REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTRE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABEDERRAHMAN MIRA – BEJAIA

FACULTE DE LA TECHNOLOGIE

Département d'Electronique

MEMOIRE

Présenté pour l'obtention du diplôme de Magister

Filière : Automatique et Traitement du signal

Option : Système

Par : Mr. SOUALMI Boussaad

Ingénieur d'Etat en Electronique

Option : contrôle

Université Abderrahmane Mira – Béjaia

Thème :

PLANIFICATION DE TRAJECTOIRES DANS UN ENVIRONNEMENT PARTIELLEMENT CONNU A BASE DES TECHNIQUES D'INTELLIGENCE ARTIFICIELLE

Soutenu publiquement le : 02 /06 /2009, devant le jury :

Président :	Mr Y. ZEBOUDJ	Professeur, Université A.MIRA –Béjaia
Examinateurs:	Mr K. MOKRANI	Maître de conférence, Université A.MIRA –Béjaia
	Mr S. BERRAH	Maître de conférence, Université A.MIRA –Béjaia
Rapporteur :	Mr B. MENDIL	Professeur, Université A.MIRA –Béjaia

Remerciements

Ce présent travail est réalisé sous la direction du Pr. B. MENDIL, qu'il trouve ici ma profonde reconnaissance et mes sincères remerciements.

Je tiens à remercier le président du jury pour avoir accepté de présider le jury.

Je remercie aussi les deux examinateurs pour l'honneur qu'ils nous font en participant à l'évaluation de ce travail.

Sommaire

Introduction générale

Introduction Générale

Les premiers travaux en robotique mobile en milieu industriel ne conféraient aux robots mobiles qu'une autonomie de déplacement très limitée et nécessitait la réalisation d'importants et coûteux travaux d'infrastructures. C'était le cas en particulier des chariots filoguidés, limitant le déplacement du robot à des voies lui étant réservées.

Depuis quelques années, un intérêt croissant est porté au sein de la communauté robotique au développement de systèmes intelligents et autonomes dans le cadre de la robotique mobile. Un tel intérêt peut être aperçu comme une conséquence à l'apparition d'applications potentielles (nettoyage, aide à la mobilité des personnes handicapées...) et au désire de mettre les robots sur des taches nouvelles telles que l'exploration de sites inaccessibles à l'homme (e.g. planètes, fonds d'océans) ou opérer dans des milieux qui lui sont hostile (milieux radioactifs).

Dans de tels contextes, munir les systèmes robotiques d'une capacité de décision et, plus particulièrement, de planifier leurs trajectoires ou naviguer d'une manière autonome, reste l'un des éléments clé dans la mise en œuvre de leur autonomie. En effet, le robot autonome doit être en mesure de prendre des décisions, quant aux mouvements à réaliser, en fonction des informations dont il dispose sur l'environnement ou lui parvenant de ces capteurs.

A cet effet, le souci major est d'élaborer des techniques de navigation efficaces, telle que la sécurité soit prioritaire par rapport à l'optimalité. La stratégie de navigation peut intégrer les informations, a priori, dont on dispose sur l'environnement, sous forme d'un modèle de ce dernier. Cependant, lorsque l'environnement devient plus complexe (partiellement connu, dynamique,…), il apparaît indispensable que le robot soit doté de capacités décisionnelles aptes à le faire réagir aux aléas qui peuvent contrarier ses mouvements.

Les architectures de contrôle en robotique mobile sont généralement classées en deux catégories principales : hiérarchiques et réactives. Les architectures hiérarchiques sont des architectures graphiques. Celles-ci génèrent une trajectoire à partir d'une carte de l'environnement de travail supposé parfaitement connu. Mais une connaissance parfaite d'un environnement réel est chose presque impossible ; sauf le cas d'un environnement dédie.

L'objectif principal de recherche sur les stratégies de la navigation réactives est la réalisation de systèmes autonomes équipés de capteurs et actionneurs de faibles coût, afin d'exécuter des tâches complexes dans des environnements incertains ou inconnus. Beaucoup d'études se sont concentrées sur l'approche comportementale dans laquelle la réactivité aux circonstances imprévisibles est accomplie par des algorithmes simples (implantant des comportements élémentaires) traitant les informations sensorielles en temps réel moyennant des inférences de haut niveau.

C'est dans cette perspective que s'inscrit notre travail. Il concerne principalement le développement d'une navigation réactive d'un robot mobile autonome dans un environnement inconnu. L'approche est basée sur la fusion de deux comportements ; à savoir, le comportement de convergence vers une cible et le comportement d'évitement d'obstacles. A cet effet, une technique de traitement de données et de prise de décisions a, tout particulièrement, retenue notre attention. Il s'agit du raisonnement approximatif à base de la logique floue.

La logique floue est souvent adoptée pour vaincre les difficultés de modélisation des environnements incertains et/ou dynamiques ; tache difficile à exprimer par des équations mathématiques. Pour vaincre les incertitudes et rehausser la robustesse de navigation, beaucoup de recherches sont basées sur des techniques d'apprentissage automatiques qui permettent au contrôleur flou d'exploiter les données sensorielles de l'environnement exploré. On trouve, en particulier, les algorithmes issus de l'intelligence artificielle tels que : les algorithmes génétiques, l'apprentissage par renforcement, etc.

Dans un premier temps, l'intégration au robot des deux comportements cités ci-dessus, est réalisée par deux contrôleurs flous dont les caractéristiques paramétriques et structurelles sont déterminées d'une manière empirique, en utilisant nos connaissances et nos estimations qualitatives. Cette approche se caractérise essentiellement par sa relative facilitée de conception. Mais, elle présente des performances moyennes. Il apparaît donc qu'un apprentissage est souhaitable si l'on désire s'affranchir des réglages empiriques. Les algorithmes génétiques sont alors utilisés pour cet objectif. Ceci fait l'objet de la deuxième partie de notre étude.

Pour une meilleure présentation de notre travail, le mémoire est organisé en trois chapitres et structuré comme suit :

Le premier chapitre est consacré à une introduction à la robotique de manière globale et à la robotique mobile en particulier, ainsi que leurs modélisations. Une grande partie du chapitre est consacrée aux deux grandes architectures de contrôle utilisées en robotique mobile autonome, à savoir l'architecture hiérarchique (délibérative) et l'architecture réactive (réflexe). Et enfin, un aperçu de l'outil utilisé pour la simulation du robot mobile dans son environnement de navigation est présenté.

Dans le deuxième chapitre la simulation d'une architecture réactive comprenant deux comportements essentiels « convergence vers une cible » et « évitement d'obstacles » est réalisé par implantation des deux comportements précédents avec deux contrôleurs flous. La première partie est consacrée à la conception d'un contrôleur flou de convergence vers une cible. Celui-ci n'opère, bien sûr, que dans un environnement dénué d'obstacles. Dans la seconde partie, on a considéré un environnement avec obstacles et un contrôleur d'évitement d'obstacles. Le chapitre est finalisé par la fusion des actions des deux contrôleurs dans le but de réaliser des taches d'atteintes de cibles dans un environnement contenant des obstacles.

Le troisième chapitre est consacré à l'introduction des algorithmes génétiques comme outil d'optimisation. Dans un premier temps, du contrôleur de convergence vers une cible, et dans un second, rechercher le gain optimal pour la fusion des actions issues des deux contrôleurs.

Nous terminons par une conclusion générale et les perspectives envisagées.

Chapitre 1 :

Généralités sur la robotique

Chapitre 1 : Généralités sur la robotique

Dans ce chapitre, nous donnerons quelques généralités sur la robotique ; modélisation, autonomie d'un robot, architecture de commande et enfin une introduction au travail.

1. Introduction

D'importantes activités de recherche se sont développées ces dernières années dans le domaine de la *planification de trajectoire* en robotique. Plusieurs applications réussies ont montré des caractéristiques de robustesse dans des milieux autres que les laboratoires [Siegwart04], quoique, bien structurés : environnement de types domotique, hôpitaux, ateliers industriels, ...etc.

Un des axes de recherche actuel est d'étendre le domaine d'application de la *planification* à toutes sortes d'environnements non structurés, dû principalement à la nécessité de remplacer l'intervention humaine dans des milieux qui lui sont hostiles ou qui ne lui sont pas accessibles.

Plusieurs méthodes de contrôle des systèmes ont été développées en robotique. Dans ce premier chapitre, nous introduirons quelque unes.

Nous nous intéresserons, par la suite, aux différentes architectures de contrôle utilisées dans la commande des robots autonomes. L'architecture retenue dans ce mémoire est celle issue de l'intelligence artificielle, faisant appel au contrôle flou et l'apprentissage de ces derniers avec les algorithmes génétiques.

Enfin, nous donnerons un aperçu de l'environnement simulé que nous avons utilisé dans ce travail.

2. Généralités sur la robotique

Le mot "ROBOT", qui veut dire - travail forcé – en tchèque, est introduit dans le vocabulaire humain pour la première fois, dans une pièce de théâtre ; R.U.R. : Rossums Universal Robots, par l'écrivain tchèque Karel Capek en 1921. Le premier robot industriel a été installé en 1961 dans une fonderie d'aluminium aux états unis d'Amérique.

Un robot est un dispositif susceptible d'accomplir une ou plusieurs fonctions. Il se compose principalement d'un socle, dans le cas d'un manipulateur ce dernier est muni d'un ou de plusieurs bras articulés, équipés de systèmes de préhension (pinces, ventouses, etc.), d'une source d'énergie, d'actionneurs, capteurs internes et capteurs externes et d'un système de traitement d'information.

Deux sortes de robots existent :

- Robots manipulateurs : robots ancrés physiquement à leur place de travail et, généralement, mis en place pour réaliser une tâche précise ou répétitive.
- Robots mobiles : robots capables de se déplacer dans un environnement et peuvent supporter des manipulateurs suivant leur utilisation.

En industrie une grande part est réservée aux robots manipulateurs ; vu leur intérêt économique. Ces derniers sont utilisés dans les chaînes d'assemblage et permettent d'accélérer la cadence de production. Les taches réalisées sont programmées de sorte que les manipulateurs reproduisent des mouvements séquentiels et répétitifs [Siegwart04].

Un intérêt particulier est attribué ces dernières années aux robots autonomes. Contrairement aux premiers, ils sont dotés d'une certaine intelligence leurs permettant de prendre des décisions ou de choisir les actions à exécuter pour accomplir les taches qui leurs sont confiées. Dans cette perspective, les robots mobiles ont suscité un intérêt particulier chez les chercheurs, puisque ces derniers ne souffrent pas de l'handicape des manipulateurs (incapacité à se déplacer).

Dans le paragraphe qui suit, nous exposerons les différents modèles utilisés dans la commande des robots.

2.1 Modélisation des robots

Les modèles utilisés en robotique décrivent le comportement du robot (système physique) sous forme d'équations mathématiques, dans le but d'élaborer des lois de commandes appropriées pour la réalisation des taches. La modélisation consiste à retrouver les relations liant les grandeurs suivantes :

$$\Gamma(t) \leftrightarrow c(t) \leftrightarrow \theta(t) \leftrightarrow x(t) \tag{1.1}$$

Tel que :

x(t) : Vecteur des variables opérationnelles.

$\theta(t)$: Vecteur des variables articulaires.

$c(t)$: Vecteur des couples articulaires.

$\Gamma(t)$: Vecteur des variables moteur.

En faisant quelques hypothèses simplificatrices de la relation (1.1), de droite à gauche, trois modèles de robot résultent [Aoughellanet06] :

- **Modèle géométrique direct et inverse**

En ne considérant que la géométrie du robot, le modèle géométrique direct (**M.G.D)** donne les coordonnées opérationnelles «X» en fonction des coordonnées articulaires « θ ». En d'autres termes, permet de connaître la position et l'orientation de l'organe terminal (effecteur), auquel on associe un repère, par rapport à un référentiel de travail, connaissant la configuration du robot. Dans cette modélisation, la relation (1.1) se réduit à :

$$X=f(\theta) \tag{1.2}$$

Pour ce faire, on établit les relations mathématiques exprimant la position et orientation de l'effecteur dans le repère de référence, et cela, par le calcul des matrices de passage. Plusieurs méthodes sont utilisées pour le calcul des matrices de passage, et ce, en se basant sur des conventions pour chaque changement de repère. Le choix de la méthode à adopter pour le calcul dépend des caractéristiques du robot et de la tache à réaliser.

Le modèle géométrique inverse (**M.G.I)** donne les coordonnées articulaires « θ » en fonction des coordonnées opérationnelles « X », i.e. : donne la configuration du robot

permettant d'obtenir une position et une orientation donnée de l'effecteur. Le M.G.I est obtenu par inversion du (M.G.D).

$$\theta = f^{-1}(X) \tag{1.3}$$

Si un M.G.I du robot existe (robot résoluble), on peut dans ce cas, faire sa commande en position ; générer les variables articulaires donnant la position désirée. Le MG n'est pas toujours utilisable. Car, il est une approximation loin de la réalité et ne prend pas en considération ni les vitesses ni la dynamique du robot. De plus, les équations résultantes (de forme trigonométriques) sont non linéaires.

- **Modèle différentiel (cinématique) direct et inverse**

Le modèle différentiel direct (**M.D.D**) est une sorte de linéarisation du M.G. Basé sur le calcul variationnel, le M.D.D permet de décrire les variations élémentaires des coordonnées opérationnelles « ΔX » en fonction des variations élémentaires des coordonnées articulaires « Δθ » i.e. donne la vitesse de l'organe terminal dans l'espace opérationnel en fonction des vitesses articulaires, d'où :

$$\Delta X = J(\theta).\Delta\theta \tag{1.4}$$

Tel que J (θ) : matrice du Jacobéen.

Dans le but de réaliser une commande cinématique, on a besoin de connaître le modèle différentiel inverse (**M.D.I**), puisque la commande se fait dans le domaine opérationnel. Le calcul du M.D.I revient à former le vecteur des vitesses articulaires « $\Delta\theta$ » en fonction des vitesses cartésiennes « ΔX » du point terminal. D'où :

$$\Delta\theta = J^{-1}(\theta) \times \Delta X \tag{1.5}$$

M.D.I peut être obtenu par les deux méthodes suivantes :

- Utilisation du modèle géométrique inverse : cette méthode consiste à dériver le MGI.
- Utilisation M.D.D : le MDI se déduit du MDD. Ce qui revient à résoudre un système d'équations linéaires. Ceci peut être fait, soit analytiquement où par calcul numérique.

Ce modèle est une approche utilisable pour une commande postulant les hypothèses suivantes :

- La géométrie du robot est parfaite et correspond à celle décrite par les équations mathématiques.
- La commande est parfaite, c'est-à-dire, instantanée et précise.
- Il n'y a pas de phénomènes de perturbations (qui sont généralement dynamiques).

En pratique, toutes ces hypothèses ne sont pas toujours vérifiées, et pour une commande rapide et précise les contraintes dynamiques doivent être prises en compte. Ce qui revient à prendre l'équation (1.1) entière.

- **Modèle dynamique**

Le modèle dynamique (**M.D**) donne les équations du mouvement du robot en tenant compte de sa géométrie et de son inertie.

Ce modèle, dans lequel l'équation (1.1) est prise en sa totalité, s'obtient par l'application des lois fondamentales de la mécanique (Lagrange-Euler, Newton-Euler ou des versions modifiées). L'objectif reste le même, mais la structure des équations diffère selon des raisons et des motivations particulières (temps de calcul, analyse, synthèse, ...).

Comme exemple des formulations précédentes, on donne, ici, la formulation de Lagrange-Euler qui fournit un modèle explicite des équations du mouvement. Mais, le modèle résultant est très complexe et conduit à des calculs intenses. Pour son calcul, on applique l'équation de Lagrange-Euler donnée par :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} = \tau_i \qquad i = 1,..,n \tag{1.6}$$

Tel que :

n : nombre de degrés de liberté du robot.

L : fonction de Lagrange = K-P (K : énergie cinétique, P : énergie potentielle)

τ_i : force (ou couple) appliquée au système à l'articulation i.

3. Autonomie d'un robot

Depuis une vingtaine d'années, un effort particulier a été consacré pour le domaine de la recherche et de l'industrie pour construire des robots autonomes capables d'évoluer avec un minimum d'intervention humaine. Une première génération de robots a consisté en des machines capable d'évoluer dans des environnements parfaitement connus : celles-ci réalisent des missions planifiées à partir d'une modélisation parfaite de l'environnement ou se contentent de suivre une trajectoire par un mécanisme de guidage. Le point commun de ces robots est qu'ils évoluent dans un environnement qui leur est totalement dédié.

Cependant, lorsque l'environnement devient plus complexe (partiellement connu, dynamique, inconnu, …), il apparaît indispensable que le robot soit doté de *capacités décisionnelles* aptes à le faire réagir aux aléas qui peuvent contrarier ses mouvements (les obstacles). Cela peut être le cas, lorsque le robot mobile évolue sans intervention humaine, soit parce que l'environnement est inconnu pour l'être humain lui-même, donc le robot effectue la première exploration des lieus (planète, fond des océans …), soit le milieu est dynamique donc change de configuration à travers le temps. Pour cela, le robot, doit suivre le schéma correspondant au paradigme *Percevoir-Décider-Agir* [Brooks86] [Lewis06] [Chen03] [Siegwart04] [Wang 08]. La manière dont un robot gère ces différents éléments est définie par son architecture de contrôle.

- **Percevoir** : le rebot doit acquérir des informations de son état interne et de l'état de l'environnement dans lequel il évolue, et ce, via ces capteurs. Dans le cas où le robot évolue dans un environnement inconnu, ces informations sont, pour lui, la seule source sur ce milieu. Ces dernières peuvent être utilisées pour avoir une représentation de l'environnement lui-même. Si l'environnement est statique, cette dernière peut être exploitée pour de prochaines missions.
- **Décider :** selon l'architecture de contrôle adoptée, le robot définit l'action que doit entamer. Pour une architecture hiérarchique ou délibérative, il s'agit d'une séquence d'actions ; puisque le robot dispose d'un modèle de l'environnement. Dans le cas d'une architecture réactive, le robot réagit d'une manière réflexe aux stimulés des capteurs.
- **Agir :** enfin, le robot exécute les séquences d'actions en envoyant des consignes aux actionneurs par l'intermédiaire des asservissements.

La problématique de l'autonomie des robots est, donc, la détermination à chaque instant, sans aucune intervention externe, les commandes qui doivent être envoyées aux effecteurs, connaissant, d'une part, *le but* à atteindre et, d'autre part, les valeurs retournées par les différents capteurs (état interne du robot et celle de l'environnement). Il s'agit *de déterminer les liens existants entre la perception et l'action connaissant le but à atteindre*.

Mais, un robot évaluant dans un environnement réel est confronté à de multiples contraintes citons [Lachekhab05] :

- l'environnement est vaste et dynamique. Contrairement à un bras manipulateur fixe travaillant dans un espace de travail assez réduit, le robot mobile évolue dans un environnement vaste non contrôlé et dans la plupart des cas non modélisable. Cela signifie que des objets (obstacles) peuvent apparaître, se déplacer ou disparaître et de manière non prévisible. Ce qui exige que le robot soit doté de capteurs lui permettant d'acquérir des informations sur son environnement, du moins, le plus proche. Pour ce, il existe une variété de capteurs (caméras vidéo, télémètre, capteurs ultrasoniques ou infrarouges, etc.) et d'un système de navigation capable de gérer les différentes situations déjà citées.

L'objectif majeur de la recherche en robotique, actuellement, est la création de robot autonome capable d'agir sur l'environnement en vue de réaliser des taches prédéfinies, tout en s'adaptant à certaines variations de leurs conditions de fonctionnement, moyennant une spécification externe *de haut niveau* de la tache à accomplir sans aucune intervention humaine. Donc, parmi l'ensemble des robots existants, nous nous intéresserons au cours de cette étude à cette sous-famille appelée *les robots mobiles autonomes*.

Qu'est-ce qu'un système autonome ?
Un système est dit autonome si [Fukuda99], [Lewis06] :

- Il est capable de percevoir son environnement.
- Il est capable d'accomplir les objectifs pour lesquels il a été conçu.
- Il est capable de choisir, sans interventions externes, ses action afin d'accomplir ces objectifs.

4. Architectures de contrôle des robots mobiles

Les architectures de contrôle sont généralement divisées en trois groupes [Brooks 86], dont les deux premières se sont longtemps opposées [Lewis06] [Siegwart04] [Kim03] :

- *Approche délibérative* : utilise une modélisation de l'environnement connu *a priori* ou obtenue à partir des données capteurs, pour planifier à l'avance les commandes que le robot doit exécuter.

- *Approche réactive* : ne suppose aucun modèle a priori de l'environnement. Cette architecture s'appuie sur le couplage étroit entre les capteurs et les actionneurs, pour générer en continu les commandes que le robot doit exécuter.

- *Approche hybride* : tente de combiner les deux approches précédentes afin de tirer partie des avantages respectifs de ces deux premières approches.

Ces trois approches (figure 1.2) ne différent pas dans leurs détails de contrôle, mais plutôt, par leur agencement et leurs relations.

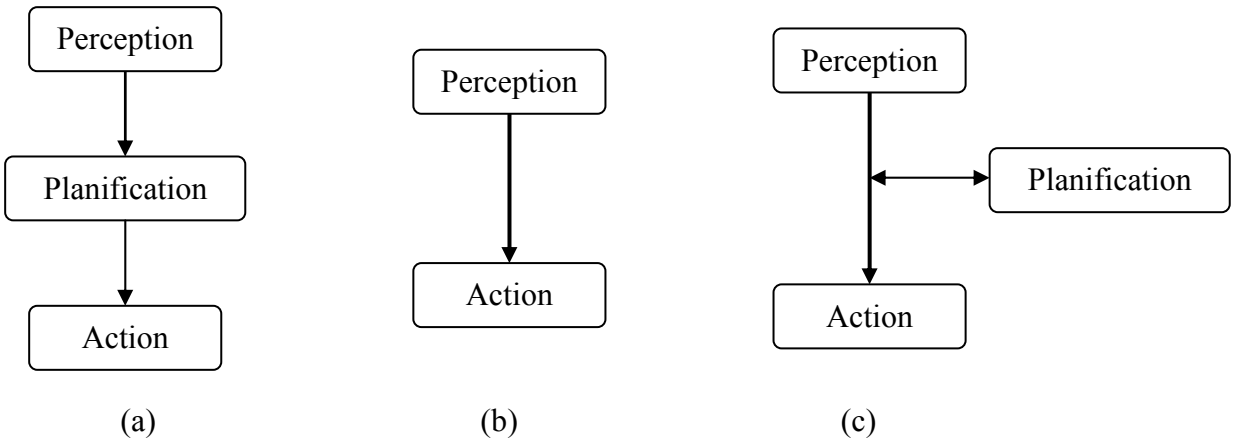

Figure 1.1 : architectures de contrôle pour les robots mobiles : (a) hiérarchique, (b) réactive, (c) hybride.

4.1 Les systèmes purement délibératifs (hiérarchiques)

Les premiers robots mobiles, obtenus des recherches en intelligence artificielle, utilisaient des contrôleurs hiérarchiques dont le fonctionnement repose essentiellement sur la capacité de décision travaillant sur un modèle de l'environnement supposé parfait. Ces architectures ont l'avantage de prouver l'existence d'une solution optimale permettant au robot d'atteindre le but assigné. Elles fonctionnent selon un cycle rigide de modélisation de l'environnement, planification des actions au sein de cette représentation, puis exécution du plan. Cette vision de la robotique autonome conduit à une décomposition séquentielle du traitement réalisé et à des systèmes fortement hiérarchiques.

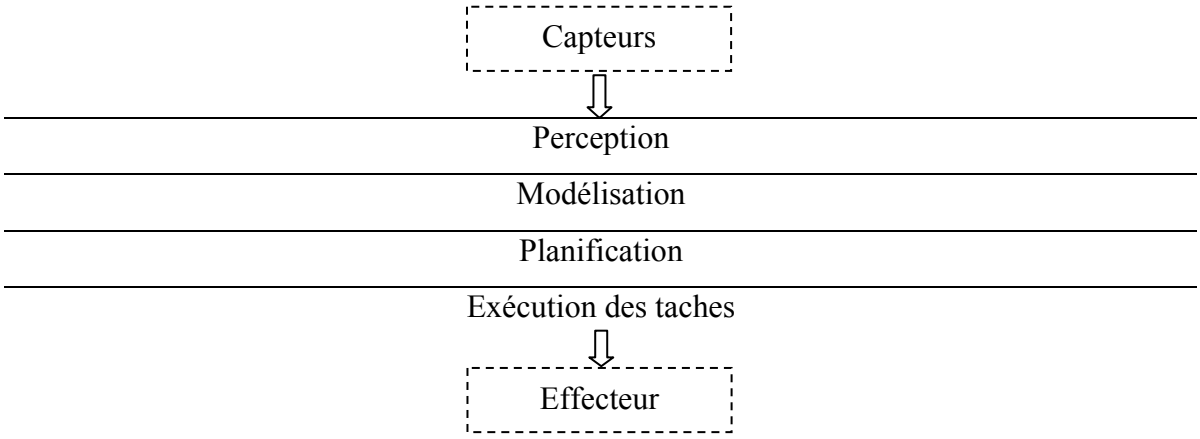

Figure 1.2 : Architecture hiérarchique.

Le traitement est décomposé en une série d'opérations successives décrites dans la figure (1.2). Cette première architecture ne garantie la réussite de la tache que dans le cas des environnements connus et statiques ; puisque la planification est faite au début. Dans le cas de changement de la configuration de l'environnement, le plan établi n'est plus valable. De plus, les temps de calcul pour le déplacement sont très grands.

C'est dans ce contexte que de nouvelles méthodes de contrôle s'imposèrent dans le domaine de la navigation autonome des robots. Ces méthodes doivent permettre de surpasser ce problème. Ceci est possible en établissant un couplage entre la perception et l'action en faisant des traitements locaux.

4.2 Les systèmes réactifs

En 1986, Brooks [Brooks 86] propose une approche radicale à tous ces problèmes, sous la forme d'une architecture *réactive* qui se distingue par l'abandon des phases de modélisation et de planification. Sans aucun modèle de l'environnement, cette architecture se compose d'un ensemble de comportements réactifs fonctionnant en parallèle pour le contrôle du robot (figure (1.3).

Cette approche, d'inspiration éthologique ou physiologique, tente de reproduire les principales propriétés du comportement des êtres vivants. Elle est basée sur plusieurs comportements (*convergence vers la cible, évitement d'obstacle, suivi de murs, exploration de l'environnement..*). Pour guider le robot, il faut choisir à chaque instant lequel de ces comportements à activer ou comment les combiner. Ce problème est connu dans la littérature sous le nom de *sélection de l'action*. La solution proposée par Brooks [Brooks86], *l'architecture de subsomption*, est devenue un classique. Elle utilise une hiérarchie des comportements qui s'exécutent selon un ordre de priorité en fonction des *perceptions* du robot.

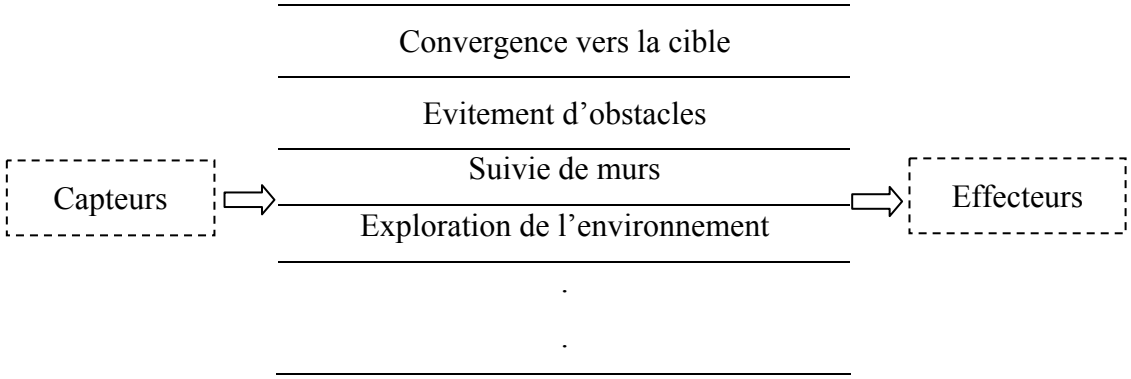

Figure 1.3 : Architecture réactive.

Les tâches sont parallélisées. Dans cette hiérarchie, le concept de base est le suivant : un problème (comportement) donné, si complexe soit-il, peut être décomposé en sous problèmes (comportements) élémentaires, représentant chacun un niveau de compétence. Ces différents niveaux sont placés en couches (hiérarchie de couches) (figure 1.4).

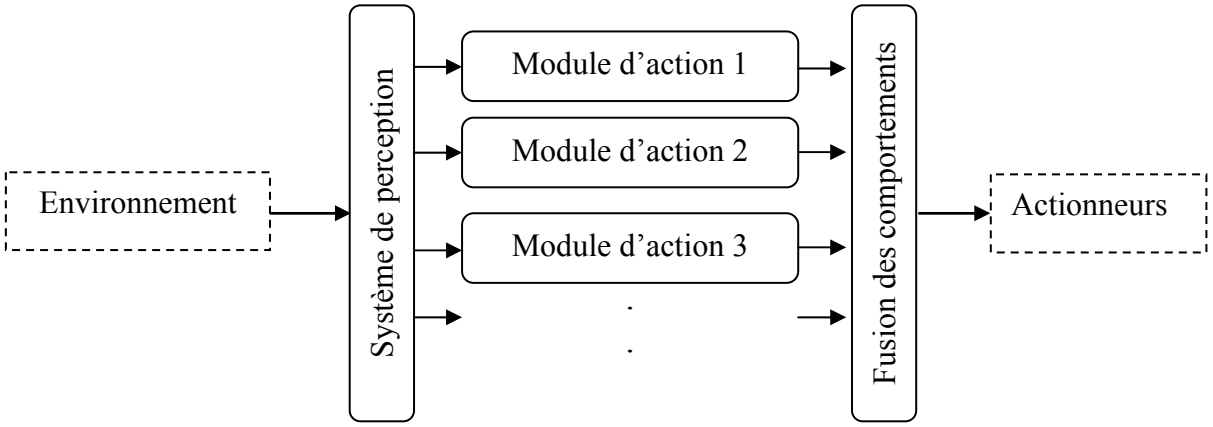

Figure 1.4 : Structure d'une architecture de « subsomption »

Cette architecture présente au moins quatre avantages :

- elle peut réagir aux éventualités en temps réel, dû au parallélisme.
- chaque comportement à une tache qui lui est confiée. Ce qui facilite leur implantation, ainsi que l'insertion ou la suppression de modules.
- elle présente une bonne robustesse ; le système peut rester fonctionnel, même si y a échec d'un ou plusieurs comportements.
- cette approche est plus adaptative à l'environnement. Même pour un environnement dynamique, on a plus de chance pour la réussite de la tache.

C'est cette architecture que nous retenons dans ce travail.

4.3 Systèmes hybrides

Cette architecture, intermédiaire entre les deux premières architectures, diffère de l'architecture réactive (celle que nous avons adoptée dans ce mémoire) par insertion de niveaux de taches de plus hauts niveau (planification,...etc.) exploitant des données à priori sur l'environnement.

5. Conception et simulation du robot dans son environnement de navigation

5.1 Contexte et hypothèses de travail

Afin de définir le cadre de notre travail, nous avons essayé d'envisager les différentes applications possibles des robots mobiles. La grande partie d'entre elles se font dans des environnements partiellement connus, inconnus ou dynamiques. Souvent, les environnements dans lesquels les robots évoluent sont difficilement modélisables et le processus de modélisation est coûteux en calculs.

Dans ce paragraphe, nous exposerons les différentes hypothèses que nous avons formulées pour réaliser le présent travail :

a. Le robot évolue dans un environnement plat et uniforme.
b. La position et l'orientation du robot sont obtenues par intégration des vitesses.
c. Les vitesses du robot sont suffisamment basses pour négliger les effets dynamiques. On peut, alors, se limiter au modèle cinématique.
d. La perception de l'environnement est parfaite ; les obstacles présents dans l'environnement sont détectés par le robot.

5.2 Simulation de l'environnement

Dans le but de procéder à la simulation de la navigation autonome d'un robot mobile, nous avons procédé, en premier lieu, à une simulation d'un environnement dans lequel on pourra effectuer nos tests.

La simulation de l'environnent consiste à générer une surface sous Matlab (400*400 grilles dans notre cas) pouvant servir de plateforme pour l'implantation des différents objets (robot et obstacles). Cette opération est assurée moyennant des procédures appropriées, à savoir : une procédure simulant un robot mobile, une autre pour la simulation d'obstacles ronds et une troisième simulant des obstacles carrés. Cependant, les procédures de perceptions servent à calculer les distances ainsi que les angles entre le robot et les différents obstacles pour donner les états des différents capteurs du robot. Dans la section ci-après, nous avons jugé utile de donner plus de détails sur cette simulation.

5.3 Simulation du robot

Le robot est représenté par les trois coordonnées (x_c, y_c, θ). Avec (x_c,y_c) sont les coordonnés du centre de gravité du robot exprimées dans le repère absolu et (θ_{robot}) est l'orientation du robot prise entre l'axe des abscisses du repère absolu et la direction du robot.
La procédure *Robot1*, ayant comme entrées les variables citées ci-dessus, donne la représentation graphique du robot mobile (figure 1.5).

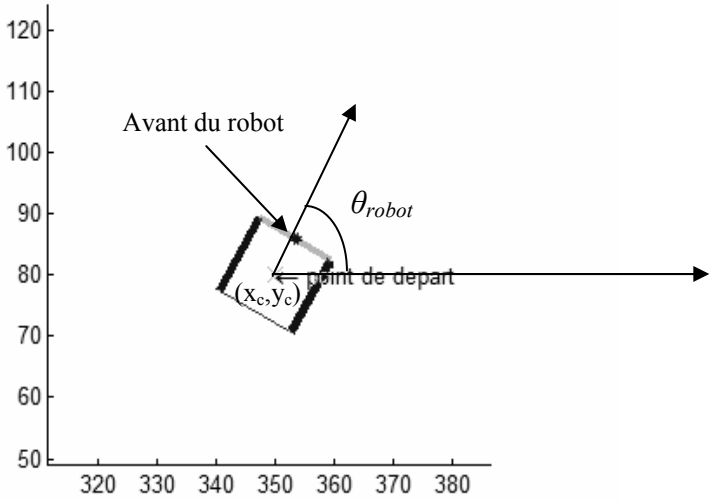

Figure 1.5 : Positon (x_c,y_c) et orientation (θ_{robot}) du robot dans le repère absolu.

5.4 Simulation des obstacles

L'environnement simulé doit être le plus proche possible de la réalité. C'est-à-dire, il doit contenir les objets pouvant se présenter dans un environnement réel. Ou encore, il doit avoir les natures de formes géométriques les plus répandues (courbées et droite). Dans notre étude, nous nous sommes limités aux deux formes suivantes (figure 1.7) : les formes rondes et les formes carrées.

- **Obstacles ronds :**

Un obstacle de forme ronde est caractérisé par son centre (x_0, y_0) et son rayon R. Pour l'insérer dans l'environnement, on introduit ces derniers comme entrées de la procédure *obscercul* qui donne dans l'environnement simulé la représentation graphique de cet obstacle.

- **Obstacles carrés :**

Un obstacle de forme carrée est caractérisé par quatre coordonnés (x_0, y_0, x_1, y_1). Pour insérer ce genre d'obstacles dans l'environnement simulé, il suffit d'introduire ces coordonnées comme entrées de la procédure *obscercare*.

5.5 Perception de l'environnement

Dans notre travail, nous avons opté pour la navigation autonome dans un environnement inconnu. Ce qui signifie que le robot n'a aucune information à priori sur la configuration de son environnement. Pour réussir les taches qui lui ont été assignées, il doit être doté d'un module de perception lui permettant de percevoir l'environnement dans lequel il évolue, du moins le plus proche, et ce, en exploitant les données issues de ses capteurs.

En s'inspirant du robot Khepera (développé par le laboratoire K-Team SA, Suisse) [Siegwart04], nous avons simulé les six capteurs infrarouge avant du robot pour la détection des obstacles présents dans l'environnement. Cette tache est assurée par une procédure de test des états des capteurs qui, à son tour, fait appel à une autre procédure permettant le calcul des distances et des orientations de robot-obstacles.

- **Répartition des capteurs**

Les capteurs du robot sont repartis comme le montre la figure 1.6. Chacun d'eux balaie une zone de l'espace couvrant un angle de 30°, et ce, à une portée de rayon *R* que nous avons choisit de telle sorte qu'il soit de l'ordre de la taille du robot. L'environnement simulé avec les obstacles et le robot est illustré sur la (figure 1.7)

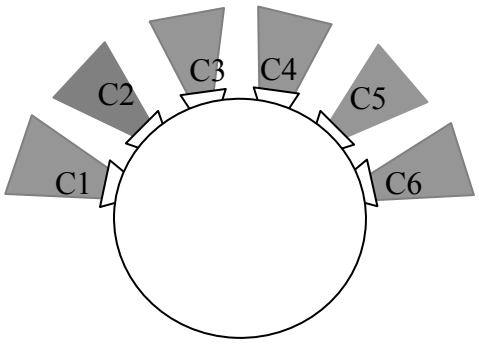

Figure 1.6 : Répartition des capteurs du robot et leurs spectres.

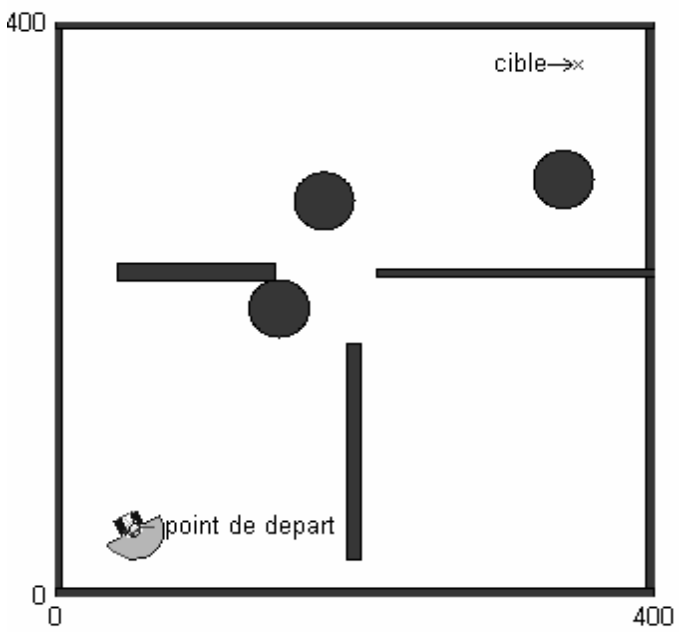

Figure 1.7 : Environnement simulé avec le robot et les différents obstacles.

6. Conclusion

Dans ce chapitre, nous avons exposé les différents modèles et architectures de contrôle (hiérarchique, réactif et hybride) utilisés en robotique en citant les avantages et les inconvénients de chacun. L'architecture retenue dans notre étude est l'architecture réactive comprenant trois modules : un module de convergence vers la cible, un module d'évitement d'obstacles et un module de fusion des deux comportements précédents. Enfin, nous avons présenté la plateforme simulée pour effectuer nos tests de navigation.

Chapitre 2 :

Navigation Floue

Chapitre 2 : Navigation floue

Dans ce chapitre, les principes de la logique floue sont utilisés pour la conception d'une commande pour un robot mobile dans un environnement inconnu.

Dans un premier temps, l'environnement considéré est sans obstacles. Ce qui n'exige que le seul comportement de convergence vers la cible. La seconde partie traite le problème de présence d'obstacles dans l'environnement de navigation. Ce qui nécessite, en plus de la convergence vers la cible, un autre comportement lui permettant d'éviter ces derniers. La gestion des consignes issues des deux contrôleurs est assurée par un module de fusion de comportements.

Un aperçu de l'outil utilisé (la logique floue) est donné en premier.

1. Introduction

La problématique à traiter dans ce mémoire est la planification de trajectoires pour un robot mobile dans un environnement partiellement connu. La tache confiée au robot est l'atteinte de cibles sans aucune intervention externe, et ce, dans un environnement sur lequel peu d'informations à priori lui sont fournies. Le robot est amené à exploiter les informations issues de ses capteurs pour pouvoir réaliser ces taches et se guider de sorte à éviter les obstacles qui peuvent surgir sur son chemin.

Dans ce contexte plusieurs applications ont été développées pour assurer une navigation réactive dans un environnement partiellement connu voire même totalement inconnu à savoir : les méthodes de champ de potentiel et la logique floue [Lewis06], [Siegwart04].

La navigation autonome d'un robot mobile est un comportement difficile à modéliser et un formalisme mathématique, s'il y est ? s'avère très complexe. Une manière de pallier la recherche explicite d'un modèle, on a fait appel à la commande floue. Celle-ci est basée sur la notion d'ensembles flous [Zadeh 65] et la perception du système par l'opérateur humain.

Dans le paragraphe qui suit, Nous donnerons un aperçu général de ladite technique ainsi que la démarche de conception de contrôleurs flous.

2. Aperçu sur la logique floue

2.1 Historique

Depuis une vingtaine d'années, l'utilisation de la logique floue dans la modélisation et la commande des systèmes complexes est devenue un outil à part entière. Le concept d'ensembles flous a été introduit par Zadeh en 1965 [Zadeh65]. Il constitue une interface de commande pour la modélisation du langage naturel, en particulier, des concepts linguistiques utilisés par l'expert d'un procédé [Guenounou03].

Le principe de la commande floue à été expérimenté pour la première fois, sur une turbine à vapeur par Mamdani et Assilian en 1974 [Mamdani 74] et la première application industrielle d'envergure, date de 1987 au Japon, dans le transport ferroviaire (métro) de Sendai (Tokyo), dont les performances rivalisent celles d'un système de commande classique **[**Sugeno 85**]**.

De nos jours, les domaines d'application de la commande floue deviennent de plus en plus importants (industrie, automobile, robotique ...), et peuvent être classés en deux catégories :

La conception de contrôleurs pour les procédés difficilement modélisables.

La conception de contrôleurs pour les procédés modélisables non linéaires.

2.2 Contrôleur flou

La logique floue a été conçue essentiellement, comme nous l'avons déjà dit, pour la commande des systèmes complexes ou mal définies et le traitement des données approximatives. Elle permet de prendre en considération des variables linguistiques dont les valeurs sont des expressions du langage naturel, telle que *négatif, positif, loin, proche, gauche droite,...*etc. A travers le nouveau concept d'ensembles flous, où un élément appartient de manière graduelle à un ensemble, elle permet, donc, de prendre en compte les états intermédiaires entre les deux états (tout ou rien) de la logique boolienne [Homaifar95].

L'idée principale de la commande floue est d'imiter le processus du *raisonnement humain* en exprimant les connaissances humaines sous forme de règles de type « *si prémisses* alors *conclusions* ». La partie prémisses correspond à une description de l'état du système qui déclanche les règles et les conclusions. Ces dernières, représentent les actions qui doivent être appliquées au système.

En résumé, les différents modules d'un contrôleur flou sont (Figure 2.1) :

- **La fuzzification** : consiste à associer pour chaque valeur d'entrée, un ou plusieurs sous- ensembles flous. Cette étape fait passer les valeurs numériques en informations floues. Généralement la fuzzification singleton est utilisée.
- **Phase d'inférence** : consiste à calculer le degré de vérité de chaque règle et associer à chacune d'elle une valeur de sortie. Cette valeur est numérique dans le cas de contrôleur de type Sugeno et symbolique dans le cas de type Mandani.
- **La défuzzification** : dernière étape où les valeurs numériques des sorties sont obtenues à partir de la fonction d'appartenance résultante de la phase d'inférence floue. Dans la littérature, plusieurs méthodes sont proposées. Mais, les plus utilisées sont : méthode du centre de gravité (CG) et la méthode de la moyenne des maximas (MM).

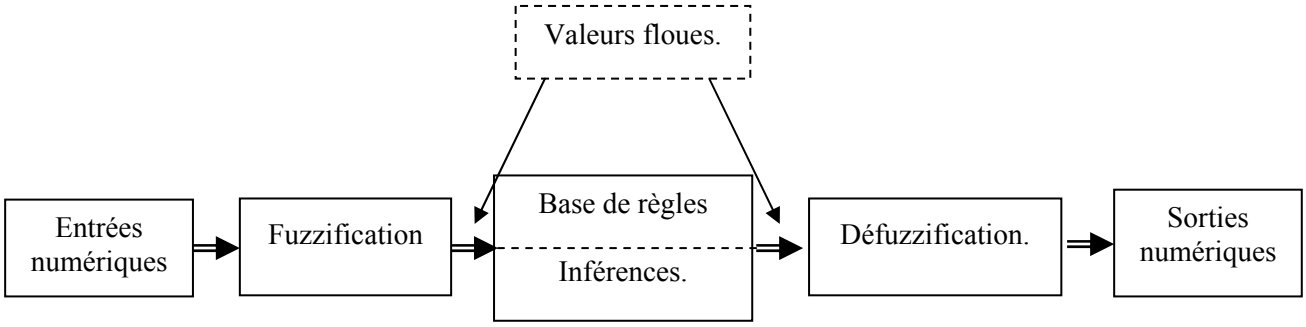

Figure 2.1 : Structure d'un contrôleur flou.

La figure (2.2) est une représentation graphique de ces étapes pour le calcul de la sortie d'un contrôleur flou de type Mamdani à deux variables d'entrée (X_1, X_2) et une variable de sortie (Y) intégrant la base de règles suivantes :

- **Si** X_1 **est** A_1 **et** X_2 **est** B_1 **alors** Y **est** C_1
- **Ou, si** X_1 **est** A_1 **et** X_2 **est** B_2 **alors** Y **est** C_2
- **Ou, si** X_1 **est** A_2 **et** X_2 **est** B_2 **alors** Y **est** C_3

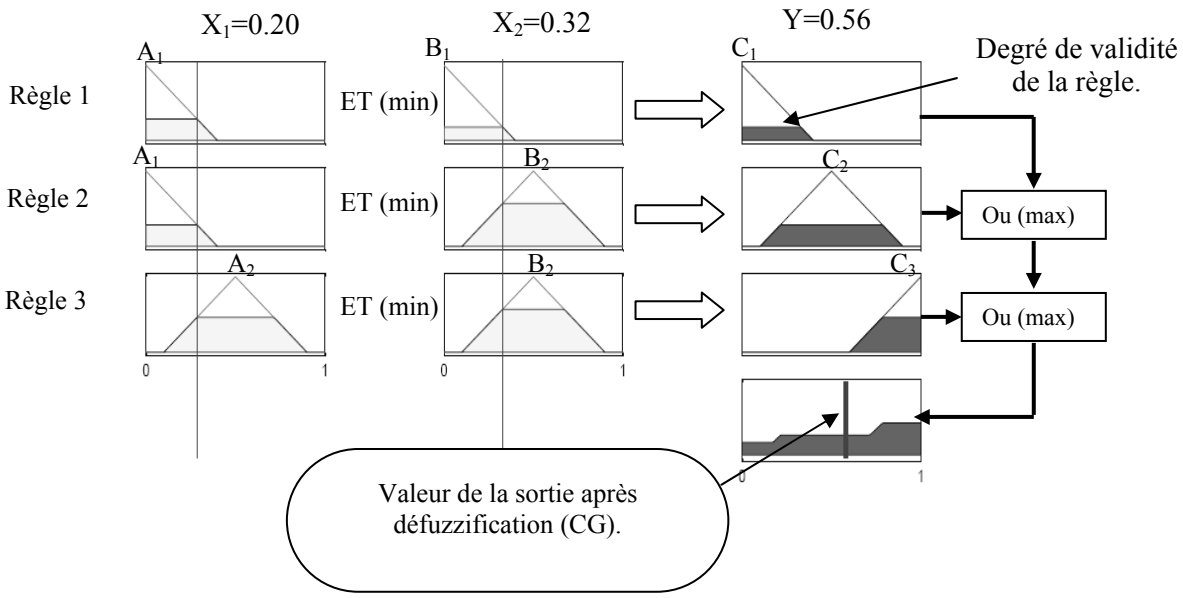

Figure 2.2 : Les étapes de calcul de la sortie d'un CF de type Mamdani.

3. Problématique de navigation floue d'un robot mobile

La résolution du problème de navigation d'un robot mobile est sujette, en grande partie, aux recherches et aux résultats obtenus en intelligence artificielle ou en informatique [Brooks86] [Wang92] [Shibata94] [Fukuda 99] [Siegwart04]. Dans cette optique, une des facultés premières que doit intégrer un robot mobile autonome est la capacité de mouvoir et de converger vers des lieux (cibles) qui lui sont indiqués au préalable sans aucune intervention externe. De plus, dans un environnement partiellement connu ou inconnu, des objets gênant peuvent surgir. Le robot doit être doté d'une seconde faculté qui lui permet d'éviter ces obstacles. Dans notre cas, ces facultés sont mises en œuvre en utilisant le raisonnement flou. Dans un premier temps, nous avons implémentée ces deux facultés exclusivement. Puis, nous les avons fusionnées par un module commun. Le schéma global du système de navigation est illustré par la figure 2.3.

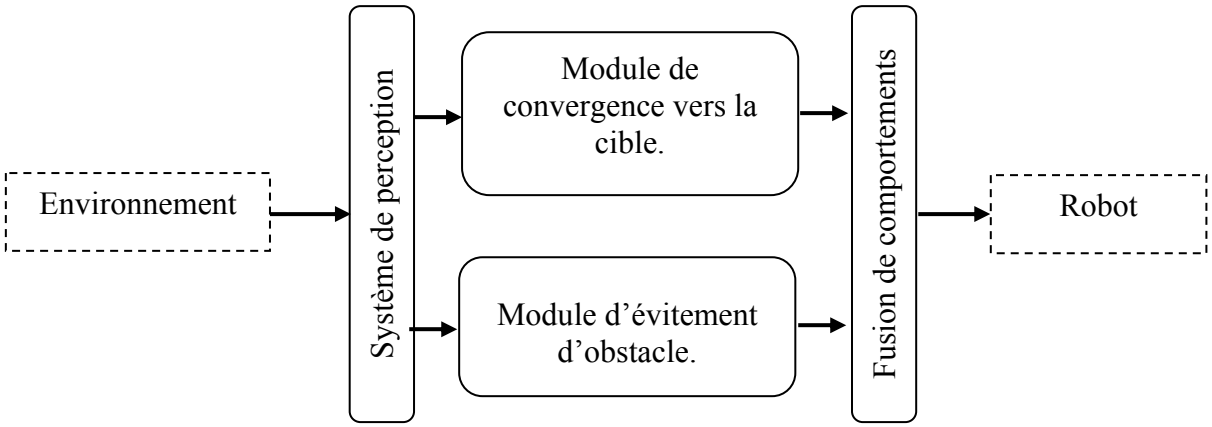

Figure 2.3 : Schéma de contrôle du robot.

4. Navigation dans un environnement sans obstacles

Dans cette section, nous allons développer un contrôleur flou permettant d'assurer la tache de «*convergence vers la cible* ». Le travail est réalisé en considérant un environnement sans obstacles.

Ce premier comportement permet au robot de se mouvoir en utilisant seulement les information sur ses position/orientation courantes et la position cible à atteindre ; sans aucune prétention aux états des capteurs et, par ce, ignore la présence des obstacles. L'unique objectif visé par ce contrôleur est de guider les déplacements du robot pour qu'il puisse atteindre le point d'arrivée. Il est bien entendu que ce comportement ne peut opérer que dans un environnement sans aucun obstacle gênant la progression du robot.

Pour que le robot puisse atteindre la cible (les cibles), les coordonnés de cette dernière, lui sont introduites au début. Par la connaissance de ses position/orientation courantes, il calcule les valeurs des deux variables : distance du robot de la cible « D_{cible}» et l'erreur d'orientation (angle entre la direction du robot et le vecteur robot-cible) « φ_{cible} » (figure 2.4).

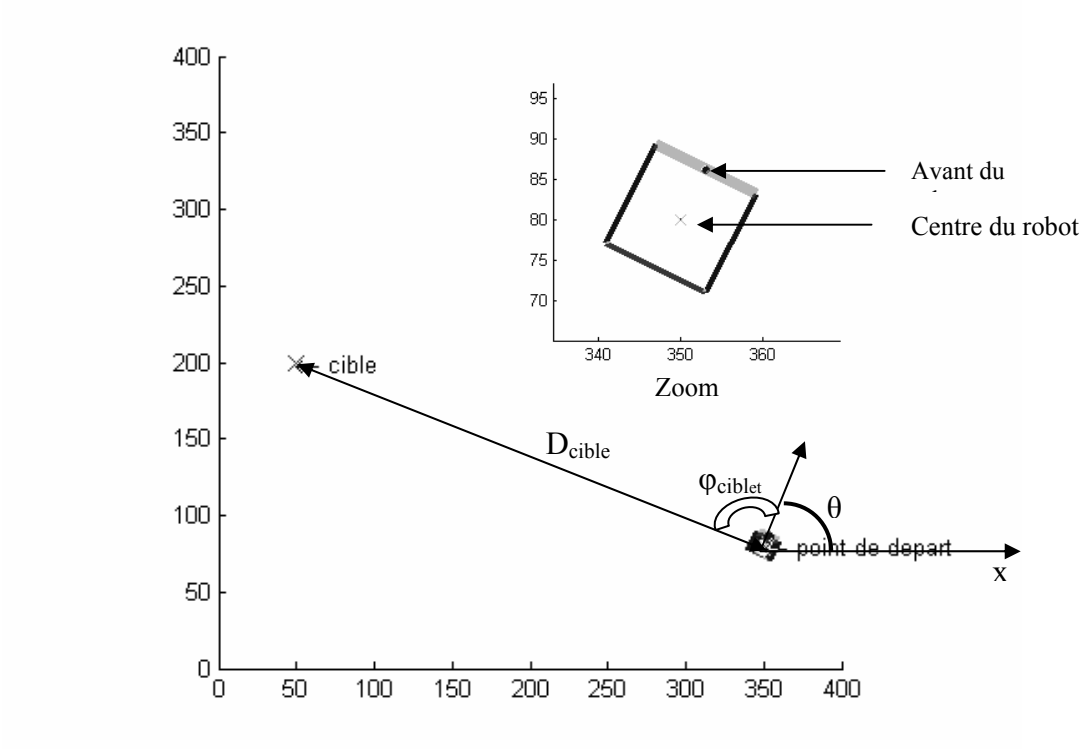

Figure 2.4 : Distance et orientation initiales du robot par rapport à la cible.

Pour une commande cinématique du robot, en utilisant le modèle d'un robot mobile à roues différentielles (équation 2.1), les variables de sortie du contrôleur sont : la vitesse de braquage (**Vb**) et la vitesse d'avancement **(Va)** du robot (Figure 2.5) où [X, Y, θ] est la position et orientation du robot.

Le modèle cinématique utilisé est donné par [Topalov98] [Siegwart04] [Lewis06] [Kim03]:

$$\begin{cases} \dot{x} = v_a * \cos(\theta) \\ \dot{y} = v_a * \sin(\theta) \\ \dot{\theta} = v_b \end{cases} \tag{2.1}$$

Tel que (x, y) coordonnées du robot dans le repère absolu et θ son orientation. La position et orientation du robot sont obtenues par intégration de l'équation (2.1), d'où :

$$\begin{cases} \theta(i+1) = \theta(i) + \dot{\theta} * Te \\ x(i+1) = x(i) + \dot{x} * Te \\ y(i+1) = y(i) + \dot{y} * Te \end{cases} \tag{2.2}$$

Tel que : Te est la période d'échantillonnage.

L'erreur d'orientation du robot est donnée par la relation :

$$\varphi(i+1) = \varphi(i) - v_b * Te \tag{2.3}$$

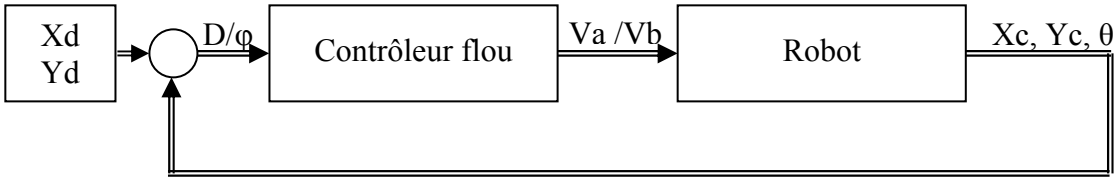

Figure 2.5 : Structure de commande utilisée.

Puisque nous avons fait appel au contrôle flou, nous avons d'abord procédé à une partition floue de l'espace opérationnel en trois orientations (gauche, avant, droite) où gauche pour $\varphi \in]\, 0\, ,\, \pi\,]$, droite pour $\varphi \in [-\pi, 0\, [$ (selon le sens horaire) [Siegwart 04].

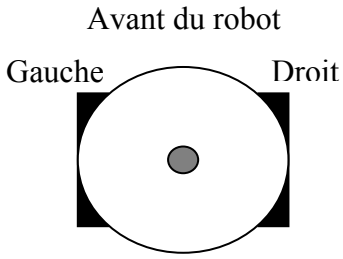

Figure 2.6 : Schéma représentatif d'un robot mobile.

Définition des fonctions d'appartenance :

Les univers de discours des variables d'entrée (D_{cible} et φ_{cible}) du contrôleur de convergence vers la cible, retenus après plusieurs essais, sont respectivement décomposés en trois et cinq sous ensembles flous selon figure (2.7) et figure (2.8). Cette partition floue est assez simple et permet d'obtenir une base de règle concise et facile à interpréter (15 règles).

Les labels utilisés pour la distance D sont : Pr (proche), Moy (moyennement proche) et L (loin) et pour la variable erreur d'orientation φ_{cible} : C-D (complètement droite), MD (moyennement droit), C (centre), MGa (moyennement gauche), C-G (complètement gauche).

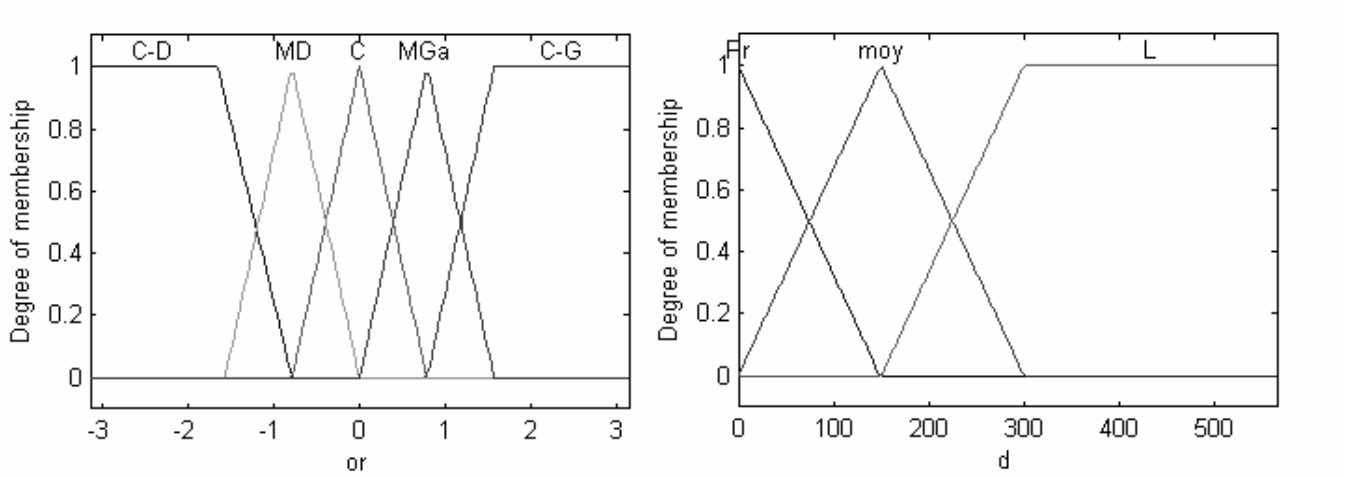

Figure 2.7 : les fonctions d'appartenances des variables d'entrée du contrôleur « convergence vers la cible »

Quant aux variables de sortie du contrôleur (la vitesse d'avancement « **Va »** et la vitesse braquage « **Vb »)** sont partitionnées comme suit : la première variable est partitionnée en trois sous-ensembles flous : Z (zéro), Rap (rapidement) et TRap (très rapidement) et la deuxième variable est partitionnée en cinq sous-ensembles flous : DrRap (droit rapide), Dlent (droit lent), Z (zéro) et Galent (gauche lent) et GaRap (gauche rapide) (figure 2.8).

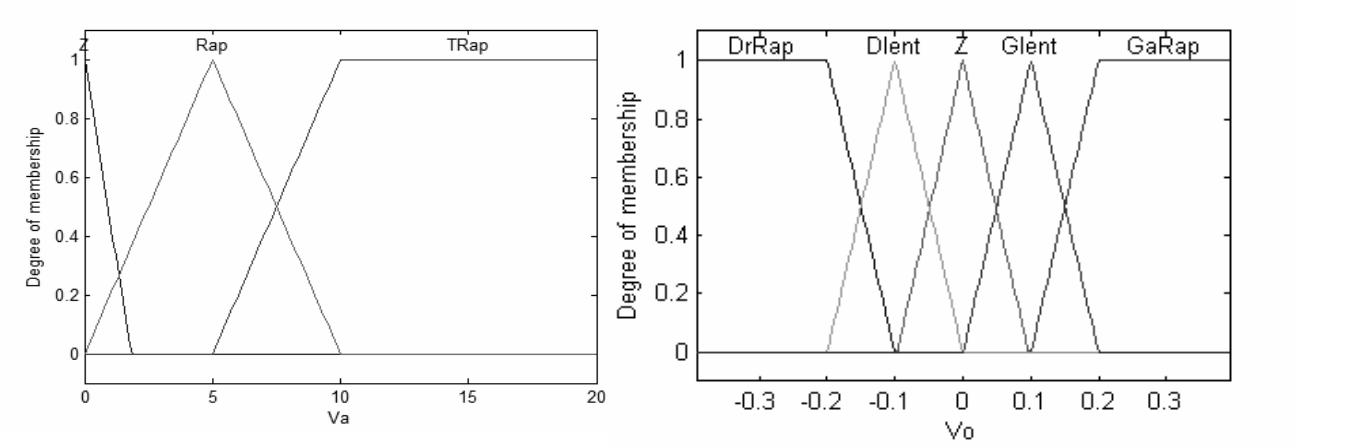

Figure 2.8 : les fonctions d'appartenance des variables de sotie du contrôleur « convergence vers la cible »

Base de règles :

Le contrôleur flou utilisé est de type Mamdani et l'étape d'inférence floue fait appel à des règles floues faisant un lien entre les variables d'entrée D_{cible} et φ_{cible} et les variables de sortie Va et Vb. Les règles floues sont des expressions formées de prémisses (conditions) et de conclusions (actions) de la forme :

- **Si** D_{cible} est L **et** φ_{cible} est C **alors** Vb est Z **et** Va est TRap.

Les deux tableaux ci-dessous représentent la base de règles floues pour les deux variables de sortie **Va (**Tableau 2.1) et **Vb** (tableau 2.2), et ce, pour toutes les combinaisons possibles des deux variables d'entrée. Ces deux tables sont établit sur la base du raisonnement suivant : à partir de sa position initiale, le robot s'oriente vers la cible puis converge vers elle avec une vitesse inversement proportionnelle à la distance robot-cible pour s'arrêter une fois la cible atteinte avec une marge d'erreur admissible.

D_{cible} / φ_{cible}	Pro	Moy	L
CD	Z	Z	Z
MD	Z	Rap	Rap
C	Z	TRap	TRap
MG	Z	Rap	Rap
CG	Z	Z	Z

Tableau 2.1 Base de règles floues de la vitesse d'avancement.

D_{cible} / φ_{cible}	Pro	Moy	L
CD	DrRap	DrRap	DrRap
MD	Dlent	DrRap	DrRap
C	Z	Z	Z
MG	GaLen	GaRap	GaRap
CG	GaRap	GaRap	GaRap

Tableau 2.2 base de règles floues de la vitesse de braquage.

Les figures ci-dessous représentent les graphes de surface du contrôleur pour une t-norme min, S-norme est max et une défuzzification par la méthode du centre de gravité.

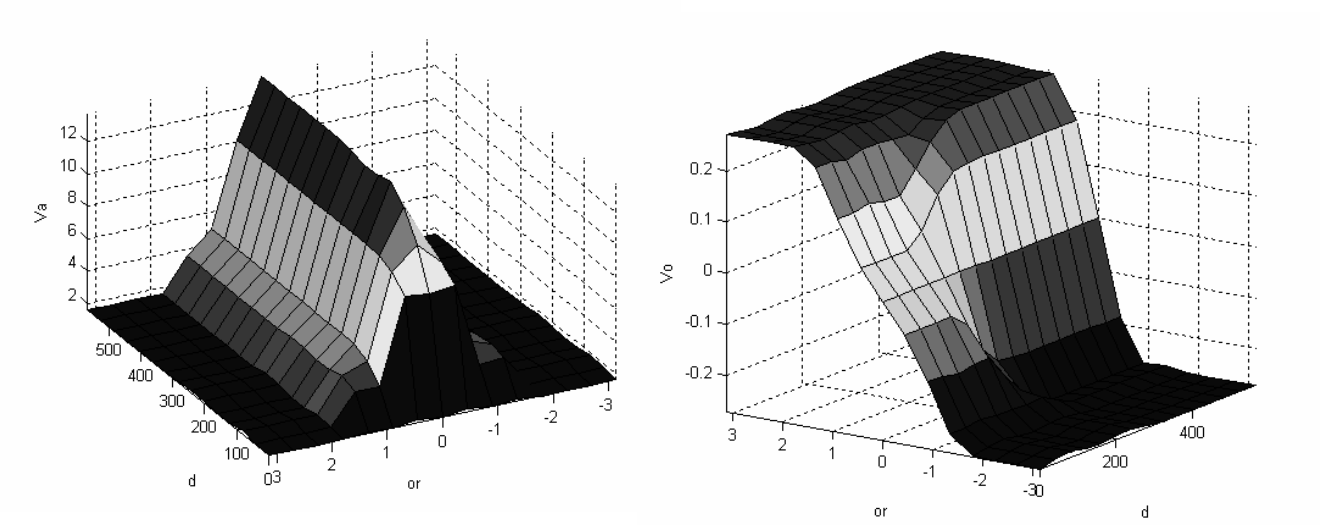

Figure 2.9 : Réponse du contrôleur en Va et Vb en fonction de D_{cible} et φ_{cible}.

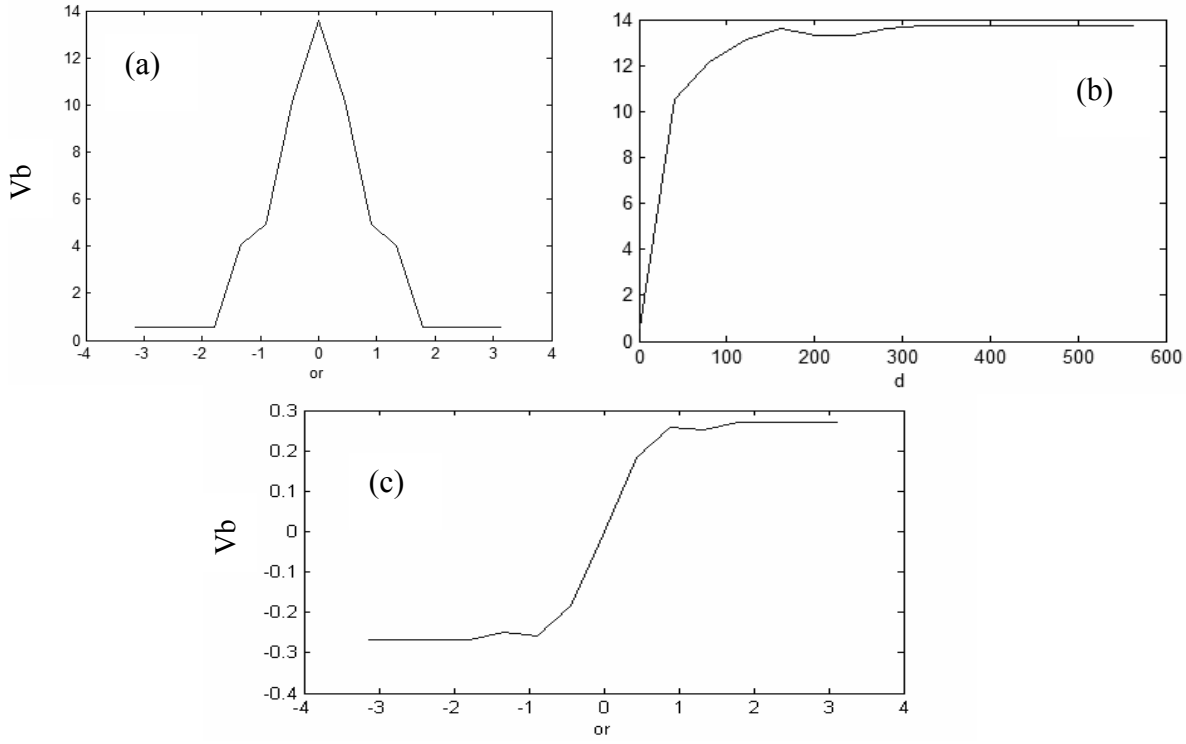

Figure 2.10 : Réponse du contrôleur : Va en fonction (a) de φ_{cible}, (b) de D_{cible} et Vb en fonction de φ_{cible} (c).

Résultats de simulation

La figure (2.11) représente la trajectoire réalisée par le robot pour atteindre la cible à une distance initiale $D_{cible} = 403.60$ u.l(unité de longueur) et une erreur d'orientation φ_{cible}=123.64° à partir du point de coordonnées (350,80) vers la cible de coordonnées (50,350). Les variations de la vitesse de braquage, vitesse de translation du robot et erreur orientation ainsi que la distance robot-cible en fonction du temps sont représentées sur la figure (2.12).

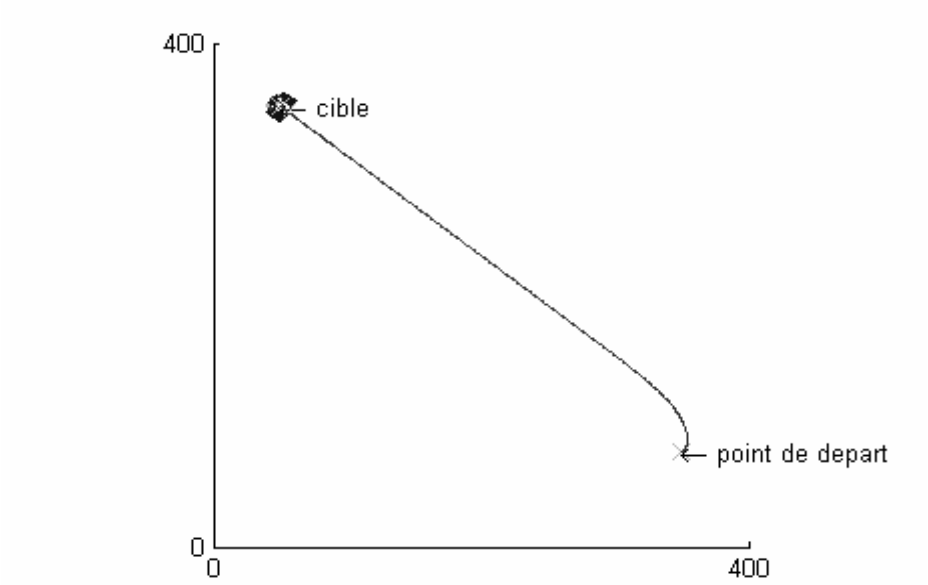

Figure 2.11 : Trajectoire réalisée par robot pour $D_{cible} = 403.60$ u.l. et φ_{cible0}= 123.64°.

Ce premier résultat, montre bien que le robot adopte le comportement souhaité, à savoir : le robot s'oriente d'abord vers la cible (vitesse de braquage maximale) tout en accélérant (avec une accélération inversement proportionnelle à l'erreur d'orientation) durant cette manœuvre. Ensuite, le robot avance avec une vitesse maximale, et enfin, à partir d'une distance jugée proche de la cible, décélère jusqu à l'arrêt total une fois cette dernière est atteinte.

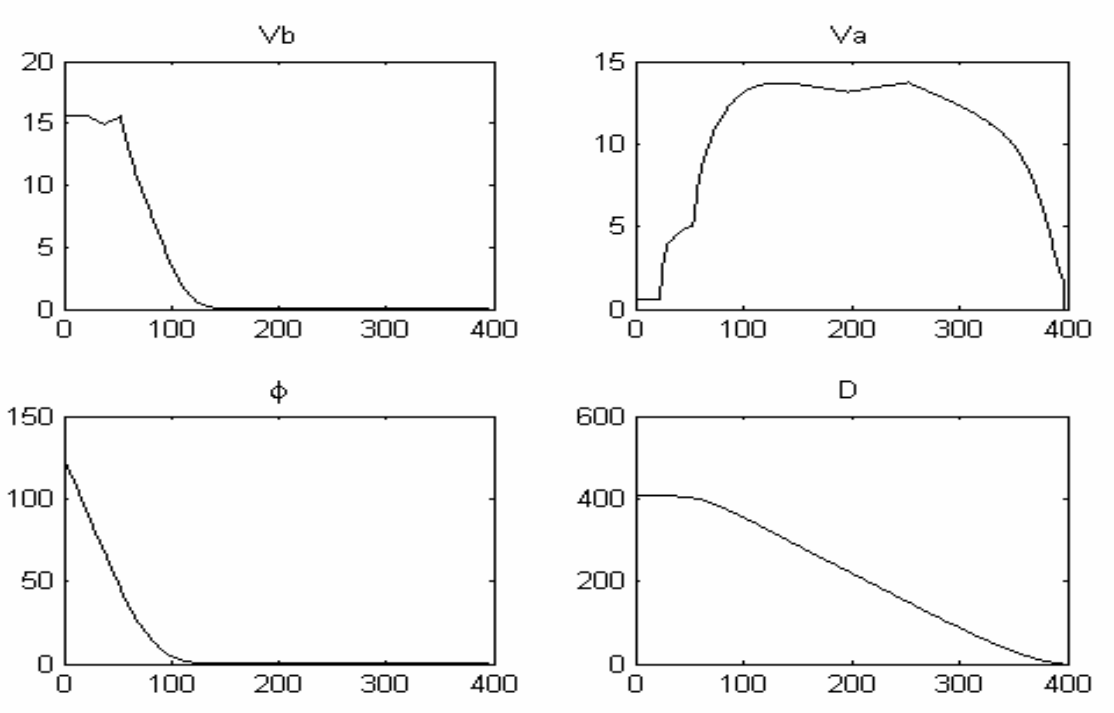

Figure 2.12 : Variation de la vitesse de braquage (Vb), vitesse d'avancement (Va), erreur d'orientation (φ) et distance (D) robot-cible en fonction de temps.

Cas de plusieurs cibles :

Pour valider le contrôleur conçu, on a procédé à un test incluant le maximum d'états de figures du robot afin exciter le contrôleur dans le maximum d'états de fonctionnement c'est-à-dire activer toutes les de règles implantées.

La figure (2.13) est la trajectoire réalisée par le robot, pour un point de départ de coordonnes (350,75) et orientation initiale de 0° (par rapport à repère absolu) en vu d'atteindre la cible4 (350, 350), tout en passant par les cibles : cible1 (50,200), cible2 (150,250) et cible3 (50,355). La figure (2.13) est la représentation de la vitesse de braquage, vitesse d'avancement, du robot et erreur orientation ainsi que la distance robot cible en fonction du temps.

Le résultat obtenu dans le cas de figure : plusieurs cibles dans un environnement sans obstacles montre l'efficacité de la technique adoptée pour un bon choix des différents paramètres du contrôleur à savoir : la définition des variables d'entrée/sortie, l'univers de discours pour chacune de ces variables ainsi que le choix des fonctions d'appartenance.

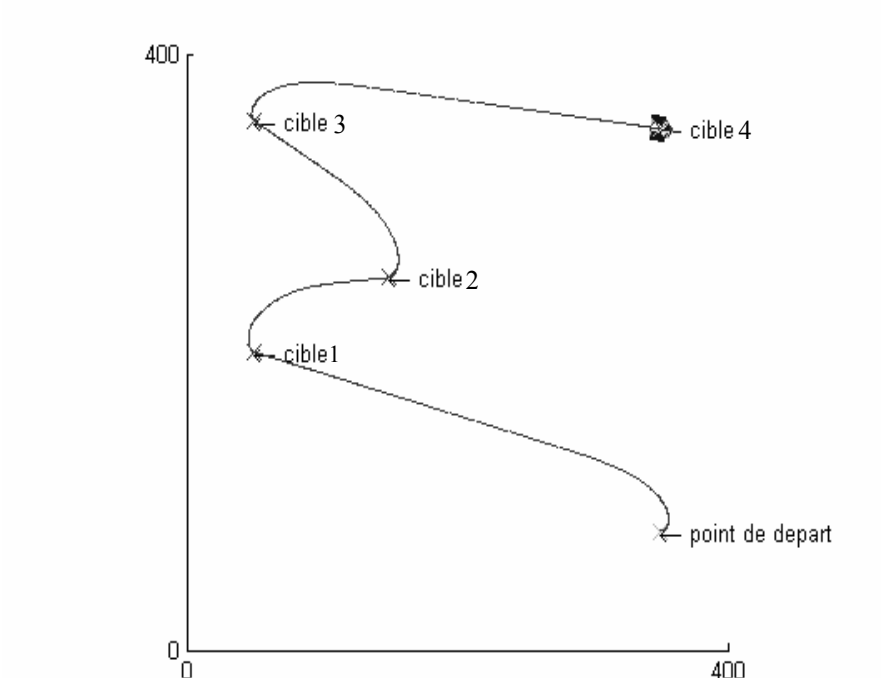

Figure 2.13 : Trajectoire réalisée par robot à partir de la configuration (350, 75,0°) pour plusieurs cibles.

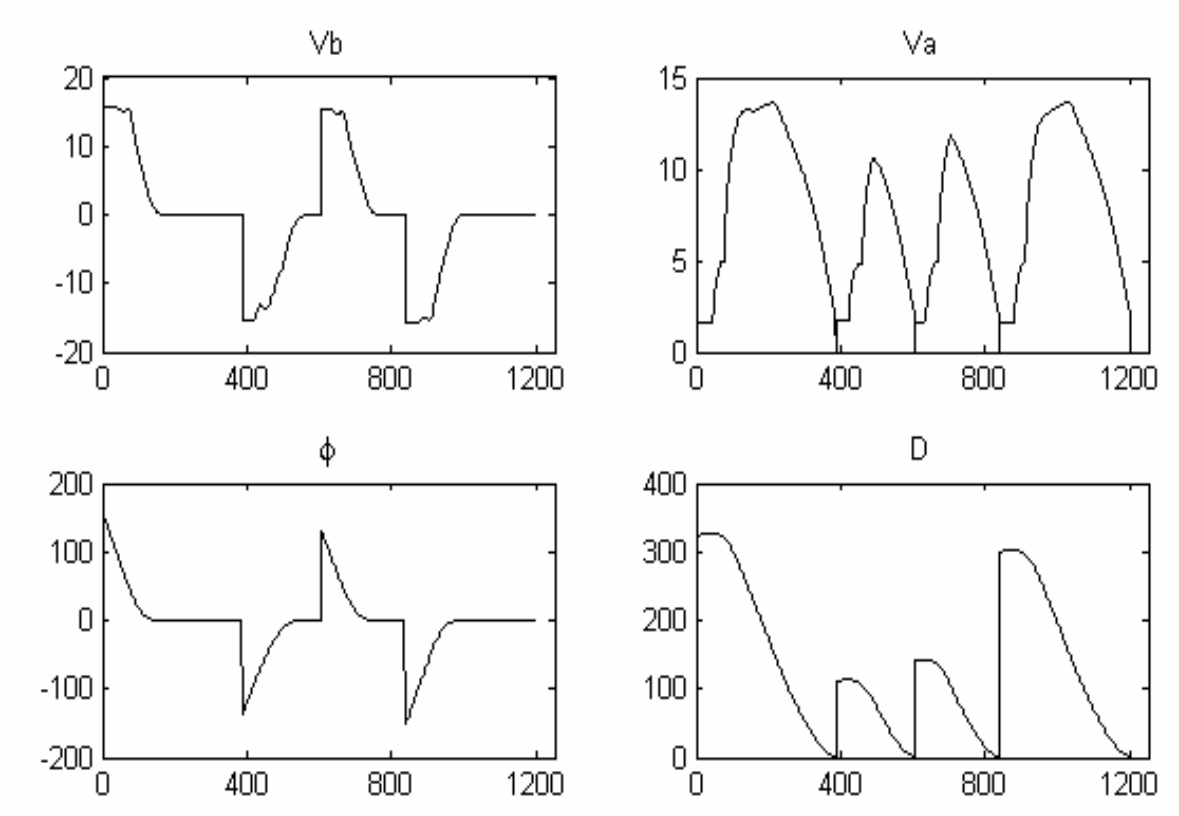

Figure 2.14 : Variation de la vitesse de braquage (Vb), vitesse d'avancement (Va), erreur d'orientation (φ) et da la distance (D) du robot-cible.

5. Navigation dans un environnement avec obstacles

Dans la première partie de ce chapitre, les résultats obtenus pour un environnement sans obstacles sont assez satisfaisants. Néanmoins, pour une approche non loin de la réalité, où des objets (obstacles) gênant la progression du robot peuvent être présents, le contrôleur de convergence vers la cible, employé seul, ne peut garantir au robot l'accomplissement de la tache qui lui est assignée (atteindre les cibles). Donc, le robot doit être munit d'un deuxième contrôleur lui permettant d'éviter la collision avec ces obstacles.

Le contrôleur dont on parle, *contrôleur d'évitement d'obstacles*, exploite les données (positions des obstacles par rapport au robot) issues des capteurs, répartis comme le montre la figure (2.15), pour guider le déplacement du robot.

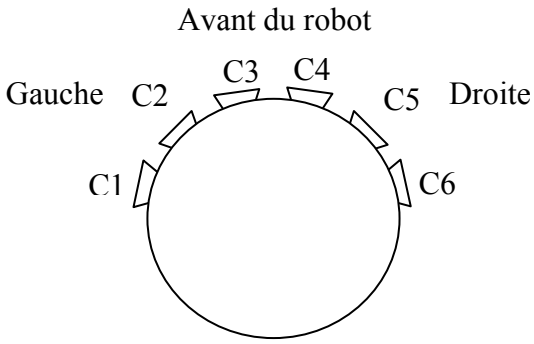

Figure 2.15 : Répartition des capteurs du robot.

Perception et détection des obstacles :

Dans notre cas, un module de perception est intégré via différentes procédures. La démarche suivie pour la simulation des capteurs de détection des obstacles est la suivante :

Calcul, à chaque instant, des distances du robot des obstacles ainsi que leurs positions. Si des obstacles se trouvent dans un périmètre de rayon « R » (simulant la portée des capteurs), on calcule l'état de chaque capteur, et ce, en fonction de la distance robot-obstacle et sa position (angle direction du robot-obstacle).

Conception du navigateur réactif flou :

Afin d'assurer la non collision du robot avec des obstacles, le contrôleur d'évitement d'obstacles est sollicité à chaque fois qu'un capteur (ou plus) détecte un obstacle situé à une distance inférieure à la distance de sécurité (Ds). Ce dernier génère une commande permettant d'orienter le robot dans le sens opposé à ce dernier, c'est-à-dire, le robot s'oriente toujours

dans la direction libre d'obstacles. Par exemple, si le capteur C1, C2 ou C3 détecte un obstacle proche, le robot tourne à droite. La figure (2.16) donne la représentation du schéma de contrôle utilisé. On a utilisé, comme entrées du contrôleur, les rapports des distances mesurées par les capteurs C1-C6 à la distance (Ds), par contre, les sorties du contrôleur sont la vitesse de braquage **(W)** et la vitesse d'avancement **Va**.

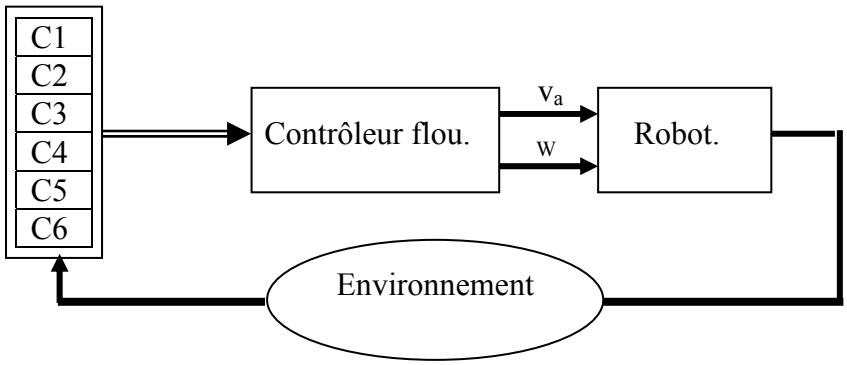

Figure 2.16 : Schéma de contrôle utilisé.

Définition des fonctions d'appartenance :

L'univers de discours des variables d'entrée du contrôleur d'évitement d'obstacles est décomposé en trois sous ensembles flous. Comme dans le premier contrôleur, ces derniers sont caractérisés par des fonctions d'appartenance triangulaires. Les nuances linguistiques utilisées pour les distances sont : Pro (proche), Moy (moyennement proche) et loin (loin) figure (2.17).

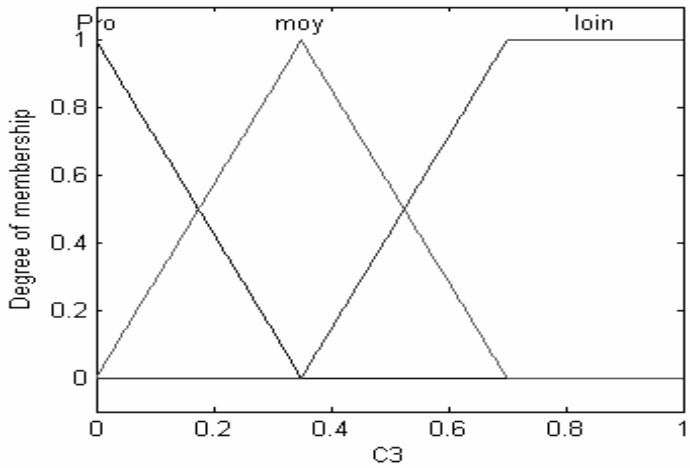

Figure 2.17 : Fonctions d'appartenance des variables d'entrée du contrôleur d'évitement d'obstacles.

Les univers de discours des variables de sortie (vitesse d'avancement Va et vitesse de braquage W), sont, à leurs tour, partitionnées respectivement en deux et trois sous ensembles flous. Les labels utilisés pour la vitesse d'avancement sont : L (lent) et Rap (rapidement) et pour la vitesse de braquage sont : Dr (droit), Z (zéro) et Ga (gauche) figure (2.18).

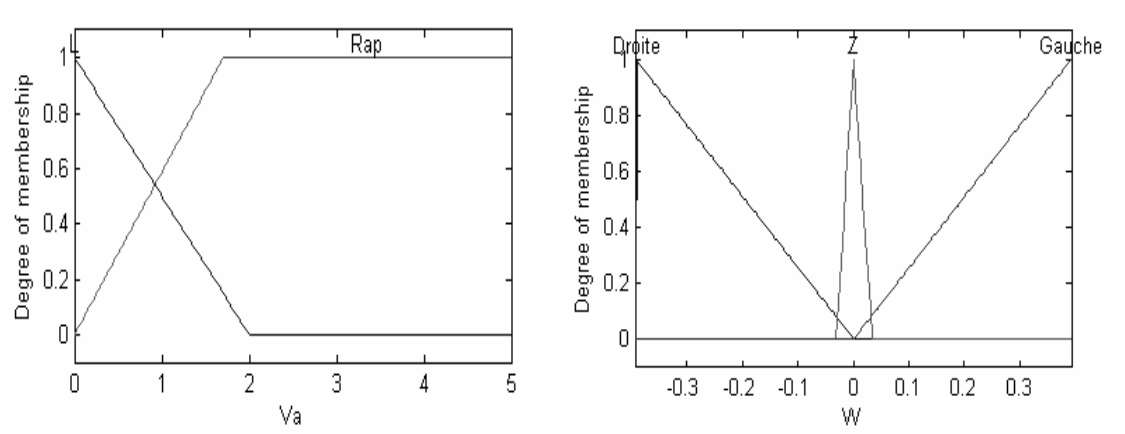

Figure 2.18 : Les fonctions d'appartenance des variables de sortie du contrôleur d'évitement d'obstacles.

Base de règles :

Pour simplifier la base de règles nous nous sommes inspirés du travail réalisé par F.CUPERTINO [Cupertino 06] où dans la partie condition, les variables d'entrée sont reliées par le « ou » au lieu de « et ». En tenant en compte de l'emplacement des capteurs (C1-C3 ; gauche et C4-C6 ; droite), nous avons obtenu une base de règles ci-dessous comprenant six règles au lieu de 276 (3^6) règles. Les règles sont de forme suivante :

Si C1 est P **ou** C2 est P **ou** C3 **est** P **alors W** est Dr **et** Va est L.

C1	C2	C3	C4	C5	C6	W	Va
P	P	P	-	-	-	Dr	L
-	M	M	-	-	-	Dr	L
L	L	L	-	-	-	Z	Rap
-	-	-	P	P	P	Ga	L
-	-	-	M	M	-	Ga	L
-	-	-	L	L	L	Z	Rap

Tableaux 2.4 : Base de règle du contrôleur d'évitement d'obstacles.

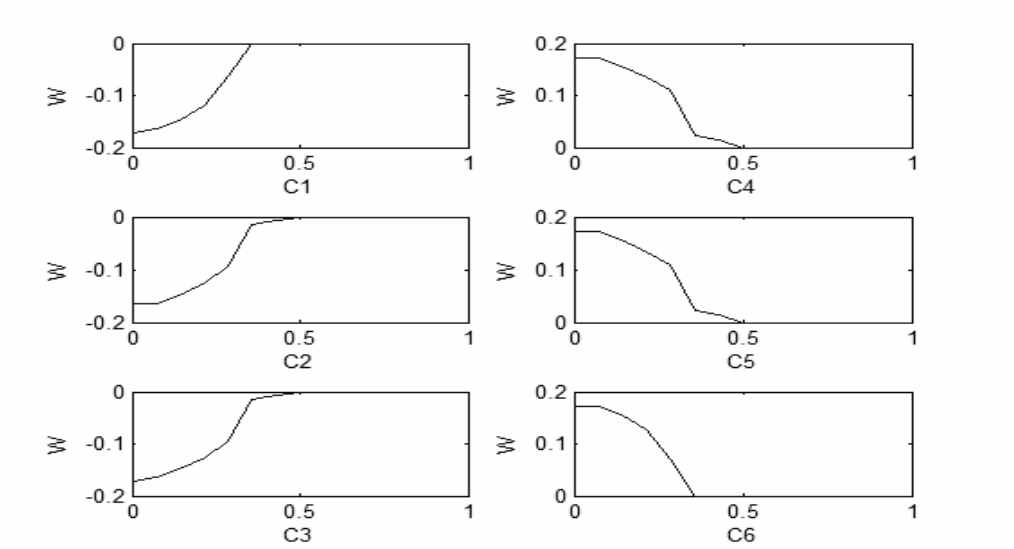

Figure 2.19 : Réponse du contrôleur en W (en radiant) en fonction des distances normalisées mesurées par les capteurs C1–C6.

Résultats de simulation :

La réponse du contrôleur en fonction des variables d'entrée C1-C6 (distances normalisées mesurées par chaque capteur) est représentée sur la (figure 2.19).

Les tests de simulation effectués, pour un temps de 2000 itérations, ont montré que le robot évolue dans l'environnement encombré où des obstacles sont éparpillés. A chaque fois qu'un obstacle est détecté par l'un des capteurs, le contrôleur d'évitement d'obstacles réagit en générant une consigne qui permet au robot d'éviter ce dernier. La trajectoire réalisée par le robot est donnée par figure (2.20).

Les résultats obtenus montrent bien que le contrôleur d'évitement d'obstacles garantit une navigation sainte ; aucune collision (figure 2.20 (a) et (b)). Sauf que, ce dernier, employé seul, ne peut garantir au robot d'atteindre une cible (figure 2.20 (b)). Dans la section qui suit, on montrera comment le robot munit des deux contrôleurs conçus auparavant peut atteindre la cible sans collision avec les obstacles.

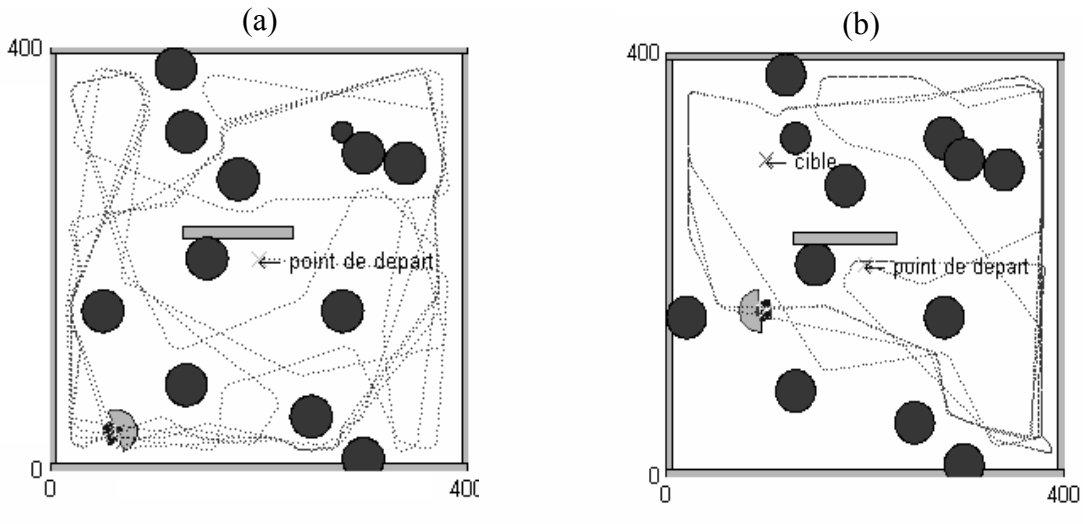

Figure (2.20) : Trajectoires réalisées par le robot.

6. Fusion de comportements

On a vu, d'après les sections précédentes, que le comportement obtenu avec le contrôleur de convergence vers la cible et celui obtenu avec le contrôleur d'évitement d'obstacles fonctionnent indépendamment. Si les deux contrôleurs sont employés ensemble, il arrive qu'à l'approche d'un obstacle, ils génèrent des consignes contradictoires. De telles situations ne peuvent être gérées qu'en fusionnant les deux comportements dictés par ces derniers afin de diriger le mouvement du robot de telle manière que l'action de convergence vers une cible n'amène pas le robot à une collision avec un obstacle, et l'action d'évitement d'obstacle ne le fait pas dévier de sa trajectoire de convergence vers la cible.

La méthode adoptée dans notre cas est l'attribution d'un gain « $g<1$ » pour les sorties du contrôleur de convergence vers la cible et un gain de « $\alpha*(1-g)$ » pour celles du contrôleur d'évitement d'obstacles, à chaque fois que la distance robot-obstacle est inférieure à la distance de sécurité Ds.

Résultats :

Pour entamer les tests, nous avons considéré dans premier temps un environnement peu encombré. Le résultat obtenu, pour Ds= 25u.l, $\alpha=2$ et $g = 0.15$, est illustré sur la (figure 2.21).

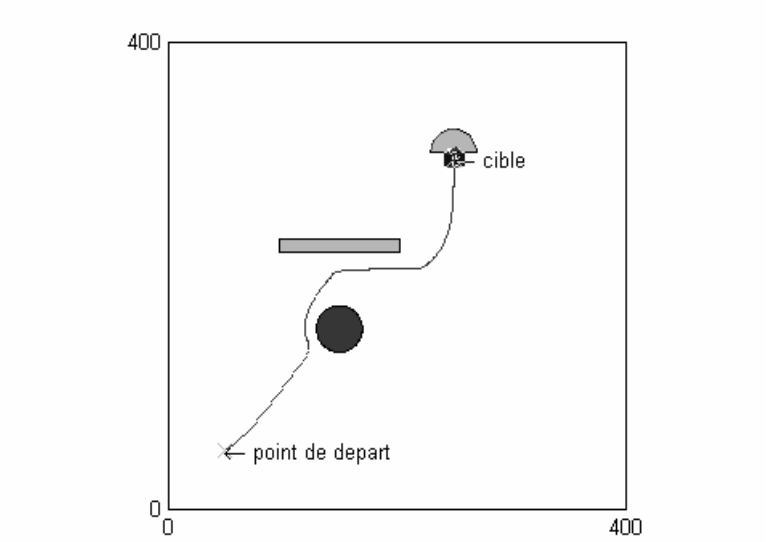

Figure 2.21 : Trajectoire réalisée par le robot dans un environnement de test.

La figure (2.22) montre les trajectoires réalisées par le robot pour l'atteinte de plusieurs cibles dans un environnement encombré. D'après ces dernières, on constate que le robot a réussi à atteindre les cibles qui lui sont désignées sans aucune collision.

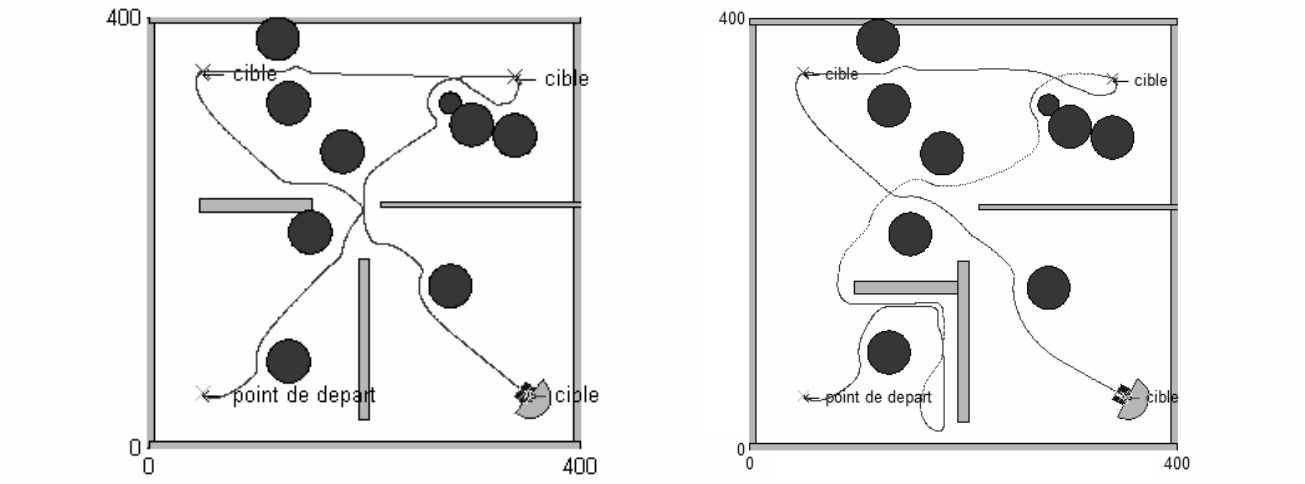

Figure 2.22 : Trajectoires réalisées par le robot dans un environnement encombré.

7. Conclusion

Dans ce chapitre, nous avons exploré le concept de navigation floue d'un robot mobile dans un environnement inconnu, et ce, par une approche comportementale à savoir : le comportement de convergence vers la cible qui assure la convergence du robot mobile vers une cible (ou plusieurs), mais sans la prise en compte des obstacles. Un second comportement d'évitement d'obstacles est introduit pour permettre au robot d'éviter les obstacles, mais sans autant pouvoir assurer l'atteinte de cibles. Enfin, une fusion des deux comportements est réalisée, par attribution de gains aux consignes des deux contrôleurs, afin d'assurer une navigation ayant comme objectif l'atteinte de cibles, tout en garantissant la non collision du robot avec les obstacles rencontrés sur son trajet.

Les résultats obtenus par cette approche sont assez satisfaisants. Mais, le temps nécessaire pour la conception des contrôleurs ainsi que la recherche des valeurs des gains optimaux pour la fusion des deux comportements (d'une manière empirique) est important.

Chapitre 3 :

Optimisation avec les algorithmes génétiques

Chapitre 3 : Optimisation avec les algorithmes génétiques

Dans ce chapitrer, nous introduirons les algorithmes génétiques afin d'améliorer les performances des contrôleurs obtenus précédemment, et ce, pour la génération du contrôleur flou de convergence vers la cible et la recherche du gain optimal pour la fusion des deux comportements (convergence vers la cible et évitement d'obstacles).

1. Introduction

Dans le chapitre précédant, les résultats obtenus par l'utilisation du contrôle flou sont assez satisfaisants. Sauf que la procédure suivie pour l'obtention des contrôleurs ainsi que pour la fusion de deux comportements est empirique. Ce qui a demandé un temps assez important pour avoir des résultats qui ne sont pas optimaux. Dans ce contexte, nous avons fait appel à un outil de recherche global et exhaustif afin d'examiner la possibilité d'automatiser la procédure de conception de contrôleurs flous et de fusion des deux comportements. L'outil choisi est l'une des techniques des algorithmes évolutionnaires ayant connu un grand succès qui est : les algorithmes génétiques (AG).

Dans la première partie de ce chapitre, nous exposerons l'utilisation des AG dans le but de la génération et optimisation du contrôleur flou assurant la convergence du robot vers une cible. Dans la seconde, la technique à laquelle nous avons fait appel, sera utilisée dans la recherche du gain optimal pour la fusion des deux comportements développés précédemment.

2. Aperçu sur les AG

Les algorithmes génétiques (AG) sont des algorithmes d'exploration fondés sur les mécanismes de la sélection naturelle et de la génétique dont les fondements théoriques ont été définis par J.H. Holland. Ils tentent de simuler le processus d'évolution naturelle suivant le modèle darwinien dans un environnement donné. C'est-à-dire, utilisent à la fois les principes de la survie des structures (individus) et les échanges d'informations pseudo-aléatoires pour former, à chaque génération, un nouvel ensemble de créatures artificielles (chaîne de caractères), en utilisant des parties des meilleurs éléments (les mieux adaptés), ainsi que des parties innovatrices à l'occasion [Goldberg94].

Les AG utilisent un vocabulaire similaire à celui de la génétique naturelle tel que : population, individu, sélection, croisement et mutation.

2.1 Concepts de base

Contrairement aux méthodes classiques d'optimisation, posant des hypothèses contraignantes sur le domaine d'exploration (continuité, dérivabilité, unimodalité…etc.) d'une fonction de coût d'un problème à optimiser, les algorithmes génétiques n'utilisent que les valeurs numériques de cette dernière et au lieu des paramètres d'origine du problème, utilisent un codage de ces derniers.

Donc pour une optimisation avec les AG, la première étape, est le codage des paramètres représentant les gènes, et pour ce, plusieurs codages existent. Les plus utilisés sont le codage binaire et le codage réel. L'ensemble des gènes est dit chromosome ou individu (point dans le domaine d'exploration ; solution potentielle). Comme nous l'avions déjà cité, les AG travaillent sur une population d'individus, générée initialement de manière aléatoire. Dans la quête de l'optimum, cette dernière subira les différentes opérations suivantes [Homaifar95], [Goldberg94] :

-*Evaluation* : un algorithme d'optimisation nécessite, généralement, la définition d'une fonction rendant compte de la pertinence des solutions potentielles, à partir des grandeurs à optimiser : c'est la fonction d'adaptation. Dans un AG, on travail toujours sur des fonctions à maximiser. Dans le cas où nous avons une optimisation d'une fonction de coût au sens de minimisation, on forme la fonction d'adaptation composée de la fonction à minimiser, combinée avec une autre fonction de sorte à revenir à un problème de maximisation.

-*Sélection* : consiste à faire un tri et des copies des individus, et ce, par ordre de satisfaction de la *fonction d'adaptation* ; plus la valeur de cette dernière, associée à un individu, est grande, plus l'individu a de chances d'être élu et multiplié pour participer dans la formation de la génération suivante. La méthode la plus utilisée est la sélection proportionnelle. Comme son nom l'indique, dans cette méthode, on attribue à chaque individu une probabilité de sélection proportionnelle à sa performance sur une roulette de loterie. Les individus sélectionnés, dits parents, subiront les deux opérations génétiques suivantes :

-*Croisement :* est la principale opération agissant sur la population des parents. Elle consiste à un échange des parties de chaînes entre deux individus sélectionnés (parents) pour former deux nouveaux individus (enfants). Cet échange peut être fait, soit, à un seul point, ou sur plusieurs points. La figure 3.1 est un exemple pour un codage binaire.

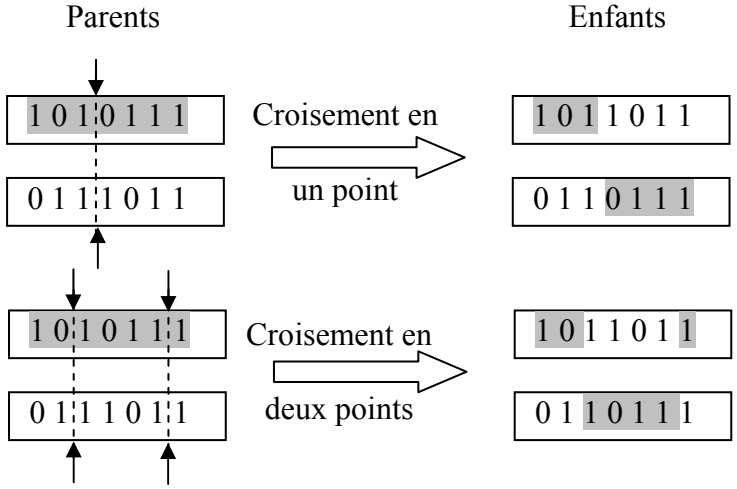

Figure 3.1 : Représentation schématique d'un croisement.

-*Mutation :* contrairement à l'opération précédente, la mutation opère sur un seul individu en modifiant d'une manière aléatoire, une partie de ce dernier. Dans le cas de codage binaire, cela est fait en inversant un ou plusieurs bits dans un chromosome (figure 3.2). D'autres méthodes peuvent être utilisées comme la mutation par soustraction fixée sur un gène ou le remplacer par une valeur aléatoire choisie dans un sous ensemble de valeurs. A la différence du croisement, le taux de mutation est généralement très faible et se situe entre 0.5% et 1% de la population totale. Ce faible taux permet d'éviter une dispersion aléatoire de la population et n'entraîne que quelques modifications sur un nombre limité d'individus assurant la diversité de la population. La mutation prend une

place de plus en plus importante dans les algorithmes génétiques, alors qu'il y a encore quelques années son rôle était encore considéré comme accessoire [De Falco02].

Les mécanismes d'AG sont résumés dans l'organigramme de figure 3.3.

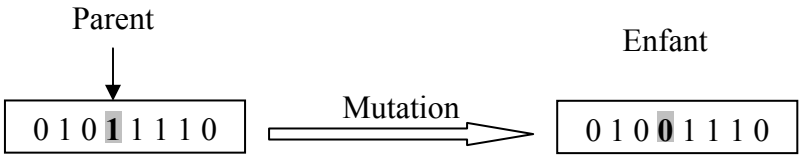

Figure 3.2 : Représentation schématique d'une mutation.

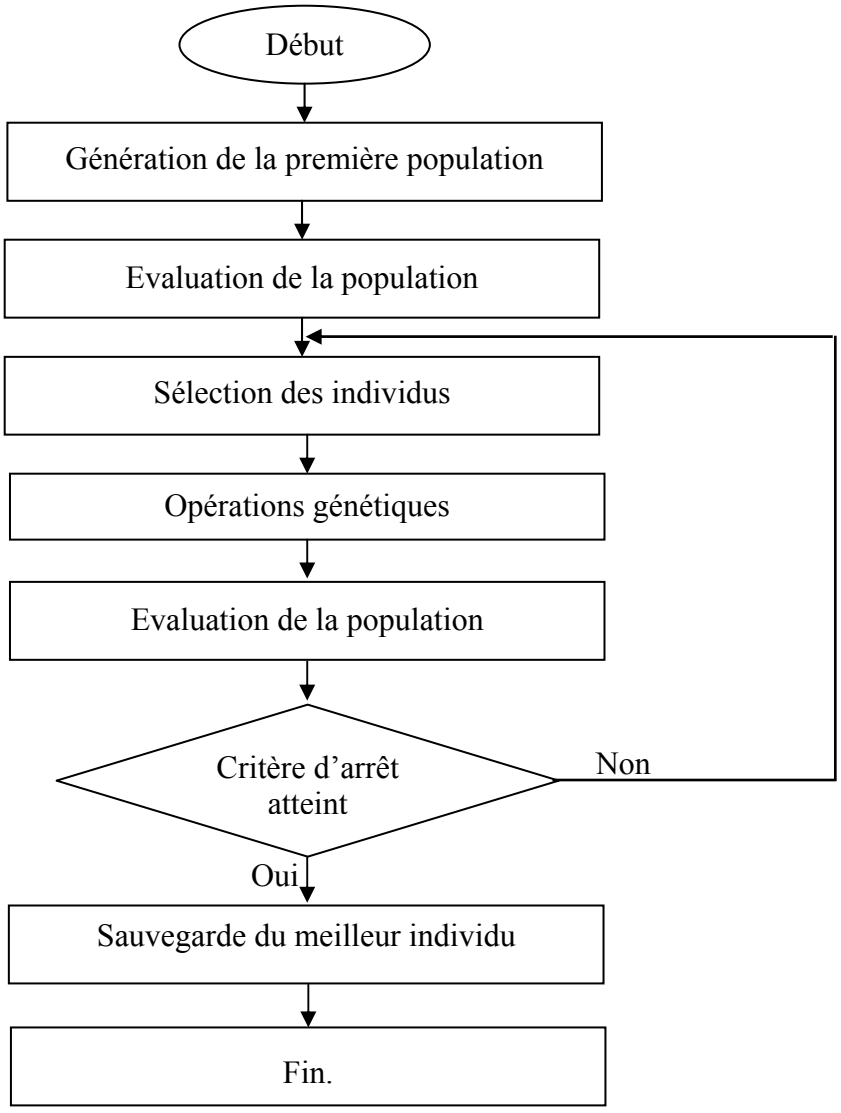

Figure 3.3 : Organigramme d'algorithme génétique.

3. Conception d'un contrôleur flou avec un algorithme génétique

Les difficultés rencontrées dans le réglage empirique du contrôleur flou, nous ont fait penser à l'utilisation d'une méthode d'optimisation automatique telle les AG. L'idée de génération et d'optimisation de contrôleurs flous par les AG est assez ancienne. Dans la littérature, on peut rencontrer deux stratégies d'application des AG simples pour la conception de contrôleurs flous [KARR93] [Homaifar95] [SIMON95] [Russo 98] :

1. Dans cette première stratégie, les fonctions d'appartenance sont prédéfinies par un expert et la base de règles est optimisée par un algorithme génétique.

2. La base de règles est prédéfinie par un expert et les fonctions d'appartenance sont générées par un algorithme génétique. Là aussi, plusieurs possibilités sont envisageables : soit seulement les centres des fonctions d'appartenance qui sont optimisés et les largeurs sont fixes, soit leurs largeurs, ou bien les deux en même temps [Fan03]. Dans le cas de fonctions ne présentant pas de symétrie, l'optimisation est faite sur les fonctions toutes entières.

Dans notre travail nous avons adopté la seconde technique pour générer le contrôleur flou de *convergence vers la cible*, en utilisant la base de règles obtenue dans le chapitre précédant.

La suite du chapitre expose la démarche suivie pour la réalisation de cet objectif.

3.1 Principe

Notre démarche pour l'apprentissage est que, chaque individu de la génération en cours est implanté comme contrôleur au robot et laissé naviguer en vu d'atteindre la cible. Cette navigation cesse pour deux conditions : soit le robot à atteint la cible, ou la durée maximale, qui lui est alloué, a écoulé. Dans notre cas, cette dernière est fixée à 1000 itérations. Enfin, l'individu est sanctionné par une valeur de la fonction d'adaptation (équation 3.1), nous renseignant sur le degré d'efficacité de ce dernier.

Le critère d'arrêt utilisé est le nombre de génération (fixé à 50) et le schéma global de l'apprentissage est rapporté sur la figure 3.4.

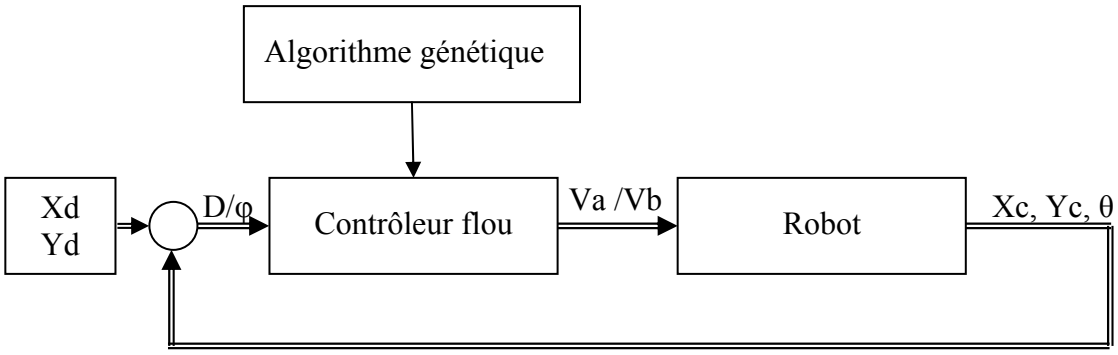

Figure 3.4 : Structure d'apprentissage utilisée.

3.2 Structure du chromosome

Les fonctions d'appartenance et les univers de discoure des variables utilisées sont donnés dans le chapitre 2. Chaque fonction est caractérisée par trois points modaux (a_{ia}, a_{ib}, a_{ic}). Donc générer les fonctions d'appartenance d'une variable, revient à générer les trois points modaux les caractérisant (figure 3.6).

Pour s'assurer que le contrôleur généré ait un comportement qui *n'est ni gauchiste ni droitier* (c'est-à-dire, pour des orientations symétriques), le robot doit avoir des comportements similaires. Nous avant exploité la propriété de *symétrie* pour la génération des points modaux des fonctions d'appartenance des deux variables « φ_{cible} » (erreur d'orientation du robot) et la variable de sortie « Vb » (vitesse de braquage). Durant l'apprentissage, on génère juste les points modaux de la partie positive des variables. Pour l'implantation du contrôleur, on transpose les paramètres pour compléter l'ensemble. Cette technique permet une réduction du temps de calcul et économise l'espace mémoire consommé.

Dans le souci d'éviter des fonctions d'appartenance non interprétables en termes linguistiques ou des zones du domaine de variation des variables d'entrée/sortie non couvertes (figure 3.5 (a)), un réarrangement des points modaux est procédé avant leurs implantation.

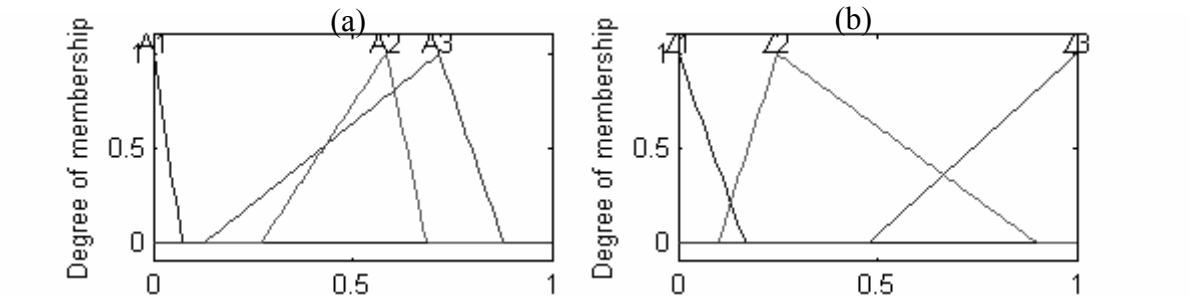

Figure 3.5 : Exemple (a) d'une mauvaise et (b) d'une bonne reprtition des sous ensembles flous

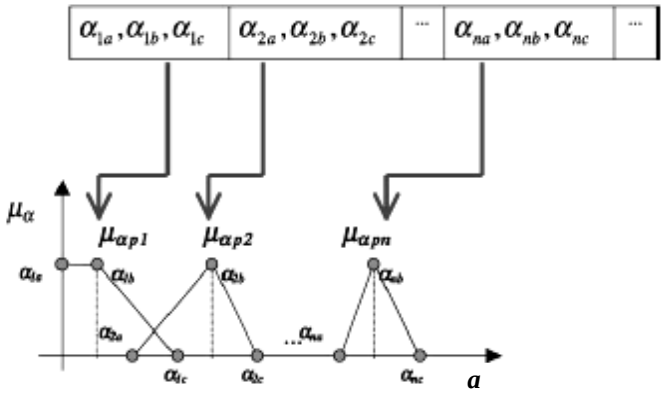

Figure 3.6 : Implantation d'un chromosome.

Dans notre cas, le contrôleur flou utilisé est à deux variables d'entrée et deux variables de sortie, le codage des leurs paramètres donne la forme de chromosomes illustrée par la figure 3.7.

Paramètres de variable d'entrée φ. Paramètres de la variable d'entrée D.

α_{1a}	α_{1b}	α_{1c}	…	α_{ma}	α_{mb}	α_{mc}	β_{1a}	β_{1b}	β_{1c}	…	β_{na}	β_{nb}	β_{nc}

Paramètres de la variable de sortie Vb. Paramètres de la variable de sortie Va.

γ_{1a}	γ_{1b}	γ_{1c}	…	γ_{pa}	γ_{pb}	γ_{pc}	δ_{1a}	δ_{1b}	δ_{1c}	…	δ_{qa}	δ_{qb}	δ_{qc}

Figure 3.7 : Structure du chromosome

Où :

$\alpha_{ia,b,c \quad i=\overline{1,m}}$: Paramètres de la i^{eme} fonction d'appartenance de l'erreur d'orientation du robot (φ).

$\beta_{ia,b,c \quad i=\overline{1,n}}$: Paramètres de la i^{eme} fonction d'appartenance de la distance robot-cible (D).

$\gamma_{ia,b,c \quad i=\overline{1,p}}$: Paramètres de la i^{eme} fonction d'appartenance de la vitesse de braquage (Vb).

$\delta_{ia,b,c \quad i=\overline{1,q}}$: Paramètres de la i^{eme} fonction d'appartenance de la vitesse d'avancement (Va).

m, n, p et q sont les nombres de sous ensembles flous des variables d'entrée et de sortie.

L'univers de discours de la variable φ est partitionné en cinq sous ensembles flous. Ce qui correspond à 15 poids modaux, avec l'exploitation de la caractéristique de symétrie et en

fixant les poids modaux des extrémités de variation (0, $\varphi_{max}=\pi$), on obtient m=6. De même pour Vb, ce qui donne n=6.

Pour les deux variables D et Va, l'univers de discours est partitionné en trois sous ensembles flous. En fixant les poids modaux des limites minimales et maximales (0, D_{max}) et (0, Va_{max}) on obtient p=q=6. Ce qui donne des chromosomes de 24 gènes.

3.3 Fonction d'adaptation

La fonction d'adaptation est la clé de la réussite d'un algorithme génétique. Puisque c'est dans cette dernière, que sont exprimés les objectifs à atteindre après apprentissage. Sa formulation exprime, sous forme mathématique, les objectifs que l'on souhaite atteindre [Acosta03] [Homaifar95] [Belarbi 00].

Dans notre cas, le contrôleur à concevoir doit répondre aux exigences suivantes :

1. Le robot doit s'orienter, d'abord, vers la cible puis avancer vers elle.
2. Atteindre la cible en empruntant le chemin le plus court possible. Plusieurs critères peuvent être formulés, en distance, en temps ou les deux en même temps. Nous avons retenu le dernier.
3. Les consignes des deux vitesses ne doivent pas contenir de variations brusques (grandes accélérations).

La fonction d'adaptation formulée est :

$$f = \lambda \exp(-\sigma[\alpha_1 \times D + \sum_{i=1}^{N}(\alpha_2 \times \varphi(i)^2 + \alpha_3 \times \left(\frac{d}{dt}Va(i)\right)^2 + \alpha_4 \times \left(\frac{d}{dt}Vb(i)\right)^2 + \alpha_5 \times Va(i) \times Vb(i)) + \alpha_6 \times T]) \quad (3.1)$$

Avec :

λ =10 et σ =10^{-4}

$\alpha_{1..6}$: coefficients de pondération.

D : distance totale parcourue par le robot pour atteindre la cible.

φ(i) : erreur d'orientation à l'itération i.

Va et Vb : vitesse d'avancement et vitesse de braquage du robot.

T : le temps total mis par le robot pour atteindre la cible.

3.4 Paramètres de l'AG utilisé

Le tableau 3.1 récapitule les paramètres choisis de l'algorithme génétique.

Paramètres	valeur
Codage	Binaire
Taille de la population	30 individus
Croisement	Plusieurs points
Probabilité de croisement	0.70
Mutation	Plusieurs points
Probabilité de mutation	0.01
Sélection	Roue de loterie
Nombre de générations	50

Tableau 3.1 : Paramètres de l'algorithme génétique.

Résultats de simulation

La figure 3.8 représente les trajectoires réalisées par les individus de la première génération, pour un point de départ de coordonnées (350,80), en vu d'atteindre la cible de coordonnées (50,350).

Comme on peut le voir, la majorité des contrôleurs, générés de manière aléatoire, ont une tendance à faire converger le robot vers la cible ; dû au fait que la base de règle est prédéfinit. Sauf que la répartition des sous-ensembles flous est quelconque, ce qui génère des trajectoires non optimales.

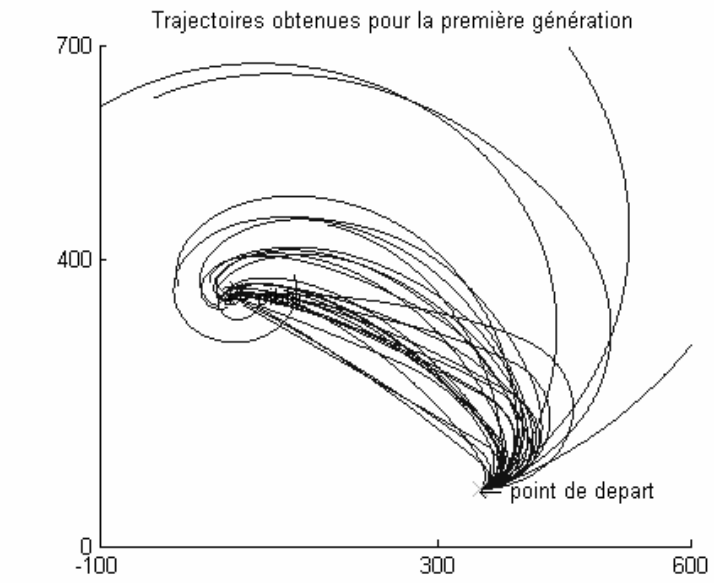

Figure 3.8 : Trajectoires réalisées par les individus de la première génération.

La figure 3.9 représente les fonctions d'appartenances des variables d'entrée/sortie du meilleur contrôleur de cette génération et la trajectoire réalisée par ce dernier est rapportée sur la figure 3.10. Les graphes de variations des différentes grandeurs en fonction du temps pour réaliser cette trajectoire, sont représentés sur la figure 3.11.

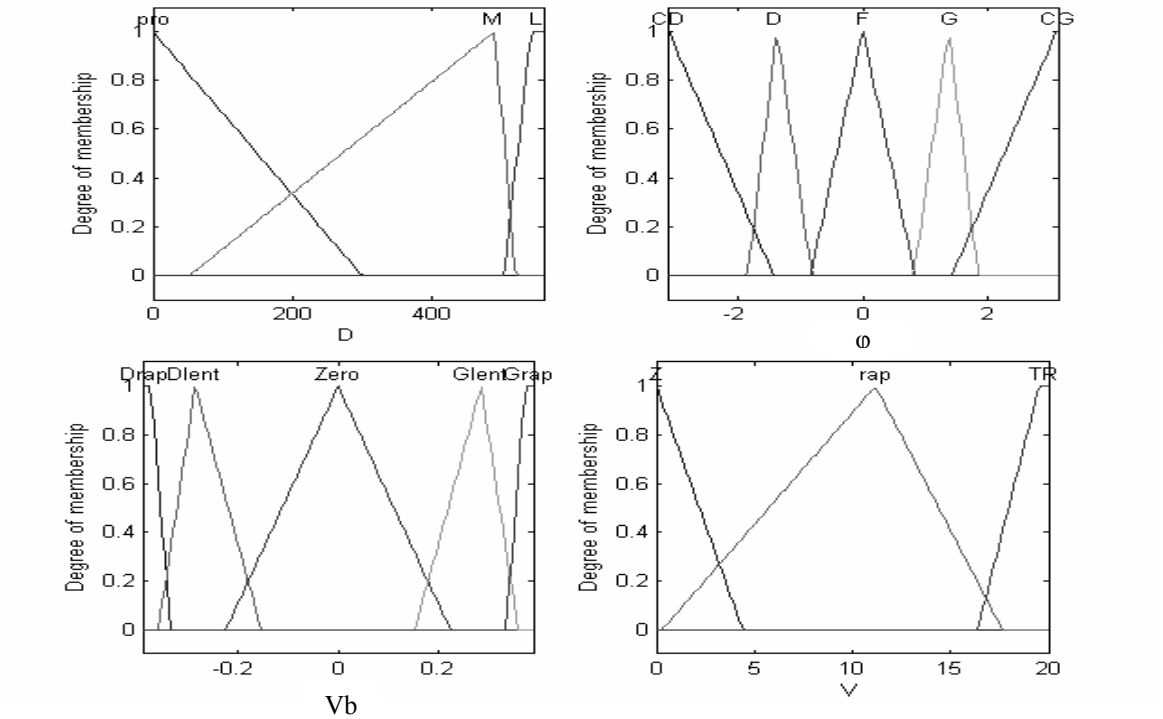

Figure 3.9 : Fonctions d'appartenance du meilleur contrôleur obtenu pour la première génération.

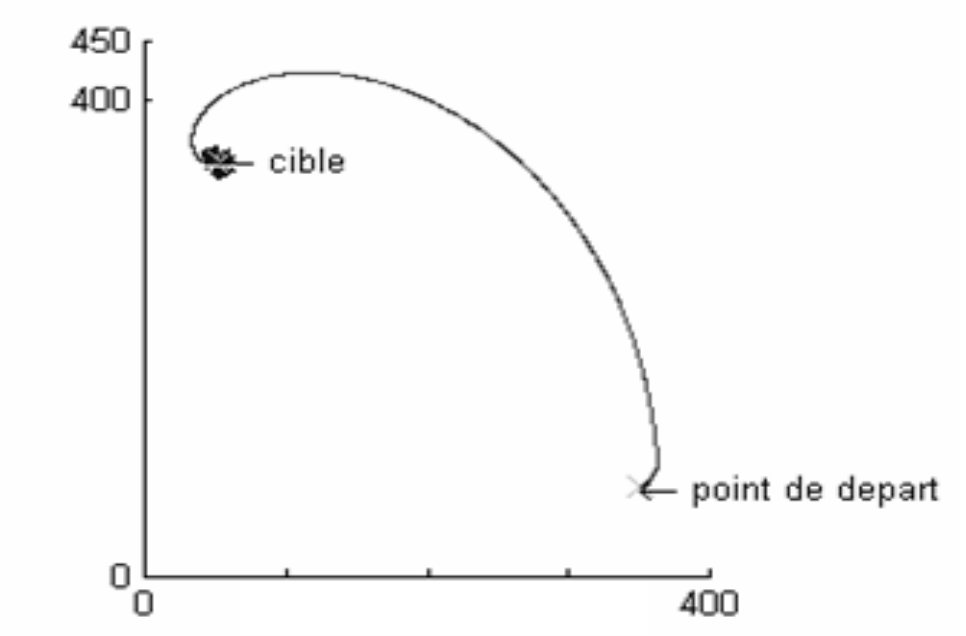

Figure 3.10 : Trajectoire réalisée par le meilleur individu de la première génération.

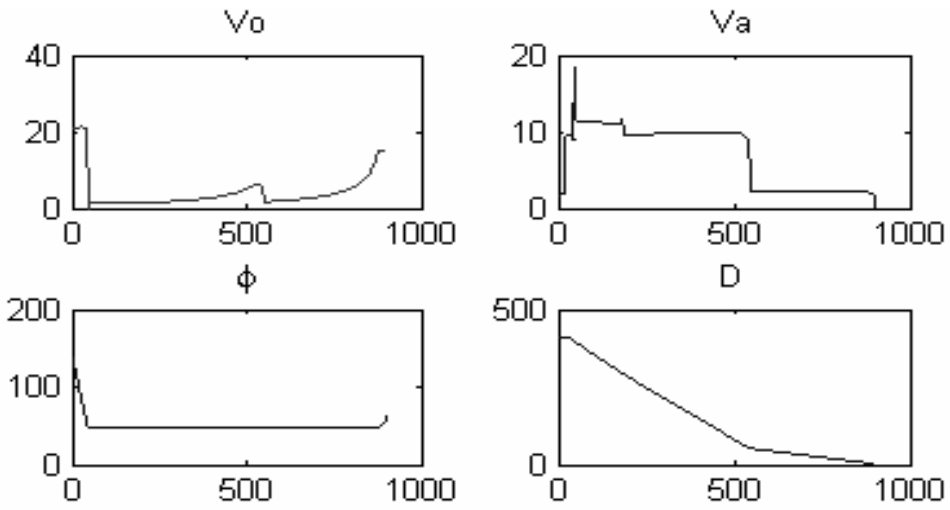

Figure 3.11 : Variations de Vb, Va, φ et D en fonction du temps

Le graphe d'évolution de la valeur de la fonction d'adaptation pour les meilleurs individus le long des générations, ainsi que, ses valeurs pour les individus de la dernière génération, sont rapportés sur les graphes de la figure 3.12.

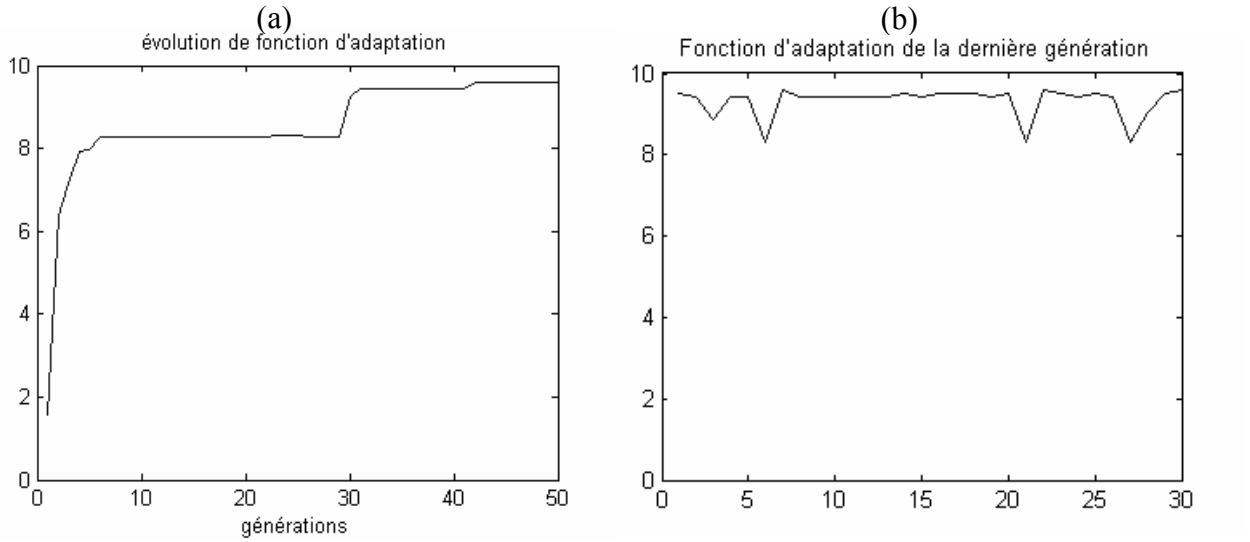

Figure 3.12 : Evolution de la valeur de la fonction d'adaptation (a) des meilleurs individus le long des générations, (b) valeurs de cette dernière pour la dernière génération.

Contrôleur flou obtenu après apprentissage

La figure 3.13 représente les fonctions d'appartenance des variables d'entrée/sortie obtenues du meilleur individu après apprentissage.

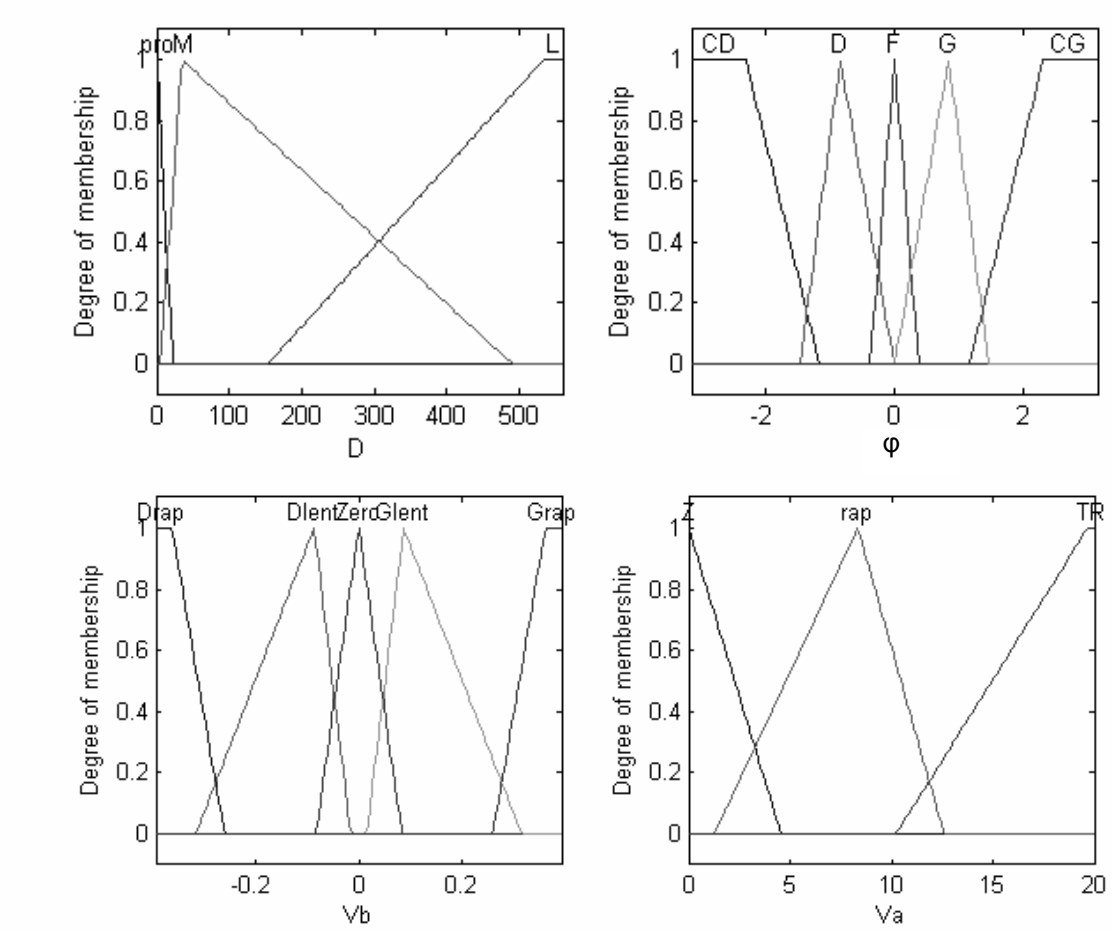

Figure 3.13 : Fonctions d'appartenance des variables d'entrée/sortie.

Les surfaces, représentant les réponses du contrôleur en fonction des entrées, sont illustrées par les graphes de la figure3.14.

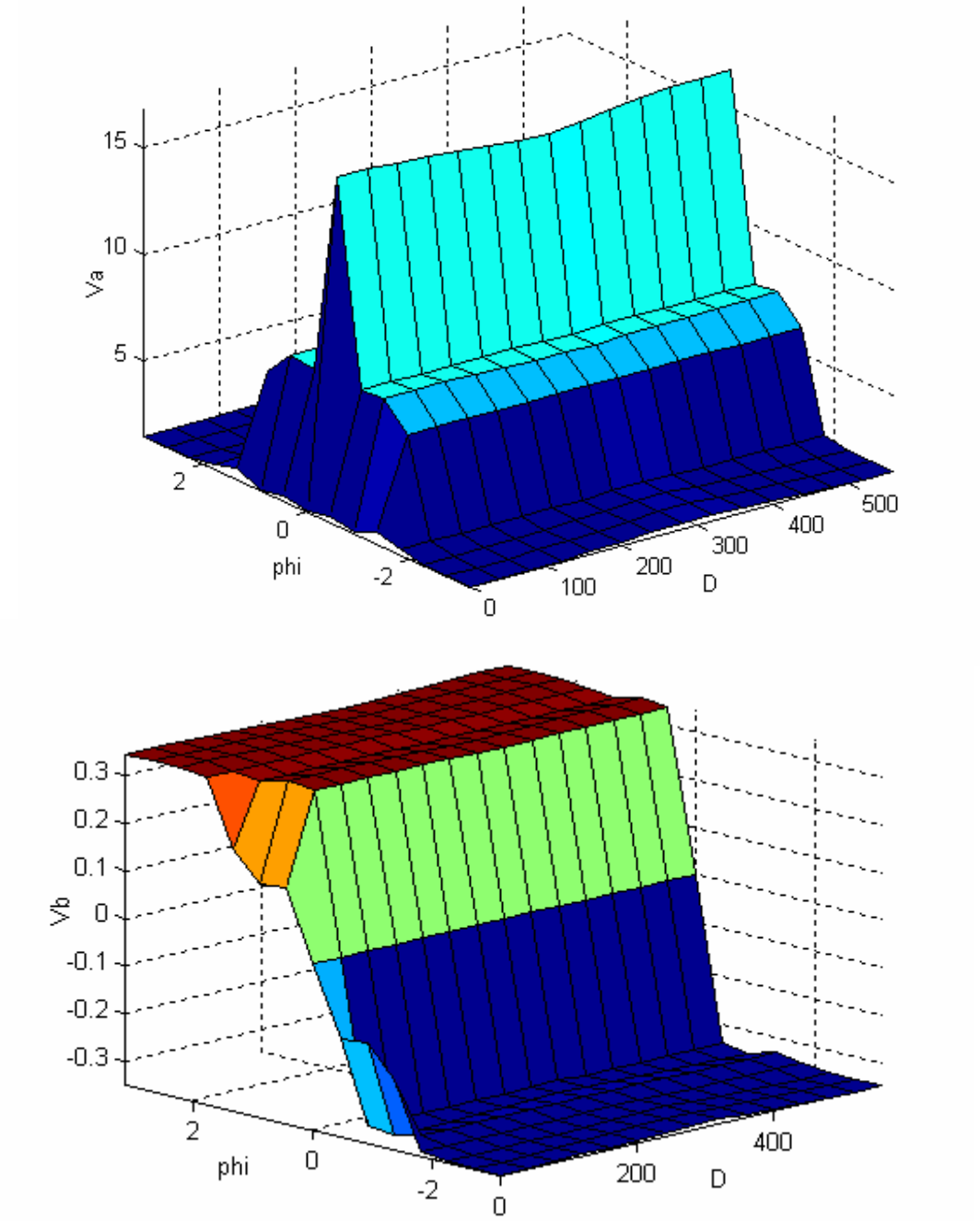

Figure 3.14 : Surfaces du contrôleur obtenu après apprentissage.

La trajectoire réalisée par cet individu est illustrée sur la figure 3.15 et les variations de la vitesse de braquage, de la vitesse d'avancement, d'erreur d'orientation et de la distance robot-cible en fonction du temps sont rapportées sur figure 3.16.

Comme on peut le constater, le robot satisfait les contraintes citées ci-haut et adopte bien le comportement souhaité ; s'orienter d'abord vers cible, tout en accélérant son allure

avec une vitesse inversement proportionnelle à l'erreur d'orientation. Une fois le robot est bien aligné à la cible, il converge vers elle avec une vitesse proportionnelle à la distance le séparant de cette dernière et s'arrête une foi la cible atteinte, et tous cela sans générer de grandes accélérations.

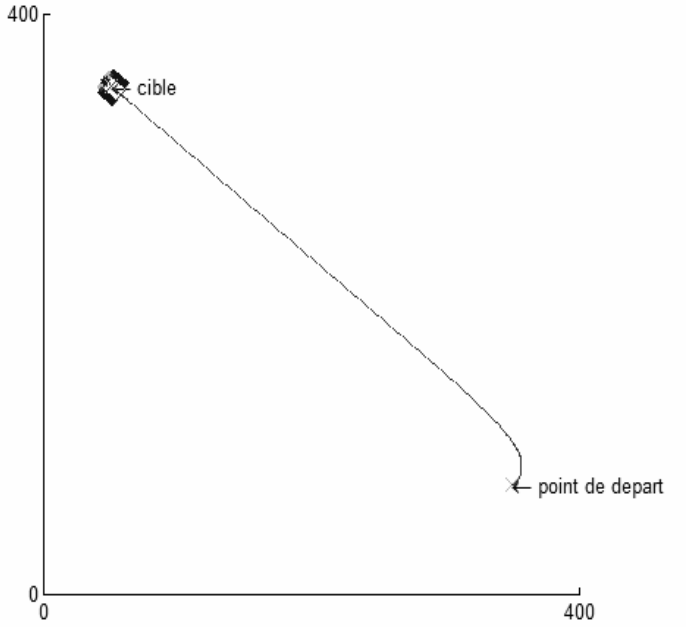

Figure 3.15 : Trajectoire réalisée par le robot après apprentissage.

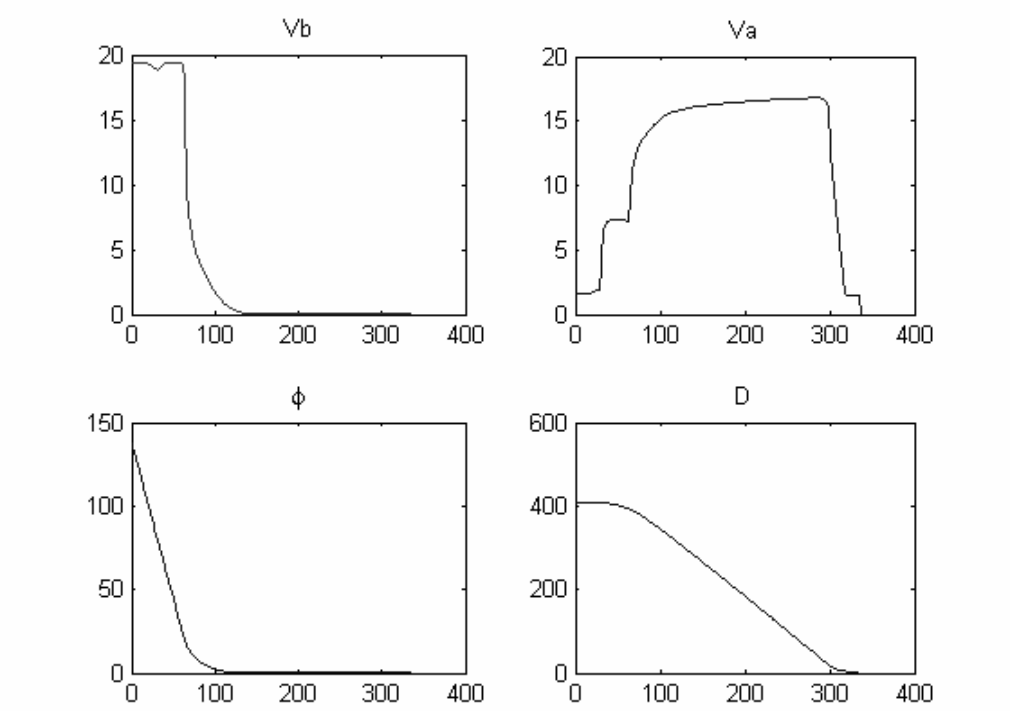

Figure 3.16 : Variation de Vb, Va, φ et D en fonction de temps.

Test pour le cas plusieurs cibles

Les figures 3.17-18 illustrent le comportement du robot dans le cas de plusieurs cibles. Ce test permet d'avoir le nombre maximal d'état de figure du robot par rapport à une cible, en vu d'exciter le contrôleur dans tous les modes de fonctionnement possible.

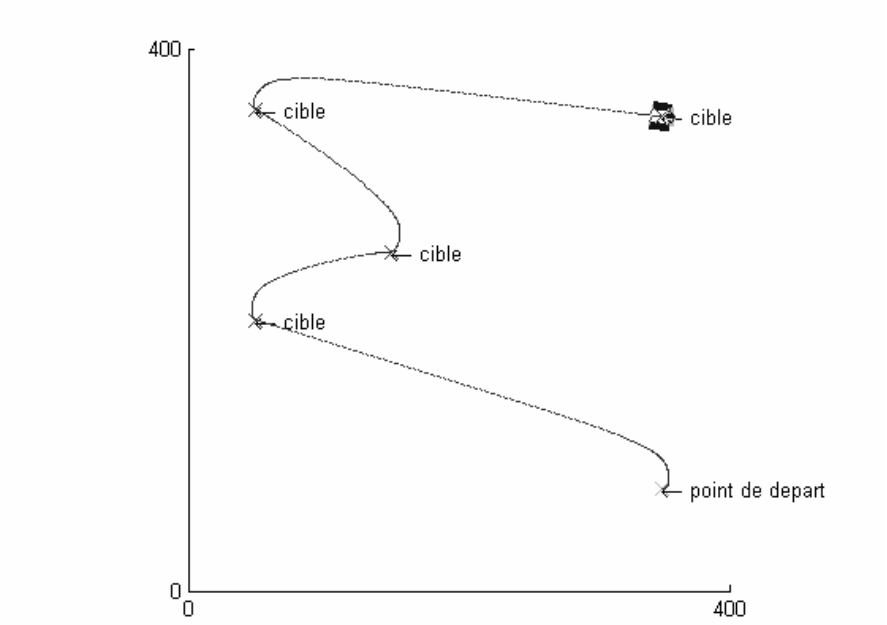

Figure 3.17 : Trajectoire réalisé par le robot pour plusieurs cibles.

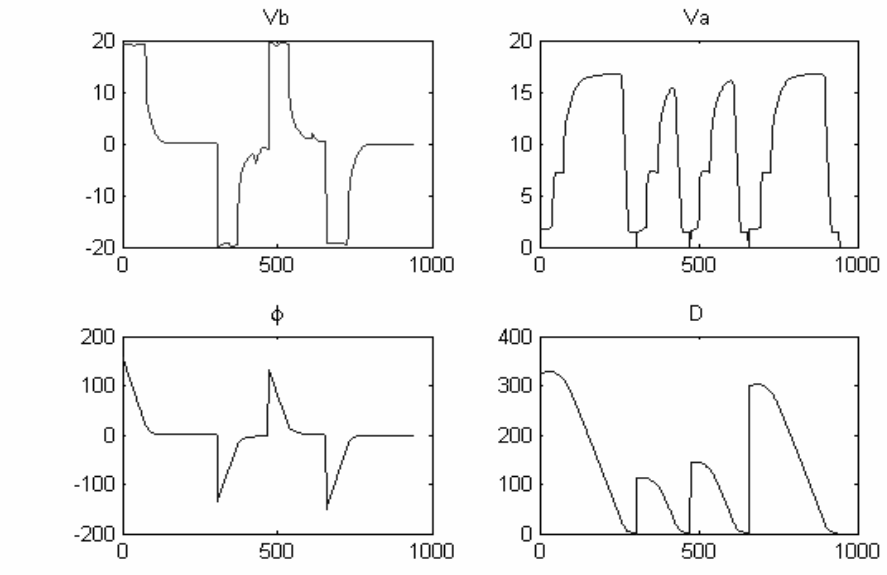

Figure 3.18 : Variations de Vb, de Va, φ et de D en fonction du temps.

4. Fusion des comportements

La méthode de fusion des deux comportements est la même que celle utilisée dans le chapitre précédent, mais en assignant, cette fois-ci, la tache de recherche d'un seul gain, *g,* à assigner aux consignes du contrôleur de convergence vers la cible, à un AG. Dans cette partie, nous aborderons la problématique de fusion des deux comportements du robot ; *convergence vers la cible* et *évitement d'obstacles*, en utilisant un AG permettant la recherche du gain « *g* » optimal dans le sens où la valeur de ce dernier, permettra la *non collision* du robot avec les obstacles présents dans l'environnement sans autant le faire trop éloigné de sorte à réaliser une *trajectoire trop longue.*

Pour ramener le problème de fusion de deux comportements sous forme d'une fonction de coût à maximiser, en introduisant les deux objectifs cités précédemment, nous avons formulé la fonction ci-dessous :

$$f = \lambda \exp\left[-\sigma \times \left(\alpha_1 \times D + \frac{\alpha_2}{d_{\min}} + \alpha_3 \times SC \right) \right] \tag{3.2}$$

Où : $\lambda = 10$, $\sigma = 10^{-3}$

α_1, α_2, α_3 : Coefficient de pondération.

D : distance totale parcourue par le robot pour atteindre la cible.

SC : variable logique indiquant s'il y a eu collision.

d_{min} : la distance minimale entre le robot et les obstacles.

Les paramètres adoptés de l'AG implanté sont récapitulés dans le tableau 3.2.

Paramètres	valeur
Codage	Binaire
Taille de la population	10 individus
Croisement	deux points
Probabilité de croisement	0.70
Mutation	un point
Probabilité de mutation	0.02
Sélection	Roue de loterie
Nombre de générations	10

Tableau 3.2 : Paramètres de l'algorithme génétique utilisés.

Résultats et commentaires

Les trajectoires réalisées par les individus de la première génération sont rapportées sur la figure 3.19. Comme on peut le remarquer, aucun individu de cette génération n'a atteint la cible ; dû aux valeurs trop grandes du gain *g*.

L'évolution de la valeur de la fonction d'adaptation est rapportée sur le graphe de la figure 3.20. Comme on peut le remarquer, la situation précédente est restée telle jusqu'à la cinquième génération, à partir de laquelle l'AG produit des individus atteignant la cible.

La trajectoire réalisée par le meilleur individu après apprentissage, correspondant à g=0.047, est rapportée sur la figure 3.21.

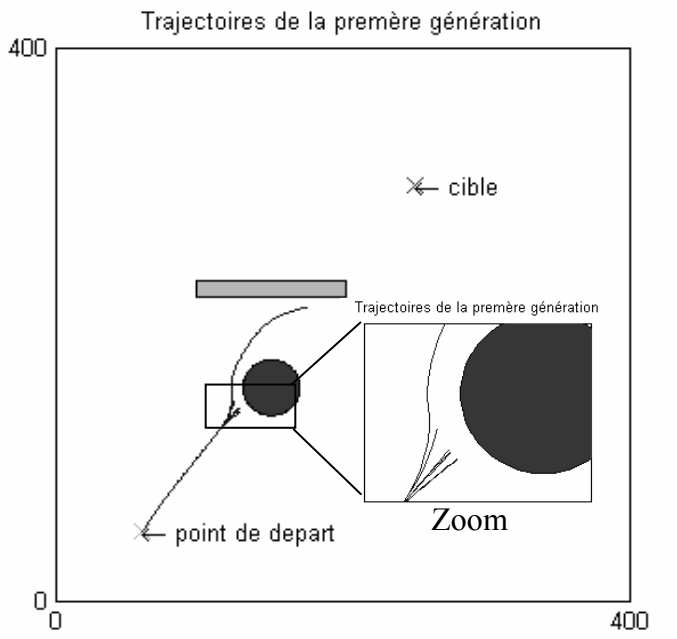

Figure 3.19 : Trajectoires réalisées par les individus de la première génération

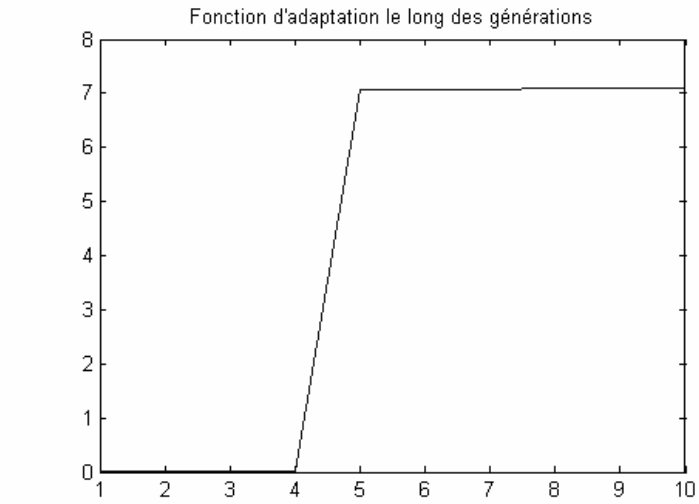

Figure 3.20 : Evolution de la fonction d'appartenance le long des générations

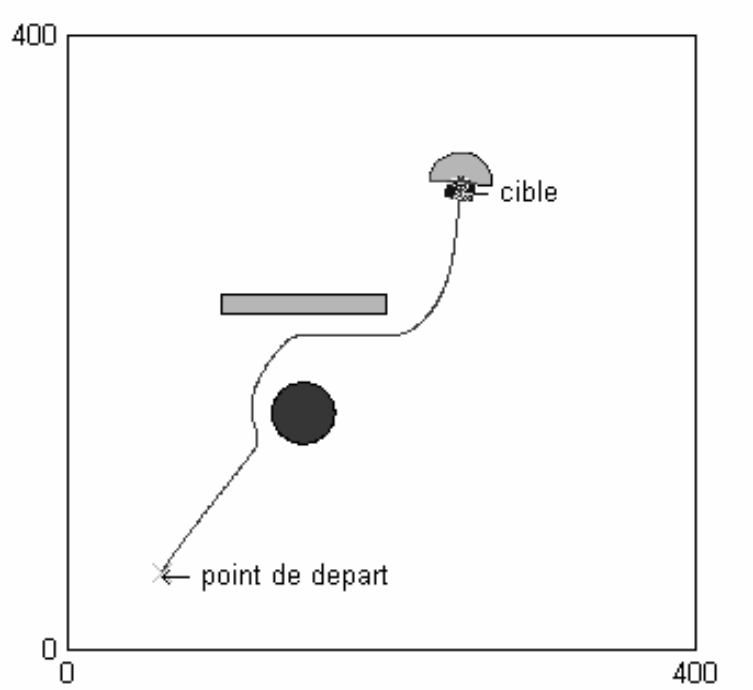

Figure 3.21 : Trajectoire obtenue après apprentissage (g = 0.047).

Cas d'un environnement encombré

Pour évaluer le résultat obtenu par apprentissage, nous avons testé ce dernier dans deux environnements encombrés. Les trajectoires réalisées sont rapportées sur la figure 3.22.

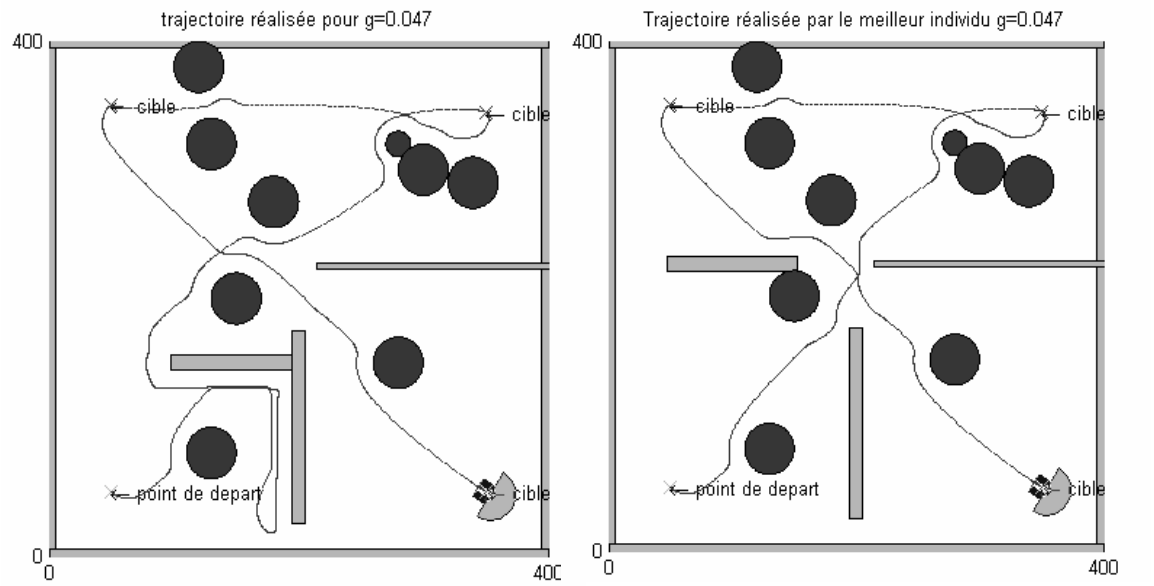

Figure 3.22 : Test dans un environnement encombré.

Comme on peut le voir, le robot navigue et atteint les cibles qui lui sont désignées sans aucune collision avec les obstacles.

5. Conclusion

Dans ce chapitre nous avons exploré l'application du contrôle flou optimisé avec les AG pour la navigation autonome d'un robot mobile dans un environnement inconnu. Dans la première partie, nous avons exploité la faculté d'exploration globale et exhaustive des AG pour l'optimisation du contrôleur de convergence vers la cible. Les résultats obtenus sont meilleurs que celles obtenus dans la chapitre 2, où la conception du contrôleur est faite par notre expertise de manière empirique. Dans la seconde partie, un AG est chargé de la recherche d'un gain de pondération des consignes issues de contrôleur de convergence vers la cible et celle du contrôleur d'évitement d'obstacles.

Conclusion générale

Conclusion générale

Le travail présenté dans ce mémoire porte sur la navigation réactive floue, basée sur l'approche comportementale, d'un robot mobile dans un environnement inconnu.

La première partie concerne la conception de deux contrôleurs flous permettant la simulation de deux comportements indispensables pour une navigation réactive d'un robot mobile : le comportement de convergence vers une cible et le comportement d'évitement d'obstacles. Dans le premier test, le comportement de convergence vers la cible a été simulé en considérant un environnement sans obstacles. Les résultats obtenus sont assez satisfaisants. Sauf que la nature de l'environnement considéré est loin d'être réelle.

Dans le souci de simuler le comportement du robot dans un environnement proche du réel, un deuxième test a été effectué. Dans ce dernier, l'environnement considéré comporte des obstacles de formes circulaire et carrée. La détection de ces obstacles est assurée par un module de perception. Ce dernier transmet les états des capteurs au contrôleur flou dit d'évitement d'obstacles. Ce dernier génère les deux sorties (vitesse de braquage et vitesse d'avancement) permettant de dévier le robot des obstacles rencontrés. Ce second comportement, utilisé seul, garantit la non collision de robot avec les obstacles rencontrés sur sa trajectoire, mais sans pouvoir autant garantir l'atteinte d'une cible. Un module, exploitant les données issues des capteurs et la position du robot par rapport à la cible, est alors implanté assurant la fusion des ces deux comportements. Les résultats obtenus montrent que le robot atteint les cibles sans collision avec les obstacles.

Dans la seconde partie de notre travail, nous avons fait appel aux algorithmes génétiques (AG) pour un apprentissage automatique et l'optimisation des résultats. Dans un premier temps, Un AG est utilisé pour l'automatisation de la conception du contrôleur de convergence vers la cible. La base de règles est conçue d'une manière empirique et les sous-ensembles flous sont générés et optimisés par l'AG. Enfin, un AG est utilisé pour la fusion des deux comportements (convergence vers une cible et évitement d'obstacles), par la recherche du gain optimal assurant la non collision et une trajectoire optimale.

Comme perspectives, l'étude proposée peut être complétée pour améliorer les performances de l'architecture adoptée en lui intégrant d'autres modules assurant l'intégration de comportements de plus haut niveau tel que :

- Perception visuelle
- Evaluation de l'environnement
- Elaboration de cartes, pouvant servir pour planifier des missions futures ou caractérisant la dynamique de l'environnement.

Bibliographie

[Acosta 03] Gerardo Acosta, Elias Todorovich : "Genetic algorithms and fuzzy control: a practical synergism for industrial applications", computers in industry V52 p183-195, 2003.

[Aoughellanet 06] Saïd Aoughellanet : "Planification de trajectoire et commande d'un bras manipulateur thèse de doctorat", thèse de doctorat en électronique, université FERHAT ABAS –SETIF 2006.

[Belarbi 00] Khaled Belarbi and Faouzi Titel :"Genetic algorithm for the design of a class of fuzzy controllers: An alternative approach", IEEE transactions on fuzzy systems, vol. 8, no. 4, P 398-405, 2000.

[Brooks 86] Rodney A. Brooks: "A layered control system for a mobile robot", IEEE JN of robotics and automation, vol. RA-2 n°1,P14-23, 1986.

[Brooks 91] Rodney A. Brooks : "Intelligence without representation", JN Artificial Intelligence 47: 139–160, 1991.

[Chen 03] Liang-Hsuan, Chen-Hsiung Chiang : "New approach to intelligent control systems with self-exploration process", IEEE, Tans. On systems man. and cybernetics-part-B, vol. 33, no 1, P56-66, February 2003.

[Chen 04] Liang-Hsuan, Chen-Hsiung Chiang :"An intelligent control system with a multi-objective self-exploration process", Fuzzy set and systems V143, P 275-294, 2004.

[Cuesta 03] F. Cuesta, A. Ollero, B.C. Arrue, R. Braunstingl : "Intelligent control of nonholonomic mobile robots with fuzzy perception" , Fuzzy Sets and Systems V134, P47–64, 2003.

[Cupertino 06] F. Cupertino, V. Giordano,D Naso:" Fuzzy control of a mobile robot", IEEE robotics & automation magazine, P74-81, December 2006.

[De Falco 02] I. De Falco, A. Della Cioppa, E.: "Tarantino Mutation-based genetic algorithm: performance evaluation", Applied Soft Computing 1, P 285–299, 2002

[Fan 03] Xingzhe Fan, Naiyao Zhang , Shujie Teng : " Trajectory planning and tracking ofball and plate system using hierarchical fuzzy control scheme", Fuzzy Sets and Systems 144 2003.

[Fukuda 99] Toshio Fukuda, Naoyuki Kubota: "An intelligent robotic system based on a fuzzy approach", proceeding of the IEEE, vol 87, no 9,P448-1470, september 1999.

[Guenounou 03] Ouahib Guenounou: " Optimisation des contrôleurs flous par algorithmes génétiques hiérarchisés. Application sur un actionneur asynchrone. Mémoire de Magister, université Abderrahman Mira –Béjaia. 2003.

[Homaifar 95] A. Homaifar, E. McCormick: "Simultaneous design of member-ship functions and rule sets for fuzzy controllers using genetic algorithms". IEEE Transactions on Fuzzy Systems Vol 3, P129-139, 1995.

[Karr 93] C. L. Karr: " Fuzzy control of PH using genetic algorithms", IEEE Transaction on fuzzy systems, Vol 1,P46-53, 1993.

[Kim 03] Dong-Han Kim, Jong-Hwan KimA : "Real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer". JN. Robotics and Autonomous Systems 42, P17-30, 2003.

[Lachekhab 05] Fadhila Lachekhab : " Planification et navigation a base de la logique floue d'un robot mobile dans un environnement partiellement connu", mémoire magister, école militaire polytechnique, 2005.

[Lewis 06] Shuzhi Sam Ge and Frank L. Lewis: "Autonomous mobile robots sensing, control, decision making and applications". CRC press taylor & francis group 2006.

[Mamdani 74] E. H. Mamdani and S. Assilian: "A case study on the application of fuzzy set theory to automatic control", Proc. IFAC stochastic control symp., Budabest, 1974.

[Reignier 98] Patrick Reignier : "Pilotage réactif d'un robot mobile étude du lien entre la perception et l'action", thèse doctorat informatique, Institut national polytechnique de Grenoble 1998.

[Russo 98] Marco Russo: "FuGeNeSys—A Fuzzy Genetic Neural System for Fuzzy Modeling" , IEEE transactions on fuzzy systems, vol. 6, no. 3, P373-388, 1998

[Saffiotti 97] A. Saffiotti : "The uses of fuzzy logic in autonomous robot navigation", Soft Computing 1, 180-197 Springer-Verlag, 1997.

[Setnes 00] Magne Setnes, Hans Roubos :" GA-fuzzy modeling and classification: complexity and performance" IEEE transactions on fuzzy systems, vol.8, no.5, P 509-522 2000.

[Shibata 94] Takanori Shibata, Toshio Fukuda: "Hierarchical intelligent control for robotic motion", IEEE Transaction on nerenal networks, vol 5 no 5, P823-833, 1994.

[Siegwart04] Roland Siegwart and Illah R. Nourbakhsh : "Introduction to Autonomous mobile robot". MIT Press Cambridge, Massachusetts, 2004.

[Simon 95] D. Simon & H. EL-Sherief : "Fuzzy logic for digital phase –locked loop filter design ", IEEE Transaction on fuzzy systems, Vol 1, P211-218, 1995.

[Sugeno 85] M. Sugeno " An introductory survey of fuzzy control", information sciences, no. 36, pp.59-83, 1985

[Topalov 98] A. Venelinov Topalov, Jong-Hwang Kim, Todor Philipov Proychev : "Fuzzy-net control of non-holonomic mobile robot using evolutionary feedback-error-learning", JN. Robotic and Autonomous System 23, P187-200, 1998.

[Wang 08] Meng Wang, James N.K Liu: "Fuzzy logic-based real-time robot navigation in unknown environment with dead ends", JN. Robotics and autonomous systems (56), P 625-643, 2008.

[Wang 92] Li-Xin Wang, Jerry M. Mendel: "Generating fuzzy rules by learning from examples", IEEE Tans on sys, man and cybernetics Vol 22 N 6, P1414-1427,1992.

[Zadeh 65] L. A. Zadeh : "Fuzzy sets", Information and Control, vol. 8, P338–353, 1965.

Annexe A

Procédure de détection des obstacles

Puisque notre travail est une simulation sur Matlab, tous les objets se trouvant dans l'environnement sont simulés par des procédures dédiées à cet effet (ch1§ 5.3-5.4).

Pour calculer l'état de chaque capteur, qui sont les entrées du contrôleur d'évitement d'obstacles, on commence par la localisation des obstacles se trouvant à une distance inférieur ou égale à la distance de sécurité D_s, considérée comme la portée des capteurs, puis on calcule l'angle que fait la direction du robot avec ce dernier (ces derniers).

La relation de calcule des états de chaque capteur est donnée par :

1. Pour les obstacles carrés

$$D_i = \min\left(\frac{D_{obs}}{D_s \times \cos(\varphi_i - \alpha)}, 1\right)$$

$D_{i=1..6}$: Etat du capteur i.

D_{obs} : Distance du robot de l'obstacle donnée par

$$D_{obs} = \begin{cases} y_1 - y_c \;\; si \;\; y_c < y_1 \\ y_c - y_2 \;\; si \;\; y_c > y_2 \\ x_1 - x_c \;\; si \;\; x_c < x_1 \\ x_c - x_2 \;\; si \;\; x_c > x_2 \end{cases}$$

α : Angle robot obstacle.

$\varphi_{i\ i=1,..,6}$: angle du capteur "i" (figure 1.6).

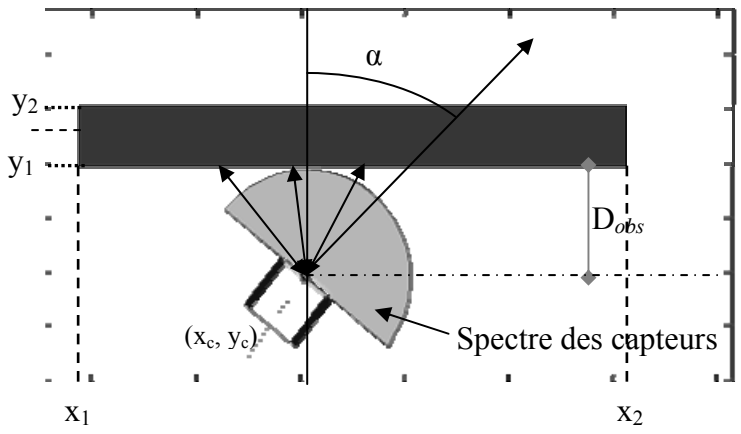

Figure A.1 : Représentation du robot devant un obstacle carré.

2. Les obstacles ronds.

Contrairement aux obstacles carrés, pour lesquels on calcule les états des tous les capteurs, pour les obstacles ronds, on calcule juste l'état du capteur le plus aligné avec l'obstacle, et la relation donnant l'état du capteur se résume alors à :

$$D_i = \min(\frac{D_{obs}}{D_s},1)$$

Tel que :

D_i : état d'obstacle pour lequel $|\varphi_i - \alpha|$ est minimal

$$D_{obs} = \sqrt{(x_c - x_{obs})^2 + (y_c - y_{obs})^2} - r_{obs}$$

$(x_{obs}, y_{obs}, r_{obs})$: coordonnées et rayon de l'obstacle rond.

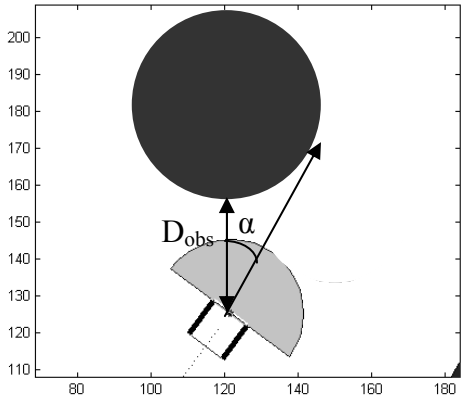

Figure A.2 : Représentation du robot devant un obstacle rond.