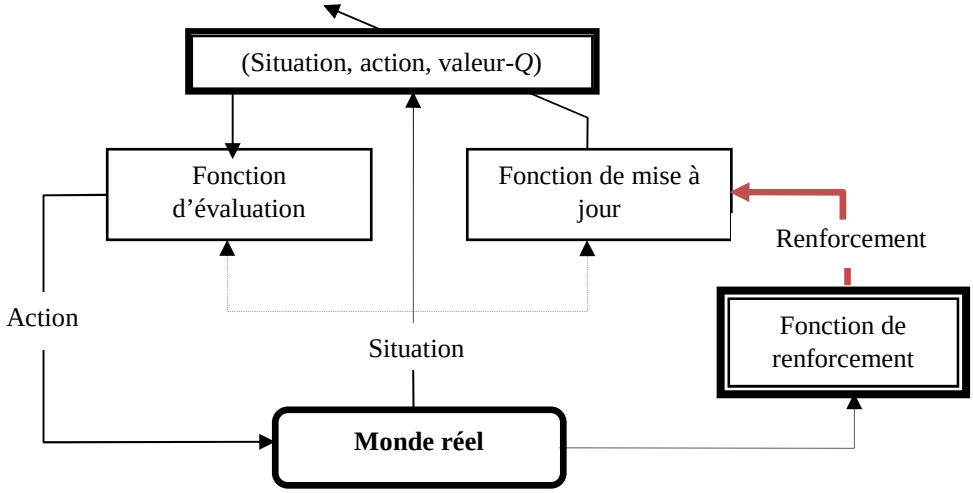


Figure III.9 Modèle de l'algorithme Q-learning

- ***Fonction d'évaluation:*** Les résultats de l'algorithme Q-learning sont stockés dans la fonction qualité (état-*action*), qui est souvent sous la forme de table à deux dimensions : situation et action. La fonction d'évaluation parcourt, pour une situation donnée, les valeurs de Q associées aux actions et sélectionne l'action avec la plus grande qualité. Au début lorsque la table ne contient pas suffisamment de données, une composante aléatoire est ajoutée de façon à ne pas restreindre les actions éligibles au petit nombre des actions déjà essayées. Au fur et à mesure que la table se remplit, cette composante aléatoire est réduite afin de permettre l'exploitation des informations reçues et d'obtenir une bonne performance.
- ***Fonction de renforcement:*** Cette fonction fournit pour chaque paire une évaluation qualitative de son intérêt par rapport au comportement désiré par l'opérateur. Il est important de remarquer que le retour de la fonction de renforcement estime l'intérêt de la situation perçue, ce retour est utilisé pour définir l'utilité d'effectuer l'action précédente dans la situation précédente.
- ***Fonction de mise à jour:*** Le principe de Q-learning est de mettre à jour itérativement la fonction courante $Q^{\pi}(s_t, a_t)$ à la suite de chaque transition (s_t, a_t, r_t, s_{t+1}), Cette mise à jour se fait sur la base de l'observation des transitions instantanées et de leur récompenses associées par l'équation suivante [SUT98] :

$$Q(s_t,a_t) \leftarrow Q^{\pi}(s_t,a_t) + \beta\left[r_t + \gamma \max_{a \in A(s)} Q(s_{t+1},a_t) - Q(s_t,a_t)\right] \qquad \text{(III.26)}$$

La convergence de cet algorithme est établie en [WAT89, WAT92]. Si l'on dispose de mise à jour de la fonction Q par l'équation III.26, et si l'on dispose d'une représentation sous

forme de table; la fonction $Q^{\pi}(s_t, a_t)$ converge presque sûrement vers $Q^{\pi}(s_t, a_t)$ optimale avec les conditions de l'équation (III.24). En comparant SARSA et Q-learning, on voit que SARSA utilise dans la mise à jour de Q l'action qui va être émise à l'itération suivante (eq. III.25). Au contraire, le Q-learning choisit une action à émettre et met à jour Q en fonction de la meilleure action dans l'état suivant (eq. III.26). Dans cet algorithme, le taux d'apprentissage $\beta_t(s,a)$ est propre à chaque paire état-action, et décroît vers 0 à chaque passage pour une convergence rapide.

$Q(s,a) \leftarrow 0, \ \forall (s,a) \in (S,A)$
Pour un grand nombre **faire**
  Initialiser l'état initial s_0
  $t \leftarrow 0$
  Répéter
    Choisir l'action à émettre a_t en fonction de la politique de Q
    Emettre l'action choisie.
    Observer r_t et s_{t+1}
    $Q(s_t,a_t) \leftarrow Q^{\pi}(s_t,a_t) + \beta\left[r_t + \gamma \max_{a \in A(s_{t+1})} Q(s_{t+1},a') - Q(s_t,a_t)\right]$
    $t \leftarrow t+1$
  Jusqu'à $s_t \in F$
Fin pour

Algorithme III.2 Q-Learning

Une autre méthode bien connue dans l'apprentissage par renforcement est l'acteur-critique (ACRL) [SUT98]. ACRL a une structure plus complexe que SARSA et Q-learning. Pour SARSA et Q-Learning, les politiques optimales sont stockées dans des fonctions qualités. Mais pour ACRL, la politique et la valeur d'état sont stockées séparément. Bien que ceci augmente la complexité de la méthode et la rendre difficile à analyser. L'unité employée pour stocker la politique est appelé (*Acteur*) et l'unité de stockage de la fonction de valeur d'état est désignée sous le nom du (*critique*). Pour une action, si la différence temporelle TD est positive, on choisit l'action qui devrait renforcer l'agent. Sinon, la préférence de choisir cette action devrait être diminuée.

III.4.10 Choix d'action (dilemme Exploration-Exploitation)

Le choix et la génération de l'action est un problème majeur en commande. Il doit garantir un équilibre entre l'exploration et l'exploitation de l'apprentissage déjà réalisé; c-à-d faire confiance à l'estimation courante de la qualité Q pour choisir la meilleure action à effectuer dans l'état courant (*exploitation*) où, au contraire, choisir une action à priori sous optimale pour observer ses conséquences (*exploration*). Naturellement, on conçoit intuitivement que l'exploration doit initialement être importante (quand l'apprentissage est encore très partiel, on explore) puis diminuer au profit de l'exploitation quand l'apprentissage a été effectué pendant une période suffisamment longue [KAE96, SUT98, PRE07]. Plusieurs stratégies sont utilisées classiquement:

III.4.10.1 ***Gloutonne (greedy):*** consiste toujours à choisir l'action estimée comme la meilleure, soit:

$$a\_gloutonne = \arg\max_{a \in A(s)} Q(s_t, a) \tag{III.27}$$

III.4.10.2 ε ***-gloutonne (*** ε ***-greedy):*** consiste à choisir l'action gloutonne avec une probabilité ε et à choisir une action au hasard avec une probabilité $1-\varepsilon$, soit:

$$a_{\varepsilon-gloutonne} = \begin{cases} \arg\max_{a \in A(s_t)} Q(s_t, a) \text{ avec probabilté } \varepsilon \\ \text{action prise au hasard dans } A(s_t) \text{ avec probabilité } 1-\varepsilon \end{cases} \tag{III.28}$$

Un cas particulier est la sélection 0-gloutonne qui consiste à choisir une action au hasard.

III.4.10.3 ***Softmax:*** consiste à choisir une action en fonction de sa qualité par rapport à la qualité des autres actions dans le même état, soit: on associe par exemple à chaque action la probabilité d'être émise selon l'équation suivante :

$$\Pr\left[a_t \middle| s_t\right] = \frac{Q(s_t, a_t)}{\sum_{a \in A(s_t)} Q(s_t, a)} \tag{III.29}$$

III.4.10.4 ***Boltzmann:*** c'est un cas particulier de sélection *softmax* : la probabilité est calculée avec l'équation suivante qui produit une distribution dite de Boltzmann:

$$\Pr\left[a_t \middle| s_t\right] = \frac{e^{\frac{Q(s_t, a_t)}{\tau}}}{\sum_{a \in A(s_t)} e^{\frac{Q(s_t, a_t)}{\tau}}} \tag{III.30}$$

Où τ est un réel positif. Si τ est grand, cette méthode tend vers la méthode gloutonne, et si τ est proche de 0, elle tend alors vers une 0-gloutonne.

III.4.11 Traces d'éligibilité

Les méthodes utilisant les différences temporelles ne prennent en compte le retour immédiat que pour la dernière transition pour modifier la fonction valeur. Donc, on peut propager ces conséquences aux transitions précédentes, le long de la trajectoire [SUT96]. Cela est fait par l'utilisation de n pas du futur, qui conduit à la méthode des différences temporelles à n pas. Dans cette méthode, le retour est estimé par une équation de la forme (eq. III.31):

$$R_t = R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^n r_{t+n+1} \gamma V(s_{t+n+1}) \tag{III.31}$$

La mise à jour se ferait alors par l'équation:

$$V(s_t) \leftarrow V(s_t) + \beta\left[R_t^n - V(s_t)\right] \tag{III.32}$$

Ce cas particulier est intéressant car il peut s'appliquer simplement en utilisant les valeurs passées des récompenses, au lieu des valeurs futures. On utilise pour cela une valeur auxiliaire appelée "*trace d'éligibilité"* que l'on définit de la manière récursive suivante [SUT98, FIL04]:

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & si \quad s \neq s_t \\ \gamma\lambda e_{t-1}(s) + 1 & si \quad s = s_t \end{cases} \tag{III.33}$$

L'utilisation de cette notion est une généralisation des algorithmes. Cette éligibilité est quantifiée par le niveau de responsabilité: le dernier état ou la dernière paire état-action est la plus responsable, donc possède l'éligibilité la plus grande. La mise à jour des valeurs se fait alors pour chaque état proportionnellement à la valeur de son éligibilité (eq. III.34):

$$V(s_t) \leftarrow V(s_t) + \beta e(s_t)[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \quad \text{(III.34)}$$

L'idée de cette mise à jour est de remplacer la mise à jour d'un état en utilisant des récompenses futures par la mise à jour des états passés en utilisant la récompense courante. L'algorithme $Q(\lambda)$ est présenté par l'algorithme III.3. Cet algorithme se décline simplement pour l'extension SARSA(λ).

Initialiser $Q^{\pi} \leftarrow 0$

Pour un grand nombre faire

$e(s,a) \leftarrow 0, \forall (s,a) \in S \times A$

$t \leftarrow 0$

Initialiser l'état initial s_0

Choisir l'action à émettre a_0

Répéter

Emettre l'action $a_t = \pi(s_t)$

Observer r_t et s_{t+1}

Choisir l'action à émettre a_0

$a^* \leftarrow \arg\max_{a \in A(s_{t+1})} Q(s_{t+1}, a)$

$\delta \leftarrow r_t + \gamma Q(s_{t+1}, a^*) - Q(s_t, a_t,)$

$e(s_t, a_t) \leftarrow e(s_t, a_t) + 1$

Pour $(s;a) \in S \times A$ **faire**

$Q(s,a) \leftarrow Q(s,a) + \beta \delta e(s,a)$

Mettre à jour de l'éligibilité de *(s,a)*

Fin pour

$t \leftarrow t+1$

Jusqu'à $s_t \in F$

Fin pour

Algorithme III.3 l'Algorithme Q(λ)

III.4.12 Applications et limitations de l'apprentissage par renforcement

L'apprentissage par renforcement a démontré des performances intéressantes et des solutions prometteuses dans plusieurs domaines et notamment en contrôle [SUT91, GLO96, BOU01]. Il est considéré comme une approche de la commande optimale qui peut être utilisée dans plusieurs domaines: la commande, les jeux, l'allocation des ressources, l'ordonnancement des tâches, prédiction,…etc. L'apprentissage par renforcement est utilisé aussi dans la robotique mobile; ou il est défini comme une modification automatique du comportement du robot, l'objectif est d'améliorer ce comportement dans son environnement pour accomplir des tâches variées: convergence vers un but, évitement d'obstacle, suivi de murs, exploration, ramassage…etc [CAN03, RIS05, CHE09, KHR11]. Pour utiliser l'apprentissage par renforcement, plusieurs approches sont possibles. La première consiste à discrétiser manuellement le problème afin de fournir des espaces d'états qui pourront être utilisés directement par des versions naïves des algorithmes (en utilisant des tableaux de *Q*). Il faut cependant bien faire attention au choix des discrétisations afin qu'elles permettent un apprentissage correct en fournissant des états et des actions contenant notamment des

récompenses cohérentes [SUT98]. La seconde méthode va permettre de travailler directement dans les espaces d'états et d'actions continus en utilisant des méthodes d'approximation des fonctions. En effet, pour utiliser l'apprentissage par renforcement, il est nécessaire d'estimer correctement la fonction qualité. Cette estimation peut se faire directement par un approximateur de fonction continu comme les réseaux de neurones ou les systèmes d'inférence flous [BER92, GLO96, JOU98]. L'utilisation de ces approximateurs permet de travailler directement dans l'espace continu et de limiter les effets de parasites qui pourraient apparaitre suite à un mouvais choix de discrétisation.

III.5 Conclusion

Ce chapitre est divisé en deux parties; dans la première partie, nous avons effectué une présentation générale des réseaux de neurones. Ils constituent une véritable solution pour la résolution de plusieurs problèmes, là ou les méthodes classiques ont montrés leurs limites. L'utilisation des réseaux de neurones pour la commande peut se justifier par la simplicité de mise en œuvre, par leur capacité d'approximation universelle et par la possibilité de considérer le processus comme une boite noire. Grâces à ces propriétés intéressantes, l'application des RNA pour la commande en robotique mobile présente un intérêt considérable. La deuxième partie est consacrée à l'apprentissage par renforcement, c'est une méthode d'apprentissage par expérience pour définir une action optimale à effectuer dans une situation donnée afin de maximiser les récompenses. C'est une méthode d'apprentissage orientée objectif qui va conduire à un contrôleur optimal pour la tâche spécifiée par les récompenses. Elle permet aussi de résoudre les problèmes de récompense retardée pour lesquels il faut apprendre à sacrifier une récompense à court terme pour obtenir une plus forte récompense à long terme, et donc apprendre de bonnes séquences d'actions qui permettront de maximiser la récompense à long terme. Les algorithmes d'apprentissage par renforcement sont basés sur les processus markoviens et demandent de discrétiser l'environnement en couples (état-action). Pour parfaitement modéliser un problème réel, la taille de la matrice V ou Q est proportionnelle au nombre de combinaisons états-actions possibles. Il est évident qu'il est impossible de stocker autant de données sur un système embarqué. De plus l'apprentissage serait extrêmement long, puisqu'il demanderait à l'agent de visiter plusieurs fois chaque combinaison état-action possible.

Même si ces méthodes d'entrainement semblent mal indiquées pour l'apprentissage de comportements réactifs continus, elles peuvent être utilisées avec d'autres méthodes pour la résolution de ces problèmes. En effet, dans la pratique, les réseaux de neurones et l'apprentissage par renforcement peuvent être des choix possibles pour plusieurs problèmes et notamment en robotique mobile (la navigation d'un robot mobile). Pour cela, on s'intéresse dans ce travail à utiliser premièrement ces deux approches seules, et par la suite on va utiliser des systèmes hybrides en utilisant les comportements flous (neuro-flous et renforcement-flous).

Dans le chapitre suivant, nous présenterons l'utilisation des systèmes d'inférences flous (structures basés comportements flous) pour contrôler et planifier la trajectoire du robot mobile en présence des obstacles pour une tâche de navigation autonome.

Chapitre IV

Navigation Autonome par des Contrôleurs Flous

IV.1 Introduction

Comme on a vu dans le chapitre 2, la théorie de la logique floue est caractérisée par la capacité de modéliser et de traiter des informations incertaines et imprécises. Dans la majorité des applications de logique floue pour la navigation des robots mobiles, le modèle mathématique de la dynamique de l'environnement du robot n'est pas nécessaire dans le processus de conception du contrôleur de mouvement. Puisque la logique floue est un outil mathématique puissant qui permet de manipuler le raisonnement humain, les concepts et les termes linguistiques [MEN95, JAN07]. La connaissance de l'opérateur humain se présentera sous la forme d'un ensemble de règles linguistiques floues de type Si-Alors. Ces règles produisent une décision approximative de la même manière comme un expert [SAF97, YAN05].

D'autre part, les systèmes de navigation basés sur les comportements (*Behavior based navigation systems*) [BRO86, ARK89, SER02, FAT06] ont été proposées comme une alternative à la stratégie la plus traditionnelle de commande qui est basée sur la construction d'une représentation approximative de l'environnement de navigation des robots mobiles [KHA86], l'information acquise conduit à une étape de raisonnement puis agir sur le monde externe. L'idée principale de la navigation d'un robot mobile basée sur les comportements est d'identifier des réponses différentes (comportements) selon les données sensorielles. Une variété de systèmes de contrôle basés sur le comportement sont inspirés par le succès de Brooks [BRO86] avec son architecture, qui est connue par l'architecture de *subsumption*. Grace à ces propriétés intéressantes, plusieurs travaux ont utilisés la logique floue comme approche de commande pour la navigation d'un robot mobile et pour la représentation des comportements [ROS89, SAF97, MAT97, MAA00, SER02, ZAV03, CAN03, SIM04, BOT05, FAT06].

Dans ce chapitre, on va présenter des architectures de navigation réactive d'un robot mobile autonome dans des environnements inconnus. Il s'agit de contrôler l'évolution du robot dans des environnements caractérisés par leurs complexité (ils peuvent être vastes, imprécis, dynamiques ou inconnus,…). Ce qui nous amène à la définition de certaines comportements élémentaires tels que: *la convergence vers un but, suivi de murs, évitement d'obstacles, poursuite de trajectoire ou d'une cible mobile,*…etc. Nous nous sommes intéressés à l'utilisation des comportements flous pour le déplacement du robot mobile d'une position initiale vers une quelconque destination désirée en évitant les obstacles, tout en respectant les contraintes cinématiques du robot et de façon autonome.

IV.2 Navigation basée sur les comportements

L'un des aspects les plus difficiles dans le domaine de la robotique mobile est la possibilité de naviguer de manière autonome pour rejoindre un but visé, en évitant les obstacles modélisés et non modélisés, et en particulier dans un environnement encombré d'objets statiques ou dynamiques imprévisibles [CUE05]. Une méthode très efficace peut être utilisée pour accomplir la tâche de navigation est la structure basée sur les comportements [BRO86, ARK89, ROS89]. Le principe de base du système de navigation basé sur les comportements est de subdiviser la tâche de navigation globale en un ensemble des comportements élémentaires d'action (*comportement 1, comportement 2,…, comportement n*); simple à concevoir et à gérer. Cette architecture peut régler le problème de conflit des actions dans les approches traditionnelles.

Ces comportements sont indispensables à l'exécution des sous-tâches spécifiques pour le robot mobile, qui peuvent être par exemple (éviter les obstacles, converger vers le but, suivre un mur,…). Dans l'architecture proposée par Brooks [BRO86] appelé *subsumption*, qui est présentée sur la figure IV.1, les comportements sont arrangés dans les niveaux de priorité. En cas de déclenchement, le comportement de plus haut niveau supprime tout les comportements de niveau inférieur. Arkin [ARK89] a décrit l'utilisation de comportements réactifs appelés "*motor-schemas*". Dans cette architecture, la méthode de champ de potentiel est utilisée pour définir la sortie de chaque bloc. Puis, toutes les sorties calculées sont combinées par une somme pondérée. Rosenblatt [ROS95] a présenté l'architecture *DAMN* (*Distributed Architecture for Mobile Navigation*). Elle consiste à utiliser une architecture distribuée multi-comportements (agents), chaque comportement représente une fonction bien particulière avec la caractéristique que ces agents ne peuvent pas discuter directement entre eux mais doivent passer par une sorte d'ordonnanceur (tableau noir ou un arbitre) qui gère leurs communications et leurs états.

L'architecture subsumption est très efficace puisque le système est de nature modulaire, ce qui simplifie à la fois la résolution de problème de planification de trajectoire du robot mobile, et offre la possibilité de supprimer ou d'ajouter de nouveaux comportements pour le système sans provoquer d'augmentation importante de la complexité. Les sorties proposées de chaque comportement actif simultanément sont ensuite rassemblés selon une règle de coordination des actions inférées. La tâche de contrôle se réduit alors à un couplage des entrées sensorielles avec les actionneurs (c-à-d trouver un lien entre la perception et l'action [REI94]) en utilisant des blocs de comportements sous formes des contrôleurs réactifs pour le robot mobile. Chaque comportement peut avoir des entrées provenant des capteurs du robot (caméra, ultrasons, infrarouges, tactiles ...) et/ou d'autres comportements dans le système, et envoyer des sorties aux actionneurs du robot (moteurs, roues, pinces, bras ...) et/ou à d'autres comportements. D'autres auteurs ont utilisés la logique floue pour la représentation et la coordination des comportements comme [CAN01, BOT05, FAT06, WAN08]. La figure IV.2 présente la structure d'un système de contrôle basé comportements flous utilisé pour la commande des robots mobiles avec un système de supervision des comportements.

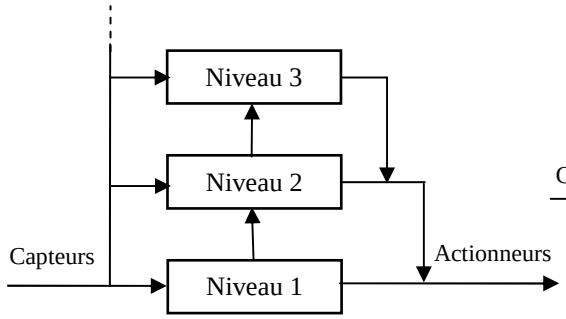


Figure IV.1 Structure subsumption

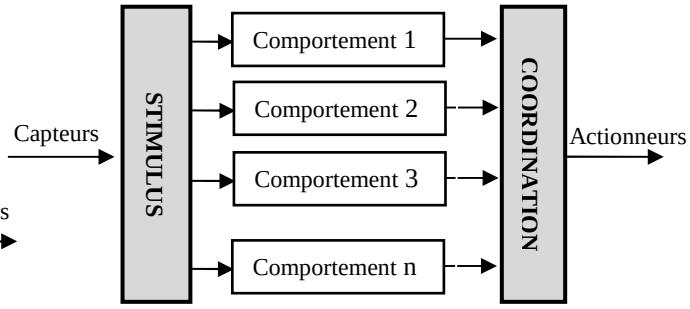


Figure IV.2 Architecture basée sur les comportements

IV.3 Navigation basée sur les comportements flous

Dans la théorie de contrôle flou, les commandes inférées sont traités en trois étapes: fuzzification, inférence et déffuzification. Le système de contrôle du robot mobile basé sur des comportements flous, est décomposé en sous-tâches simples (comportements indépendants), ou chaque bloc est considéré comme un contrôleur flou définit par un ensemble de règles floues visant à atteindre un objectif bien défini (éviter obstacle, chercher la cible, suivre une trajectoire,…) [YAN05, FAT06, WAN08]. Les règles linguistiques peuvent être par exemple:

$\boldsymbol{R_1}$: Si le but est Loin Alors la vitesse est Grande et le braquage est Zéro,
$\boldsymbol{R_2}$: Si l'obstacle est à droite Alors la vitesse est Moyenne et le braquage est à Gauche,
…
$\boldsymbol{R_n}$: Si *le but* est Proche Alors *la vitesse* est Faible et *le braquage* est Zéro.

Dans ce travail, on utilise des stratégies de contrôle flou pour la navigation d'un robot mobile de type tricycle [CHE12b, CHE13c]. On adopte des structures basées comportements flous comme présentées sur la figure IV.2 illustrant la décomposition de la tâche de navigation globale en un ensemble des comportements élémentaires. Grace aux propriétés importantes et efficacité pour la commande des robots mobiles, les avantages des comportements flous viennent notamment de leurs capacités à :

- ✓ Formaliser et simuler l'expertise d'un opérateur dans la conduite et la commande des procédés où les connaissances sont formulées sous forme d'une base de règles de type: *Si-Alors*,
- ✓ Maitriser les systèmes non linéaires et difficiles à modéliser par ces propriétés d'approximation universelles,
- ✓ Prendre en compte sans discontinuité des cas ou exceptions de natures différentes, et les intégrer au fur et à mesure dans l'expertise,
- ✓ Prendre en compte plusieurs variables et effectuer la fusion pondérée des grandeurs d'influences, et grâce à des processeurs efficaces, on peut régler des processus rapides.

Les contrôleurs flous utilisés sont de type Takagi-Sugeno d'ordre zéro (TS0) qui permettent une simplification de calcul et efficacité de contrôle. La sortie obtenue n'est pas floue, ce qui supprime une étape dans l'inférence.

Pour cela, afin de faciliter l'application, il est préférable d'utiliser quelques contraintes qui ne sont pas restrictives mais permettent une simplification :

- La structure du contrôleur flou qui définit le comportement est fixée à priori par l'utilisateur (le type et le nombre des ensembles flous, les règles et les conclusions,..),
- Les partitions floues sur chaque domaine d'entrée sont des partitions floues fortes triangulaires, c'est-à-dire : $\forall x, \sum_{i=1} \alpha_i(x) = 1$,
- La T-norme est utilisée pour la conjonction des prémisses.

IV.4 Modèle du robot mobile

Le robot mobile utilisé dans la simulation est de type tricycle (figure IV.3), possédant deux roues motrices arrière commandées par un seul moteur muni d'un encodeur odométrique pour la mesure de la position et la vitesse longitudinale. La roue avant est une roue orientable commandée par un deuxième moteur. Cette roue assure la stabilité du robot et elle est équipée par un capteur d'orientation permettant la mesure de l'angle de braquage de châssis du robot. On suppose que le robot mobile dans le cas réel est muni d'un ensemble des capteurs pour la mesure des variables à utiliser selon la tâche à accomplir. Dans ce travail, nous utilisons deux systèmes de coordonnées; le système XOY dans l'espace et le système xoy du robot mobile. La mission de base du robot mobile est d'atteindre un but désiré dans son environnement de navigation en évitant les obstacles qu'il contourne de façon autonome. Le système de navigation autonome doit générer les actions de mouvement: angle de braquage de la roue avant orientable α et la vitesse linéaire du châssis V_r.

La figure IV.3 présente la configuration du robot mobile utilisé et les variables calculées dans l'espace de configuration.

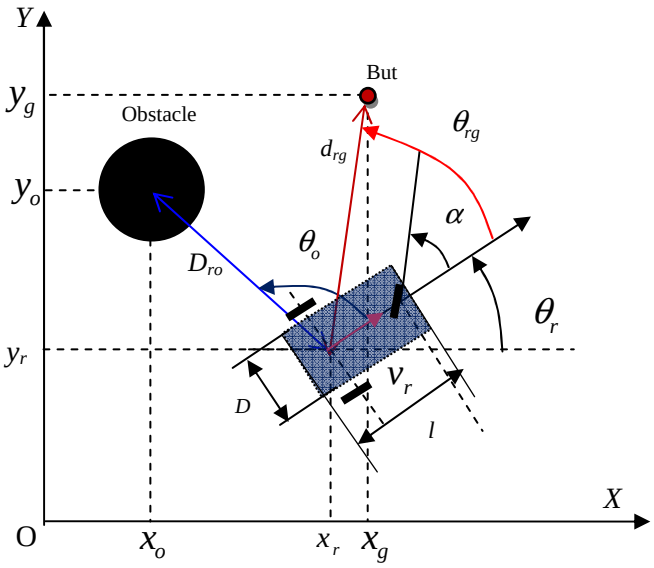


Figure IV.3 Configuration du robot mobile utilisé

Le modèle cinématique du robot mobile est donné par [CAN03, SIM04] :

$$\begin{aligned} \dot{x}_r &= V_r \cos(\theta_r) \\ \dot{y}_r &= V_r \sin(\theta_r) \\ \dot{\theta}_r &= \frac{V_r}{l} tg(\alpha) \end{aligned} \qquad \text{(IV.1)}$$

Dans notre travail, on utilise la version discrète de ce modèle qui est comme suit :

$$\begin{aligned} x_r(k+1) &= x_r(k) + \Delta t V_r(k) \cos(\theta_r(k)) \\ y_r(k+1) &= y_r(k) + \Delta t V_r(k) \sin(\theta_r(k)) \\ \theta_r(k+1) &= \theta_r(k) + \Delta t \frac{V_r(k)}{l} tg(\alpha(k)) \end{aligned} \qquad \text{(IV.2)}$$

où : - θ_r : l'orientation du robot,

- d_{rg} : la distance entre le robot et la cible (but),
- θ_{rg} : l'angle entre l'orientation actuelle du robot et celle de la cible,
- l: la longueur du châssis,
- D: la largeur du robot,
- D_{ro}: la distance entre le robot et l'obstacle,
- θ_o : l'angle entre l'orientation du robot et de l'obstacle,
- α et V_r sont les commandes du robot mobile (l'angle de braquage et la vitesse linéaire de translation du robot respectivement).

IV.5 Conception des comportements flous

Pour permettre au robot mobile de se déplacer de manière autonome dans des environnements inconnus encombrés d'obstacles, dans cette section, on va présenter l'approche étudiée et les structures de navigation du robot mobile à roues en utilisant des comportements flous de Takagi-Sugeno d'ordre zéro. Cette méthode de conception est motivée par l'efficacité de commande et la simplicité pour rendre les contrôleurs plus appropriés en implémentation temps réel. Le système de commande global du robot mobile est basé sur le modèle cinématique du centre du robot (eq. IV.1); où les contrôleurs flous permettent de générer les deux actions de mouvement (angle de braquage de la roue avant α et vitesse de déplacement du robot V_r). Ces commandes sont liées par des comportements élémentaires de base pour le robot mobile: recherche de la cible et l'évitement d'obstacles. Les actions générées sont basées sur les mesures des capteurs du robot mobile pour la détection de la position de but ou pour la mesure des distances aux obstacles. Les comportements conçus sont à structures fixes, comportent un minimum de règles floues pour être simple à interpréter, lisible et avec efficacité de commande. Chaque comportement est représenté par une base de règle avec deux entrées x_1, x_2 et deux sorties sous la forme suivante:

$$Si \;\; x_1 \; est \; A^i_1 \;\; Et \;\; x_2 \; est \; A^i_2 \;\; Alors \;\; \alpha \; est \; B_1{}^i \;\; Et \;\; V_r \; est \; B_2{}^i \qquad \text{(IV.3)}$$

Pour le contrôleur flou de type T-S d'ordre 0, les valeurs symboliques des conclusions sont réduites à des constantes ou valeurs numériques pour α et V_r. Les sorties inférées deviennent alors :

$$V_r = \sum_{i=1}^{N} w_i V_{ri} \quad \text{et} \quad \alpha = \sum_{i=1}^{N} w_i \alpha_i \tag{IV.4}$$

où w_i représente la valeur de vérité de la règle *i* calculée par la T-norme selon l'équation suivante:

$$w_i = \mu_{A_1}(x_1) \times \mu_{A_2}(x_2) \tag{IV.5}$$

Avec A^i_1 et A^i_2 sont les fonctions d'appartenances des entrées x_1 et x_2.

IV.5.1 Comportement de navigation vers un but (recherche d'une cible)

Ce comportement permet de réaliser l'action "*convergence vers le but*" par le robot à partir de la connaissance de sa position courante et la définition d'une position relative à atteindre dans l'environnement de navigation (c-à-d, faire aligner le robot vers la direction de but). La définition du point à atteindre est effectuée par l'intermédiaire de deux variables d'entrées; la distance entre le robot et le but (d_{rg}) et l'angle entre l'orientation actuelle du robot et celle de but θ_{rg} (figure IV.3). Le schéma fonctionnel de contrôle proposé pour le robot est donné par la figure IV.4. Le module de calcul compare les coordonnées réelles du robot avec les coordonnées de but afin de calculer la distance robot-but et l'angle désiré θ_d pour le rejoindre. Puis cette valeur d'angle est comparée avec l'orientation actuelle du robot pour calculer l'angle entre l'axe du robot et de but (θ_{rg}) en utilisant les équations suivantes :

$$d_{rg} = \sqrt{(x_g - x_r)^2 + (y_g - y_r)^2} \tag{IV.6}$$

$$\theta_d = arctg\left(\frac{y_g - y_r}{x_g - x_r}\right) \tag{IV.7}$$

$$\theta_{rg} = \theta_d - \theta_r \tag{IV.8}$$

Le contrôleur flou proposé utilise ces deux variables (d_{rg} et θ_{rg}) pour générer les deux valeurs de commande (u_1 est l'angle de braquage de la roue avant α, et u_2 est la vitesse de déplacement du robot V_r). Les fonctions d'appartenances utilisées pour fuzzifier les variables d'entrée sont données sur les figures IV.5 et IV.6 respectivement avec des formes triangulaires et trapézoïdale. Les deux actions de commande sont réduites à des singletons (figures IV.7 et IV.8). Les labels désignant les noms des variables linguistiques entrée-sortie sont: Z(Zéro), P(Petite), M(Moyenne), G(Grande), TG(Très Grande), NG(Négative Grande), NM(Négative Moyenne),NP(Négative Petite), PP(Positive Petite), PM(Positive Moyenne) et PG(Positive Grande), F(Faible).

Sur la base de la description du système à commander avec des variables linguistiques et de la définition des fonctions d'appartenance pour les variables d'entrée et de sortie, on peut établir les règles d'inférence. L'étape d'inférence fait appel à 35 règles représentant le lien entre les différentes variables entrée-sortie. La table (IV.1) décrit les règles floues utilisées pour ce comportement.

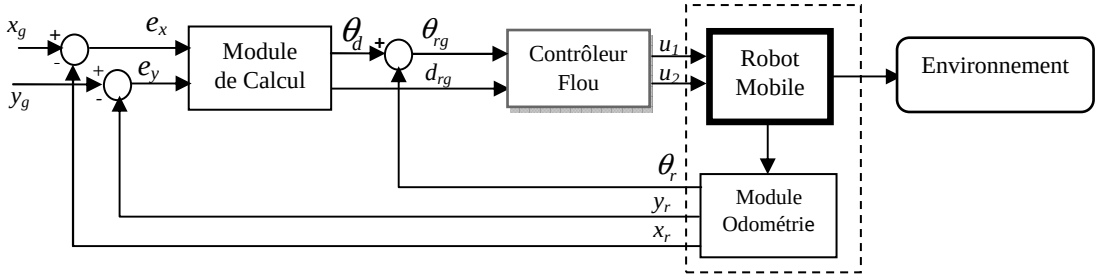


Figure IV.4 Structure de comportement flou pour la convergence vers un but

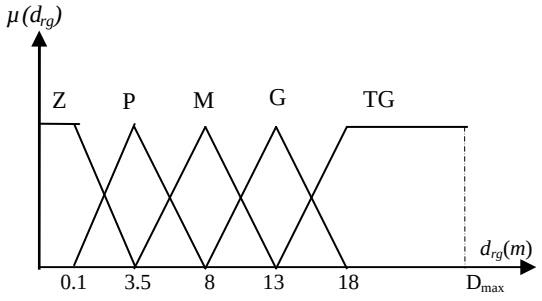


Figure IV.5 Fonctions d'appartenance de d_{rg}

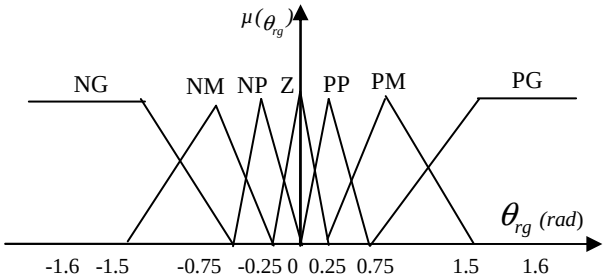


Figure IV.6 Fonctions d'appartenance de θ_{rg}

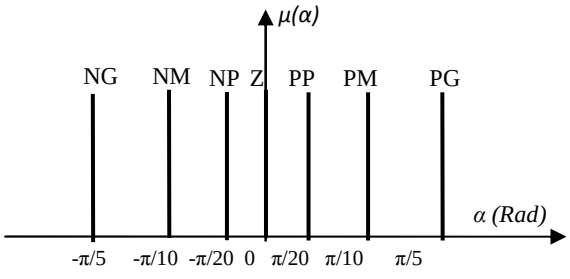


Figure IV.7 Angle de braquage

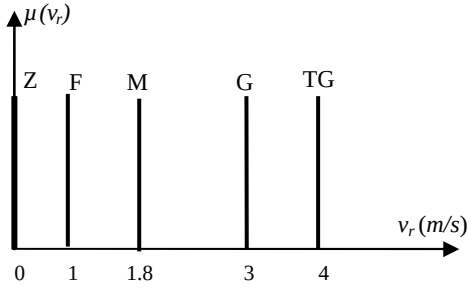


Figure IV.8 Vitesse de translation du robot

Les Actions α/V			θ_{rg}						
			NG	**NM**	**NP**	**Z**	**PP**	**PM**	**PG**
d_{rg}	**Z**	α	*PM*	*PP*	*Z*	*Z*	*Z*	*NP*	*NM*
		V_r	*Z*	*Z*	*Z*	*Z*	*Z*	*Z*	*Z*
	P	α	*PG*	*PG*	*PM*	*Z*	*NM*	*NG*	*NG*
		V_r	*F*	*F*	*F*	*F*	*F*	*F*	*F*
	M	α	*PM*	*PM*	*PP*	*Z*	*NM*	*NG*	*NG*
		V_r	*F*	*F*	*M*	*M*	*M*	*F*	*F*
	G	α	*PM*	*PP*	*PP*	*Z*	*NP*	*NP*	*NM*
		V_r	*F*	*M*	*G*	*G*	*G*	*M*	*F*
	TG	α	*PM*	*PM*	*PP*	*Z*	*NP*	*NM*	*NM*
		V_r	*F*	*M*	*G*	*TG*	*G*	*M*	*F*

Tableau IV.1 Les règles floues pour le comportement convergence ver un but

Le comportement présenté dans la section précédente permet de réaliser la navigation autonome d'un robot mobile vers un but si le robot ne reçoit aucune information de ses capteurs de perception, où l'environnement est très peu encombré (c-à-d aucun obstacle n'a été détecté autour de lui). Une telle condition est très restrictive vu la nature de l'environnement dans lequel

le robot est amené à réaliser ses missions. Dans le cas de présence d'obstacles (statiques ou dynamiques) qui empêchent le mouvement du robot mobile vers son objectif, le robot doit avoir une capacité efficace d'évitement d'obstacles.

IV.5.2 Comportement d'évitement d'obstacles

Dans ce qui suit, on présente la conception du deuxième comportement de base qui s'intéresse à la génération des actions adéquates pour l'évitement des collisions avec les obstacles si un ou plusieurs objets sont observés dans un certain voisinage du robot mobile par ses moyens de perception (en face, à droite et à gauche). Dans notre travail, on propose deux structures de commande pour l'évolution du robot mobile dans un environnement encombré d'obstacles. La première est basée sur un seul comportement d'évitement d'obstacles en utilisant les mesures de distance et de l'angle entre le centre du robot et l'obstacle, cela est fait si on suppose que les obstacles sous formes circulaires (la zone de risque est prise en compte par un champ autour du centre de l'obstacle (figure IV.3)). Dans la deuxième application, on va utiliser un système de contrôle basé sur 5 comportements flous pour accomplir la tâche de navigation autonome d'un robot mobile pour les différents types d'environnements.

IV.5.2.1 *Evitement d'obstacles par les mesures de distance et l'angle*

La première stratégie de navigation réalisée est basée sur l'utilisation d'un contrôleur flou qui peut, à partir de la distance D_{ro} et l'angle θ_o (figure IV.3), de générer les commandes nécessaires pour l'évitement d'obstacles notées (U_o). Où D_{ro} est la distance entre le robot et le plus proche obstacle, θ_o est l'angle relatif entre l'axe du robot et l'obstacle, U_o est un vecteur contient un angle de braquage (α) et vitesse de déplacement (V_r). Ce contrôleur est de type T-S d'ordre zéro, qui peut être utilisé dans les environnements moins compliqués avec des obstacles uniformes. La figure IV.9 représente l'architecture proposée pour la navigation autonome du robot mobile. On a utilisé un système de contrôle basé sur deux comportements flous (le comportement précédent pour la convergence vers le but et un comportement pour l'évitement d'obstacles) avec un bloc de coordination simple pour le choix entre les deux contrôleurs. La tâche de navigation globale se réalise par l'activation de l'un des comportements; soit converger vers le but ou éviter les obstacles dans les trois cotés (en avant, à droite ou à gauche). En utilisant les mesures des capteurs, le premier bloc comme un module de calcul permet de définir les variables d'entrées des comportements flous.

Les univers de discours de chaque entrée du comportement d'évitement d'obstacles sont partitionnés en 3 et 7 ensembles flous respectivement (figures IV.10 et IV.11). Les termes linguistiques utilisés sont: P(Petite), M(Moyenne), G(Grande), NG(Négative Grande), NM(Négative Moyenne), NP(Négative Petite), Z(Zéro), PP(Positive Petite), PM(Positive Moyenne) et PG(Positive Grande). Ce qui donne une base de 21 règles floues illustrées sur la table IV.2. Avec un principe de tourner à gauche si l'obstacle est à droite, et braquer à droite si l'obstacle se situe à gauche,…etc. Ces règles sont de la forme:

R₁: Si *l'obstacle* est en Face Alors *la vitesse* est Faible et *le braquage* est à Droite,
R₂: Si *l'obstacle* est à Droite Alors *la vitesse* est Moyenne et *le braquage* est à Gauche,
…

Les variables de sortie sont définies par les mêmes types et labels présentés dans les figures (IV.7 et IV.8).

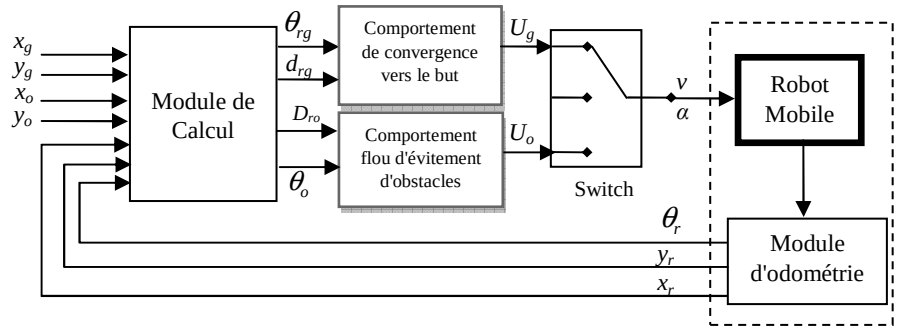


Figure IV.9 Navigateur flou

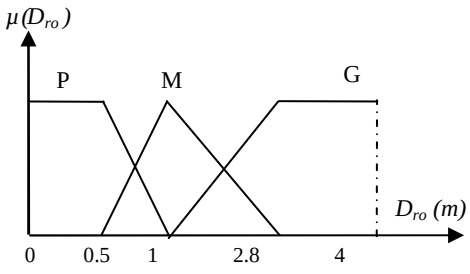


Figure IV.10 La distance robot-obstacle D_{ro}

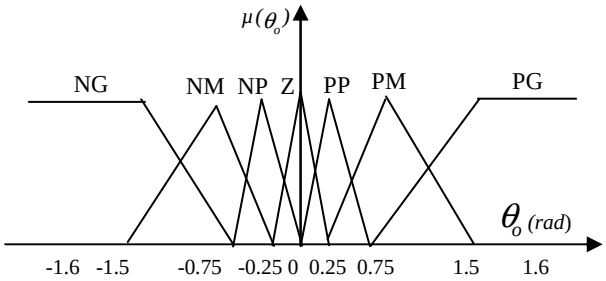


Figure IV.11 Fonctions d'appartenance de θ_o

α/V_r			L'angle θ_o						
			NG	**NM**	**NP**	**Z**	**PP**	**PM**	**PG**
Distance D_{ro}	**P**	α	*NM*	*NM*	*NG*	*NG*	*PG*	*PM*	*PM*
		V_r	*F*	*F*	*Z*	*Z*	*Z*	*F*	*F*
	M	α	*NP*	*NM*	*NG*	*NG*	*PG*	*PG*	*PP*
		V_r	*M*	*M*	*F*	*F*	*F*	*M*	*M*
	G	α	*Z*	*Z*	*NM*	*NM*	*PM*	*Z*	*Z*
		V_r	*M*	*M*	*M*	*M*	*M*	*M*	*M*

Tableau IV.2 Base de règles pour le comportement d'évitement d'obstacles

IV.6 Résultats de simulation

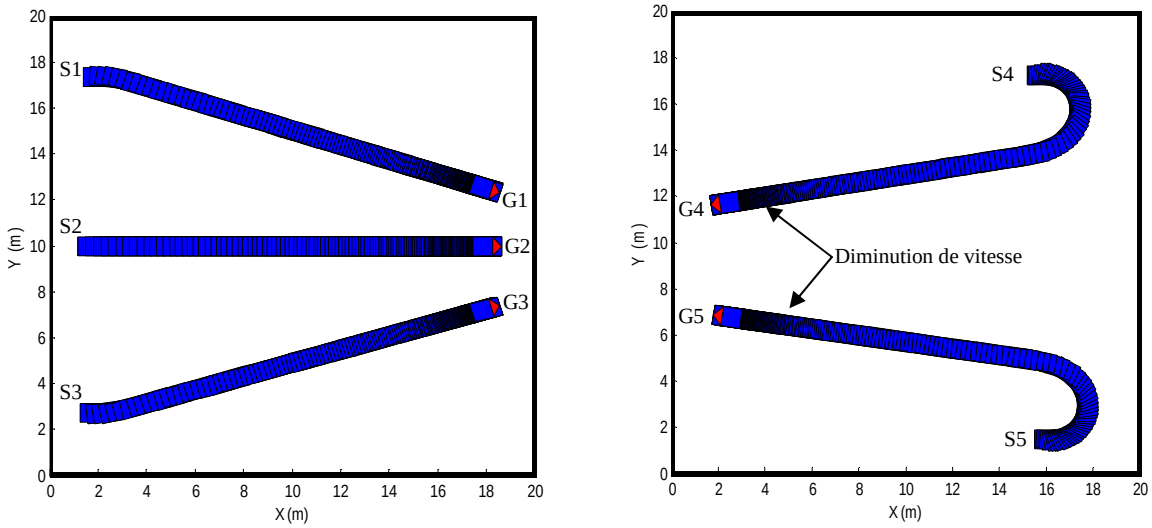
Dans cette section, des exemples de navigation du robot mobile dans des environnements d'intérieur seront présentés pour vérifier la validité et l'efficacité des schémas de commande proposés. L'environnement utilisé prend en compte les contraintes de modélisation et de mouvement du robot utilisé dans plusieurs situations telles que: l'espace libre et l'environnement avec des obstacles statiques.

IV.6.1 Comportement de convergence vers un but

Lorsque les capteurs de perception du robot ne détectent aucun obstacle près de lui, la tâche devienne alors une orientation directe vers la cible pour l'atteindre. Elle s'appelle aussi navigation libre vers un but. En choisissant différents points de départ et d'arrivés du robot mobile, les figures IV.12 (a-b) présentent les trajectoires du robot pour différentes

configurations. Comme présenté sur les figures, dans tous les cas, le robot se déplace de sa position initiale (*Start:* S_i) vers son objectif (*Goal:* G_i) où ($i = 1,\ldots 5$) d'une manière autonome quelques soit sa situation. Le contrôleur flou élaboré est capable de générer les actions les plus appropriées pour accomplir cette tâche montrant l'efficacité du comportement proposé. La vitesse linéaire du robot mobile comme montrée sur les figures se diminue quand il s'approche de son but.

La figure IV.13 présente un autre exemple de navigation libre vers un but. Si on observe la variation des actions exécutées (figure IV.14), au départ lorsque le robot se trouve loin de son objectif, le contrôleur flou génère une action de braquage maximale et positive pour tourner à gauche vers la direction de but. Puis elle se diminue pour être nulle (le robot a la même direction de but). La vitesse de déplacement pour les premières étapes de braquage est faible, elle s'augmente jusqu'à une valeur max lorsque le but est loin, puis elle se diminue pour être nulle si le robot s'approche à l'objectif. Les résultats obtenus sont très satisfaisants pour ce comportement.



(a) (b)

Figure IV.12 Convergence vers un but en utilisant un contrôleur flou TS0

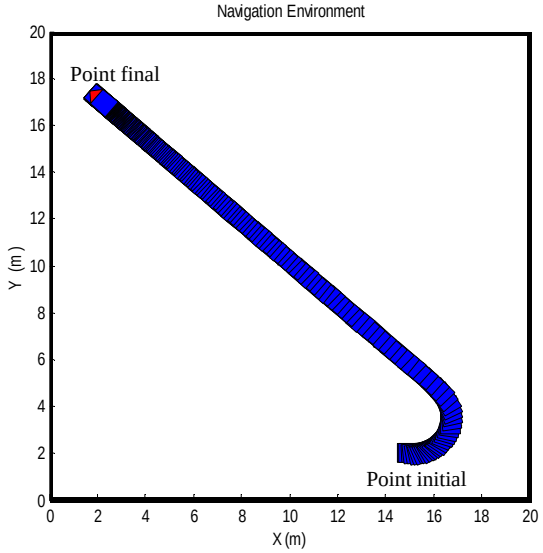


Figure IV.13 Exemple de navigation libre

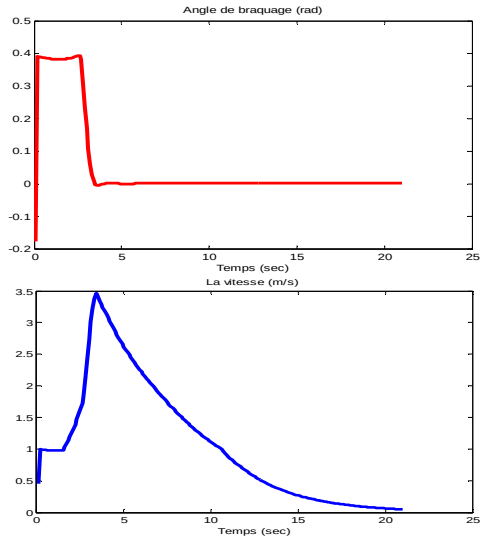


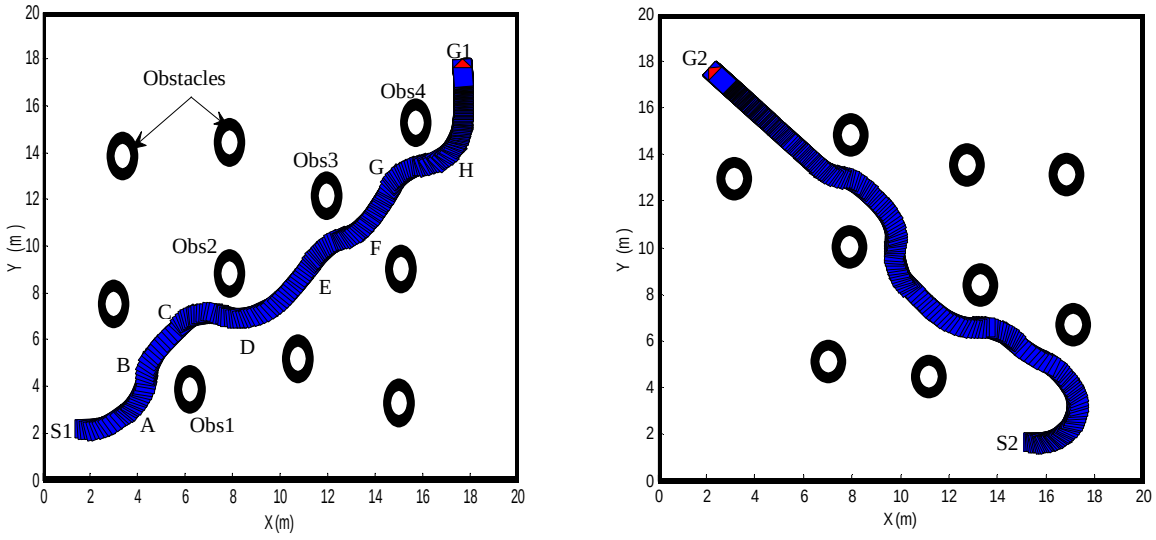
Figure IV.14 Les commandes générées α(rad) et V_r(m/s)

IV.6.2 Comportement d'évitement d'obstacles

Si l'environnement du robot contient un ou plusieurs obstacles, le robot doit pouvoir éviter les collisions avec ces objets. Comme il est présenté précédemment, le système de navigation autonome contient deux comportements élémentaires (figure IV.9): comportement de recherche de but et autre pour l'évitement d'obstacle. Le robot exécute l'action adéquate pour atteindre la destination finale en toute sécurité sans risque de collision avec les objets en déclenchant l'un des deux comportements selon la situation perçue.

Les figures IV.15 (a-b) montrent des exemples de navigation du robot mobile en présence d'obstacles dans l'environnement. Le robot se déplace vers l'objectif, lorsqu'un obstacle est détecté dans l'un des trois côtés (en face, à droite ou à gauche), le comportement d'évitement d'obstacle est activé afin de générer les actions appropriées pour éviter ces collisions. Comme montré sur les figures IV.15 (a-b), dans tous les cas, le robot est capable de naviguer de manière autonome et peut atteindre son but efficacement en évitant les obstacles avec succès quelques soit sa position initiale. La trajectoire de mouvement obtenue et les actions générées montrent que le système de commande proposé donne des meilleures performances et d'efficacité.

Dans l'exemple présenté sur la figure IV.15.a, le robot doit atteindre le but (*G1*) avec un point de départ (*S1*). Le robot commence à exécuter l'action suivant le comportement actif; en fonction du contexte actuel. Tout d'abord, le robot se déplace vers l'objectif en activant le comportement (contrôleur) de convergence vers le but avec une vitesse maximale jusqu'à ce qu'il détecte l'obstacle 1 (Obs 1) sur son côté droite, puis il change son comportement à l'évitement d'obstacles au point *A* jusqu'au point *B* au cours de lequel le robot se situe dans un espace libre et il se déplace vers l'objectif jusqu'à le point *C*. A ce moment, le robot détecte en avant l'obstacle 2 (Obs 2) et exécute le comportement d'évitement d'obstacle jusqu'au point *D* en tournant vers la droite. Les points *E*, *F*, *G* et *H* subissent aux mêmes actions. Enfin, le comportement de convergence à l'objectif est activé pour atteindre la configuration finale avec la diminution de la vitesse de déplacement quand il se rapproche vers le but. La coordination des deux comportements se fait par le module présenté auparavant.



(a) (b)

Figure IV.15 Navigation avec évitement d'obstacles

L'utilisation de ce type des contrôleurs donne des résultats acceptables dans les environnements encombrés d'obstacles. Mais dans les environnements compliqués ou les obstacles peuvent prendre différentes formes; l'utilisation d'autres types de capteurs et des contrôleurs est nécessaire.

Dans la section suivante, on va présenter un système de contrôle pour la navigation autonome d'un robot mobile basé sur 5 comportements flous pour l'évitement des obstacles de différentes formes dans les trois directions en se basant sur les mesures de distances par des capteurs de détection de type ultrasons ou infrarouge.

IV.7 Navigation avec évitement d'obstacles (2ème application)

Dans cette application, l'environnement de navigation du robot mobile peut contenir des obstacles de différentes formes (polygones, murs, ellipses,...), le robot doit donc avoir une capacité réactive et efficace d'évitement de collisions. Il a besoin d'acquérir des informations correctes et précises sur l'environnement pour être capable de naviguer correctement suivant la forme de ces obstacles. L'architecture du navigateur proposé est représentée sur la figure IV.16. Elle se compose de 5 comportements flous de type Takagi-Sugeno d'ordre zéro: un comportement de recherche d'un objectif (*Goal Seeking Behavior GSB*), comportement d'évitement d'obstacle en avant (*Front Obstacle Avoider FOA*), comportement d'évitement d'obstacle à droite (*Right Obstacle Avoider ROA*), comportement d'évitement d'obstacle à gauche (*Left Obstacle Avoider LOA*) et un comportement de réduction de vitesse de déplacement (*Velocity Reducing Behavior VRB*). Tous ces comportements sont liés par un superviseur de coordination simple pour transmettre les commandes appropriées aux actionneurs du robot mobile.

Pour cela, on suppose que le robot mobile est équipé par sept capteurs de type ultrasons pour la détection des obstacles dans les trois directions (en avant, à droite et à gauche). Les comportements proposés emploient les mesures de ces capteurs pour générer l'action adéquate d'évitement des obstacles. Les capteurs sont regroupés en trois modules (comportements) et pour chaque module, on utilise trois capteurs afin de générer l'action appropriée de mouvement. La figure IV.17 représente les positions des capteurs sur le robot mobile ainsi que l'arrangement sous forme des modules:

- Le module 1 d'évitement d'obstacle en avant (FOA) utilise les distances d_1, d_2, d_3.
- Le module 2 d'évitement d'obstacle à droite (ROA) utilise les distances d_2, d_4, d_6.
- Le module 3 d'évitement d'obstacle à gauche (LOA) utilise les distances d_3, d_5, d_7.

Où le capteur C_i mesure la distance la plus proche d_i, avec $i = 1,..,7$.

Pour les trois contrôleurs d'évitement, on utilise la même subdivision de l'univers de discours pour les différents capteurs, où la variable linguistique distance est définie par deux ensembles flous**:** P(Proche) et L(Loin) comme montré sur la figure IV.18. On peut définir ces fonctions d'appartenances par trois paramètres:

- d_m : indiquant la distance minimum par rapport à un obstacle, distance pouvant varier avec la vitesse du robot,
- d_s : distance de sécurité, au-delà de laquelle le robot peut circuler à vitesse élevée,
- d_{max}: la distance maximale mesurée par le capteur.

Pour la variable de sortie (angle de braquage), on utilise les mêmes ensembles flous de type singletons adoptés pour le comportement de navigation vers un but (figure IV.7).

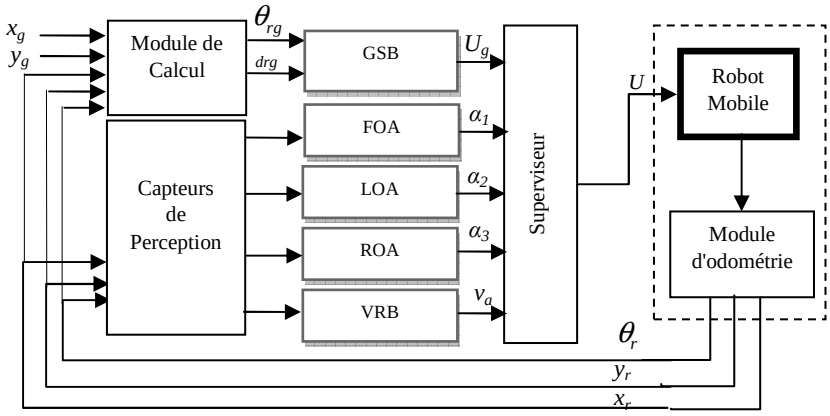


Figure IV.16 Structure de navigation basée comportements flous proposée

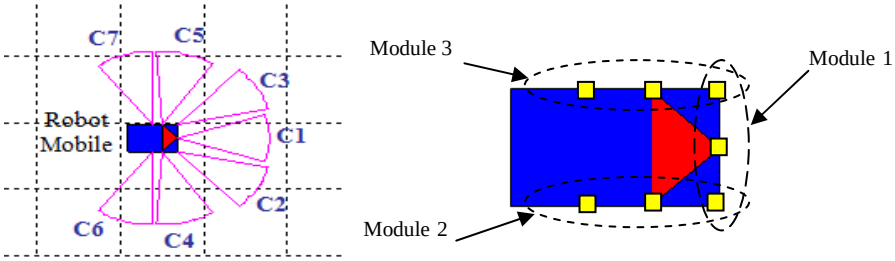


Figure IV.17 Positions et arrangement des capteurs

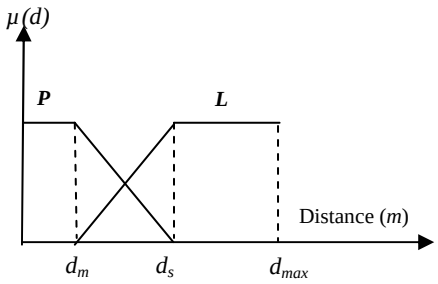


Figure IV.18 Fonctions d'appartenances de la distance d_i

Les trois distances d'entrées de chaque contrôleur permettent de définir 8 situations pour chacun, immédiatement interprétables et pour lesquelles la conduite à tenir découle du simple bon sens. Les règles floues utilisées sont représentées dans les tableaux (IV.3, IV.4 et IV.5).

D'après la structure retenue (figure IV.16), si l'environnement est libre d'obstacles, le superviseur déclenche seulement le contrôleur de convergence vers le but afin d'inférer les actions de mouvement (α et V_r). En cas de présence des obstacles, un des comportements sera activé selon la situation perçue pour éviter la collision, et en parallèle avec lui un comportement élaboré pour la génération de vitesse de translation sera activé. Le robot doit réduire sa vitesse en cas de contournement d'obstacle. A cet effet, l'architecture globale du navigateur est modifiée en ajoutant un contrôleur flou qui représente le comportement de réduction de la vitesse de déplacement (*Velocity Reducing Behavior VRB*). L'objectif est de générer une valeur d'ajustement de la vitesse courante du robot noté (V_{ajust}). Cette valeur est ajoutée à la valeur actuelle pour une réduction de vitesse selon l'équation IV.9.

$$V_r(k+1) = V_r(k) + V_{ajust}(k) \quad \text{(IV.9)}$$

Pour cela, les entrées sont : la vitesse actuelle du robot mobile V_r et la distance minimale à un obstacle proche (D_{min}). La fuzzification des entrées se fait comme suit :

- Pour la vitesse, trois ensembles flous sont utilisés (figure IV.19) avec les variables linguistiques suivantes: F(Faible), M(Moyenne) et G(Grande).
- Les distances sont fuzzifiées par deux fonctions d'appartenances: P(Proche) et L(Loin) comme montré sur la figure IV.20.

La valeur de correction de la vitesse de translation du robot (V_{ajust}) est distribuée par des constantes singletons comme présentées sur la figure IV.21. Les significations des labels de sortie sont: DI(Diminution Importante), D(Diminution), AC(Aucun Changement), A(Augmentation) et AI(Augmentation Importante). En utilisant cette répartition, la table d'inférence de la variation de vitesse regroupe seulement 6 règles floues (tableau IV.6).

Braquage α			*Distance* d_3			
			P		**L**	
			Distance d_1			
			P	**L**	**P**	**L**
d_2	**L**	α	*NG*	*NM*	*NG*	*Z*
	P	α	*NG*	*NM*	*PG*	*PG*

Tableau IV.3 Règles du module 01 (FOA)

Braquage α			*Distance* d_6			
			P		**L**	
			Distance d_4			
			P	**L**	**P**	**L**
d_2	**L**	α	*PM*	*Z*	*PM*	*Z*
	P	α	*PG*	*PM*	*PG*	*PM*

Tableau IV.4 Règles du module 02 (ROA)

Braquage α			*Distance* d_7			
			P		**L**	
			Distance d_5			
			P	**L**	**P**	**L**
d_3	**L**	α	*NM*	*NM*	*NM*	*Z*
	P	α	*NG*	*NM*	*NG*	*NM*

Tableau IV.5 Règles du module 03 (LOA)

Vitesse de Robot			*Distance* D_{min}	
			P	**L**
Vitesse	**Petite**	V_r	*AC*	*A*
	Moyenne	V_r	*D*	*AC*
	Grande	V_r	*DI*	*DI*

Tableau IV.6 Règles de comportement (RVB)

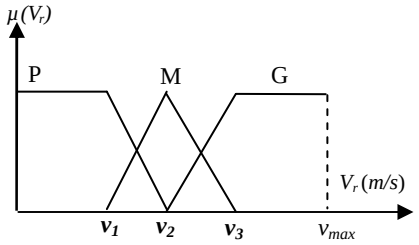


Figure IV.19 Vitesse courante V_r

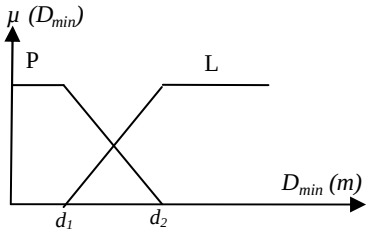


Figure IV.20 Distance minimale D_{min}

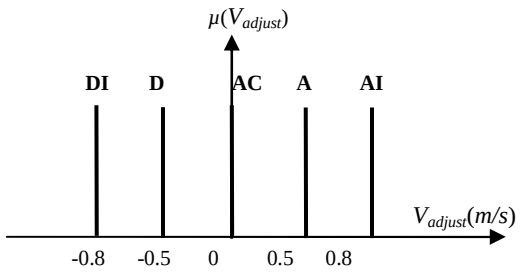
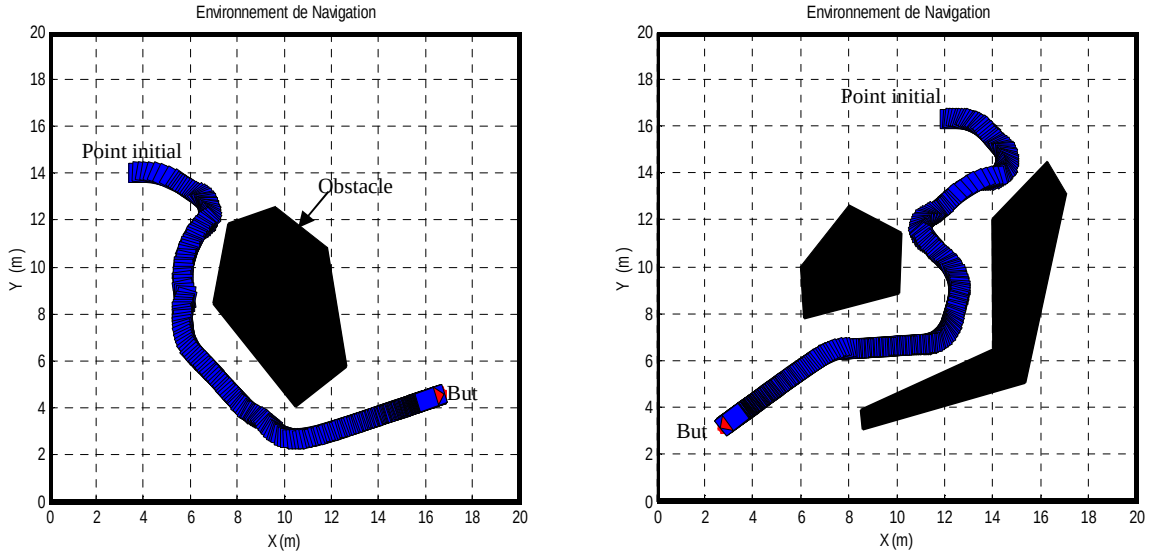


Figure IV.21 Fonctions d'appartenance de V_{ajust}

IV.7.1 Résultats de simulation

Dans les figures IV. 22 (a-b), différents exemples de navigation autonome du robot mobile seront présentés dans des environnements inconnus. Durant le mouvement, la mission de base du robot est d'atteindre un but désiré. Lorsque le robot rencontre un obstacle dans son chemin vers le but, le robot doit avoir une capacité d'évitement d'obstacles en activant les comportements selon les distances mesurées. Ces figures montrent l'efficacité des contrôleurs utilisés, le robot mobile, dans toutes les configurations, peut rejoindre la cible en évitant la collision avec les obstacles. Les comportements (contrôleurs) flous proposés donnent des résultats acceptables et satisfaisants pour réaliser la tâche de navigation du robot mobile. Ce système de contrôle est réactif, intelligent, modulaire et efficace.

Dans la figure IV. 23, la tâche du robot est d'atteindre un objectif à l'intérieur des parois de forme U. Au cours de la séquence de déplacement, le robot passe parallèlement au mur latéral droite jusqu'à ce que l'erreur d'orientation entre dans les limites d'un intervalle prédéfinit de vision. Le robot commence à se diriger vers le but à atteindre.



(a) (b)

Figure IV.22 Navigation du robot dans des environnements inconnus

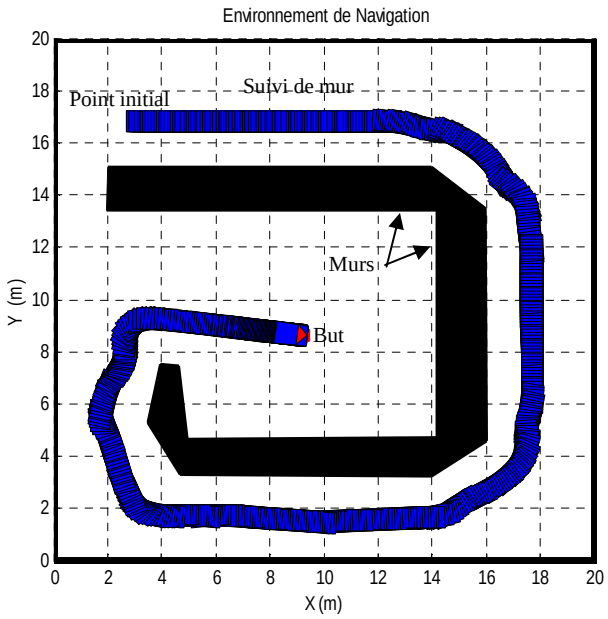


Figure IV.23 Recherche d'un but en poursuivant les murs

IV.8 Conclusion

Dans ce chapitre, une solution du problème de mouvement du robot mobile dans des espaces avec des obstacles a été proposée. Les systèmes de commande présentés sont des structures basées comportements flous où la tâche de navigation globale est décomposée en un ensemble des sous-tâches simples. La logique floue peut être utilisée efficacement pour concevoir ces contrôleurs grâce à ces propriétés intéressantes. Elle permet l'inférence des actions en cas d'incertitudes des mesures. Pour raison de simplification et d'efficacité, on a utilisé l'architecture basé comportement pour que le système de commande soit modulaire et adapté à une mission visée. Des blocs de supervision simples sont utilisés pour la coordination entre ces comportements. Les résultats de simulation présentés donnent des solutions prometteuses pour la commande autonome du robot mobile surtout dans les environnements moins compliquées. Dans tous les cas, le robot est capable d'atteindre son objectif en évitant les obstacles.

La mise en place d'un système de commande floue nécessite une expérience pour une meilleure conception de ce type des contrôleurs. La logique floue est un outil puissant et assez simple à implémenter. Mais ses contrôleurs présentent quelques inconvénients, entre autres: la nécessité d'évaluer et d'affiner les contrôleurs (les fonctions d'appartenance d'entrée et de sortie et les tables d'inférence). Le réglage des différents paramètres dans un environnement pouvait fournir de bons résultats pour cet environnement et ne pas répondre correctement face à une situation nouvelle. De manière à augmenter l'adaptabilité du système de navigation, nous nous sommes tournés vers le domaine de l'apprentissage automatique. Une solution possible est l'application des techniques d'apprentissage pour l'ajustement et la génération des paramètres du contrôleur flou.

Dans les deux chapitres suivants, on va présenter les possibilités offertes par les réseaux de neurones artificiels et l'apprentissage par renforcement pour l'adaptation des paramètres des comportements flous destinés à la commande automatique d'un robot mobile.

Chapitre V

Navigation Autonome par des Contrôleurs Neuro-Flous

V.1 Introduction

La commande floue a montré son efficacité pour la navigation des robots mobiles, mais la construction d'un système flou performant n'est pas toujours facile. Son principal inconvénient est le manque d'une méthodologie systématique pour la conception, due au nombre important de paramètres à régler (les paramètres des fonctions d'appartenances, les paramètres de la partie conclusion et les règles d'inférences). Une solution possible est l'intégration des propriétés des systèmes flous avec d'autres approches de l'intelligence artificielle comme les réseaux de neurones [JAN93, GOD99], l'apprentissage par renforcement [BER92, JOU98], les algorithmes génétiques [HER95, ABD04], colonie de fourmis [BOU09],…etc. Ces méthodes hybrides combinent les propriétés de chaque approche afin d'optimiser les paramètres des systèmes d'inférence flous [HOF00].

La capacité d'apprentissage des réseaux de neurones peut être utilisée pour automatiser le processus d'ajustement des paramètres des systèmes d'inférence flous, en réduisant considérablement le temps de développement, et d'améliorer les performances des contrôleurs flous. Cette hybridation a permis la création des contrôleurs neuro-flous qui sont actuellement l'un des domaines de recherche les plus populaires. Les systèmes hybrides neuro-flous [FUL95] permettent de tirer les avantages de ces deux approches, les réseaux de neurones (RN) offrent une capacité d'apprentissage et de généralisation permettant une représentation efficace de la connaissance, et la logique floue (LF) permet de traduire l'expérience humaine en un ensemble de règles linguistiques et facilite le traitement des connaissances imprécises. Ces systèmes hybrides sont dérivés des technologies émergentes qui sont beaucoup plus efficaces dans le développement des systèmes intelligents que les méthodes classiques, et ont démontrés son efficacités dans plusieurs domaines d'applications et notamment en commande des processus.

Plusieurs combinaisons des RNA et la LF ont été développées dans la littérature [JAN93, GOD97, GLO99]. Elles ont donnés naissance des systèmes neuro-flous, qui sont le plus souvent orientées vers la commande des systèmes complexes et les problèmes de classification. Les systèmes hybrides neuro-flous sont classés généralement en deux catégories: les systèmes neuro-flous de type (ANFIS) [JAN93], et les systèmes hybrides comportant des réseaux de neurones et des systèmes flous [FUL95, KHA09]. La première catégorie est la plus utilisée, et elle est conçue pour combiner les capacités d'apprentissage des réseaux de neurones et les propriétés de raisonnement de la logique floue. La fonction principale du réseau de neurones dans ce type est

d'apprendre davantage sur le comportement du système d'inférence flou, et utilise cette connaissance pour modifier de manière adaptative ces paramètres. L'adaptation du SIF est réalisée par la modification de la base de règles et/ou de fonctions d'appartenance (les paramètres de la partie condition et conclusion des règles floues). Les règles peuvent être générés, modifiés ou éliminés, tandis que les fonctions d'appartenance des variables entrée-sortie peuvent être ajustées par des mécanismes de mise à jour [JAN93]. L'idée de base d'utiliser les systèmes de la deuxième catégorie est de remplacer toute ou une partie des systèmes flous par des réseaux de neurones ou inversement [DON03, AYA07, KHA09]. Le système de commande global est une association des RNA et SIFs. L'objectif de ces arrangements est de combiner les avantages de ces deux approches, réduisent la vitesse de traitement et présentent une efficacité de commande.

Ces méthodes hybrides sont des solutions prometteuses pour la tâche de navigation d'un robot mobile, ou la mission présente une connaissance incomplète et incertaine en raison de l'imprécision inhérente du système sensoriel du robot, et la difficulté rencontrée par l'utilisateur pour générer les règles floues représentant les relations entré-sortie. Le robot mobile pour être autonome, doit être doté par une capacité de décision intelligente qui lui permet de se déplacer vers un objectif désiré en exécutant quelques fonctionnalités dans son environnement. Les robots mobiles, connaissant leurs positions, se déplacent d'un point à un autre avec diverses contraintes. Dans certains cas, seule l'atteinte du point final est importante, sans se préoccuper du trajet. Dans d'autres cas, le trajet parcouru est important tout comme sa dépendance du temps et la trajectoire à suivre par le robot doit être définie précisément. Ce comportement est appelé poursuite de trajectoire (suivi de chemin). Le comportement de poursuite de trajectoire est la tâche qui va nous intéresser dans ce chapitre.

Dans ce chapitre, on va présenter l'application des contrôleurs flou, et neuronal pour la commande d'un robot mobile afin de réaliser une tâche de poursuite d'une trajectoire de référence. Pour améliorer les performances du robot mobile, un contrôleur hybride neuro-flou de type ANFIS sera utilisé pour une mission de poursuite de trajectoire. Les trois contrôleurs proposés seront appliqués aussi pour la tâche de poursuite d'une cible mobile. Les résultats sont comparés et discutés.

V.2 La tâche de poursuite d'une trajectoire

En robotique mobile, la poursuite de trajectoire est une tâche élémentaire qui représente la base de toute autre tâche du robot. C'est une fonction importante que doit exécuter un robot mobile avec le minimum d'erreurs. La trajectoire est un ensemble de points à parcourir par le robot pour atteindre son but final [GAU99, BEN07]. Dans les stratégies que nous avons utilisées [CHE11, CHE13a], la trajectoire à suivre est représentée par un certain nombre de points de passage entre le point de départ et le point d'arrivée. Le chemin à suivre peut être une trajectoire d'évitement d'obstacle issue d'un planificateur, ou un nombre de points de passage qui peuvent être des postes de travail dans l'environnement de navigation. Le robot exécute sa trajectoire point par point, en corrigeant son orientation en fonction de sa position par rapport au point désiré. Le rôle des systèmes de contrôle proposés est de régler l'orientation du robot sur la trajectoire en générant les actions appropriées. Généralement pour exécuter cette mission, le robot se déplace avec une vitesse fixe où la commande inférée est seulement l'angle de braquage α de la roue avant du robot mobile.

Dans ce travail, la trajectoire est représentée par un ensemble de points de passage reliant entre deux points dans l'espace de travail. Pour cela, on considère que la trajectoire est un chemin stockée en mémoire sous la forme des vecteurs des points formant la trajectoire (x_p, y_p, θ_p). Dans notre application, on considère seulement les deux coordonnées d'abscisses et d'ordonnées (x_p, y_p) générés par un module de génération de trajectoire. La valeur de θ_p est calculée durant le mouvement du robot mobile. Le robot dans sa position courante de coordonnées (x_r, y_r, θ_r) comme le montre la figure V.1, doit calculer l'orientation désirée θ_d qui va servir par la suite au calcul de l'erreur angulaire en position θ_{er} , en utilisant les équations suivantes:

$$e_x = x_p - x_r \tag{V.1}$$

$$e_y = y_p - y_r \tag{V.2}$$

$$\theta_d = arctg\left(\frac{e_y}{e_x}\right) \tag{V.3}$$

$$\theta_{er} = \theta_d - \theta_r \tag{V.4}$$

Pour conserver l'aspect simpliste des systèmes de contrôle, le robot doit calculer l'angle d'orientation θ_d qui va le mener directement de sa position courante vers le prochain point sur la trajectoire. Le calcul de l'orientation désirée se fait selon la position du robot par apport au point de destination. Les différentes formules utilisées pour le calcul de θ_d sont données sur la figure V.2.

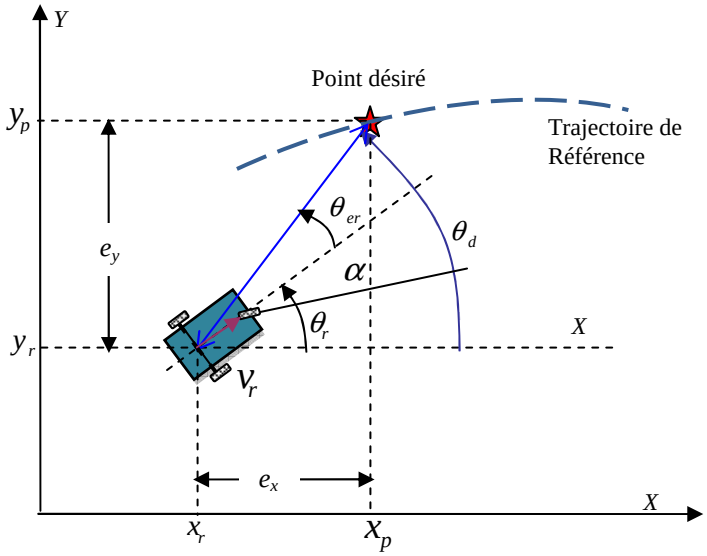


Figure V.1 Principe de la poursuite

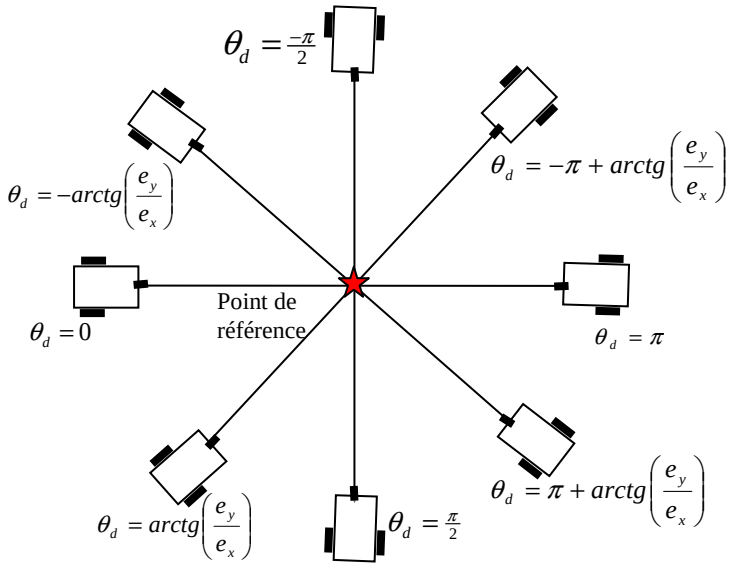


Figure V.2 Calcul de l'orientation désirée

V.3 Conception des contrôleurs de poursuite d'une trajectoire

Dans ce qui suit, on va proposer des contrôleurs intelligents pour une mission de poursuite de trajectoire par un robot mobile autonome. Ils sont basés sur des méthodes de conception simples en utilisant des contrôleurs flous de type TS, neuronal de type réseau multicouches et neuro-flou de type ANFIS [CHE11, CHE13a]. Ces comportements seront utilisés aussi pour une tâche de poursuite d'une cible mobile par un robot. Le développement est motivé par la simplicité et l'efficacité des structures étudiées pour accomplir cette mission d'une manière autonome.

V.3.1 Contrôleur flou

Pour accomplir la tâche de poursuite, on utilise un contrôleur flou de TS d'ordre zéro dont la structure est donnée sur la figure V.3 [CHE11]. Les informations effectives du système de contrôle sont les coordonnées du point désiré, à partir des quelles on calcule les composantes de l'erreur en position et sa variation. Le module de calcul compare les coordonnées actuelles du robot (x_r, y_r) avec les coordonnées du prochain point de référence à atteindre (x_p, y_p) stockés dans la mémoire "*bloc de génération de trajectoire*". La sortie de ce bloc est un angle désiré noté θ_d pour rejoindre le point de trajectoire. Cette valeur est comparée avec l'orientation courante du robot fournit par le capteur de localisation θ_r afin de déterminer l'erreur angulaire θ_{er} entre le robot et le point de référence du trajectoire, qui va être utilisée avec sa dérivée noté $d\theta_{er}$ comme des entrées du contrôleur flou conçu. Le contrôleur flou de poursuite doit inférer une seule action de braquage α pour corriger l'orientation du robot en suivant sa trajectoire, où la vitesse de déplacement est constante.

Dans ce travail, nous avons choisi la structure la plus utilisée pour la commande des processus, qui est un contrôleur à deux entrées : l'erreur et la variation de l'erreur. La sortie du contrôleur est la commande du processus. Dans notre cas c'est l'angle de braquage de la roue

avant du robot mobile de type tricycle (figure V.1). Les variables d'entrées sont l'erreur angulaire θ_{er} et sa variation $d\theta_{er}$. Ces entrées sont partitionnées sur 5 ensembles flous triangulaires représentées sur les figures V.4 et V.5. La forme triangulaire est la plus utilisée en commande des systèmes. La sortie est représentée par des singletons comme le montre la figure V.6 avec les valeurs linguistiques des variables floues entré-sortie suivantes: Z(Zero), PM(Positive Moyenne), PG(Positive Grande), NG(Négative Grande) et NM(Négative Moyenne).

Nous avons utilisé une base de règle que l'on trouve dans la plupart des applications où la dynamique du système étudié présente une certaine symétrie autour d'un certain état stable ou l'objectif est de corriger cette variation. La table V.1 représente les règles d'inférences retenues dans cette application. La base de règle est définie comme suit:

Si (*Condition*) Alors (*Conclusion*):

R₁: *Si* θ_{er} *est NG Et* $d\theta_{er}$ *est PG Alors* α *est Z*

R2: *Si* θ_{er} *est NM Et* $d\theta_{er}$ *est Z Alors* α *est PM*

…

R₂₅: *Si* θ_{er} *est PG Et* $d\theta_{er}$ *est PG Alors* α *est NG*

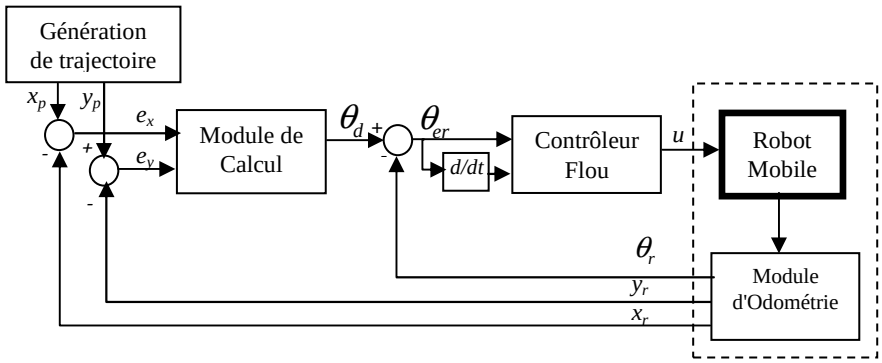


Figure V.3 Contrôleur flou de poursuite de trajectoire

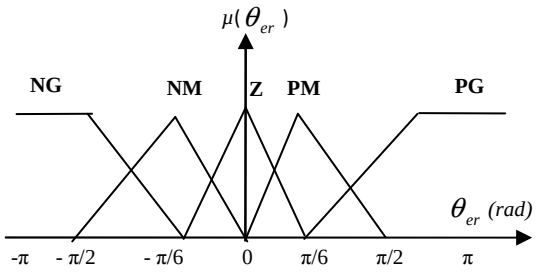


Figure V.4 Fonctions d'appartenances de θ_{er}

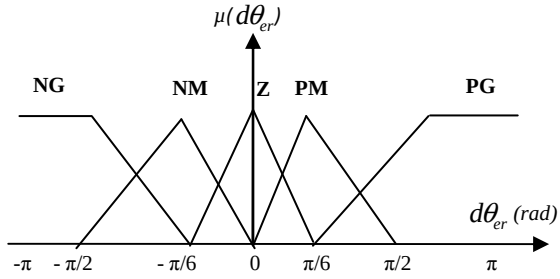


Figure V.5 Fonctions d'appartenances de $d\theta_{er}$

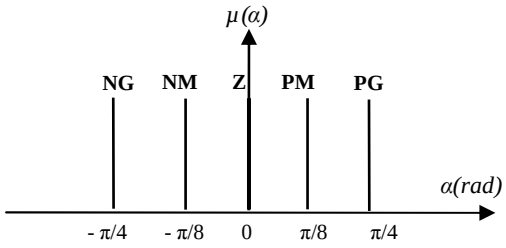


Figure V.6 Fonctions d'appartenances de la commande α

$d\theta_{er}$	θ_{er}				
	NG	**NM**	**Z**	**PM**	**PG**
NG	PG	PG	PG	PM	Z
NM	PG	PG	PM	Z	NM
Z	PG	PM	Z	NM	NG
PM	PM	Z	NM	NG	NG
PG	Z	NM	NG	NG	NG

Tableau V.1 Règles floues de poursuite de trajectoire

V.3.1.1 *Résultats de poursuite de trajectoire*

Pour démontrer l'efficacité du contrôleur intelligent proposé, des exemples de différents types de trajectoires seront utilisées. Ces chemins de référence sont choisis pour fournir plusieurs changements de direction et des courbes. Ils sont composés de plusieurs segments (courbe discontinue qui ne respecte pas les contraintes cinématiques du robot). Ces chemins ramènent le robot vers la destination finale désirée.

La surface de décision du contrôleur flou est illustrée dans la figure V.7. Elle présente la non linéarité et la souplesse de la commande floue élaborée indiquant la meilleure représentation des sorties inférées en fonctions des deux variables d'entrées.

Les résultats de simulation pour la poursuite de trajectoire de type droite sont présentés sur les figures V.8 (a-b). La tâche commence par un point initial de la trajectoire et se termine en un point final. D'après ces résultats, on observe que le robot peut suivre les trajectoires de référence avec minimum d'erreurs. Le contrôleur flou agit de manière correcte en générant les commandes appropriées pour cette mission. Dans les premières étapes, le robot exécute l'action appropriée, afin de rattraper le chemin (braquage à droite ou à gauche), puis, lorsque l'erreur angulaire est nulle, il se déplace sur les points de référence en générant un signal de sortie nul (aucun changement de direction selon la règle d'inférence choisie). Pour chaque figure, nous présentons les premiers mouvements exécutés. Les valeurs de commande générées par le contrôleur flou pour chaque situation sont données sur les figures V.9 (a-b).

Pour différentes trajectoires rectilignes, les résultats de la poursuite autonome sont présentés sur la figure V.10. Dans ce cas, le robot est initialement situé au point de l'environnement avec les coordonnées (10,10) avec une orientation nulle. Comme montré sur cette figure (V.10), le robot peut suivre ces trajectoires de référence de manière efficace dans toutes les directions possibles. Le robot a tendance à éloigner la trajectoire par quelques différences dans les premières étapes de mouvement. L'erreur de poursuite existe aux points de changement de la trajectoire, due à la discontinuité des segments formant le trajet complet qui conduit au changement brusque d'orientation. Ces premiers mouvements sont supposés comme des zones de tolérance autour des points de passage. Pendant son déplacement, si le robot entre dans la zone de tolérance du point auquel il se dirige, il commence à s'orienter vers le prochain point. Ce fait va donner un aspect souple au déplacement du robot, et va éviter les arrêts et les changements d'orientation sur le même point, puisque ces rotations sur places sont coûteuses en terme de consommation énergétique. Le robot doit arriver au point final avec minimum d'erreur. L'application de ce contrôleur pour des trajets de forme N et trapèze composés de plusieurs segments donne les résultats représentés sur les figure V.11 (a-b).

Le contrôleur flou utilisé a démontré son efficacité pour accomplir la tâche de poursuite de trajectoire de manière autonome. La qualité et la stabilité des mouvements exécutés par le robot mobile montrent que les contrôleurs basés sur la décomposition en un ensemble des comportements donnent des meilleures performances.

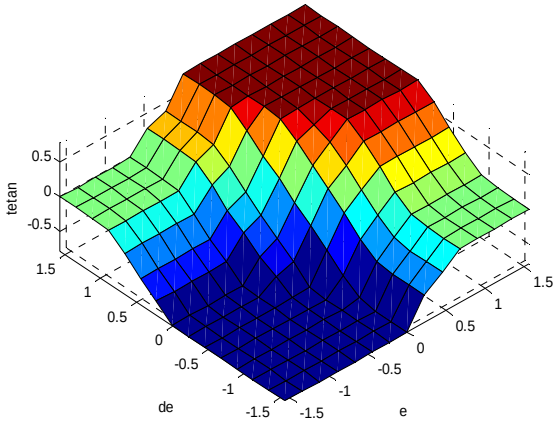
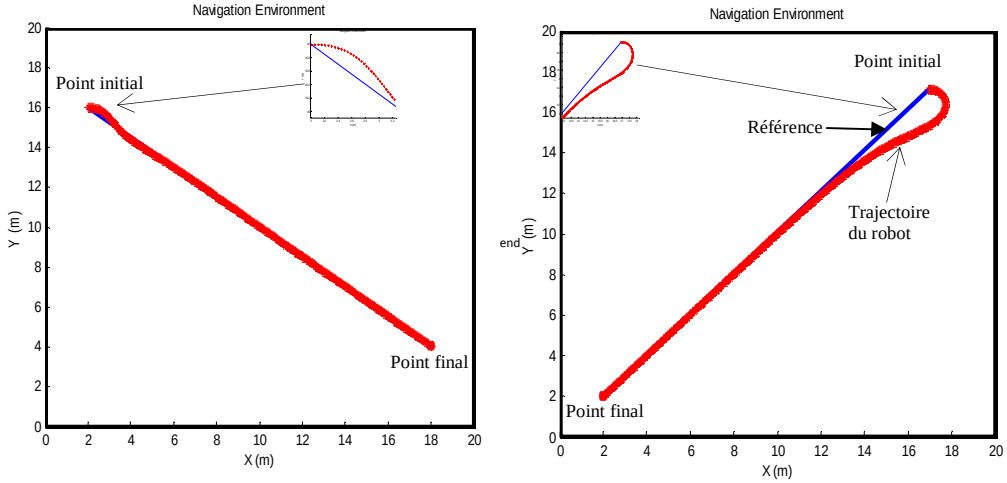
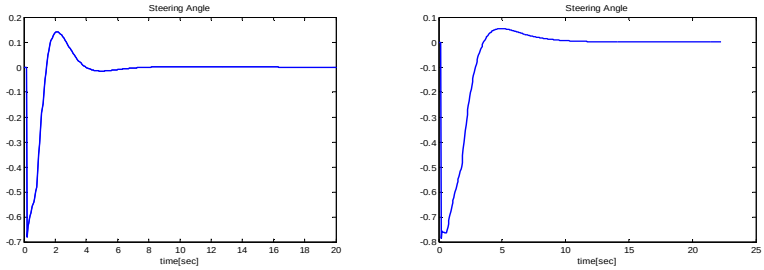


Figure V.7 Surface de commande du contrôleur flou



(a) (b)

Figure V.8 Poursuite de trajectoire de type droite par le contrôleur flou



(a) (b)

Figure V.9 Valeurs de commande générées (α en rad)

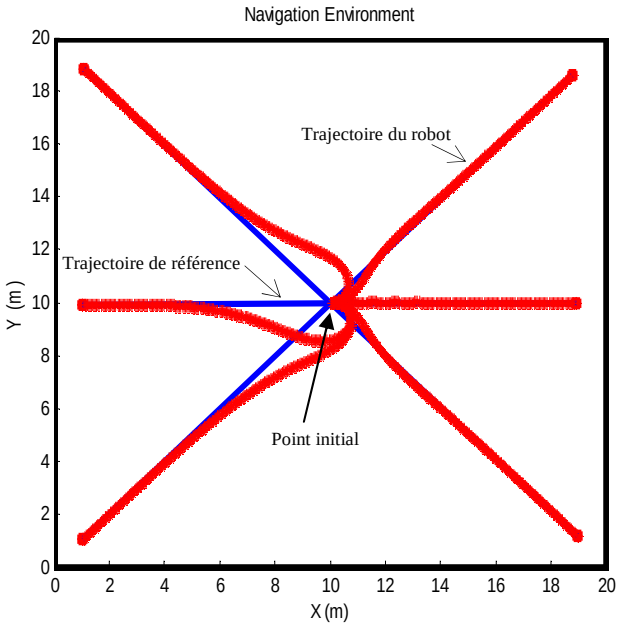
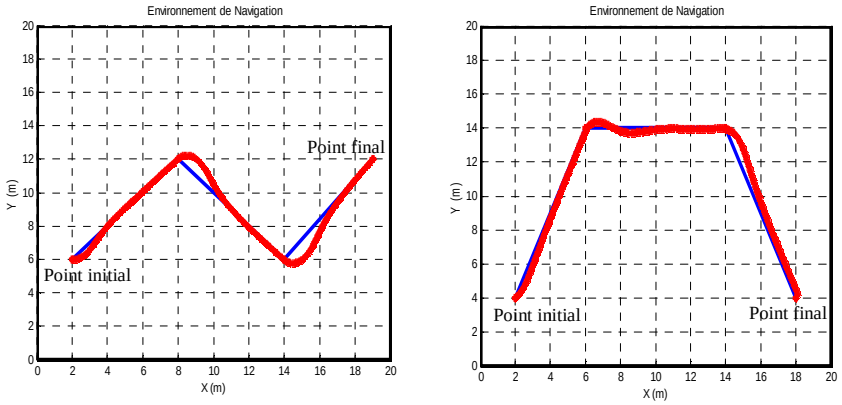


Figure V.10 Poursuite des trajectoires rectilignes



(a) (b)

Figure V.11 Poursuite de trajectoire de type N et trapèze par le contrôleur flou

Pour simplifier la stratégie étudiée, nous pouvons employer seulement l'erreur angulaire θ_{er} entre la position du robot et celle du point de la trajectoire de référence comme entrée du contrôleur de poursuite. A cet effet, on suppose que le robot se localise initialement sur la trajectoire à suivre (aucune capacité de rattrapage). Cette simplification sera considérée dans les deux implémentations que nous les verrons par la suite (neuronale et neuro-floue).

V.3.2 Contrôleur neuronal

Dans la littérature, on trouve de nombreux travaux sur l'apprentissage et l'adaptation des comportements du robot mobile [THR95, GAU99, JAN04]. La commande par réseaux de neurones est une solution possible pour un robot mobile. Elle nécessite la disponibilité d'une base de données sur la tâche étudiée. La commande neuronale se fait généralement par deux structures: la commande par modèle inverse ou par un modèle direct. La stratégie de commande proposée dans cette section emploie une structure en boucle fermée qui permet une navigation autonome d'un robot mobile pour accomplir la tâche de poursuite de trajectoire (suivi d'un chemin) en se basant sur l'élaboration d'un modèle neuronal inverse (MNI) du robot [CHE11],

dont l'apprentissage se compose de deux étapes: l'apprentissage généralisé et l'apprentissage spécialisé. Dans notre travail, on n'a utilisé que la première étape qui est l'apprentissage généralisé. Le modèle neuronal inverse est élaboré pour générer après une phase d'apprentissage et adaptation des paramètres, les commandes nécessaires aux actionneurs du robot mobile pour suivre sa trajectoire de référence. Le schéma fonctionnel de la commande proposée est illustré sur la figure V.12. A chaque pas de temps Δt, et à partir de l'erreur angulaire θ_{er} entre l'orientation actuelle du robot θ_r et l'angle désiré θ_d pour se mouvoir vars le point de trajectoire, le contrôleur neuronal génère l'angle de braquage α comme valeur de commande pour atteindre le point de référence. Après l'exécution de cette action, le robot sera à la nouvelle position $(x_r^{'}, y_r^{'}, \theta_r^{'})$.

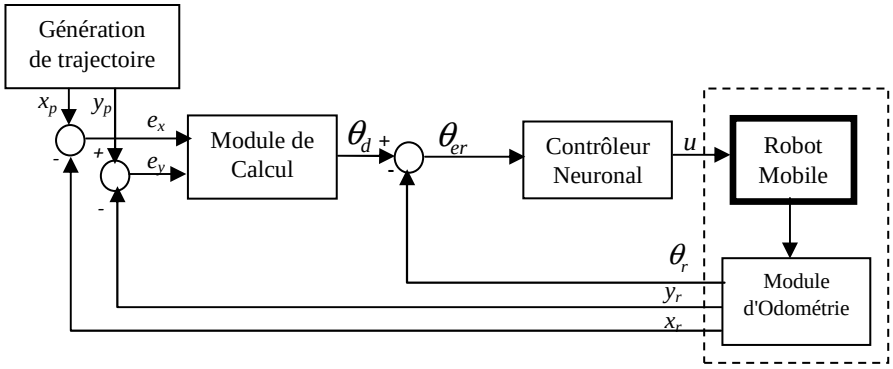
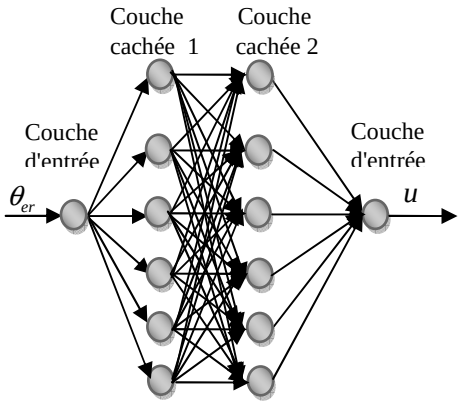


Figure V.12 Contrôleur neuronal de poursuite de trajectoire

Pour l'élaboration de ce contrôleur neuronal inverse, nous procédons en premier lieu, en choisissant des exemples d'apprentissage pour le comportement étudié. La base de données est composé de deux vecteurs: l'erreur angulaire θ_{er} comme entrée et l'action de braquage appropriée α . Ces données seront utilisées ensuite pour l'entrainement du réseau en modifiant progressivement les valeurs des poids d'interconnexion entre les neurones, à travers d'un algorithme d'apprentissage. L'architecture du réseau à entrainer est déterminée après plusieurs essais selon l'évolution de l'erreur d'apprentissage. Si l'erreur décroit lentement ou se trouve être dans un minimum local, l'apprentissage est relancé avec des poids obtenus précédemment, jusqu'à aboutissement du résultat. Si l'erreur observée est trop importante, la structure est revue en augmentant ou en diminuant le nombre des neurones ou des couches cachées.

Un réseau de neurone artificiel de type perceptron multicouches est utilisé dans ce travail avec deux couches cachées et une conche de sortie contient un seul neurone de fonction d'activation linéaire. La figure V.13 représente l'architecture du contrôleur neuronal proposé après des essais effectués pour obtenir le comportement le plus convenable. La fonction d'activation utilisée est la tangente hyperbolique (*tansig*), c'est la fonction la plus utilisée et adaptée pour l'algorithme d'apprentissage. Puisque nous disposons d'exemples entrée-sortie, ce qui correspond à l'apprentissage supervisé, nous cherchons alors à construire un modèle capable de prédire la valeur de sortie connaissant celle de l'entrée. La figure V.14 présente la structure d'apprentissage adopté pour ce modèle, qui est l'apprentissage généralisé. Pour l'adaptation du model neuronal et l'ajustement des ces paramètres, l'algorithme d'apprentissage de rétro-propagation de l'erreur est utilisé.



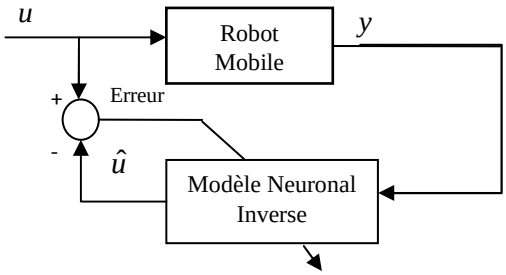
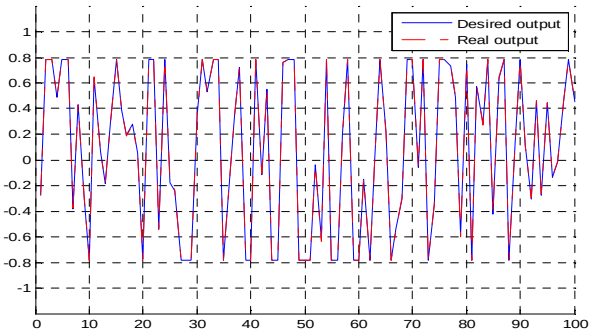
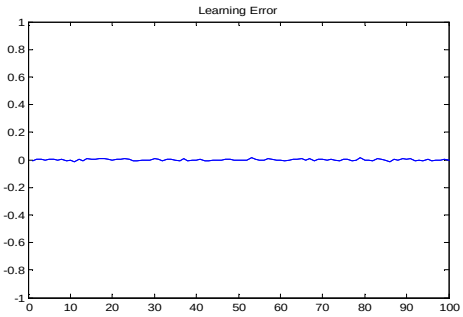


Figure V.13 Architecture de contrôleur neuronal **Figure V.14** Structure d'apprentissage généralisé

Pour vérifier l'efficacité du réseau entrainé, on test le réseau avec des nouvelles valeurs entrée-sortie qui n'ont pas été utilisées lors de l'apprentissage. La figure V. 15 (a) présente une comparaison entre les deux sorties du réseau (réelle après apprentissage et désirée). L'erreur entre les deux réponses est illustrée sur la figure V. 15 (b), comme présenté sur cette figure, l'erreur est nulle montrant la capacité d'apprentissage et représentation de la base de données.





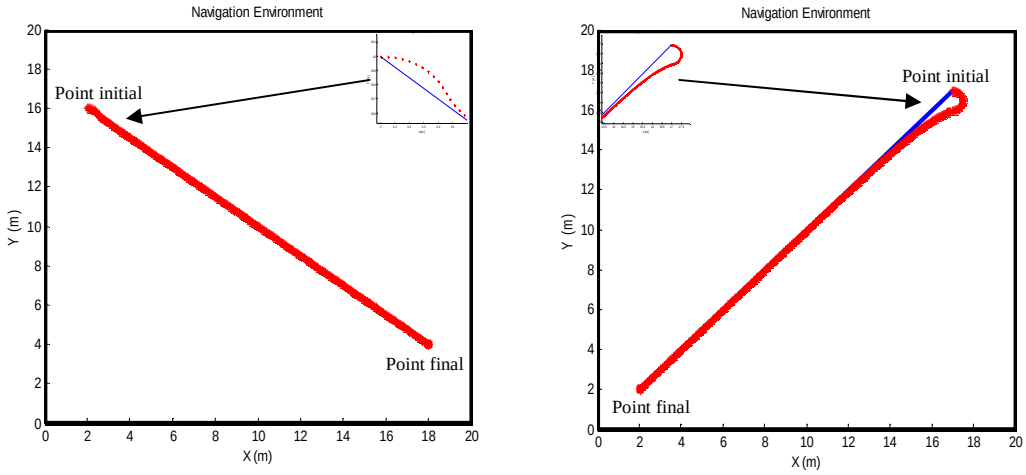
(a) (b)

Figure V.15 (a) Sortie réelle de RN et sortie désirée (b) Erreur entre les deux réponses

V.3.2.1 *Résultats de poursuite de trajectoire*

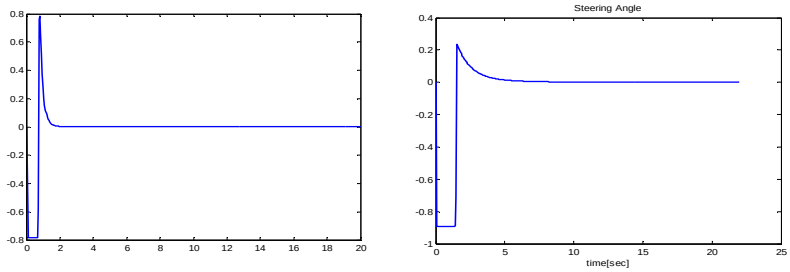
Une fois le contrôleur neuronal est testé, celui-ci est appliqué pour la tâche visée de poursuite des trajectoires de référence selon la structure de commande illustrée sur la figure V.12. Les résultats de poursuite de trajectoire de type droite sont présentés sur les figures V.16 (a-b) avec les actions de braquage générés par ce contrôleur neuronal (figure V.17 (a-b)). Les résultats montrent que le robot est capable de suivre les trajectoires de référence. Le contrôleur neuronal peut générer les commandes adéquates selon la situation du robot sur les trajectoires à suivre. Cela est montré aussi dans la figure V. 18 avec différents types de trajectoires. Dans cette figure, à partir de la position initiale du robot, il peut se déplacer de manière autonome et efficace pour rejoindre les positions finales de chaque chemin. Il est évident que l'erreur entre la trajectoire du robot et de référence est apparue toujours aux premiers pas de mouvement. Puis il exécute une action nulle de poursuite (figure V.17 (a-b)). Comme montré sur la figure V.18, le robot arrive toujours au point final avec une erreur minimale. Pour des trajectoires de forme N et trapèze composés de plusieurs segments, les résultats sont représentés sur les figure V.19 (a-b).

Le comportement obtenu est acceptable; le contrôleur neuronal est capable de commander le robot de manière autonome pour accomplir cette tâche.



(a) (b)

Figure V.16 Poursuite de trajectoire de type droite par le contrôleur neuronal



(a) (b)

Figure V.17 Valeurs de commande générées de α en rad

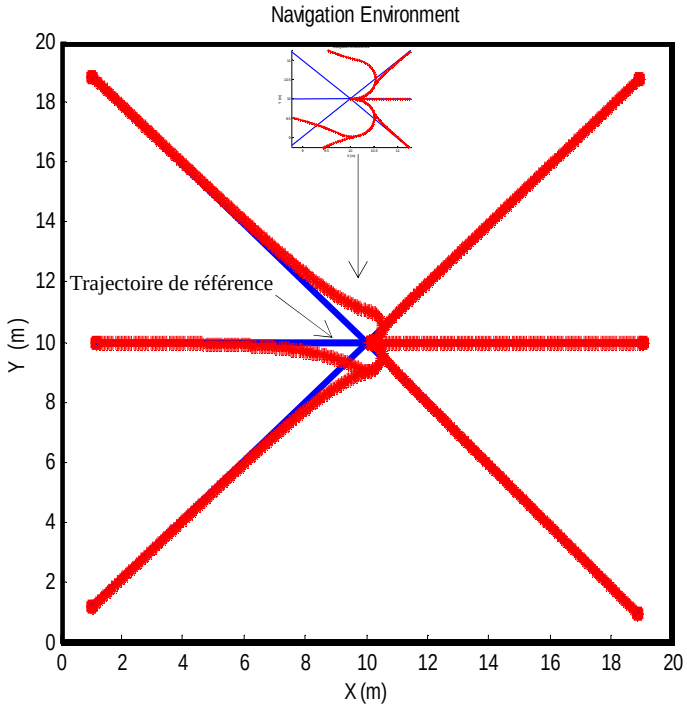
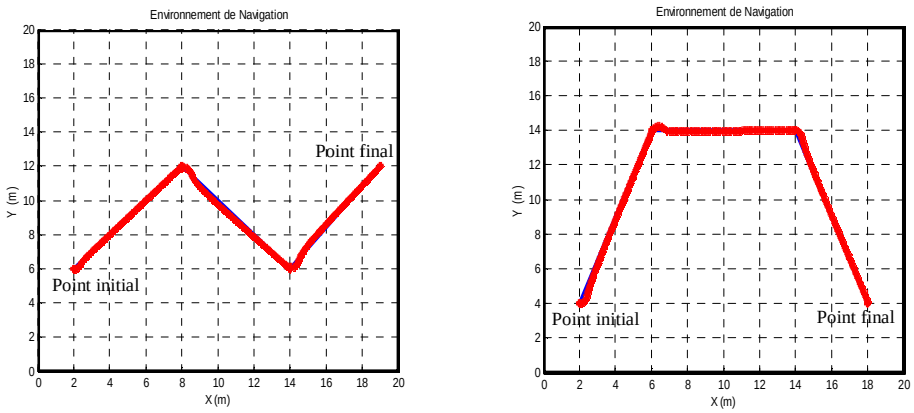


Figure V.18 Poursuite des trajectoires rectilignes



(a) (b)

Figure V.19 Poursuite de trajectoire de type N et trapèze par le contrôleur neuronal

Dans ce qui suit, nous utiliserons un contrôleur hybride neuro-flou de type ANFIS pour la poursuite d'une trajectoire par un robot mobile autonome. Pour cela, on commence par un bref exposé sur les systèmes neuro-flous et plus particulièrement sur le modèle qu'on va l'utiliser; le modèle ANFIS. Puis on va présenter notre application.

V.3.3 Systèmes Neuro-Flous

Les systèmes d'inférences flous (SIFs) et les réseaux de neurones artificiels (RNA) ont chacun des avantages particuliers. Les méthodes hybrides neuronales et floues permettent de tirer les avantages de ces deux approches, principalement la capacité d'apprentissage des RNA et de la lisibilité et la souplesse des SIFs. Plusieurs combinaisons de ces deux méthodes ont été développées dans la littérature [JAN93, GOD97, TSA97]. Elles ont données naissance aux systèmes neuro-flous, qui sont le plus souvent orientés vers la commande des systèmes complexes et les problèmes de classification.

Il existe généralement quatre grandes catégories de combinaisons des réseaux de neurones avec la logique floue :

- ***Réseau flou neuronal:*** dans ces réseaux, les techniques floues sont employées pour augmenter les capacités d'apprentissage et d'application des réseaux de neurones.
- ***Système neuronal-flou simultanément:*** le réseau de neurone et le système flou fonctionnent ensemble pour la même tâche, mais indépendamment, c-à-d ni l'un ni l'autre n'est employé pour déterminer les paramètres de l'autre. Habituellement le réseau neuronal traite les entrées, ou les sorties du système flou.
- ***Modèles neuro-flous coopératifs:*** le réseau de neurone est employé pour déterminer les paramètres (les règles et les ensembles flous) d'un système flou. Après la phase d'apprentissage, le système flou fonctionne sans le réseau de neurone. C'est une forme simple des systèmes neuro-flous.
- ***Modèles neuro-flous hybrides:*** les approches neuro-floues modernes sont de cette forme. Un réseau neuronal et un système flou sont combinés dans une architecture homogène. Le système peut être interprété comme un réseau neuronal spécial avec des paramètres flous, ou comme un système flou sous une forme distribuée parallèle.

V.3.4 Le Systèmes Neuro-Flou de type ANFIS

Le système ANFIS (Adaptative Network based Fuzzy Inference System) est un réseau adaptatif proposé par Jang en 1993 [JAN93]. Ce système peut être vu comme un réseau de neurones non bouclé pour lequel chaque couche est un composant d'un système flou. Le modèle ANFIS est le modèle le plus utilisé en pratique. Des applications notamment dans le traitement du signal, le filtrage adaptatif et la commande des systèmes ont été réalisées avec cette architecture. Plusieurs ouvrages et articles présentent des meilleures performances du modèle ANFIS lorsqu'il est utilisé dans la commande et en particulier la navigation des robots mobiles [KIM98, GLO99, RUS03, NUR06, BOS08].

Cette architecture neuro-floue affine les règles floues obtenues par des experts humains pour décrire le comportement entrée-sortie d'un système complexe en utilisant une base de donnée pour l'apprentissage. La sortie globale dans ce modèle est donnée par la moyenne pondérée de chaque sortie des règles actives (le produit ou minimum des degrés d'activation) et les fonctions d'appartenance de sortie. Il s'agit d'une technique neuro-floue hybride qui apporte les capacités d'apprentissage des réseaux de neurones au système d'inférence flou de type Takagi-Sugeno. Le rôle de l'apprentissage est l'ajustement des paramètres de ce système d'inférence flou (partie prémisse et partie conclusion des règles). La force du système ANFIS est la possibilité de génération automatique des règles floues en utilisant le *substractive clustering* ou le *grid partitioning* [JAN93]. Le système hybride neuro-flou de type ANFIS se compose de cinq couches où les nœuds adaptatifs sont situés à la première et la quatrième couches (figure V.20).

Afin de présenter l'architecture de base et le fonctionnement d'un modèle neuro-flou adaptatif de type ANFIS utilisé dans ce travail, on considère un système d'inférence flou de type Takagi-Sugeno du premier ordre. Le modèle illustré sur la figure V.20 définit un SIF à deux entrées et une seule sortie. Chaque entrée possède deux fonctions d'appartenances. Nous supposons que la base de règles contient deux règles floues de type Si-Alors, les règles sont:

Règle 1: Si x_1 *est* A_1 *et* x_2 *est* B_1 *Alors* $y_1 = f_1(x_1, x_2) = p_1x_1 + q_1x_2 + r_1$ (V.5)

Règle 2 : Si x_1 *est* A_2 *et* x_2 *est* B_1 *Alors* $y_2 = f_2(x_1, x_2) = p_2x_1 + q_2x_2 + r_2$ (V.6)

Où : x_1 et x_2 sont les variables d'entrée. A_1, A_2, B_1 et B_2 les ensembles flous. y_i : les sorties de chaque règle. p_i, q_i et r_i : sont des paramètres du conséquent de la règle *i* déterminés pendant le processus d'apprentissage.

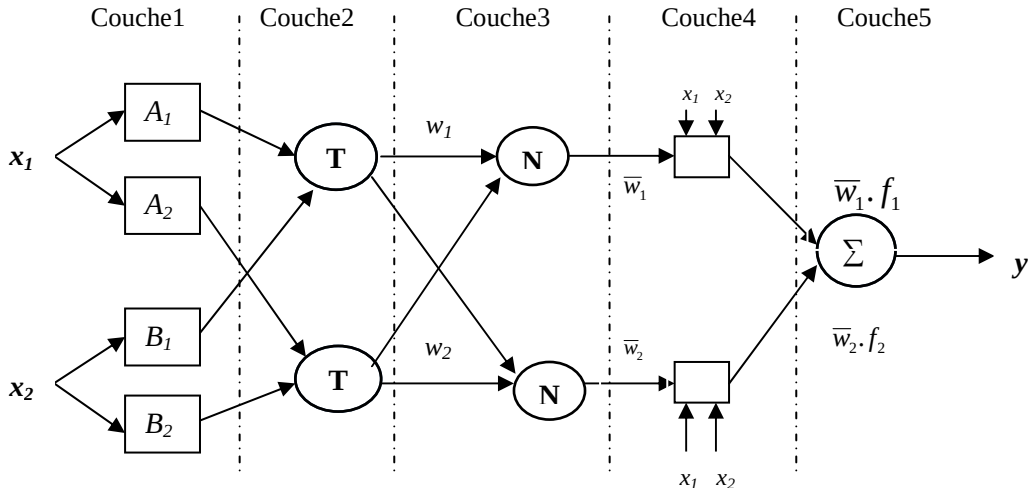


Figure V.20 Réseau ANFIS lié au modèle TS

Le réseau ANFIS est un réseau multicouches dont les nœuds sont de deux types différents selon leur fonctionnalité, ceux qui contiennent des paramètres (nœuds carrés) et ceux qui ne contiennent pas (nœuds circulaires). L'architecture de l'ANFIS contienne deux couches adaptatives:

- La première couche présente trois paramètres modifiables (a_i, b_i, c_i) liées aux fonctions d'appartenances des entrées, appelés paramètres des prémisses.
- La quatrième couche contienne également trois paramètres modifiables (p_i, q_i, r_i) appelés paramètres conséquents (conclusion).

Dans ce qui suit, on donne les fonctions réalisées par chaque couche du réseau neuro-flou:

- Première couche pour la fuzzification: Chaque nœud de cette couche fait le calcul des degrés d'appartenance des valeurs d'entrées. Ces degrés sont donnés par:

$$\begin{aligned} O_i^1 &= \mu_{A_i}(x) \qquad i = 1,2 \\ O_i^1 &= \mu_{B_i}(x) \qquad i = 3,4 \end{aligned} \tag{V.7}$$

x_1, x_2 : sont les entrées des nœuds (1, 2) et (3, 4) respectivement.

A_i, B_i : Les termes linguistiques associées aux fonctions d'appartenance $\mu_{A_i}(x)$ et $\mu_{B_i}(x)$.

Les sorties O_i^1 de la première couche représente donc les degrés d'appartenance des variables d'entrée aux ensemble flous. Dans le modèle de Jang, les fonctions d'appartenance sont des fonctions gaussiennes continues et dérivables avec les paramètres (a_i, b_i, c_i), données par :

$$\mu_{Ai}(x1) \ ou \ \mu_{Bi}(x1) = \frac{1}{1+\left[\left(\frac{x_i - c_i}{a_i}\right)^2\right]^{b_i}} \tag{V.8}$$

- Deuxième couche pour les règles floues: Cette couche est formée d'un nœud pour chaque règle floue et génère les poids synaptiques. Ces nœuds sont de type fixe notés *Π* et chacun d'eux engendre en sortie le produit de ses entrées (opérateur ET de la logique floue), ce qui correspond au degré de vérité de la règle considérée (eq. V.9):

$$O_i^2 = w_i = \mu_{A_i}(x) \times \mu_{B_i}(y) \ \text{ pour } , i = 1,2 \tag{V.9}$$

- *Troisième couche pour la Normalisation:* Les nœuds de cette couche sont également fixes et réalise la normalisation des poids des règles floues selon la relation :

$$O_i^3 = \overline{w}_i = \frac{w_i}{w_1 + w_2} \text{pour}, i = 1,2 \tag{V.10}$$

Chaque nœud *i* de cette couche est un nœud circulaire appelé *N*. La sortie du nœud *i* est le degré d'activation normalisé de la règle *i*.

- *Quatrième couche de Conséquence:* Chaque nœud de cette couche est adaptatif et calcule les sorties des règles en réalisant la fonction:

$$O_i^4 = \overline{w}_i \times f_i = \overline{w}_i (p_i x + q_i y + r_i) \text{ pour }, i = 1,2 \tag{V.11}$$

Les paramètres (p_i, q_i, r_i) sont les paramètres de sortie de la règle *i* (la partie conclusion).

- *Cinquième couche pour la Sommation:* Elle comprend un seul neurone qui fournit la sortie de l'ANFIS en calculant la somme des sorties précédentes. Sa sortie qui est également celle du réseau est déterminée par la relation suivante:

$$O_i^5 = f = \sum_i \overline{w}_i \times f_i \text{ pour }, i = 1,2 \tag{V.12}$$

V.3.4.1 *Apprentissage de l'ANFIS*

L'apprentissage est la phase de développement du réseau neuro-flou en optimisant les paramètres d'un système flou: les paramètres de la partie prémisses (fonctions d'appartenance) et la partie conclusion (les coefficients de sortie). Cette adaptation est basée sur la capacité d'apprentissage des réseaux de neurones artificiels multicouches à partir d'un ensemble de données. Pour l'identification des paramètres, la structure du réseau doit être fixée et les paramètres des fonctions d'appartenances et des conclusions seront ajustés en utilisant l'algorithme d'apprentissage de rétro-propagation avec une combinaison avec l'algorithme des moindres carrés. Le système ANFIS est définit par deux ensembles de paramètres: S_1 et S_2 tels que:

- S_1: représente les paramètres des ensembles flous utilisés pour la fuzzification dans la première couche de l'ANFIS:

$$S_1 = \left((a_{11}, b_{11}, c_{11}), (a_{12}, b_{12}, c_{12}), \ldots, (a_{1p}, b_{1p}, c_{1p}), \ldots, (a_{np}, b_{np}, c_{np})\right) \tag{V.13}$$

Ou *p* est le nombre des partitions floues de chacun des variables d'entrées et *n* est le nombre de variables d'entrées.

- S_2: représente les coefficients des fonctions linéaires (les paramètres conséquents):

$$S_2 = (p_1, p_2, p_3 \ldots, q_1, q_2, q_3 \ldots, r_1, r_2, r_3 \ldots) \tag{V.14}$$

Jang [JAN93] a proposé que la tâche d'apprentissage de L'ANFIS se faite en deux passages en utilisant un algorithme d'apprentissage hybride, comme illustré sur la table V.2.

	Passage vers l'avant	*Passage en arrière*
Paramètres des prémisses	Fixe	Rétro- propagation
Paramètres des conséquents	Moindres carrés	Fixe

Tableau V.2 Méthodes utilisées pour l'apprentissage de l'ANFIS

V.3.5 Conception du contrôleur Neuro-Flou

Après cet aperçu sur les systèmes hybride neuro-flous, dans cette section, on utilise un contrôleur hybride de type ANFIS pour une tâche de poursuite de trajectoire par un robot mobile [CHE13a]. Avec le même principe cité précédemment, la trajectoire est un chemin stockée en mémoire sous la forme de deux vecteurs (x_p, y_p) des points de la trajectoire (figure V.1). La mission du robot mobile est d'atteindre la configuration finale de façon efficace et autonome.

V.3.5.1 *Description du contrôleur ANFIS*

Il s'agit d'un modèle ANFIS de Takagi-Sugeno d'ordre un formé par une base de données regroupées pour cette mission [CHE13a]. Le schéma fonctionnel de ce système de commande est représenté sur la figure V.21. Avec le même principe du contrôleur neuronal, le contrôleur utilise l'erreur angulaire comme variable d'entrée et entrainé pour générer l'angle de braquage α de la roue avant orientable. Cette commande va permettre de diriger le robot vers le point de la trajectoire de référence. Pour la variable d'entrée θ_{er}, on a utilisé 5 fonctions d'appartenances gaussiennes puisque ce type est le plus adapté au système ANFIS. La structure du contrôleur neuro-flou proposé est donnée sur la figure V.22.

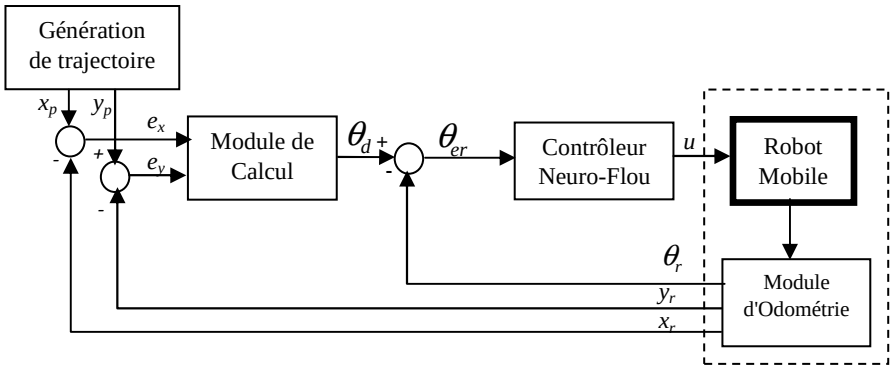


Figure V.21 Contrôleur Neuro-Flou de poursuite de trajectoire

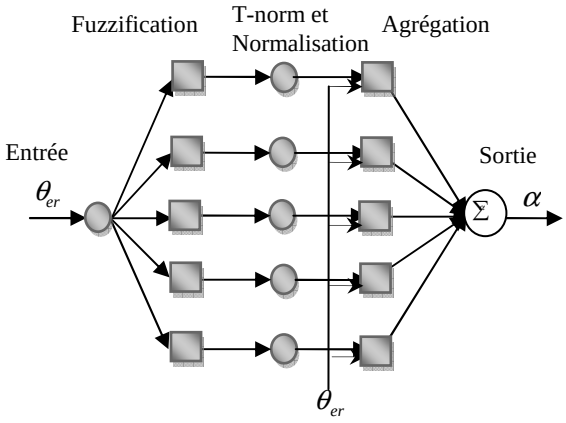
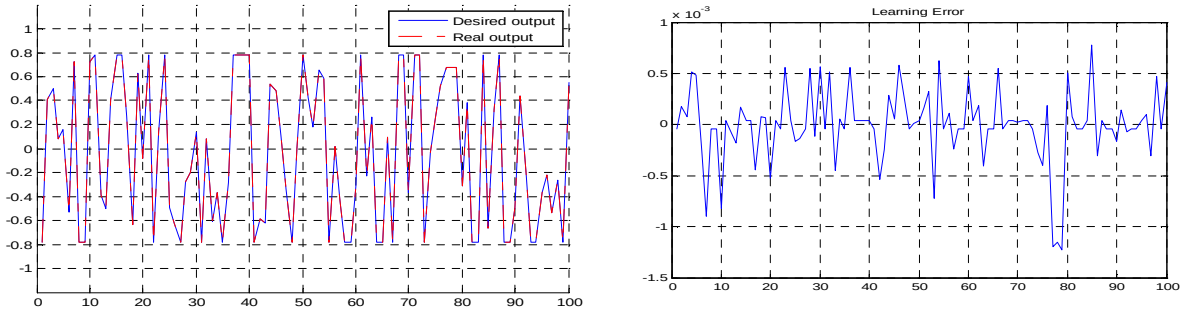


Figure V.22 Structure de réseau ANFIS

V.3.5.2 *Phase d'apprentissage*

Cette étape consiste à ajuster les paramètres des fonctions d'appartenances de la variable d'entrée représentant la partie prémisse de chaque règle floue (les paramètres non linéaires (a_i, b_i, c_i)), et les paramètres des conclusions (paramètres linéaires (p_i, q_i, r_i)). L'optimisation se fait pour apprendre la base de données pour la tâche de poursuite de trajectoire. Les données d'entrainement sont des vecteurs contient les valeurs de l'erreur angulaire θ_{er} et ces commandes correspondantes α.

Après une phase d'apprentissage, on présente sur les figures V. 23 (a), une comparaison entre les deux commandes réelle et désirée du système ANFIS. On observe une grande similarité entre les deux réponses. La figure V. 23 (b) représente l'erreur entre les deux sorties. Après cette phase de développement, le contrôleur est mis en série avec le robot pour une phase d'exécution afin d'accomplir la tâche de poursuite de trajectoire comme montré précédemment sur la figure V.21.

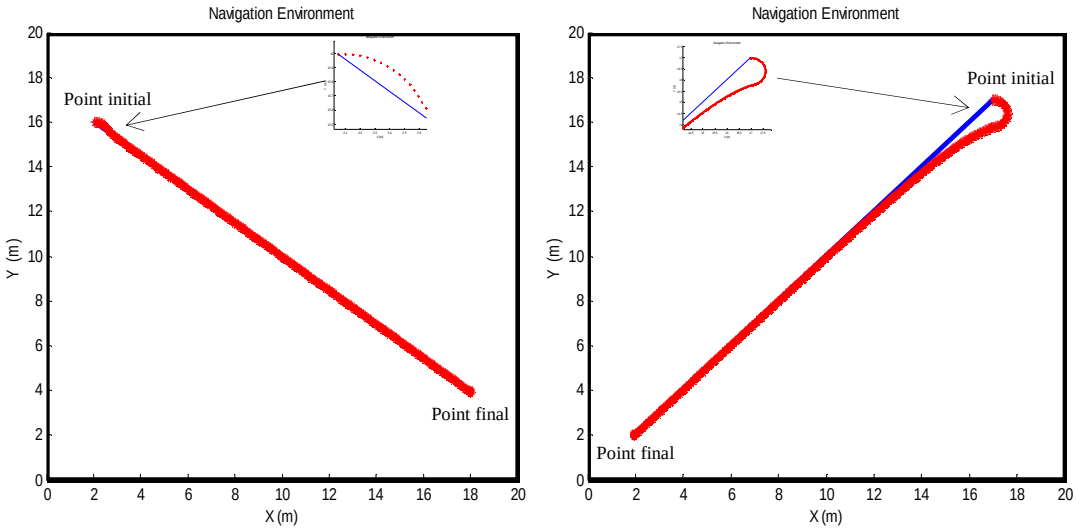


(a) (b)

Figure V.23 (a) la sortie réelle de l'ANFIS et désirée (b) l'erreur entre les deux réponses

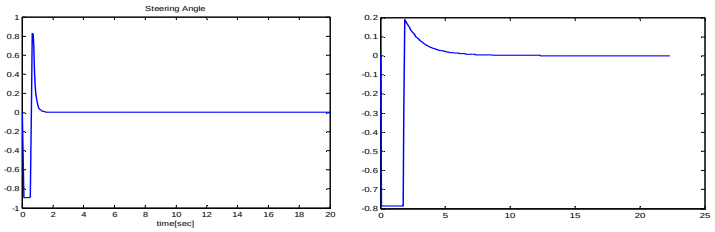
V.3.5.3 *Résultats de poursuite de trajectoire*

Les résultats pour un chemin avec un seul segment sont donnés sur les figures V.24 (a-b). Comme montré sur les figures des commandes appliquées (figures V.25 (a-b)), le robot s'oriente dans les premiers pas puis il garde son orientation sur la trajectoire qui est définie par une action de braquage nulle. L'efficacité de ce contrôleur est testée aussi pour différentes trajectoires (*Ti, i=1…6*) (figures V.26). Comme illustré sur la figure V.26, le contrôleur proposé définit un comportement correct et effectif en exécutant des actions les plus adaptées à la mission visée. Le robot a tendance à s'éloigner la trajectoire de référence dans les premières étapes de mouvement. Les figures V.27 (a-b) représentent les trajectoires du robot mobile afin de poursuivre des trajectoires de type N et trapézoïdale. La poursuite est très satisfaisante dans ces deux cas.



(a) (b)

Figure V.24 Poursuite de trajectoire de type droite par le contrôleur neuro-flou



(a) (b)

Figure V.25 Valeurs de commande exécutées de α en rad

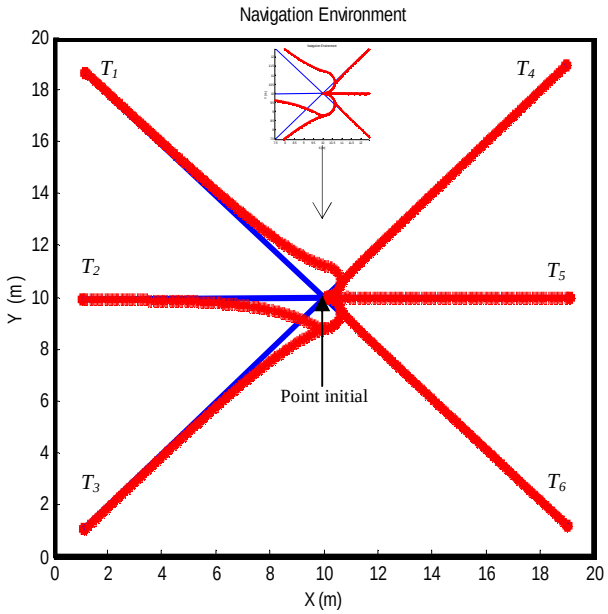
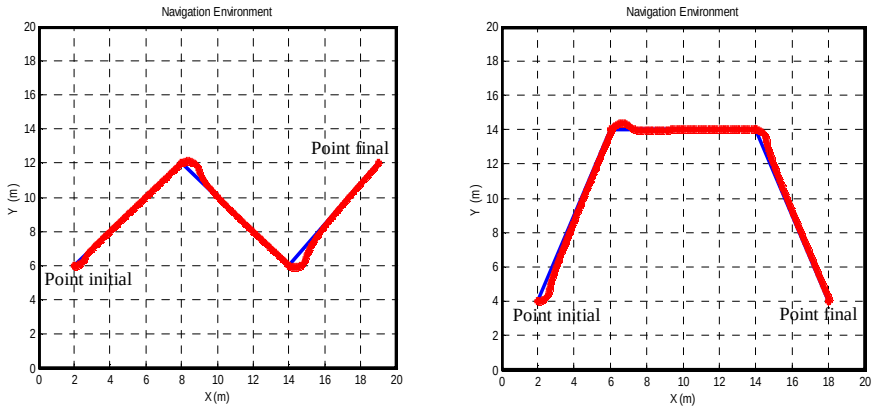


Figure V.26 Poursuite des trajectoires rectilignes

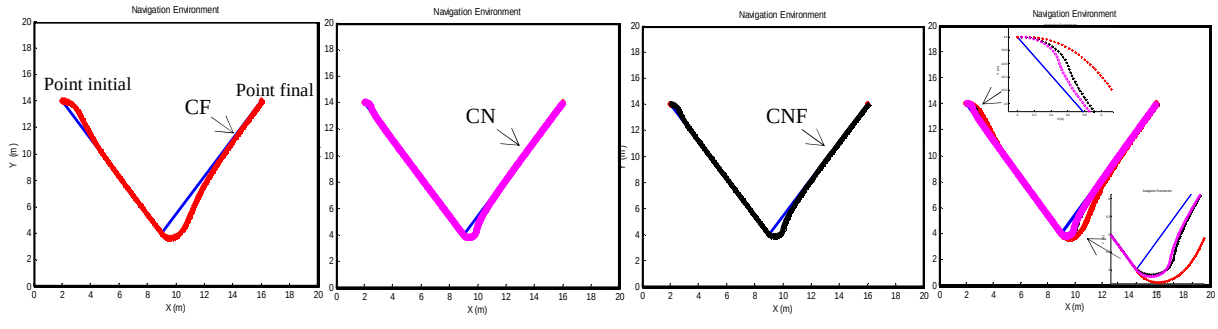


(a) (b)

Figure V.27 Poursuite de trajectoire de type N et trapèze par le contrôleur neuro-flou

V.4 Etude comparative

Dans cette section, on fait une comparaison entre les résultats obtenus par le contrôleur flou, neuronal et neuro-flou pour accomplir la tâche de poursuite de trajectoire. Sur les figures V.28 (a-b-c), on donne les comportements obtenus en utilisant les trois contrôleurs proposés pour un chemin de forme V. On remarque que le robot dans tout les cas peut suivre sa trajectoire de référence en toute autonomie. Une erreur est apparue lors de changement de direction (figure V.28 (d)), due à la contrainte de non holonomie du robot utilisé et la limitation des commandes exécutées. Cette erreur est minimale en cas de l'application des systèmes neuronal et neuro-flou grâce à la base de données utilisée pour l'apprentissage. Dans le cas des contrôleurs flous; les tables d'inférences doivent être établies par un utilisateur expert ayant une connaissance sur le système à commander (c-à-d définir la meilleure conclusion pour chaque prémisse). Les systèmes neuronal et ANFIS ne requiert pas cette connaissance, mais nécessite une base de données pour la phase d'apprentissage. Les comportements obtenus sont acceptables et très satisfaisants pour cette mission. Le contrôleur ANFIS est efficace et intègre les avantages des réseaux de neurones et les systèmes flous.



(a) (b) (c) (d)

Figure V.28 Poursuite de trajectoire de type V en utilisant: (a) contrôleur flou CF, (b) contrôleur neuronal CN, (c) contrôleur neuro-flou CNF, (d) comparaison entre les trajectoires

V.5 Poursuite d'une cible mobile

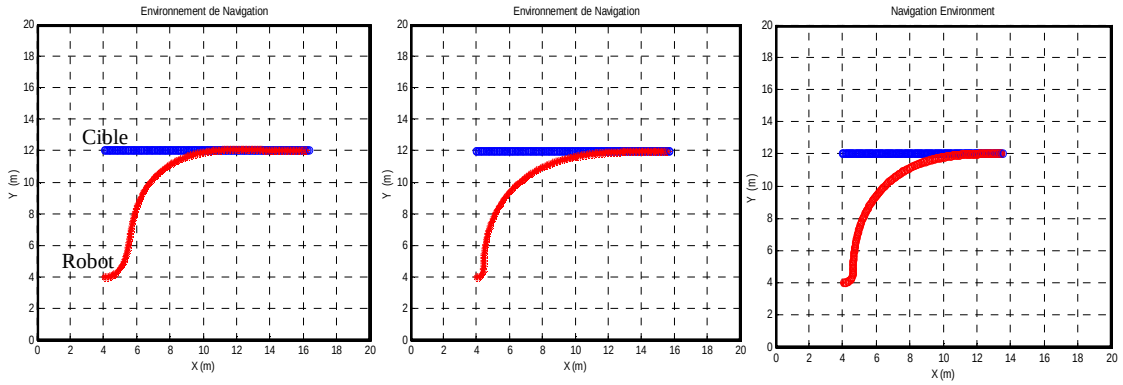
Dans cette section, nous avons appliqués les contrôleurs précédents (flou, neuronal et neuro-flou) pour une tâche de poursuite d'une cible mobile. Si on considère une cible en mouvement (objet ou autre robot) dans l'environnement de navigation, la mission du robot est de poursuivre cet objet. La tâche commence avec des positions initiales différentes de la cible et du robot et se termine lorsque le robot atteint cet objet. Alors ses coordonnées sont connues à n'importe quel moment du mouvement. Les systèmes de contrôle proposés doivent guider le robot vers la position de cet objet mobile en générant la commande de correction nécessaire.

Les figures suivantes illustrent les trajectoires du robot et de la cible. Dans le premier exemple présenté sur les figures V. 29 (a-b-c), le robot poursuit une cible mobile en ligne droite. Le robot mobile est situé initialement au point de coordonnées (4,4) dans les repères de l'environnement avec une orientation nulle, et la cible est située au point (4,12). Lorsque la cible commence à déplacer le long d'une ligne droite; le robot mobile déplace vers elle pour la rejoindre. La tâche est terminée si le robot rejoint la cible. Les contrôleurs proposés flou, neuronal et neuro-flou peuvent guider le robot vers la cible et de suivre son mouvement de manière efficace. Comme illustré sur les figures, la tâche est rapide lorsqu'on utilise les deux contrôleurs neuronal et ANFIS.

D'autres exemples sont présentés sur les figures V.30 et V. 31, qui illustrent l'efficacité des contrôleurs précédents avec des trajectoires circulaires et sinusoïdale. D'après ces résultats, on peut montrer que les contrôleurs proposés sont des solutions possibles pour cette tâche. Ces contrôleurs peuvent guider le robot vers la direction de la cible mobile avec autonomie.

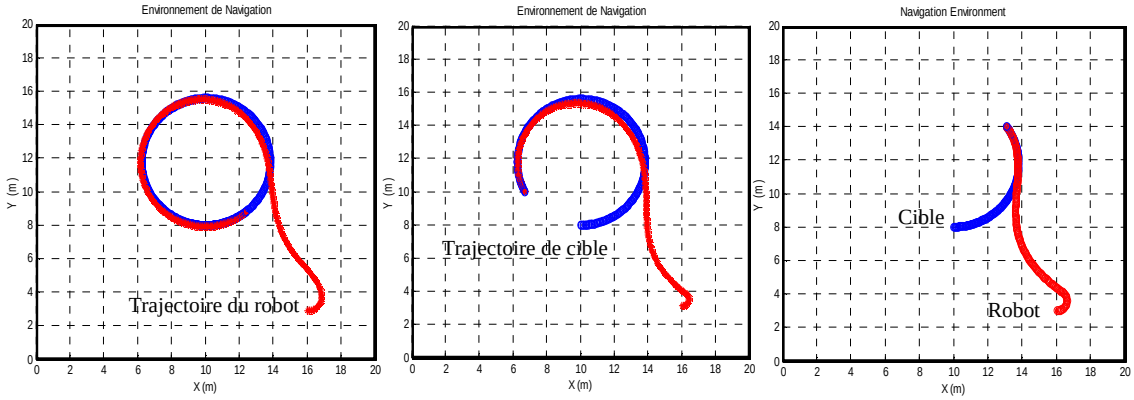
V.5.1 Comparaison

Dans ce qui suit, on présente une comparaison en donnant des résultats de poursuite d'une cible en utilisant les trois contrôleurs (flou, neuronal et neuro-flou) pour différentes types des trajectoires. Les figures V.32 (a-b-c-d) illustrent les trajectoires exécutées par le robot en utilisant: (a) le contrôleur flou, (b) le contrôleur neuronal, (c) le contrôleur de type ANFIS. Sur les figures V.33 et V.34, d'autres exemples sont présentés pour des trajectoires rectiligne, circulaire et sinusoïdale. On a montré que les systèmes de commande utilisés donnent des meilleures performances pour cette fonctionnalité.



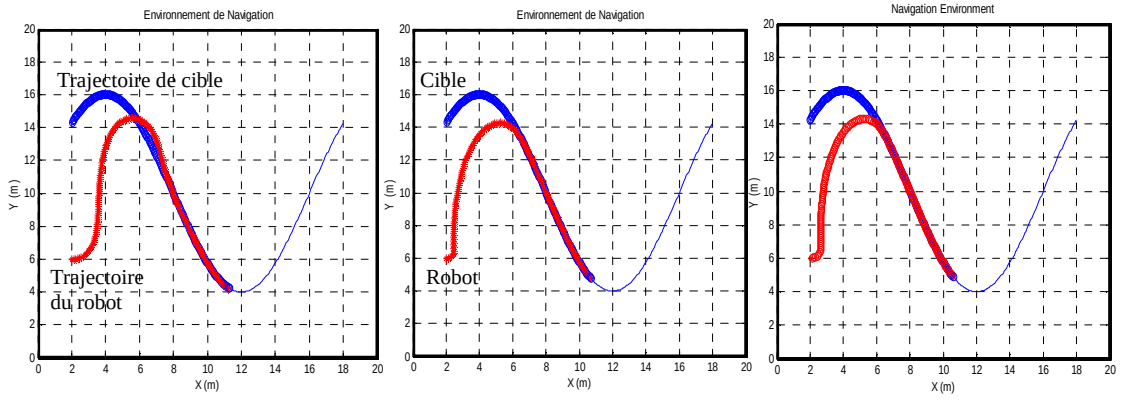
(a) (b) (c)

Figure V.29 Poursuite d'une cible mobile: (a) contrôleur flou CF, (b) contrôleur neuronal CN, (c) contrôleur neuro-flou CNF



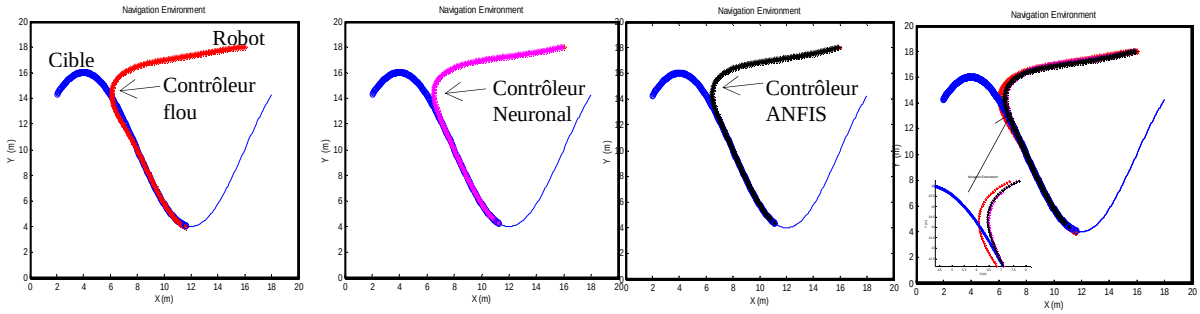
(a) (b) (c)

Figure V.30 Poursuite d'une cible mobile avec une trajectoire circulaire: (a) contrôleur flou CF, (b) contrôleur neuronal CN, (c) contrôleur neuro-flou CNF



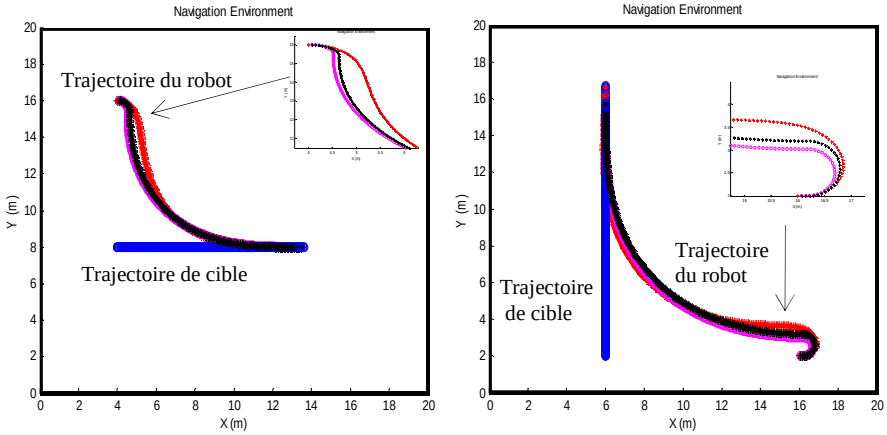
(a) (b) (c)

Figure V.31 Poursuite d'une cible mobile avec une trajectoire sinusoïdale: (a) contrôleur flou CF, (b) contrôleur neuronal CN, (c) contrôleur neuro-flou CNF



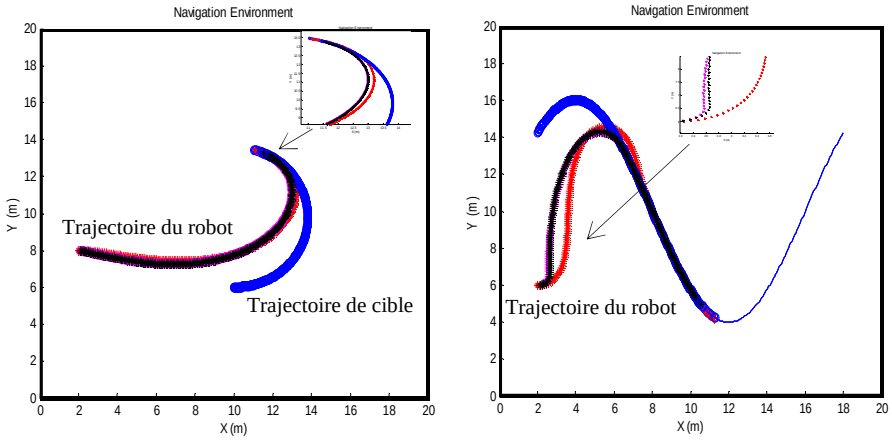
(a) (b) (c) (d)

Figure V.32 Poursuite d'une cible mobile et comparaison entre les trajectoires



(a) (b)

Figure V.33 Poursuite d'une cible mobile avec des trajectoires rectilignes



(a) (b)

Figure V.34 Poursuite d'une cible mobile avec des trajectoires circulaire et sinusoïdale

V.8 Conclusion

Dans ce chapitre, nous avons présenté des systèmes de contrôle intelligents basés sur les techniques de l'intelligence artificielle les plus populaires pour la navigation autonome d'un robot mobile. La première technique utilisée est basée sur l'élaboration d'un contrôleur flou de Takagi-Sugeno caractérisé par la capacité d'utiliser les connaissances d'un expert humain, et ne nécessite pas un modèle analytique du processus. Le deuxième système est un réseau de neurones multicouches distingué par une capacité d'apprentissage, et le troisième est un système hybride neuro-flou de type ANFIS qui intègre les deux propriétés. L'avantage de ce système hybride ANFIS est la génération des règles floues de manière automatique et l'ajustement des paramètres des contrôleurs flous en utilisant une base de données pour l'apprentissage. Ces contrôleurs sont utilisés pour réaliser la tâche de poursuite de trajectoire de référence et d'une cible mobile. Différentes types de trajectoires sont simulées. Les résultats de simulation obtenus montrent l'efficacité des contrôleurs proposés pour le contrôle du robot. Les résultats obtenus présentent une grande similarité entre les approches étudiées. L'avantage des contrôleurs est la simplicité et l'efficacité pour le contrôle du robot mobile.

L'utilisation les systèmes d'inférences flous et les réseaux de neurones artificiels nécessite l'acquisition des pairs état-action pour une application donnée. Dans le cas où ces connaissances ne sont disponibles, l'apprentissage par renforcement est une solution possible qui ne requiert pas une base de donnée, ou seulement un signal scalaire est utilisé par le robot mobile pour évaluer à chaque instant la qualité de l'action exécutée afin d'indiquer la performance du robot.

Dans le chapitre suivant, l'apprentissage par renforcement est utilisé pour la navigation d'un robot mobile en utilisant premièrement une discrétisation des espaces d'états et d'actions, puis une extension aux espaces d'état et d'action continus en utilisant les systèmes flous. Ce sont des systèmes flous entrainés par apprentissage par renforcement.

Chapitre VI

Navigation Autonome par des Contrôleurs Flous Entrainés par Renforcement

VI.1 Introduction

Nous avons vu dans le chapitre 3, que l'apprentissage par renforcement est une méthode adaptative basée sur le principe de punition-récompense pour résoudre un problème de décision séquentielle. C'est une méthode adaptative parce qu'on part d'une solution inefficace que l'on améliore progressivement en fonction de l'expérience acquise. On peut la considérer aussi comme une modification automatique du comportement où l'action optimale à effectuer pour chacune des situations est celle qui permet de maximiser l'espérance de gain [SUT98]. A chaque étape le robot doit définir l'état dans lequel il se trouve. A partir de cet état, il doit prendre une décision sur l'action à exécuter. En fonction du résultat obtenu lors de l'exécution de cette action, il est soit puni, pour diminuer la probabilité d'exécution de la même action dans le futur, soit récompensé, pour favoriser ce comportement dans les situations pareilles.

On a mentionné précédemment que deux approches sont possibles pour l'utilisation de la méthode d'apprentissage par renforcement. La première consiste à discrétiser le problème afin de fournir des espaces d'états qui pourront être utilisés directement par des algorithmes utilisant des tableaux de qualités Q. L'utilisation est difficile et exige le stockage des valeurs de la fonction qualité pour tous les couples (état, action) [GLO94, KAE96]. Dans les problèmes discrets de faible dimension, on peut utiliser des tableaux: une ligne correspond aux qualités des différentes actions pour un état donné. Mais dans le cas des espaces d'états et d'actions continus, le nombre de situation est infini et la représentation de la fonction Q par des tableaux est impossible. La deuxième va permettre de travailler directement dans les espaces d'états et d'actions continus en utilisant des méthodes d'approximation des fonctions comme les réseaux de neurones et les systèmes d'inférence flou qui offrent des solutions prometteuses pour ce problème et de limiter les effets de parasites qui pourraient apparaitre suite à un mouvais choix des espaces état-action pour la discrétisation [LIN91, BER92, TOU97, GLO97].

Dans ce chapitre, l'apprentissage par renforcement est utilisé pour la navigation autonome d'un robot mobile. Tout d'abord, nous utilisons l'algorithme Q-learning avec une discrétisation des espaces d'états et d'actions. Puis afin d'optimiser les contrôleurs flous et d'étendre la représentation aux espaces d'états et d'actions continus, une optimisation par le Q-learning flou sera étudiée (renforcement-flou). Elle est considérée comme une stratégie de commande avec une capacité d'apprentissage caractérisée par la possibilité d'introduction des connaissances à

priori pour accélérer l'apprentissage et de maximiser la moyenne des renforcements reçus. La tâche d'apprentissage consiste à optimiser les conclusions des règles floues en utilisant un signal de renforcement comme information de retour.

VI.2 Le Q-learning pour le comportement de convergence vers un but

Dans cette section, nous utilisons l'algorithme de Q-learning pour la tâche de convergence vers un but par un robot mobile [CHE13b]. Le Q-learning est l'algorithme le plus utilisé des algorithmes des différences temporelles. Dans cette application, le robot construit une fonction qualité Q qui définit la valeur de chaque action appliquée dans un état donné sous forme d'un tableau. Ce tableau comporte des coefficients représentant la qualité de telle ou telle action pour un état perçu. Après l'exécution d'une action choisie, les qualités doivent être ajustées en fonction du résultat obtenu. Cela se fait par le biais de la formule (VI.1) afin d'obtenir la valeur Q^* correspondant au comportement optimal. La fonction de mise à jour est la suivante :

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \beta\left[r_t + \gamma \max_{a \in A(s_{t+1})} Q(s_{t+1},a') - Q(s_t,a_t)\right] \quad \text{(VI.1)}$$

Où, β: le coefficient d'apprentissage, et $Q(s_t,a_t)$: la qualité en instant t de l'action a_t et l'état courant s_t.

Pour que le robot puisse à se déplacer vers son but, il doit calculer la distance par rapport à la configuration finale *(d_{rg})* et l'angle entre l'axe du robot et de but (θ_{rg}) ou l'erreur angulaire. Pour cela, on discrétise l'environnement de navigation en secteurs selon l'angle et des zones selon la distance robot-but. A partir de ces deux valeurs de mesures, le comportement est définit sous forme d'une matrice (tableau) de qualité. En utilisant une politique d'exploration-exploitation (*PEE)*, le robot doit choisir une action lui permettant de se mouvoir vers la direction de but après une phase d'apprentissage. L'action générée est l'angle de braquage α. Après l'exécution de cette action de déplacement, le robot est puni ou récompensé selon la nouvelle situation. Le comportement va changer jusqu'à une valeur optimale.

Les valeurs d'actions utilisées sont: avancer (α_1) , tourner à droite (α_2) et tourner à gauche (α_3). Ces actions sont choisies par la politique *(PEE* de type ε *-gloutonne)*. Ce qui permet au robot d'explorer l'espace d'états. Pendant le déplacement, le robot reçoit les valeurs suivantes comme signaux de renforcement (récompense ou punition):

$$r = \begin{cases} 4, & \text{Si le robot atteint la cible,} \\ 3, & \text{Si } d_{rg} \text{ déminue et } \theta_{rg} = 0, \\ 2, & \text{Si } d_{rg} \text{ et } \theta_{rg} \text{ déminuent,} \\ -1, & \text{Si } d_{rg} \text{ déminue et } \theta_{rg} \text{ augmente,} \\ -2, & \text{Si } d_{rg} \text{ augmente,} \\ -3, & \text{Si le robot percute les murs de son environnement.} \end{cases} \quad \text{(VI.2)}$$

Dans le premier essai, on a entrainé le robot en le plaçant dans la même position initiale par rapport au but pour chaque épisode. D'après la figure qui représente l'évolution des valeurs moyennes des récompenses obtenues par le robot dans chaque cycle de mouvement (figure VI.1), on remarque que le comportement de convergence vers le but s'améliore au cours du temps d'apprentissage. A partir de l'episode 1000, le robot converge vers le meilleur

comportement et peut atteindre son but efficacement en exécutant des action discrètes, comme illustré sur la figure VI.2 donnant la trajectoire du robot mobile.

Afin de généraliser la navigation du robot mobile pour toutes les configurations possibles (toutes les positions du robot et de la cible), l'apprentissage se fait avec une position initiale aléatoire du robot et de but dans chaque épisode. Les figures VI.3 (a-b-c-d) représentent les trajectoires du robot obtenues après une phase d'entrainement, le robot est capable de rejoindre son but final quelque soit sa position initiale en exécutant des actions discrètes. Les valeurs moyennes du renforcement comme un indicateur d'apprentissage sont maximixées au cours de l'apprentissage pour obtenir un comportement acceptable pour la convergence vers un but (figure VI.4).

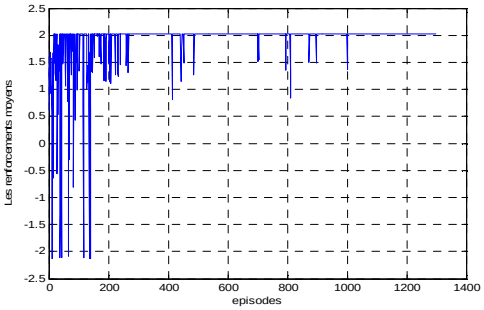


Figure VI.1 Valeurs moyennes de renforcement

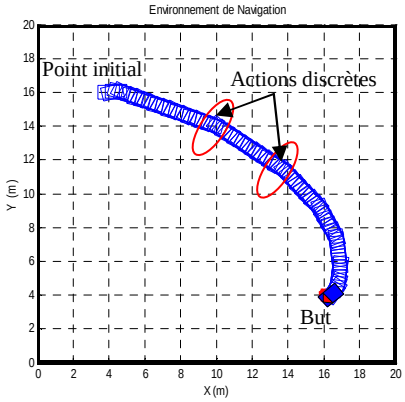
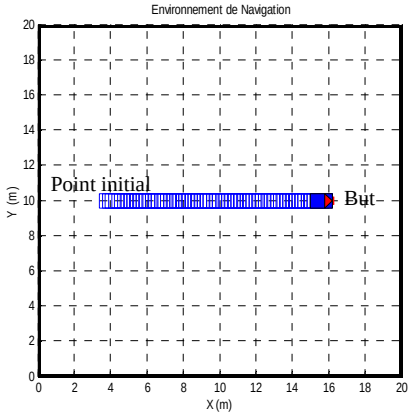
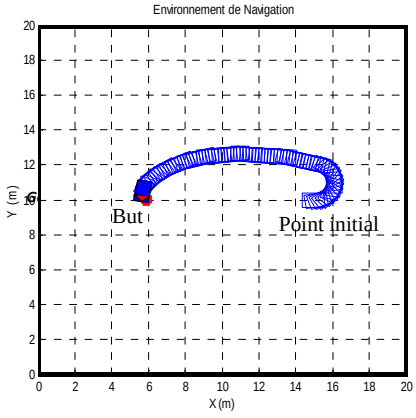


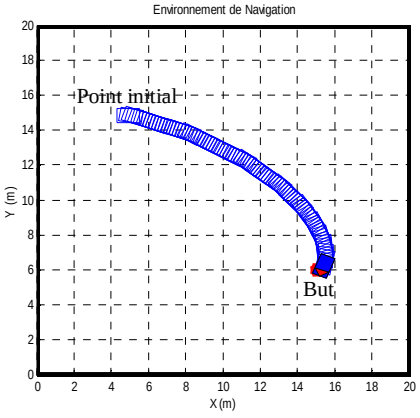
Figure VI.2 Trajectoire après apprentissage



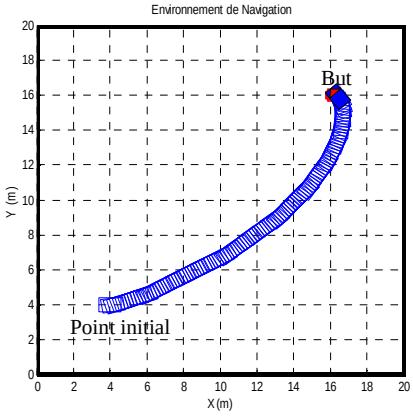
(a)



(b)



(c)



(d)

Figure VI.3 Convergence vers un but par le Q-learning

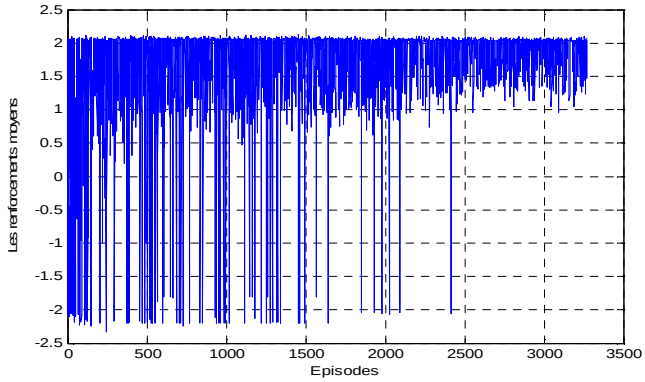


Figure VI.4 Renforcements reçus dans chaque épisode

Dans ce travail, pour obtenir un comportement acceptable, nous avons appliqué plusieurs fois l'algorithme Q-learning pour le comportement de convergence vers un but en faisant varier le nombre d'états et les actions proposées. L'augmentation du nombre de paires état-action permet d'améliorer le comportement du robot, mais nécessite une capacité de mémoire plus importante et un temps d'apprentissage plus long illustré par le nombre des épisodes nécessaires pour la convergence de l'algorithme (voir Tableau VI.1) [CHE12a, CHE13b].

Nombre d'état	Actions proposées	Nombre des épisodes
09	03	3260
11	03	4100
13	03	6200
19	05	8950
…	…	…

Tableau VI.1 Augmentation des épisodes en fonction des paires état-action

L’utilisation des algorithmes d’apprentissage par renforcement nécessite deux aspects temporels et structurels. Le problème temporel est résolu par le signal de renforcement donné au robot mobile, mais le problème structurel est lié par la méthode de représentation des espaces d’états et d’actions. Le Q-learning est un algorithme itératif, son inconvénient est la difficulté d'obtenir une convergence permettant la distribution des séquences d’action aux régions d'états. L’apprentissage par renforcement est parfaitement adapté aux processus dont le nombre d’états et d’actions est fini, mais en robotique mobile, les espaces d'entrés et de sortie des capteurs et des actionneurs sont rarement discrets, ou, si ils le sont, le nombre d'état est très grand. Or l'apprentissage par renforcement tel que nous l'avons décrit utilise des espaces discrets qui doivent être de taille raisonnable pour permettre aux algorithmes de converger en un temps raisonnable pratiquement. Dans ce cas l’utilisation d’autres méthodes comme les approximateurs universels tels que les systèmes d’inférences floues et les réseaux de neurones peut donner des solutions prometteuses pour approximer les tableaux de qualités qui sont les bases de ces algorithmes.

Afin de profiter des nombreux développements théoriques associés aux méthodes d'apprentissage par renforcement et les systèmes flous, il est nécessaire d'y apporter des modifications pour permettre l'ajustement de ces paramètres de telle manière qu'elles prennent en considération les caractéristiques spécifiques des SIFs, avec notamment l'utilisation des

algorithmes d'apprentissage par renforcement, et de travailler avec des espaces d'état et d'action continus [GLO00, GU04]. L'utilisation des SIFs apporte plusieurs avantages [JOU98, GLO99]:

- La possibilité de représenter la valeur d'une politique, en raison de la propriété d'approximation universelle,
- La possibilité de traiter simplement des espaces d'état et d'action continus,
- L'intégration des connaissances à priori et l'interprétation des connaissances acquises à l'issue de l'apprentissage.

Dans ce qui suit, pour objectif d'améliorer les performances des contrôleurs du robot mobile et afin d'introduire une connaissance préalable à la conception de ses comportements, nous utilisons les systèmes d'inférence flous pour le passage aux espaces d'état et d'action continus et pour que le comportement initial soit acceptable. Avant de présenter l'approche qu'on va l'utiliser dans ce travail, nous consacrons le paragraphe suivant à une représentation de la fonction qualité et les systèmes flous entrainés par apprentissage par renforcement.

VI.3 Représentation de la fonction qualité *Q*

La fonction *Q* doit être stockée pour tous les couples (état, action). Pour les problèmes discrets de petite dimension, comme présenté précédemment, on peut envisager l'utilisation de matrices. Mais dans le cas continu, l'algorithme Q-learning doit être étendu aux espaces d'état et d'actions continus. Le Q-learning a été utilisé au cas d'un espace d'entré continu avec des actions discrètes. Dans ce cas, il n'est plus possible de visiter tous les états ni de représenter *Q* par des tableaux. Dans la structure *Q-CON* proposée par Lin [LIN91], l'auteur a proposé d'utiliser autant de réseaux de neurones qu'il y a d'actions. Cette représentation de *Q* exploite les propriétés d'approximation des réseaux de neurones, mais il n'y a plus de convergence de Q vers Q^* optimale. Dans cette architecture, on utilise *N* réseaux de neurones ou *N* est le nombre d'actions. Le réseau de neurone RN_i est utilisé pour l'approximation de la fonction $Q(s,a_i)$ de l'action a_i [GLO94, TOU97]. L'inconvénient majeur est de cumuler deux approches assez lentes: le Q-learning et l'apprentissage neuronal. Après l'exécution de l'action a_k à l'état s, la différence $Q_{t+1}(s,a_k)-Q_t(s,a_k)$ entre les évaluations de qualité aux pas de temps t et $t+1$ peut être considérée comme un signal d'erreur dans une implémentation neuronale. L'algorithme de rétro-propagation est utilisé pour l'optimisation des réseaux de neurones en minimisant le critère:

$$E_t(s,a_k)=\tfrac{1}{2}\left[Q_{t+1}(s,a_k)-Q_t(s,a_k)\right]^2 \qquad \text{(VI.3)}$$

Une version floue de cette représentation pour un espace d'état continu et des actions discrètes, appelée *Q-FUZ* par analogie avec le modèle de Lin, a été proposée par Jouffe [JOU98]. La fonction qualité est implémentée par un système d'inférence flou à N sorties. Une fois les fonctions d'appartenance sont choisies, la tâche d'apprentissage consiste à optimiser les conclusions des règles. Ces conclusions étant les valeurs $s \rightarrow Q(s,a_j)$ cherchées, pour $j=1...N$. Cette structure n'exploite pas les propriétés d'interpolation des SIFs et utilise des sorties discrètes.

Il existe plusieurs extensions du Q-learning pour les systèmes d'inférence flous [BER92, GLO94, BEO95, MYK05]. Deux extensions autres que la structure *Q-FUZ* qui permet d'utiliser des actions discrètes, infèrent des sorties continues: la première est utilisée pour l'extraction des

connaissances et la deuxième pour la fusion des connaissances [GLO99, GLO00]. L'algorithme Q-Learning associé aux systèmes flous est appelé le Q-learning flou [GLO97], c'est un algorithme d'optimisation des conclusions des règles floues pour les systèmes de Takagi-Sugeno [GLO99, MYK05] ou de Mamdani [BOU01]. Cet algorithme est bien adapté pour le problème de navigation des robots mobiles [GLO96, GU04, RIS05, CHE13b].

Dans la section suivante, on va présenter le principe de cet algorithme pour le mettre en œuvre dans notre travail comme une méthode d'optimisation des contrôleurs flous de TS. Cet algorithme peut être considéré comme un système hybride renforcement-flou.

VI.4 Q-learning flou (*QLF*)

Le principe de cet algorithme consiste à proposer plusieurs conclusions pour chaque règle d'inférence floue, et à associer à chaque conclusion une fonction qualité qui sera évaluée incrémentalement avec des fonctions d'appartenances fixes. Il s'agit alors de trouver la conclusion appropriée pour chaque règle [GLO94, JOU98]. C'est une extension naturelle du Q-learning de base aux espaces d'état et d'action continus et une méthode d'optimisation des contrôleurs flous. Lors de l'apprentissage par l'algorithme du Q-learning flou, le contrôleur doit donc ajuster les conclusions de ses règles sur la base du renforcement [GLO99, MYK05]. La tâche consiste à approcher la fonction qualité Q par la fonction suivante qui est un SIF:

$$s \to y = \hat{Q} = SIF(s) \tag{VI.4}$$

Comme la représentation choisie est un SIF de Takagi-Sugeno d'ordre zéro (TS0), cette fonction est définie par les règles de la forme:

$$\begin{aligned} &\text{Si } s \text{ est } S_1 \text{ Alors } y = c_1 \\ &\text{Si } s \text{ est } S_2 \text{ Alors } y = c_2 \\ &\quad \dots \\ &\text{Si } s \text{ est } S_m \text{ Alors } y = c_r \end{aligned} \tag{VI.5}$$

Ou m est le nombre des règles, S_i est donnée par: x_1 est A_1^i et... et x_n est A_n^i . S_i est la prémisse de la règle i, appelée aussi le vecteur modal correspondant de la règle i ou le prototype de la règle i.

Pour un vecteur d'entrée s_t, l'évaluation de la valeur d'action est donnée par l'équation suivante:

$$\hat{Q} = SIF(s_t) = \sum_{i=1}^{m} w_i(s_t).c_i \tag{VI.6}$$

En prenant $c_i = q[S_i, a]$ dans l'équation précédente, on trouve:

$$\hat{Q} = \sum_i w_i(s_t).q[S_i, a] \tag{VI.7}$$

Les conclusions des règles floues $(c_i)_{i=1}^m$ peuvent être interprétées comme la restriction de la fonction $s \to \hat{Q}(s,a)$ vers les vecteurs modaux (S_i) telle que:

$$q[S_i, a] \equiv \hat{Q}(s,a) \text{ si } s = S_i, \text{ pour } i = 1,...r \tag{VI.8}$$

où $q[S_i, a]$ est appelée la fonction valeur de l'action a à l'état S_i (pour la règle *i*).

Le processus d'apprentissage de l'algorithme Q-learning flou pour l'extraction de la connaissance (l'algorithme VI.1) permet de déterminer l'ensemble des règles maximisant les renforcements futurs [GLO00, CHE13b]. La base des règles initiale est composée de m règles de la forme suivante:

$$\begin{aligned} &\text{Si } s \text{ est } S_i \text{ Alors} \quad && y = a[i,1] \text{ avec } q[i,1] = 0 \\ &\text{ou} && y = a[i,2] \text{ avec } q[i,2] = 0 \\ &&& \dots \\ &\text{ou} && y = a[i,N] \text{ avec } q[i,N] = 0. \end{aligned} \tag{VI.9}$$

Ou $q(i, j)$ avec $i = 1...m$ et $j = 1...N$ sont des solutions potentielles dont la valeur est initialisée à 0. Durant l'apprentissage, la conclusion de chaque règle est choisie au moyen d'une politique d'exploration-exploitation donnée (ε *-gloutonne* par exemple) : $PEE(i) \in \{1...N\}$. Dans ce cas, la sortie inférée est donnée par:

$$A(s) = \sum_{i=1}^{m} w_i(s).a[i, PEE(i)] \tag{VI.10}$$

La qualité de cette action sera alors:

$$\hat{Q}(s, A(s)) = \sum_{i=1}^{N} w_i(s).q[i, PEE(i)] \tag{VI.11}$$

1- Choisir la structure de SIF.

2- Initialiser $q[i, j] = 0$ $\quad i = 1,...m$ *Nombre des règles,*
$\quad j = 1,...N$ *Nombre des conclusions proposées,*

3- **Répéter** (pour chaque épisode)
   a. Observer l'état s
   b. **Répéter** (pour chaque pas de l'épisode)
      i. Pour chaque règle i, calculer $w_i(s)$
      ii. Pour chaque règle i, choisir une conclusion à l'aide d'une *PEE*
      iii. Calculer la sortie $A(s)$ et sa qualité correspondante $Q(s, A(s))$
      iv. Appliquer l'action $A(s)$, soit s' le nouvel état
      v. Recevoir le renforcement r
      vi. Pour chaque règle i, calculer :

$$V^*(s') = \sum_{i=1}^{m} w_i(s').q[i, \max(i)].$$

      vii. Calculer $\Delta Q = \beta[r + \gamma V^*(s') - \hat{Q}(s, A(s))]$
      viii. Mettre à jours les qualités élémentaires des conclusions utilisées

$$q[i, j] = q[i, j] + \Delta Q[i, j]$$

$$s \leftarrow s'$$

Jusqu'à $s_t \in F$

Algorithme VI.1 Algorithme Q-learning flou à un pas

VI.5 Q-learning flou pour la navigation d'un robot mobile

Dans ce travail, on utilise l'algorithme *Q-learning flou à un pas* présenté dans l'algorithme VI.1 pour les différents comportements d'un robot mobile autonome (recherche d'un but, évitement d'obstacles, suivi de murs,…) [CHE13b]. Dans cet algorithme, le robot possède une base de règles initiale de type TS0, qui définit les situations possibles pour le comportement désiré. Le principe consiste à proposer plusieurs conclusions de type singletons pour chaque règle, et à associer à chaque conclusion une fonction qualité qui sera optimisée au cours du temps. L'objectif de la phase d'apprentissage est de déterminer l'ensemble des conclusions des règles maximisant la moyenne des renforcements. Après le choix d'une conclusion, l'action inférée sera exécutée et les qualités seront adaptées en fonction du renforcement reçu et la nouvelle situation perçue.

VI.5.1 Comportement de convergence vers un but (Recherche de but)

Pour atteindre un but dans un environnement libre d'obstacles, le robot mobile doit avancer, tourner à droite ou à gauche avec des vitesses différentes selon les variables d'entrée du contrôleur. Dans ce cas, comme présenté dans les exemples précédents pour le même comportement, le robot doit mesurer la distance robot-but (d_{rg}) et l'erreur angulaire (θ_{rg}). En utilisant ces deux valeurs, le contrôleur flou optimisé par le QL va générer les deux actions de déplacement vers le but (angle de changement de direction et vitesse de déplacement). Pour cela, le comportement est définit par un SIF de type TS0. Pour les variables d'entrée, on utilise les fonctions d'appartenances présentés sur les figures VI.5 et VI.6 avec les variables linguistiques suivantes: P(Proche), M(Moyenne), G(Grande), NG(Négative Grande), NM(Négative Moyenne), Z(Zéros), M(Positive Moyenne) et PG(Positive Grande). Ce qui donne 15 règles. Pour chaque règle floue, on propose 3 conclusions pour la variable de sortie angle de braquage selon la forme suivante:

$$\begin{aligned} &\text{Si } s \text{ est } S_i \text{ Alors} && \alpha=\alpha_{i1} \text{ avec la qualité } q[i,1] \\ &\text{ou} && \alpha=\alpha_{i2} \text{ avec la qualité } q[i,2] \\ &\text{ou} && \alpha=\alpha_{i3} \text{ avec la qualité } q[i,3] \end{aligned} \qquad \text{(VI.12)}$$

pour : $i=1...15$. $\alpha_1=-\pi/5$, $\alpha_2=0$ et $\alpha_3=\pi/5$.

Dans notre cas, pour entrainer le robot mobile pour le comportement de convergence vers un but, on a appliqué l'algorithme Q-learning flou deux fois pour les deux commandes, pour l'apprentissage de l'angle de braquage et la deuxième pour la vitesse de déplacement. Pour la première, durant la phase d'exploration, le robot reçoit les mêmes valeurs de renforcement utilisées dans la section précédente (eq. VI.2). Après une phase d'apprentissage, le robot choisit pour chaque règle la conclusion avec la meilleure fonction qualité $q[i,j]_{j=1}^{N}$.

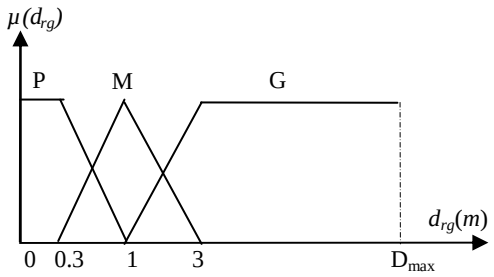
Lorsque le robot est capable de rejoindre son but final avec une vitesse fixe, on applique le deuxième algorithme pour optimiser le contrôleur de vitesse, les prémisses des règles sont les mêmes utilisées pour le calcul de l'angle de braquage mais la différence est dans la partie conclusion. La base de règles est sous la forme donnée par l'équation VI.13.

$$\begin{aligned} &\text{Si } s \text{ est } S_i \text{ Alors} && V_r = V_{i1} \text{ avec la qualité } q[i,1] \\ &\text{ou} && V_r = V_{i2} \text{ avec la qualité } q[i,2] \\ &\text{ou} && V_r = V_{i3} \text{ avec la qualité } q[i,3] \end{aligned} \qquad \text{(VI.13)}$$

pour: $i = 1...15$.

Pendant l'apprentissage de la vitesse, le robot mobile reçoit les renforcements suivants:

$$r = \begin{cases} -1, & \text{Si } d_{rg}, \theta_{rg} \text{ déminuent et la vitesse augmente,} \\ -1, & \text{Si } \theta_{rg} \text{ déminue, la vitesse et } d_{rg} \text{ augmentent,} \\ 1, & \text{Si le robot atteint la cible avec une faible vitesse,} \\ 0, & \textit{ailleurs}. \end{cases} \qquad \text{(VI.14)}$$



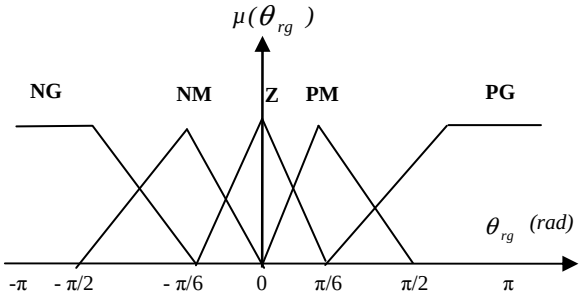
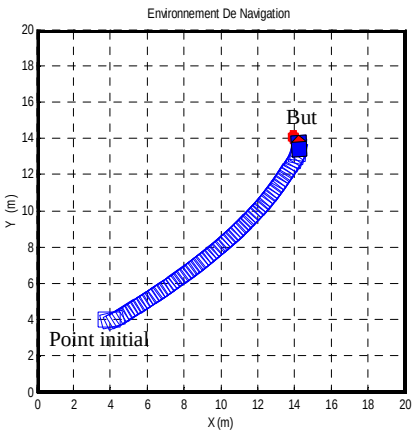


Figure VI.5 Fonctions d'appartenance de d_{rg} **Figure VI.6** Fonctions d'appartenance de θ_{rg}

VI.5.1.1 *Résultat de convergence vers un but par le QLF*

Pour un apprentissage avec une position fixe du robot et de son but, le comportement obtenu est représenté sur la figure VI.7, donnant la trajectoire du robot pour accomplir la tâche de recherche de but. Après un temps d'apprentissage suffisant, l'algorithme est arrêté. On remarque que le comportement du robot s'améliore pendant la phase d'apprentissage comme le montre la figure VI.8 représentant la valeur moyenne des renforcements reçus pour chaque épisode. A partir de l'episode 180, la valeur moyenne des renforcements est constante indiquant que l'algorithme d'apprentissage a convergé vers une solution optimale pour cette position initiale. Les actions inférées et exécutées sont continues (espace d'état et d'action continus), et l'apprentissage est plus rapide que l'algorithme Q-learning si on les compare avec les figures VI.1 et VI.3.



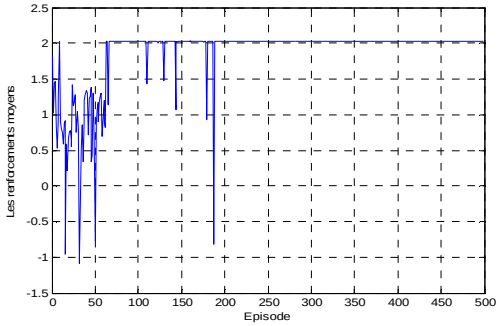
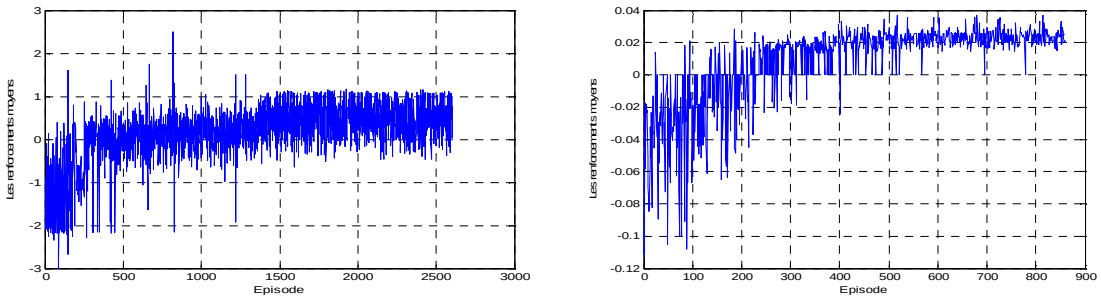


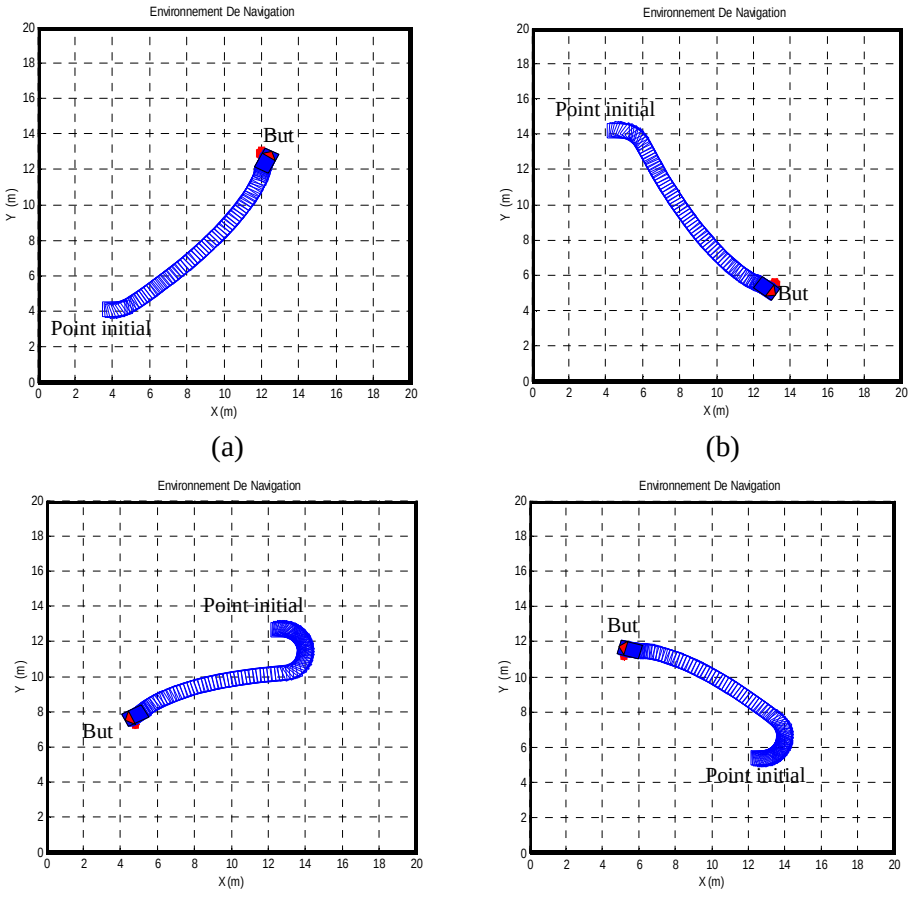
Figure VI.7 Trajectoire du robot après apprentissage **Figure VI.8** Renforcements moyens

Afin d'optimiser le système de navigation renforcement-flou, les positions initiales sont choisies aléatoirement pendant la phase d'apprentissage. Chaque épisode commence par une position aléatoire et se termine lorsque le robot atteint le but ou percute les limites de son environnement. Les figures VI.9 (a-b) représentent les valeurs moyennes du renforcement obtenues dans chaque épisode pour apprendre les deux valeurs d'actions (angle de braquage et vitesse de translation). Comme montré sur ces deux courbes, le comportement du robot s'améliore au cours de temps. Les renforcements sont maximisés illustrant la convergence des algorithmes QLF utilisés. Les trajectoires du robot mobile pour cette mission avec des différentes situations après apprentissage seront représentées sur les figures VI.10 (a-b-c-d). On remarque que le robot se dirige vers le but et s'arrête lorsqu'il atteint la cible quelque soit sa position initiale de façon autonome, en exécutant des actions continues.



(a) (b)

Figure VI.9 Valeurs de renforcement avec 15 règles pour: (a) le braquage (b) la vitesse



(a) (b)

(c) (d)

Figure VI.10 Trajectoires de convergence vers un but (15 règles - 3 conclusions)

Dans le deuxième essai, on va optimiser le comportement flou utilisé dans le chapitre 4 pour le comportement de convergence vers un but. Les fonctions d'appartenances des variables d'entrées sont présentées sur les figures VI.11, VI.12, avec les variables linguistiques suivantes: Z(Zéro), P(Petite), F(Faible), M(Moyenne), G(Grande), TG(Très Grande), NG(Négative Grande), NM(Négative Moyenne), NP(Négative Petite), PP(Positive Petite), PM(Positive Moyenne) et PG(Positive Grande). Ce système d'inférence flou est composé de 35 règles, on propose pour chacune 5 conclusions uniformément réparties dans l'intervalle $[-\pi/5, \pi/5]$ avec : $\alpha_1 = -\pi/5$, $\alpha_2 = -\pi/10$, $\alpha_3 = 0$, $\alpha_4 = \pi/10$ et $\alpha_5 = \pi/5$. On commence par l'optimisation du contrôleur de braquage en supposant que la vitesse est constante. La base de règle est donnée par l'équation VI.15. Ce contrôleur est utilisé ensuite dans la phase de l'apprentissage du contrôleur de vitesse avec les mêmes renforcements donnés au eq. VI.14.

$$\begin{aligned} \text{Si } s \text{ est } S_i \text{ Alors} \quad & \alpha = \alpha_{i1} \text{ avec la qualité } q[i,1] \\ \text{ou} \quad & \alpha = \alpha_{i2} \text{ avec la qualité } q[i,2] \\ \text{ou} \quad & \alpha = \alpha_{i3} \text{ avec la qualité } q[i,3] \\ \text{ou} \quad & \alpha = \alpha_{i4} \text{ avec la qualité } q[i,4] \\ \text{ou} \quad & \alpha = \alpha_{i5} \text{ avec la qualité } q[i,5] \end{aligned} \tag{VI.15}$$

Les actions inférées sont données par :

$$\alpha_k(s) = \sum_{i=1}^{35} w_i(s).\alpha[i,k] \tag{VI.16}$$

$$V_r(s) = \sum_{i=1}^{35} w_i V_r[i,k] \tag{VI.17}$$

Les figures VI.13 (a-b) représentent les renforcements reçus dans chaque épisode pour les deux valeurs de sorties. Le comportement du robot s'améliore au cours de l'apprentissage, mais la convergence est plus lente que pour l'essai précédent utilisant 15 régles floues, due au nombre important des règles floues utilisées. Des exemples des trajectoies du robot pour des différentes situations initiales après la phase d'apprentissage sont présentées sur les figure VI.14 (a-b-c-d).

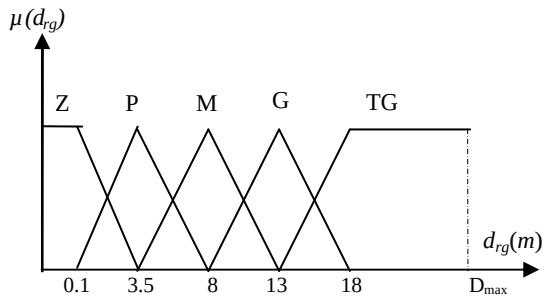


Figure VI.11 Fonctions d'appartenance de d_{rg}

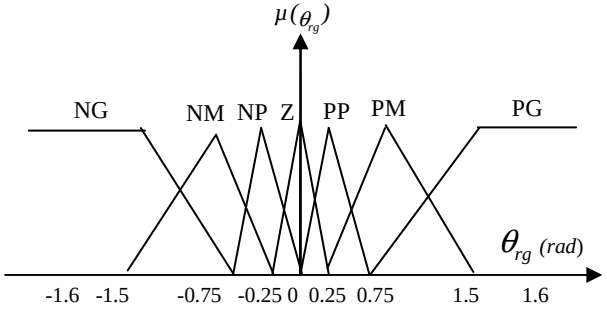
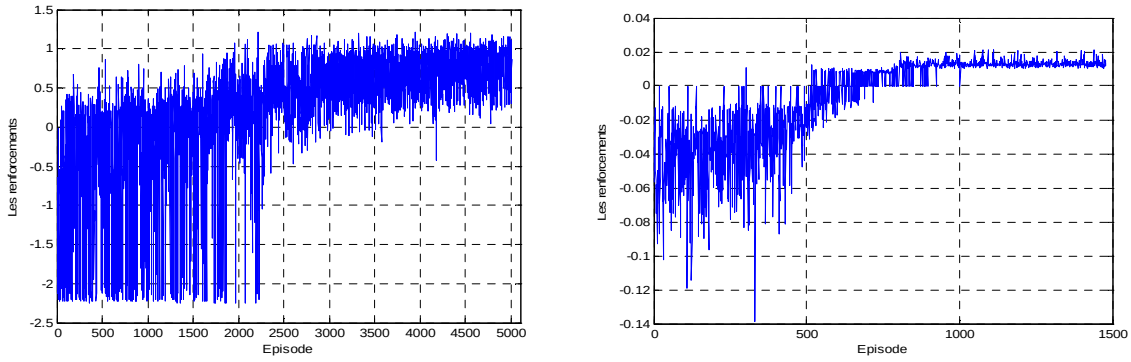
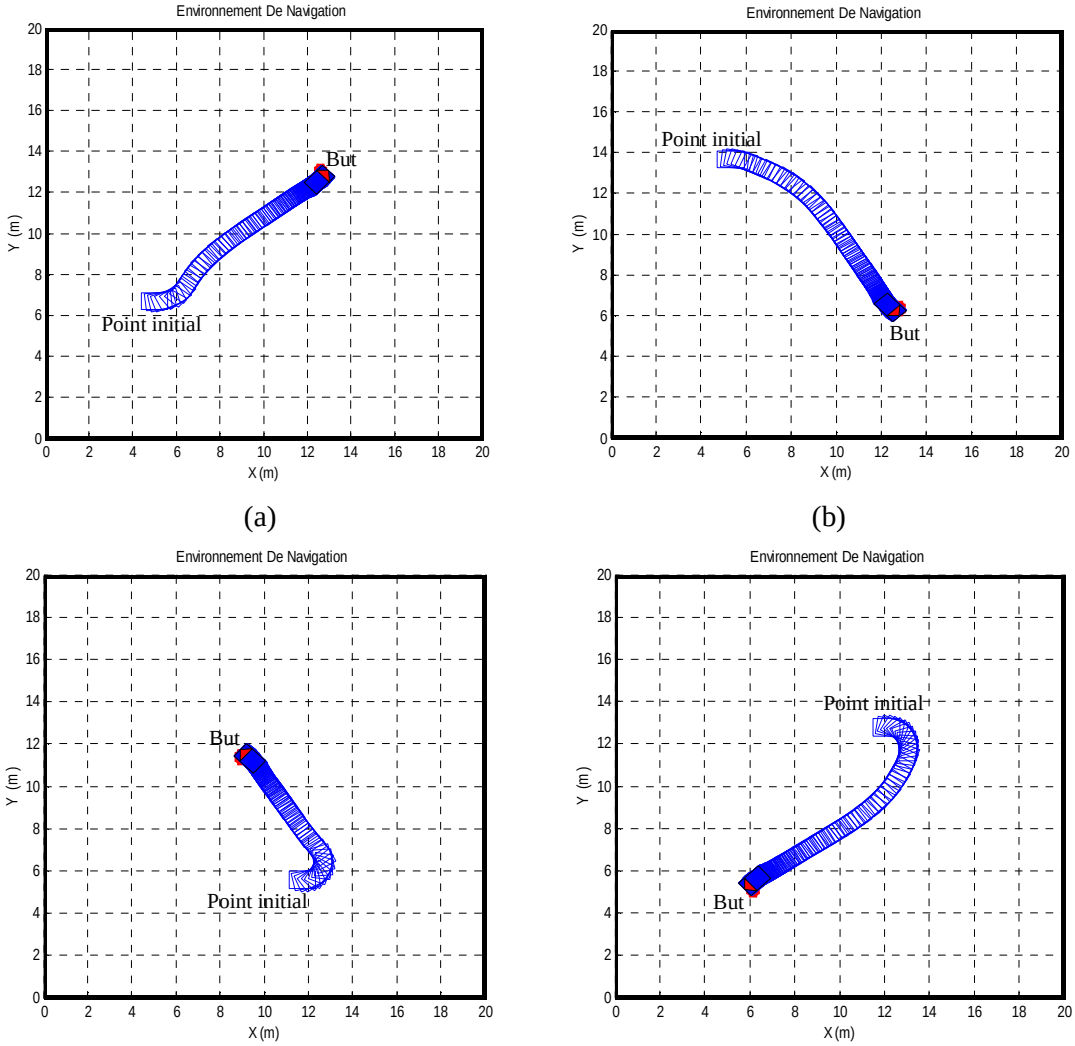


Figure VI.12 Fonctions d'appartenance de θ_{rg}



(a) (b)

Figure VI.13 Valeurs de renforcement avec 35 règles pour: (a) le braquage (b) la vitesse



(a) (b)

(c) (d)

Figure VI.14 Convergence vers un but (35 règles-5 conclusions)

Une autre amélioration du système de contrôle flou optimisé par renforcement est possible en utilisant les traces d'éligibilités. Les trajectoires du robot pour une tâche de recherche d'un but avec 15 règles floues et 3 conclusions proposées sont représentées sur les figures VI.15 (a-b-c-d). On observe l'amélioration du comportement du robot comme le montre la figure des valeurs moyennes de renforcement reçues (figure VI.16). Dans tous les cas, le robot se dirige vers la

cible quelque soit sa position initiale pour des espaces état-action continus. L'apprentissage en utilisant les traces d'éligibilités est plus rapide que pour le Q-learning flou précédent (figures VI. 10) et de même pour le Q-learning (figure VI.4).

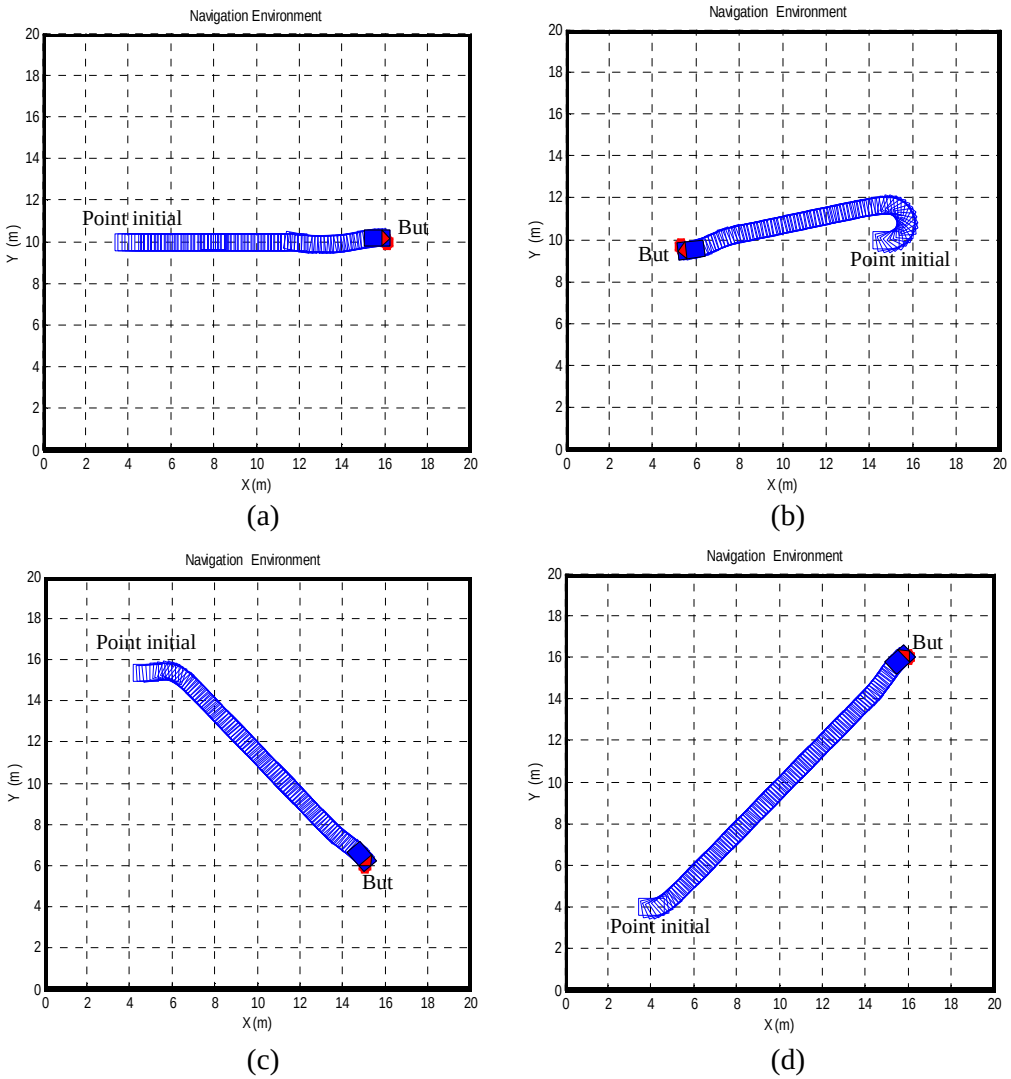


Figure VI.15 Convergence vers un but par QLF avec les traces d'éligibilités (15 règles-3 conclusions)

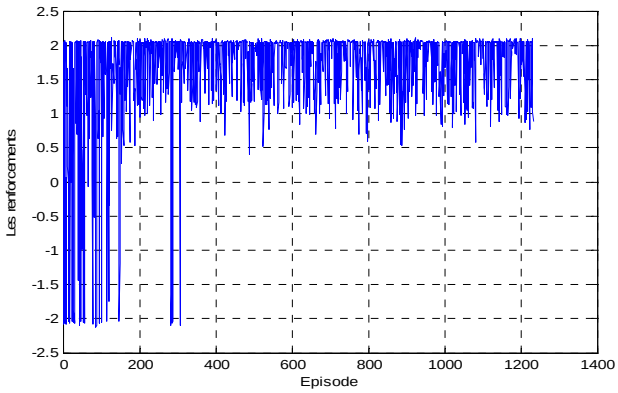


Figure VI.16 Moyenne des récompenses obtenues

VI.5.2 Comportement d'évitement d'obstacles

L'évitement d'obstacles est une tâche de base que doivent posséder tous les robots mobiles, puisque ceci permet au robot de se déplacer dans un environnement inconnu en toute sécurité sans collisions avec les objets qu'il contourne. Dans ce travail, On suppose que le robot est capable de mesurer les distances à un obstacle en trois cotés (en face, à droite et à gauche) à base d'un système de perception efficace.

Le système de navigation autonome basé sur le Q-learning flou utilise comme entrées, les distances aux obstacles dans les trois directions, et doit générer après une phase d'apprentissage l'angle de braquage de la roue avant (α) et la vitesse de translation du robot (V_r). Deux variables linguistiques, Proche (P) et Loin (L), sont utilisées pour décrire chacune des trois distances. Ces variables sont représentées par les fonctions d'appartenances de la figure VI.17. En utilisant ces fonctions d'appartenances, la base de règles floues se compose de 8 situations de base. L'algorithme du Q-learning est utilisé pour l'optimisation du navigateur flou en générant un nombre des valeurs candidates pour chaque conclusion. Pour chaque règle et chaque conclusion, on associe une fonction qualité $q[i,j]_{j=1}^{J}$ à l'interprétation j de la règle i, avec : $i=1...8$ et $j=1,2,3$. La base de règle est de la forme suivante:

$$\begin{aligned} &\text{Si } s \text{ est } S_i \text{ Alors} \quad \alpha=\alpha_{i1} \text{ avec la qualité } q[i,1] \\ &\qquad\qquad \text{ou} \quad \alpha=\alpha_{i2} \text{ avec la qualité } q[i,2] \\ &\qquad\qquad \text{ou} \quad \alpha=\alpha_{i3} \text{ avec la qualité } q[i,3] \end{aligned} \tag{VI.18}$$

Avec : $i=1...8$ et $j=1,2,3$, $\alpha_1=-\pi/5$, $\alpha_2=0$ et $\alpha_3=\pi/5$.

La qualité globale de la sortie du contrôleur flou est donnée par :

$$Q(s,\alpha)=\sum_{i=1}^{8} w_i(s).q[i,j^*(i)] \tag{VI.19}$$

On utilise un signal de renforcement qui puni toute conclusion de règle activée quand le robot rentre en collision avec un obstacle :

$$r=\begin{cases} -4, \text{ Si le robot percute un obstacle,} \\ -1, \text{ Si } d_i \text{ diminue et } d_i<\dfrac{D_{\max}}{2}, \\ 0, \text{ ailleurs,} \end{cases} \tag{VI.20}$$

Les valeurs de renforcements utilisées pour l'apprentissage de la vitesse sont :

$$r=\begin{cases} -4, & \text{Si le robot percute un obstacle,} \\ -1, & \text{Si } d_i>D_{\max} \text{ et } V_r \text{ est faible, avec } i=1...3, \\ -1, & \text{Si } d_i<D_{\max} \text{ et } V_r \text{ augmente, avec } i=1...3, \\ 0, & \text{ailleurs.} \end{cases} \tag{VI.21}$$

Dans cette application, la valeur de renforcement va servir à déterminer la meilleure conclusion parmi les trois conclusions proposées pour les deux commandes. On suppose que le robot au départ est dans une zone libre d'obstacle et s'oriente vers la direction de but pour le rejoindre en évitant la collision.

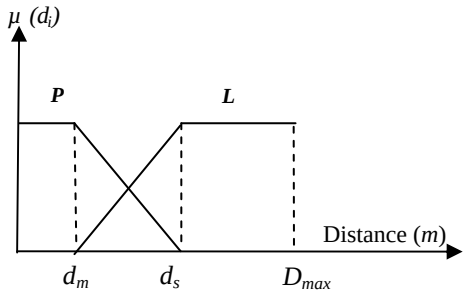


Figure VI.17 Fonctions d'appartenance de la distance à un obstacle

Durant l'apprentissage, le robot doit explorer les actions possibles pour chaque situation perçue. Dans les figures VI.18 et VI.19, on présente les trajectoires du robot mobile dans les premiers épisodes (épisode 1 et 20). On observe des collisions avec les obstacles sont produites au début de l'apprentissage. A partir de l'episode 250, le robot obtient le meilleur comportement pour converger vers le but en évitant l'obstacle comme le montre la figure VI.20. Après la phase d'apprentissage, l'algorithme est arrêté et les conclusions des règles avec les meilleures qualités sont choisies. Le robot peut suivre les murs pour atteindre un objectif dans un environnement fermé (figure VI.21). Les récompenses moyennes obtenues pour les deux phases d'apprentissage sont représentées sur les figures VI.22 (a-b).

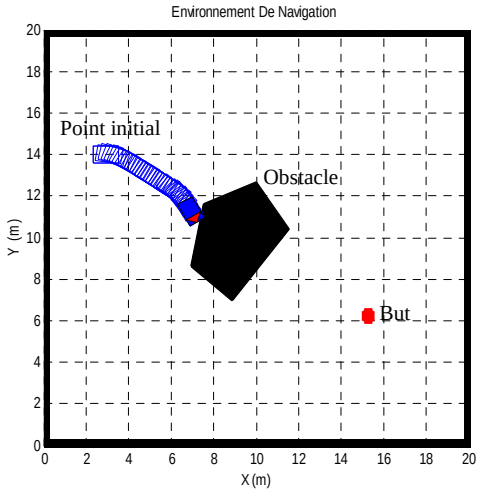


Figure VI.18 Trajectoire à l'epiosde 1

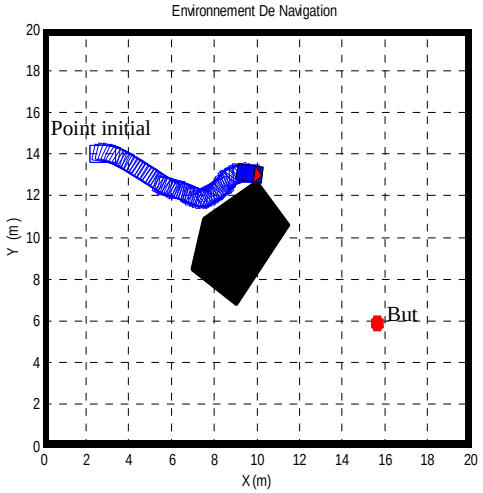


Figure VI.19 Trajectoire à l'episode 20

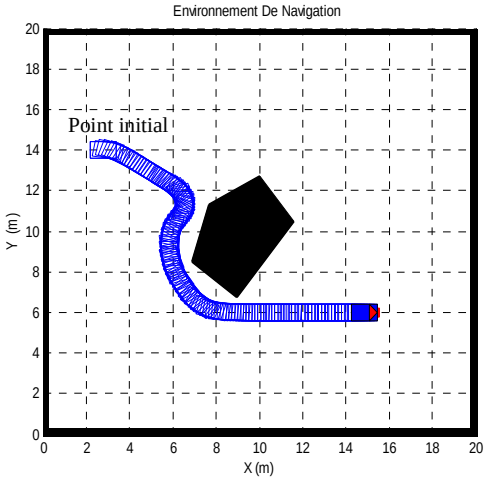


Figure VI.20 Trajectoire du robot après apprentissage

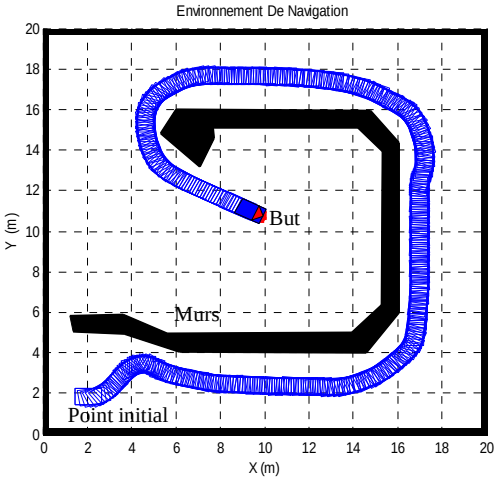
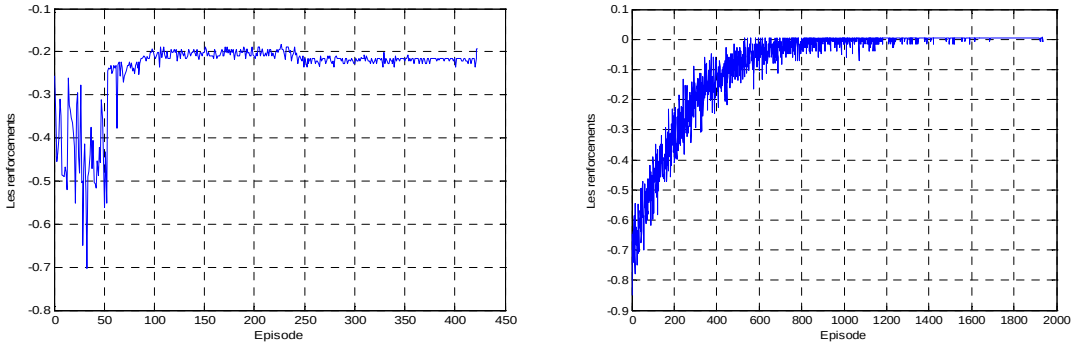


Figure VI.21 Suivi de mur



(a) (b)

Figure VI.22 Valeurs de renforcement: (a) le braquage (b) la vitesse

VI.5.2.1 *Comparaison avec un contrôleur flou*

Dans cette section, on présente une comparaison entre le navigateur renforcement-flou proposé et le contrôleur flou présenté sur le chapitre IV. Pour le même environnement de navigation, le comportement du robot dans les deux situations (flou et renforcement-flou) sont illustrés sur les figures VI.23 et VI.24 respectivement. Le robot se déplace vers le but, s'il détecte l'obstacle, il doit activer le comportement approprié afin d'éviter la collision. Les résultats sont acceptables et le contrôleur renforcement-flou réagit d'une manière très satisfaisante pour cette mission.

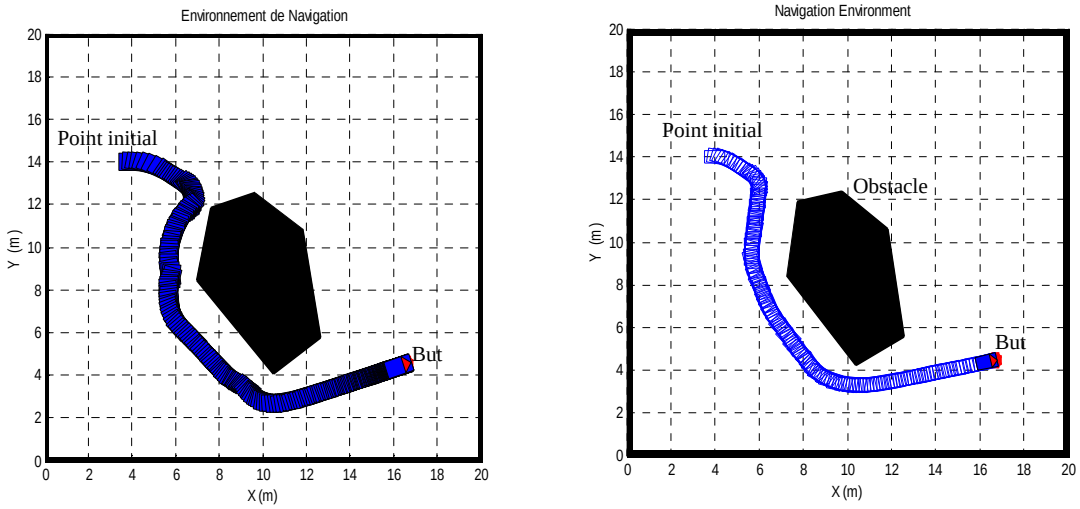


Figure VI.23 Evitement d'obstacle par le contrôleur flou **Figure VI.24** Evitement d'obstacle par QLF

VI.5.3 Comportement de suivi de mur

L'objectif du comportement de suivi de mur est de garder le robot à une distance de sécurité avec les murs détectés sur sa droite ou de sa gauche en effectuant un mouvement de type droite. Pour cela, le robot doit acquérir les informations nécessaires pour cette mission. Dans notre travail, on utilise deux types des contrôleurs pour la tâche de suivi de murs: le premier est un contrôleur flou TS0, et le deuxième est un navigateur flou entrainé par apprentissage par renforcement.

VI.5.3.1 *Navigateur flou*

Pour des raisons de simplicité, on utilise un contrôleur flou de Takagi-Sugeno d'ordre zéro dont le fonctionnement est imposé tel que:

- les règles de conduite sont générales, peuvent être utilisées pour tous les robots mobiles. Elles peuvent être individualisées par apprentissage, pour tenir compte des caractéristiques mécanique et dynamiques des robots,
- l'introduction de connaissances rend le robot immédiatement efficace dès le début de l'apprentissage et sert alors à améliorer la conduite,
- enfin, le comportement du robot est toujours prévisible puisque, pour une situation donnée les règles actives seront interprétables.

Ce contrôleur utilise les valeurs de distances dans les trois directions du robot (d_d, d_g et d_f) pour inférer les deux commandes du robot (α et V_r). Les distances sont évaluées par rapport aux sous-ensembles flous utilisés précédemment: Proche et Loin (figure VI.17). Les sorties sont des singletons données sur les figures VI.25 et VI.26. La politique adoptée pour cette mission est exprimée symboliquement par les règles présentées au tableau VI.2.

Les résultats obtenus sont présentées sur les figures VI.27 (a-b). Le contrôleur flou utilisé donne des résultats acceptables pour accomplir cette mission. Le robot peut suivre les murs efficacement. Mais dans les cas ou l'obstacle contient des pointes (coins), le comportement est mauvais et le robot est incapable d'éviter la collision (figure VI.28 (a-b)).

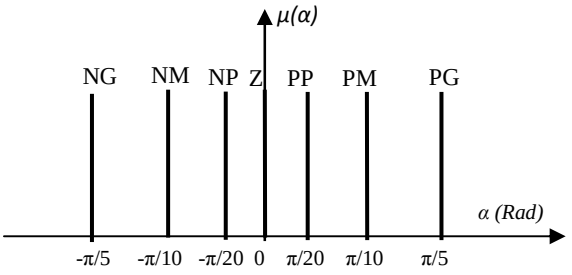


Figure VI.25 Ensembles flous de braquage

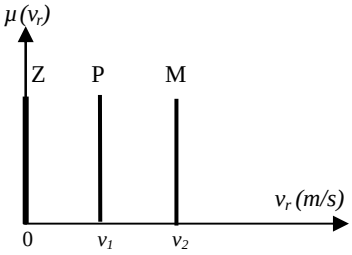
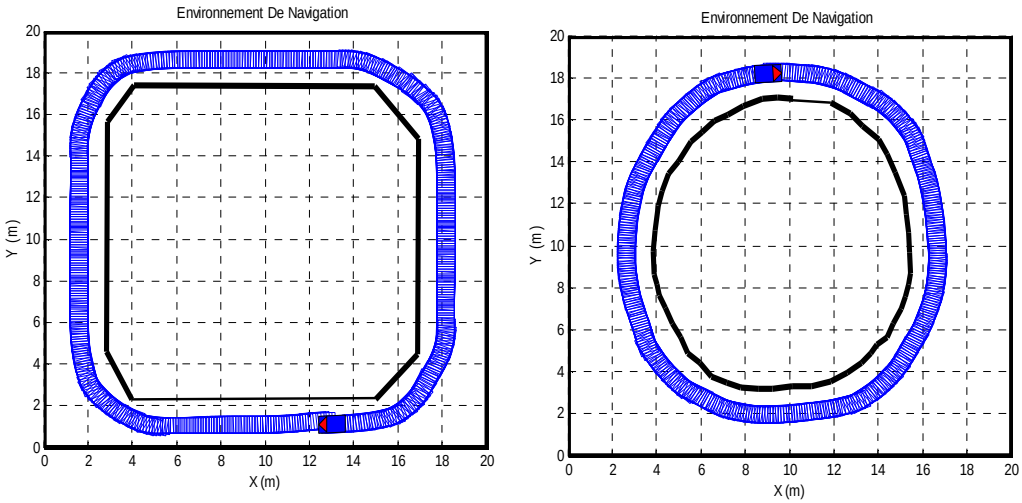


Figure VI.26 Ensembles flous de vitesse

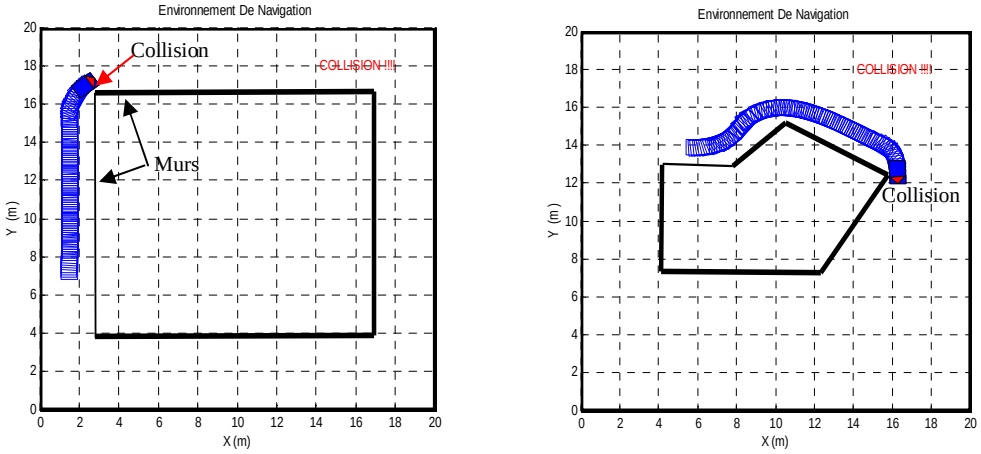
Angle de Braquage / V_r			*Distance d_g*			
			P		**L**	
			Distance d_f			
			P	**L**	**P**	**L**
Distance d_d	**L**	α	*NG*	*NP*	*PG*	*NM*
		V_r	*Z*	*M*	*Z*	*M*
	P	α	*PG*	*Z*	*PG*	*PP*
		V_r	*Z*	*M*	*P*	*P*

Tableau VI.2 Règles d'inférences du suivi de mur



(a) (b)

Figure VI.27 Suivi de mur par le contrôleur flou



(a) (b)

Figure VI.28 Collisions avec les murs

VI.5.3.2 *Navigateur flou optimisé par renforcement (QLF)*

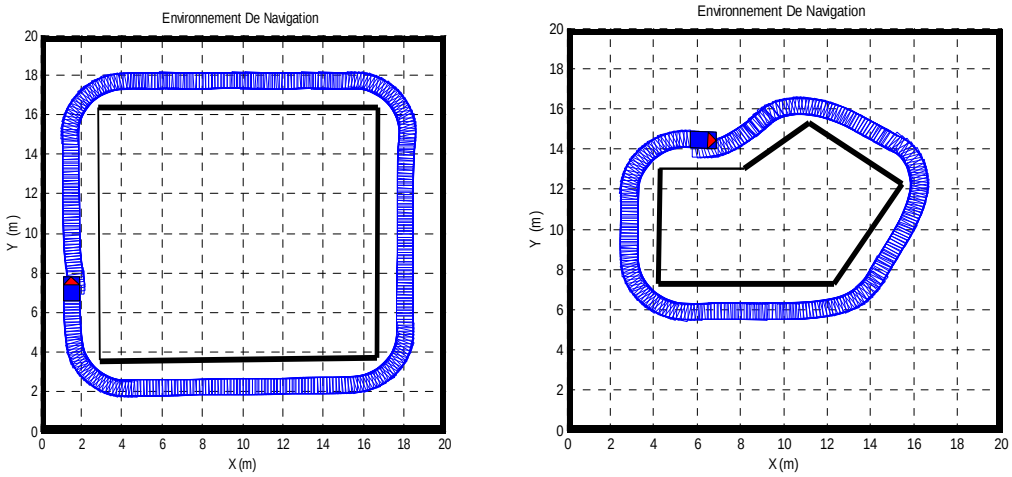
Le contrôleur précédent est optimisé en ligne par le l'algorithme Q-learning. Le signal de renforcement donné au robot mobile est définit par :

$$r = \begin{cases} -2, & \text{Si le robot percute un obstacle,} \\ -1, & \text{Si } d_i < d_m,\ i = 1...3, \\ 0, & \text{ailleurs.} \end{cases} \qquad \text{(VI.21)}$$

Ce signal va servir à déterminer la meilleure interprétation numérique des termes linguistiques utilisés. L'algorithme de Q-learning flou en extraction de connaissances est utilisé, en proposant trois interprétations pour chaque label de sortie (changement de direction). Par exemple : $\alpha = PG$ peut être interprété par $\alpha = 45°$, $\alpha = 55°$ ou $\alpha = 35°$. Les valeurs qualitatives de renforcements données au robot pour l'apprentissage de la vitesse sont données au équation VI.22.

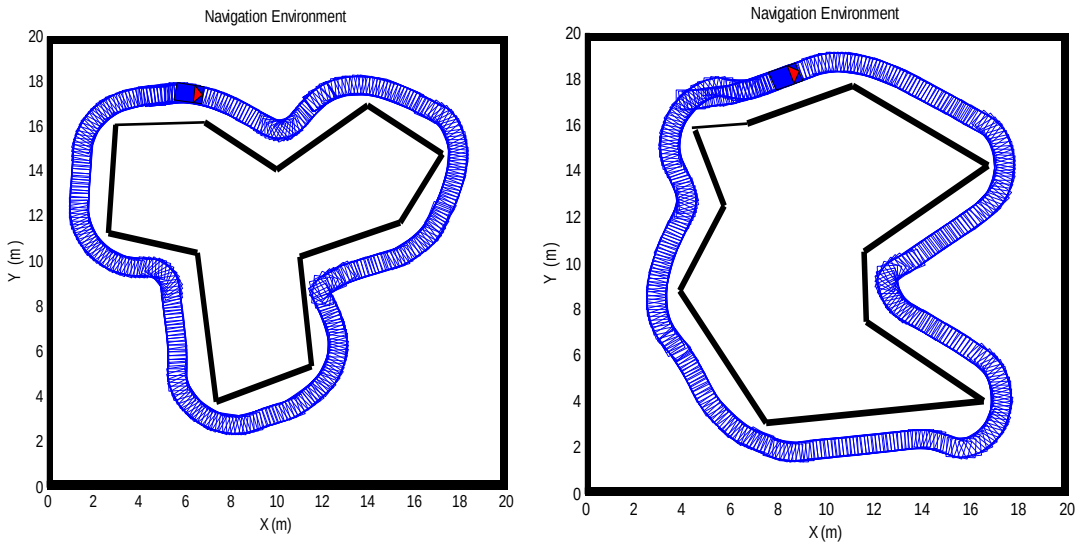
$$r = \begin{cases} -2, & \text{Si le robot percute un obstacle,} \\ -1, & \text{Si } d_i < d_s \text{ et } V_r \text{ augmente, avec } i = 1...3, \\ 0, & \text{ailleurs.} \end{cases} \qquad \text{(VI.22)}$$

Après une phase d'apprentissage, les résultats de l'optimisation sont présentés dans des environnements inconnus. On observe une amélioration du comportement du robot mobile (figure VI.29 (a-b)). Les figures VI.30 (a-b) montrent des exemples de ce résultat d'optimisation. Le robot est capable d'évoluer dans son environnement sans collision avec les obstacles en utilisant la même base de règle initiale. Le comportement du robot est satisfaisante dans tous les cas présentés.



(a) (b)

Figure VI.29 Comportements précédents améliorés



(a) (b)

Figure VI.30 Suivi des murs de différentes formes

VI.6 Conclusion

Dans ce chapitre, nous avons présenté l'application des algorithmes d'apprentissage par renforcement pour la navigation autonome d'un robot mobile, avec les possibilités offertes par ces algorithmes pour ajuster les paramètres des contrôleurs flous. L'algorithme Q-learning flou utilisé représente une extension de l'algorithme Q-learning aux espaces d'états et d'actions continus en utilisant les systèmes d'inférence flou, et d'autre part c'est une méthode d'optimisation des règles floues en maximisant une fonction de performance. Le formalisme flou est un choix possible pour le passage de l'espace discret vers l'espace continu, et permet d'accélérer la phase d'apprentissage. Cet algorithme est très approprié dans les applications des contrôleurs en ligne. De plus, l'usage des connaissances disponibles et l'introduction de connaissances réduit considérablement la période d'apprentissage qui est un inconvénient de l'apprentissage par renforcement. La représentation de la fonction qualité globale Q par un SIF est équivalente à la détermination de la fonction qualité élémentaire pour chaque vecteur modal (c'est à dire pour chaque règle), puis par interpolation pour chaque vecteur d'entrée. La tâche d'apprentissage consiste à améliorer en ligne la base de règles en optimisant la partie conclusion. L'objectif est de maximiser la moyenne des renforcements reçus par le robot mobile pour chaque mission étudiée.

Nous avons appliqué les algorithmes Q-learning et Q-learning flou pour les différents comportements du robot mobile (convergence vers un but, évitement d'obstacles, suivi de mur..). Une comparaison avec les contrôleurs flous a été abordée pour vérifier l'efficacité des comportements obtenus. D'après les résultats présentés dans ce chapitre, on a montré que l'apprentissage par renforcement et plus particulièrement le Q-learning est une solution possible pour l'optimisation des contrôleurs flous destinés à la tâche de navigation d'un robot mobile. Le système dit renforcement-flou qui représente un contrôleur flou optimisé par apprentissage par renforcement montre son efficacité de commande avec une capacité d'apprentissage en ligne.

Conclusion Générale

Dans ce travail, le problème de la navigation autonome d'un robot mobile a été abordé en utilisant les techniques neuro-floues. La tâche de base que doit réaliser chaque robot mobile avec un minimum d'erreur est d'atteindre la configuration d'arrivée à partir d'une configuration de départ sans collision avec les obstacles et sans intervention humaine. On a utilisé des contrôleurs flous, neuro-flous et renforcement-flous (contrôleurs flous entrainés par apprentissage par renforcement).

Le travail présenté dans ce mémoire est divisé en deux parties:

Dans une première partie, nous avons présenté un aperçu sur le domaine de la robotique mobile et les concepts de base des approches utilisées à savoir; la commande floue, les réseaux de neurones artificiels et l'apprentissage par renforcement.

Au début, des définitions et des concepts généraux sur l'autonomie du robot mobile ont été présentés. Les types des robots mobiles, les capteurs utilisés et les architectures de contrôles ont été exposés.

Par la suite, nous avons exposés la théorie de la commande par logique floue et l'architecture de base d'un contrôleur flou. Le fonctionnement d'un contrôleur flou dépend d'un nombre important de paramètres qu'il faut déterminer afin d'optimiser ses performances. Les contrôleurs flous présentent la possibilité d'incorporer des connaissances expertes dans leurs structures. L'intérêt majeur de la logique floue en commande réside dans sa capacité à traduire une stratégie de contrôle d'un opérateur qualifié en un ensemble de règles linguistiques facilement interprétables.

Dans le troisième chapitre, on a présenté les principes des réseaux de neurones et l'apprentissage par renforcement, ces deux méthodes permettent d'explorer de façon très efficace l'espace des solutions possibles pour la commande et les problèmes d'optimisation. En faisant l'état de l'art de ces méthodes, on a constaté que plusieurs variantes de ces algorithmes on été proposés et appliqués pour la commande des robots mobiles.

Dans la deuxième partie, les contributions de ce travail sont présentées:

Dans un premier lieu, nous nous sommes intéressés par l'utilisation des contrôleurs flous réactifs pour la navigation autonome d'un robot mobile. Nous avons utilisé des structures basé comportements avec des modèles de type Takagi-Sugeno d'ordre zéro. La commande à base de la logique floue a démontré son efficacité pour les différentes missions du robot mobile (convergence vers un but, évitements d'obstacles, suivi de murs…). L'inconvénient majeur de cette technique est la nécessité d'avoir suffisamment des connaissances à priori pour déduire le nombre des règles linguistiques. En l'absence d'expertise, on utilise généralement les techniques d'apprentissage pour la construction de la base de règles.

En second lieu, notre travail s'est orienté vers les méthodes de réglage des paramètres des contrôleurs flous. L'utilisation des techniques d'apprentissage est un choix possible à ce problème. Nous nous sommes intéressés en particulier aux réseaux de neurones et l'apprentissage par renforcement. Dans le quatrième chapitre, on a utilisé des contrôleurs flou, neuronal et hybride neuro-flou pour accomplir une fonctionnalité particulière du robot mobile, qui est la poursuite de trajectoire de référence et de cible mobile. Les comportements obtenus sont très satisfaisants à partir des structures proposées et des résultats présentés pour cette tâche. Ces contrôleurs agissent d'une manière efficace, mais l'inconvénient majeur des réseaux de neurones est la nécessitée des paires état-action définissant la base de donnée pour l'apprentissage en particulier pour les autres comportements (convergence vers un but, évitement d'obstacles, suivi de murs,..).

Dans le dernier chapitre, on a utilisé l'algorithme Q-learning d'apprentissage par renforcement seul et avec une combinaison avec les systèmes flous pour la commande automatique du robot mobile. L'apprentissage par renforcement et plus particulièrement le Q-learning est un outil performant pour obtenir un comportement optimal. L'application de Q-learning flou se résume en l'implémentation d'un contrôleur flou, la base de règles initiale est améliorée en ligne par une procédure d'apprentissage utilisant un signal de renforcement. C'est une solution prometteuse pour le problème de navigation d'un robot mobile où cet algorithme permet l'optimisation des paramètres d'un SIF et représente une extension de Q-learning au cas continus en se basant sur un signal de renforcement. Ce signal de retour permet au navigateur d'ajuster sa stratégie pour améliorer ses performances. L'utilisation de cette technique introduit plusieurs avantages parmi lesquelles :

- La possibilité de représenter la valeur qualité, grâce à la propriété d'approximation universelle des SIFs,
- La possibilité de traitement des espaces d'états et d'actions continus,
- L'intégration des connaissances à priori pour que le comportement initiale soit acceptable et l'apprentissage plus rapide,
- L'interprétation est possible après la phase d'apprentissage,
- Efficacité de commande dotée par une capacité d'apprentissage.

Des exemples de simulation sont fournis pour montrer les performances des systèmes de commande proposés pour les différents comportements du robot mobile. Une comparaison entre les navigateurs renforcement, flous et hybrides renforcement-flous a été présentée. Les résultats obtenus montrent que les structures étudiées avec des meilleures performances pour la navigation autonome d'un robot mobile.

Enfin, nous pensons que le travail présenté dans cette thèse ouvre des nouvelles perspectives selon les principales directions suivantes:

- La mise en œuvre pratique des méthodes étudiées sur un robot mobile réel,
- L'application des architectures présentées pour des environnements avec des obstacles dynamiques,
- La comparaison des méthodes développées avec d'autres algorithmes stochastiques tels que les algorithmes génétiques,
- L'optimisation des contrôleurs flous par les algorithmes de colonies de fourmis.

[ABD04] F. Abdessemed, K. Benmahammed and E. Monacelli, "A Fuzzy-based Reactive Controller for Non-holonomic Mobile Robot", *Robotics and Autonomous Systems*, vol. 47, pp. 31-46, 2004.

[AND88] J. A. Anderson, E. Rosenfeld, "*Neurocomputing: Foundations of Research*", MIT Press, Cambridge, 1988.

[AND95] J. A. Anderson, *"An Introduction to Neural Networks*", Bradford - MIT Press, 1995.

[ARK89] R. C. Arkin, "Motor Schema-based Mobile Robot Navigation", *International Journal of Robotic Research*, vol. 8, pp. 92-112, 1989.

[AYA07] I. Ayari and A. Chatti, "Reactive Control using Behavior Modeling of a Mobile Robot", *International Journal of Computers, Communications & Control*, vol. 2, no. 3, pp. 217-228, 2007.

[AZO06] O. Azouaoui, "*Planification et Contrôle des Comportements en Groupe des Systèmes Robotiques Autonomes*", Thèse de doctorat d'état en Robotique, ENP, 2006.

[BAY07] B. Bayle, "*Robotique Mobile*", Ecole Nationale Supérieure de Physique de Strasbourg, Université Louis Pasteur, 2007.

[BEL57] R. E. Bellman, *"Dynamic Programming"*, Princeton University Press, Princeton, New Jersey, USA, 1957.

[BEN07] A. Benmakhlouf, D. Djarah and A. Louchene, "A Novel Approach for intelligent Control Systems: Application to a path following By a Mobile Robot", *International Conference on Electrical Engineering Design and Technology (ICEEDT'07), Hammamet*, Tunisia, 2007.

[BEO95] H. R. Beom and H. S. Cho, "A Sensor-based Navigation for a Mobile Robot using Fuzzy Logic and Reinforcement Learning", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no.3, pp. 464-477, 1995.

[BER92] H. R. Berenji and P. Khedkar, "Learning and Tuning Fuzzy Logic Controllers Through Reinforcements", *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 724-740, 1992.

[BOR91] J. Borenstein and Y. Koren, "The Vector Field Histogram Fast Obstacle Avoidance for Mobile Robots", *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp.278-288, June 1991.

[BOR96] J. Borenstein, H. R. Everett, and L. Feng, "*Where am I, Sensors and Methods for Mobile Robot Positioning*", University of Michigan, 1996.

[BOS08] P. Boskoski, M. Stankovski, "Neuro-Fuzzy Controllers and Application to Autonomous Robots", *Mechanics, Automatic Control and Robotics*, vol. 7, no.1, pp.123-132, 2008.

[BOT05] M. Botros, "*Application of Fuzzy Logic in Mobile Robots Control*", Springer-Verlag Berlin Heidelberg, StudFuzz, pp.163-193, 2005.

[BOU93] M. Boumehraz, "Identification et Contrôle avec Réseaux de Neurones", Thèse de Magister, Département d'Electronique, Université de Sétif, 1993.

[BOU01] M. Boumehraz et al, "Fuzzy Inference Systems Optimization by Reinforcement Learning", *Courrier du Savoir Scientifique et Technique*, Université de Biskra, vol. 1, pp. 09-15, 2001.

[BOU09] H. Boubertakh, "*Contribution à l'Optimisation par Algorithmes Evolutionnaires des Contrôleurs Flous*", Thèse de Doctorat en Automatique, ENP, 2009.

[BRO86] R. Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation,* vol. 2, no.1, pp.14-23, 1986.

[BUH94] H. Bühler, *"Réglage par Logique Floue"*, Première édition, Presses Polytechniques et Universitaires, Romandes, 1994.

[CAN01] Y. Cang and D. Wang, "A Novel Behavior Fusion Method for the Navigation of Mobile Robots", *IEEE International Conference on Systems, Man, And Cybernetics*, pp.3526-3531, *Nashville*, 2001.

[CAN03] Y. Cang, N. H. C. Yung, D. Wang, "A Fuzzy Controller with Supervised Leaning Assisted Reinforcement Learning Algorithm for Obstacle Avoidance", *IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 33, no.1,pp.1-11, 2003.

[CHE09] **L. Cherroun**, "Application de l'Apprentissage par Renforcement pour la Navigation d'un Robot Mobile", Mémoire de Magister, Université de Biskra, 2009.

[CHE11] **L. Cherroun**, R. Mechgoug and M. Boumehraz, "Path Following Behavior for an Autonomous Mobile Robot using Fuzzy Logic and Neural Networks", *Revue Courrier du Savoir Scientifique et Technique,* Université de Biskra, vol. 12, pp.63-70, 2011.

[CHE12a] **L. Cherroun** and M. Boumehraz, "Intelligent Systems based on Reinforcement Learning and Fuzzy Logic Approaches, Application to Mobile Robotic", *The IEEE International Conference on Information Technology and e-Services* (*ICITeS'2012*), Tunisia, pp. 431-436, 2012.

[CHE12b] **L. Cherroun** and M. Boumehraz, "Designing of Goal Seeking and Obstacle Avoidance Behaviors for a Mobile Robot Using Fuzzy Techniques", *Journal of Automation and Systems Engineering (JASE)*, vol. 6, no. 4, pp.164-171, 2012.

[CHE13a] **L. Cherroun** and M. Boumehraz, "Path Following Behavior for an Autonomous Mobile Robot using Neuro-Fuzzy Controller", *Accepted in International Journal of Systems Assurance Engineering and Management, (IJSA)*, Springer.

[CHE13b] **L. Cherroun** and M. Boumehraz, "Fuzzy Logic and Reinforcement Learning based Approaches for Mobile Robot Navigation in Unknown Environment", *The Mediterranean Journal of Measurement and Control (MEDJMC)*, vol. 9, no. 3, pp. 109-117, 2013.

[CHE13c] **L. Cherroun** and M. Boumehraz, "Fuzzy Behavior Based Navigation Approach for Mobile Robot in Unknown Environment*", Journal of Electrical Engineering (JEE),* vol. 13, no. 4, pp. 284-291, 2013.

[CUE05] F. Cuesta, A. Ollero, "*Intelligent Mobile Robot Navigation*", Springer-Verlag, Berlin Heidelberg, 2005.

[DON03] G. Dongbing, H. Huosheng, S. Liber, "Learning Fuzzy Logic Controller For Reactive Robot Behaviours", *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*,pp. 1-6, Japan, 2003.

[FAT06] A. Fatmi, A. Al Yahmadi, L. Khriji, and N. Masmoudi, "A Fuzzy Logic Based Navigation of a Mobile Robot", *World Academy of Science, Engineering and Technology*, vol. 22, pp. 169-174, 2006.

[FER95] J. Ferber, "*Les Systèmes Multi Agents: Vers une Intelligence Collective*", InterEdition, Université Pierre et Marie Curie Paris 6, 1995.

[FIE98] R. Fierro and F. L. Lewis, "Control of a Nonholonomic Mobile Robot Using Neural Networks", *IEEE Transactions On Neural Networks*, vol. 9, no. 4, pp. 589-600, 1998.

[FIL04] D. Filliat, "*Robotique Mobile*", Cours à l'école Nationale Supérieur des Techniques Avancées ENSTA, Octobre 2004.

[FUL95] R. Fuller, "*Neural Fuzzy Systems*", Abo Academy University, 1995.

[GAU99] E. Gauthier, "*Utilisation des Réseaux de Neurones Artificiel pour la Commande d'un Véhicule Autonome*", Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1999.

[GLO00] P. Y. Glorennec, "Reinforcement Learning: an Overview", *ESIT 2000, Aachen, Germany*, September 2000.

[GLO96] P. Y. Glorennec, L. Jauffe, "A Reinforcement Learning Method for an Autonomous Robot", *Proceedings of the First Online Workshop on Soft Computing*, 1996.

[GLO97] P. Y. Glorennec and L. Jouffe, "Fuzzy Q-learning", *6th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'97)*, pp. 659-662, 1997.

[GLO99] P. Y. Glorennec, "*Algorithmes d'Apprentissage pour Systèmes d'inférence Floue*", Edition Hermès, 1999.

[GOD97] J. Godjevac, "*Neuro-Fuzzy Controllers, Design and Application*", Collection Meta, Presses Polytechniques et Universitaires Romandes, Lausanne, 1997.

[GOD99] J. Godjevac, "*Idées Nettes sur la Logique Floue*", Collection Informatique, Presses Polytechniques et Universitaires Romandes, Lausanne, 1999.

[GUE88] A. Guez, J. L. Elibert and M. Kam, "Neural Network Architecture for Control", *IEEE Control Systems Magazine*, vol. 8, no. 3, pp. 22-25, April 1988.

[GU04] D. Gu, H. Hu, "Accuracy Based Fuzzy Q-Learning for Robot Behaviors", *IEEE International Conference on Fuzzy Systems (IEEE-FUZZY 2004)*, Budapest, pp.25-29, July 2004.

[HAC08] O. Hachour, "Path planning of Autonomous Mobile Robot", *International Journal of Systems Applications, Engineering & Development*, vol. 2, no.4, pp.178-190, 2008.

[HAG00] S. T. Hagen and B. Krose, "Q-Learning for Systems with Continuous State and Action Spaces," *in Proc. BENELEARN, 10th Belgian-Dutch Conference in Machine Learning*, 2000.

[HER95] F. Herrera, M. Lozano and J. L. Verdegay, "Tuning Fuzzy Controllers by Genetic Algorithms", *International Journal of Approximate Reasoning*, vol. 12, pp. 299-315, 1995.

[HOF00] F. Hoffman, "Soft Computing Techniques for the Design of Mobile Robot Behaviors", *Information Sciences*, vol. 122, pp. 241-258, 2000.

[HON10] F. Hong, S. Sam Ge, C. Pang, T. Lee, "Robust Adaptive Neuro-Fuzzy Control of Uncertain Nonholonomic Systems", *Journal of Control Theory Applications*, vol. 8, no. 2, pp.125-138, 2010.

[JAM92] A. F. James, David M. Skapura, "*Neural Networks Algorithms, Applications and Programming Techniques*", Addison Wesly, 1992.

[JAN04] D. Janglová, "Neural Networks in Mobile Robot Motion", *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp.15-22, 2004.

[JAN07] J. Jantzen, *"Foundations of Fuzzy Control"*, West Sussex (England), John Wiley & Sons Ltd, 2007.

[JAN93] J.-S.R Jang, "ANFIS: Adaptive-Network-based Fuzzy Inference System", *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 23, pp.665-685, 1993.

[JOU98] L. Jouffe, "Fuzzy Inference System Learning by Reinforcement Methods", *IEEE Transactions on Systems, Man, and Cybernetics*, vol.28, no.3, pp. 338-355, 1998.

[KAE96] L. P. Kaelbling, M. L. Littman, and A.W Moore, "Reinforcement Learning: A Survey", *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[KHA86] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90-98, 1986.

[KHA09] K. Khaldoun, S. N. Al-Din Munaf, "A Neuro-Fuzzy Reasoning System for Mobile Robot Navigation", *Jordan Journal of Mechanical and Industrial Engineering*, vol. 3, no. 1, pp.77-88, 2009.

[KHR11] L. Khriji and Al Yahmedi, "Mobile Robot Navigation Based on Q-learning Technique", *International journal of advanced Robotic System*, vol. 8, no. 1, pp 45-51, 2011.

[KIM98] C. Ng Kim, M. M. Trivedi, "A Neuro-Fuzzy Controller for Mobile Robot Navigation and Multirobot Convoying", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 28. no.6, pp. 829-840, 1998.

[KIS85] J.B. Kiszka, M.M. Gupta, and P.N. Nikiforuk, "Energetic Stability of Fuzzy Dynamic Systems", *IEEE Transaction on Systems, Man Cybern.*, vol. 15, no. 5, pp.783-792, 1985.

[KOS93] B. Kosko, "Fuzzy Systems as Universal Approximators", *IEEE Transactions on Computers*, 1993.

[LAT91] J. C. Latombe, "*Robot Motion Planning*", Kluwer Academic Publishers,Norwell, 1991.

[LAU00] E. Laurin, "*Système Intelligent d'Assistance à la Perception dans la Conduite de Véhicul*e", Thèse de Doctorat, Université de Sherbrooke, 2000.

[LAU01] J.P. Laumond, "*La Robotique Mobile*", Editions Hermès, 2001.

[LEF06] O. Lefebvre, "*Navigation Autonome sans Collision pour Robots Mobiles non holonomes*", Thèse de Doctorat de l'Institut National Polytechnique de Toulouse, 2006.

[LIN91] L. J. Lin, "Self Improvement based on Reinforcement Learning, Planning and Teaching", *Proc. of 8th Workshop on Machine Learning ML'91*, 1991.

[MAA00] H. Maaref and C. Barret, "Sensor Based Navigation of an Autonomous Mobile Robot in an Indoor Environment", *Control Engineering Practice*, vol. 8, pp. 757-768, 2000.

[MAM74] E.H. Mamdani, "Application of Fuzzy Algorithms for Control of Simple Dynamic Plant", *Proc. of IEEE*, vol. 121, no. 12, pp. 1585-1588, 1974.

[MAM75] E. H. Mamdani and S. Assilian, "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller", *International Journal of Man-Machine Studie*s, vol. 7, no. 1, pp. 1-13, 1975.

[MAT97] M. J. Mataric, "Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior", *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, no.2, pp. 323-336, 1997.

[MEN95] J. M. Mendel, "Tutorial: Fuzzy Logic Systems for Engineering", *Proceeding of the IEEE*, vol. 83, no. 3, pp. 345-377, March 1995.

[MYK05] K. Mykhaylo et al, "Using fuzzy Inference System as a Function Approximator of a State Action Table", *Proceedings of Advanced Aspects of Theoretical Electrical Engineering*, Sozopol, Bulgaria, October 2005.

[NUR06] B. H. Nurmal et al, "Time-Optimal Collision-Free Navigation of a Car like Mobile Robot using Neuro-Fuzzy Approaches", *Fuzzy Sets and Systems*, vol. 157, pp. 2171-2204, 2006.

[PAS98] K. M. Passino and S. Yurkovich, "*Fuzzy Control*", Addison Wesley Longman, 1998.

[PAR04] M. Parizeau, "*Réseaux de neurones*", Edition université de Laval, 2004.

[PRE07] Ph. Preux, "Apprentissage par renforcement, Notes de cours", *GRAPPA, cours de l'université de Lille*, octobre 2007.

[PER03] L. Personnaz, "*Réseaux de Neurones Formels pour la Modélisation, la Commande et la Classification*", CNRS Edition, Paris, 2003.

[RAO95] N. S. V. Rao, "Robot Navigation in Unknown Generalized Polygonal Terrains Using Vision Sensors", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 6, pp. 947-962, 1995.

[REI94] P. Reignier, "*Pilotage Réactif d'un Robot Mobile, Etude de Lien entre la Perception et l'Action*", Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1994.

[RIS05] P. Rishikesh1, L Janardhanan and Er M Joo, "Goal Seeking of Mobile Robots using Dynamic Fuzzy Q-learning", *Journal of the Institution of Engineers*, Singapore, vol. 45, no. 5, 2005.

[ROS89] J. Rosenblatt and D. W. Payton, "A Fine-Grained Alternative to the Subsumption Architecture for Mobile Robot Control", *Proceedings of the IEEE/INNS International Joint Conference on Neural Networks, Washington DC*, vol. 2, pp. 317-324, 1989.

[ROS95] J. Rosenblatt, "DAMN: A Distributed Architecture for Mobile Navigation", *Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agent*s, Menlo Park, CA, 1995.

[RUS03] P. Rusu et al, "Behavior-Based Neuro-Fuzzy Controller for Mobile Robot Navigation", *IEEE Transactions on Instrumentation and Measurement*, vol. 52, no. 4, pp.1335-1340, 2003.

[SAF97] A. Saffiotti, "The Uses of Fuzzy Logic for Autonomous Robot Navigation", *Soft Computing*, vol. 1, no. 4, pp.180-197, 1997.

[SER02] H. Seraji & A. Howard, "Behavior-based Robot Navigation on Challenging Terrain: A Fuzzy Logic Approach", *IEEE Transaction on Robotic and Automation*, vol. 18, no. 3, pp. 308-321, 2002.

[SIM04] Simon X. Yang et al, "An Embedded Fuzzy Controller for a Behavior-Based Mobile Robot With Guaranteed Performance", *IEEE Transaction on Fuzzy Systems*, vol. 12, no. 4, pp. 436-446, 2004.

[SHU06] S. G. Shuzhi, F. L. Lewis, "*Autonomous Mobile Robots, Sensing, Control, Decision, Making and Applications*", Taylor and Francis Group, 2006.

[SUG84] M. Sugeno and K. Murakami, "Fuzzy Parking Control of Model Car", *The 23 rd IEEE Conference on Decision and Control*, 1984.

[SUT88] R. S. Sutton, "Learning to Predict by the Methods of Temporal Differences", *Machine Learning*, vol. 3, pp.9-44, 1988.

[SUT91] R. S. Sutton and A. G. Barto, R.J. Williams, "Reinforcement Learning is Direct Adaptive Optimal Control", *Proceeding of ACC*, Boston, 1991.

[SUT96] R. S. Sutton, "Reinforcement Learning with Replacing Eligibility Traces", *Machine Learning*, vol. 22, pp. 123-158, 1996.

[SUT98] R. S. Sutton and A. G. Barto, "*Reinforcement Learning: An Introduction*", MIT Press, Cambridge, MA, 1998.

[TAK85] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and its Applications to Modeling and Control", *IEEE Transactions on Systems, Man, Cyber*, vol. 15, pp. 116-132, 1985.

[THR95] S. Thrun, "An Approach to Learning Mobile Robot Navigation", *Robotics and Autonomous Systems*, vol. 15, pp.301-319, 1995.

[TOU92] C. Touzet, "*Les Réseaux de Neurones Artificiels, Introduction au Connexionnisme*", Paris: Masson, 1992.

[TOU97] C. Touzet, "Neural Reinforcement Learning for Behaviour Synthesis", *Robotics and Autonomous Systems*, vol. 22, no 3, pp. 251-281, 1997.

[TOU98] C. Touzet, "*L'apprentissage par Renforcement*", Paris: Masson, 1999.

[TSA97] S. G. Tzafestas, K. D. Blekas, "Hybrid Soft Computing Systems: A Critical Survey with Engineering Applications", *National Technical University of Athens*, Greece, 1997.

[WAT89] C. J. C. H. Watkins, "Learning from Delayed Rewards", PhD Thesis, University of Cambridge, England, 1989.

[WAT92] C. J. C. H. Watkins and P. Dayan, "Technical Note, Q-learning", *Machine Learning*, vol. 8, pp. 279-292, 1992.

[WAN94] L.X. Wang, *"Adaptive Fuzzy Systems and Control: Design and Stability Analysis"*, Prentice-Hall, Englewood Clifs, NJ, 1994.

[WAN08] M. Wang, N. K. Liu James, "Fuzzy Logic based Real-time Robot Navigation in Unknown Environment with Dead Ends", *Robotics and Autonomous Systems*, vol. 56, pp.625-643, 2008.

[YAG94] R. R. Yager and D. P. Filev, "*Essential of Fuzzy Modeling and Control"*, John Wily & Sons Inc., 1994.

[YAN05] X. Yang et al, "A Layered Goal-Oriented Fuzzy Motion Planning Strategy for Mobile Robot Navigation*", IEEE Transaction on Systems, Man, and Cybernetics, B*, vol. 35, no.6, pp.1214-1224, 2005.

[YUN99] N. H. C. Yung and Cang Ye, "An Intelligent Mobile Vehicle Navigation Based on Fuzzy Logic and Reinforcement Learning", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 29. no.2, pp.314-321, 1999.

[ZAD65] L. A. Zadeh, "Fuzzy Sets", *Information and Control*, vol. 8, pp. 338-353, 1965.

[ZAV03] P. G. Zavlangas and S. G. Tzafestas, "Motion Control for Mobile Robot Obstacle Avoidance and Navigation: A Fuzzy Logic-Based Approach", *Systems Analysis Modeling Simulation*, vol. 43, no. 12, pp. 1625-1637, 2003.

[ZHO07] Y. Zhou, M. Joo Er and Y. Wen, "A Hybrid Approach for Automatic Generation of Fuzzy Inference Systems without Supervised Learning", *Proceedings of the 2007 American Control Conference*, USA, pp. 3371-3376, 2007

ملخص:

هذه الأطروحة تعالج مشكل الإبحار الذاتي للآلي المتحرك باستعمال التقنيات العصبية و الغامضة الهجينة. الهدف من العمل المقدم هو دراسة وإنشاء مخططات تحكم فعالة للإبحار الذاتي للآلي المتحرك في محيط غير معروف، باستعمال من جهة التقنية الوظيفية، و من جهة أخرى منهج التعليم. التقنيات المتبعة لدراسة هذا المشكل تعتمد على الأنظمة الغامضة، الشبكات العصبية الاصطناعية و التعلم بالتدعيم. استعملنا أولا الأنظمة الوظيفية الغامضة من اجل التخطيط المحلي و الفعال، و بهدف تحسين قيم الوظائف الغامضة قدمنا الأشكال العصبية و الغامضة الهجينة للإبحار الذاتي. طريقة التعليم الثانية هي تقنية مهيأة للآليات المتحركة، تسمح بإيجاد الأمر المثالي المراد تطبيقه في وضعية الآلي عن طريق التجربة و الخطأ من اجل تضخيم القيم العائدة. لإدماج خصائص تقنيتي التعلم بالخبرة و المنطق الغامض، استعملنا إستراتيجية تحكم بإمكانية التعلم. وهي عبارة عن تمديد للخوارزم Q-learning للقيم المتصلة و طريقة لتحسين الأنظمة الغامضة. الغرض من استعمال المنطق الغامض هو تقديم معارف مبدئية ليكون السلوك الأولي للمتعلم مقبولا. أثبتنا فعالية المخططات المقترحة و المدروسة بتطبيقها على مختلف المهام للإبحار الذاتي للآلي المتحرك.
كلمات مفتاحية: آلي متحرك، إبحار، نظام عصبي غامض، التعلم بالتقوية، تجنب الحواجز.

Résumé:

Cette thèse traite le problème de navigation autonome d'un robot mobile par les techniques hybrides neuro-floues. L'objectif de travail présenté est d'étudier et développer des architectures de commande efficaces pour une navigation réactive d'un robot mobile autonome dans un environnement inconnu, en utilisant d'une part l'approche comportementale et d'autre part les méthodes de l'apprentissage. Les techniques employées pour aborder ce problème sont basées sur les systèmes d'inférence flous, les réseaux de neurones artificiels et l'apprentissage par renforcement. On a utilisé premièrement, les systèmes basés sur les comportements flous pour la planification locale et réactive, puis pour l'ajustement des paramètres des comportements flous, on a introduit les modèles hybrides neuro-flous pour la navigation autonome. La deuxième méthode d'apprentissage est particulièrement adaptée à la robotique, qui permet de trouver, par un processus d'essais et d'erreurs, l'action optimale à effectuer pour chacune des situations que le robot va percevoir afin de maximiser ses récompenses. Pour combiner les avantages de la logique floue et l'apprentissage par renforcement, une stratégie de commande avec une capacité d'apprentissage est utilisée, c'est une extension de Q-learning aux cas continus et une méthode d'optimisation des systèmes flous. L'avantage des systèmes flous est l'introduction des connaissances disponibles à priori pour que le comportement initial soit acceptable. L'efficacité des architectures proposées et étudiées sont démontrées par diverses applications de navigation autonome d'un robot mobile.

Mots-clés : *robot mobile, navigation, système neuro-flou, apprentissage par renforcement, évitement d'obstacles.*

Abstract:

This thesis deals with the autonomous navigation problem of a mobile robot using hybrid neuro-fuzzy techniques. The objective of the presented work is to study and develop effective architectures for a reactive navigation of an autonomous mobile robot in unknown environment, by using on the one hand the behavior based approach, and on the other hand the learning paradigm. The techniques employed to tackle this problem are based on the fuzzy inference systems, artificial neural networks and the reinforcement learning. Firstly, we used fuzzy behavior based navigation approach for the reactive and local planning, then for tuning the fuzzy behaviors parameters; we introduced the hybrid neuro-fuzzy models for autonomous navigation. The second type of learning method is particularly adapted to mobile robotic, which makes it possible to find by a process of tests and errors, the executed optimal action for each situation which the robot will perceive in order to maximize its rewards. To combine the advantages of fuzzy logic and reinforcement learning, a control strategy with a learning capacity is used, it is an extension of Q-learning to the continuous spaces and considered as and optimization method of fuzzy systems. The advantage of the fuzzy systems is the introduction of a priori knowledge in order to make the first behavior is acceptable. The effectiveness of the proposed and the studied architectures are demonstrated by various applications of the autonomous mobile robot navigation.

Key words: *mobile robot, navigation, neuro-fuzzy system, reinforcement learning, obstacle avoidance*