République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri de Tizi-Ouzou
Faculté de Genie Electrique et d'Informatique
Département d'Automatique

Thèse

Présenter pour obtenir le diplôme de

Doctorat 3ème Cycle LMD

Spécialité : *Automatique et Robotique*

Par :

BRAHAMI Amal

Thème

Navigation d'un robot mobile à distance à base d'un système virtuel intelligent

---

Devant le jury :

Président :	Mr. HAMMOUCHE Kamal	*Professeur Université Mouloud MAMMERI de Tizi-Ouzou*
Rapporteur :	Mr. MELLAH Rabah	*Professeur Université Mouloud MAMMERI de Tizi-Ouzou*
Examinateur :	Mr. LAHDIR Mourad	*Professeur Université Mouloud MAMMERI de Tizi-Ouzou*
Examinateur :	Mr. ARDJAL Aghiles	*MCA Université Mouloud MAMMERI de Tizi-Ouzou*
Examinateur :	Mr. RAHOUI Adel	*MCA Ecole Nationale Supérieure des Travaux Publics*
Examinateur :	Mr. CHAIB Ahmed	*Professeur Université M'Hamed BOUGARA de Boumerdes*

Soutenue 23/01/2025

Résumé

Résumé : Le suivi de trajectoire et l'évitement d'obstacles sont des problèmes principaux de la navigation autonome des robots mobiles. Cette thèse décrit la navigation virtuelle d'un robot mobile de type entrainement différentiel (Pioneer P-3dx) qui contient deux roues gauche et droite entrainer par des moteurs DC et une roue libre pour assurer sa stabilité. Pour ce faire on a conçus nos environnements de navigations sur la plateforme virtuelle V-REP (Virtual Robot Experimental Platform) ; En premier lieu on a réalisé une navigation dans un environnement sans obstacles en cherchant une cible, deuxièmement dans un environnement sans obstacles en cherchant plusieurs cibles. Ensuite on a réalisé une navigation dans un environnement encombré d'obstacles qui permet de chercher une cible tout en évitant des obstacles sans collision. Deux stratégies de commandes sont utilisées la première par l'algorithme ANFIS et la deuxième par l'algorithme hybride ANFIS-PSO écrient en code Matlab. L'objectif de l'algorithme PSO (Particule Swarm Optimisation) est la minimisation de la fonction objective qui est l'erreur quadratique moyenne entre les valeurs réelles et prédites des commandes du contrôleur ANFIS qui sont les vitesses gauche et droite du robot mobile, pour donner des valeurs de vitesse optimales qui seront appliqué au robot pour donner un meilleur suivi de trajectoire et évitement d'obstacles. La Plateforme d'Expérimentation des Robots Virtuels V-REP a été utilisé pour réaliser les environnements virtuels de simulation proche de la réalité et qui prend en considération les caractéristiques physiques du robot et de l'environnement afin de prendre une vision globale sur les problèmes et les résultats attendu avant l'implémentation de ces techniques de commande sur un robot réel dans un environnement réel. Une étude comparative entre la navigation par la méthode ANFIS et la navigation par la méthode hybrides ANFIS-PSO montre bien l'efficacité du PSO qui assure une bonne réponse au robot dans ces limitations physiques (vitesses, couples) et assurer un suivi optimal pour la navigation automatisée du robot à roues Pioneer P-3dx. En regardant les résultats expérimentaux, on observe que la méthode proposée fournit une solution efficace et optimale au problème de recherche de cible et l'évitement d'obstacles.

Mots-clés : Robot Mobile, Pioneer P-3dx, Navigation Virtuelle, V-REP, MATLAB, ANFIS, PSO.

Abstract : Trajectory following and obstacle avoidance are the main issues for autonomous navigation for mobile robots. This thesis describes the virtual navigation of a Pioneer

P-3dx differential drive mobile robot with two left and right wheels driven by DC motors and a freewheel to ensure its stability in three environments designed on the Virtual Robot Experimental Platform (V-REP). At first, we realized navigation in an environment without obstacle by tracking a target at second by tracking a several targets, first using the Adaptive Neuro Fuzzy Inference System (ANFIS) algorithm, then using the hybrid algorithm ANFIS-Particle Swarm Optimization (ANFIS-PSO) written in MATLAB code, and then we realized navigation in a cluttered environment that allows searching a target by avoiding obstacles with the ANFIS algorithm then with ANFIS-PSO. The objective of PSO algorithm is the minimization of the objective function which is the root mean square error between the actual and the predicted values of the ANFIS controller commands which are the left and right velocity of the mobile robot, to give optimum velocity values that will be applied to the robot for better trajectory tracking and obstacle avoiding. The Virtual Robot Experimentation Platform Software V-REP was used to create virtual simulation environments close to reality and which take into consideration the physical characteristics of the robot and the environment ; in order to take a global view of the problems and expected results before implementing these control techniques on a real robot in a real environment. A comparative study between navigation with ANFIS and navigation with ANFIS-PSO clearly shows the effectiveness of the PSO which ensures a good response to the robot within these physical limitations (velocities, torques) and ensures optimal tracking for the virtual navigation of the mobile robot. Looking at the experimental results, it is observed that the proposed method provides a more efficient and optimal solution to the problem of target search and obstacle avoidance.
Keywords : Mobile Robot, Pioneer P-3dx, Virtual navigation, V-REP, MATLAB, ANFIS, PSO.

Remerciements

Je remercie d'abord DIEU le tout puissant pour la santé, le courage et la foi qu'il m'a donnés pour arriver à ce jour.
Avant tous je remercie ma chère maman ALLAH YARHEMHA décédé le 15/10/2023 qui m'a donnée l'amour et l'éducation et conseils... pour réaliser ce noble travail.
Je tiens, tout d'abord, à exprimer ma gratitude, ma plus grande reconnaissance et mon profond respect à mon directeur de thèse, Monsieur Rabah MELLAH, Professeur à l'université Mouloud MAMMERI de Tizi-Ouzou, qui a dirigé mes travaux de recherche et qui m'a apporté son savoir scientifique indéniable, sa confiance, ses judicieux conseils et sa disponibilité, tout au long de l'élaboration de ce travail de thèse. Qu'il trouve ici l'expression de ma profonde reconnaissance.
Je tiens particulièrement à remercier le chef de département d'Automatique, Monsieur Mohand Achour TOUAT, Maître de conférences à l'université Mouloud MAMMERI de Tizi-Ouzou, pour sa disponibilité, son soutien et ses conseils précieux.
Ma reconnaissance va également à Monsieur Said GUERMAH, Professeur à l'université Mouloud MAMMERI de Tizi-Ouzou, qui m'a aidé à corriger mon travail avant la publication.
Je remercie aussi Monsieur Ahmed MAIDI, Professeur à l'université Mouloud MAMMERI de Tizi-Ouzou, pour son aide et ses précieux conseils.
Je souhaite ensuite remercier Monsieur Kamal HAMMOUCHE, Professeur à l'université Mouloud MAMMERI de Tizi-Ouzou, qui m'a fait l'honneur de présider le jury de cette thèse.
Je tiens à remercier Monsieur Mourad LAHDIR, Professeur à l'université Mouloud MAMMERI de Tizi-Ouzou, d'avoir accepté de juger et d'évaluer mon travail de thèse.
Je tiens à remercier également Monsieur Aghiles ARDJAL, MCA à l'université l'université Mouloud MAMMERI de Tizi-Ouzou, de m'avoir fait l'honneur d'examiner mon travail.
Je tiens à remercier également Monsieur Adel RAHOUI, MCA à ENSTP d'Alger, de m'avoir fait l'honneur d'examiner mon travail.
Je tiens à remercier également Monsieur Ahmed CHAIB, Professeur à Université de Boumerdes, de m'avoir fait l'honneur d'examiner mon travail.
Mes remerciements les plus chaleureux vont à mon mari le bras droit pour réaliser ce travail, mes chères filles : BAYA et IMEN qui m'ont données le courage de combattre

pour arriver à cette thèse, à mes chers frères : AMEZIANE, MENAD et KHELIFA et mes sœurs : DIHIA et OUALIDA et toute ma famille pour leurs encouragements, leur patience, et leur grand soutien durant toutes ces années de préparation de ce travail de thèse.

Je remercie tous ceux qui ont contribué, de près ou de loin, à la réalisation de cette thèse.

Table des matières

Liste des tableaux

Table des figures

Introduction générale

Robot ou robotique est un terme fascinant pour désigner l'attention qui attire beaucoup l'être humain, soit sous la forme de robots amateurs, soit sous la forme de robots industriels ou militaires. Aujourd'hui, la vie des gens est fortement attirée par différents types de robots. L'intérêt a été accru par les écrivains de fiction et les cinéastes qui présentent des idées innovantes et futuristes. La fiction a toujours fasciné les concepteurs/innovateurs dans le domaine des sciences et de l'ingénierie. En raison de cette fascination, une vaste communauté de scientifiques et de chercheurs s'est concentrée ces dernières années sur la conception, la modélisation et le contrôle des robots pour diverses applications.

Historiquement, les robots peuvent être classés en deux catégories : les robots stationnaires et les robots non stationnaires ou mobiles. Les robots stationnaires sont des robots manipulateurs qui peuvent se déplacer dans un châssis fixe avec une enveloppe de travail limitée et être utilisés pour amener l'effecteur final à la position souhaitée avec l'orientation souhaitée. Contrairement aux robots non stationnaires, la robotique mobile est la branche concernée par les robots qui ont la capacité de se déplacer dans leur environnement et ne sont pas fixés à un emplacement physique. Cette capacité permet d'introduire des expériences difficiles (contrôle de position et formation, suivi de trajectoire, suivi de chemin, navigation, exploration, évitement d'obstacles, etc.) [1]. Un robot mobile autonome est équipé de capteurs adéquats pour percevoir son environnement et agir en conséquence [2]. Les robots mobiles peuvent être aériens, sous-marins ou terrestres. Parmi les robots terrestres, on retrouve les robots mobiles à roues, qui font l'objet de notre travail.

Les robots mobiles à roues (WMR) ont de nombreuses applications dans le monde réel, ils sont utilisés pour aider ou remplacer les humains dans de nombreux domaines militaires, industrielles et commerciales comme moyen de logistique [2], agricole ou de transport de passagers [3], d'inspection, de surveillance et d'opérations spécialisées/non spécialisées en raison de leur efficacité et flexibilité. Les robots mobiles à roues sont également utiles pour réaliser des tâches dites 4D (Difficile, Ennuyeuse, Dangereuse et Sale) comme la manipulation de matières radioactives, la décontamination de réacteurs nucléaires, le déminage, l'inspection de tunnels, etc.

De plus, WMR est une excellente plateforme pour tester un variété d'applications éducatives et de recherche. Il existe des recherches sur ce sujet depuis les années 1970 [4], il s'agit

donc d'un domaine mature avec de nombreux algorithmes publiés et outils disponibles, est un domaine multidisciplinaire qui comprend un combinaison d'informatique, d'électronique, de mécanique, d'intelligence artificiel, contrôle, entre autres sujets. Cependant, les applications réelles dans des conditions réelles restent un défi.
De manière générale, la navigation autonome des WMR s'avère être un domaine de recherche crucial parmi les différentes problématiques de cette catégorie de robots.
Les outils de simulation sont très répandus en robotique avant la mise en œuvre réelle : on simule le robot et son environnement, via des objets en 3 dimensions soumis au lois de la physique par un moteur physique.Cependant, ces simulations manquent de souvent de réalisme. En effet, Ivaldi et al. [5] ont interrogé les chercheurs en robotique sur les outils qu'ils utilisent, et leurs attentes vis à vis de ces outils pour l'avenir : leur Graal consiste en la simulation au plus proche de la réalité. A défaut d'avoir un niveau de réalisme satisfaisant pour l'évaluation, la simulation est utilisée pour faire de la preuve de concept.
En fait, notre travail illustre les apports notables de l'Intelligence Artificielle, discipline relativement récente dans la synthèse de lois de commande, appropriées à la navigation robuste des robots mobiles. L'Intelligence Artificielle , a permis en particulier de lever une contrainte en Contrôle relativement à la nécessité de disposer d'un modèle le plus fidèle possible, sachant que les erreurs de modélisation et les imperfections des modèles, contribuent à dégrader notablement les performances des lois de commande.
Des outils tels que la logique floue qui permet de faire le lien entre modélisations numérique et symbolique à partir d'algorithmes très simples, et les réseaux de neurones qui consistent en la mise au point et l'optimisation de méthodes d'apprentissage, ont permis la conception de contrôleurs neuro-flous. Cela a permis de mettre en évidence l'interaction entre la logique floue et les réseaux de neurons en hybridation avec l'algorithme d'optimisation Meta-heuristique par essaim de particule (PSO) avec une analyse de robustesse approfondie, dans le but de réaliser des avancées significatives.
Notre intérêt s'est porté sur la navigation virtuelle d'un robot mobile Pioneer P-3dx à l'aide d'un simulateur robotique V-REP sur l'environnement MATLAB. Par la suite, deux stratégies de commande ont été synthétisées et appliquées dont l'une à base du contrôleur neuro-flou ANFIS (Adaptive Neuronal Fuzzy Inference Systems) et l'autre à base de la combinaison du contrôleur ANFIS et l'algorithme PSO. La démarche proposée consiste à concevoir trois environnements sur la plateforme d'expérimentation des robot virtuel (V-REP), le premier sans obstacles et chercher une cible, le deuxième sans obstacles pour chercher plusieurs cibles et le dernier avec obstacles pour chercher une cible.
Les performances des différents schémas de commande proposés sont illustrées à l'aide de nombreux résultats de simulation pour le robot mobile Pioneer P-3dx. Les résultats obtenus montrent bien l'efficacité du contrôleur hybride ANFIS-PSO, par rapport au contrôleur ANFIS.

Problématique

Récemment la navigation d'un robot mobile autonome utilise des contrôleurs intelligents qui permet au robot de chercher la cible et suivre un chemin le plus cours en évitant des obstacles statique ou dynamique [6]. La navigation dans le domaine robotique est divisée en deux, principalement locale et globale, les zones locales sont précédemment reconnues par les robots, tandis que les zones globales représentent les zones que les robots rencontrent pour la première fois, les obstacles dans les zones globales sont un problème car ils ne sont pas connus par les robots, par conséquent la navigation dans les zones globales reste l'un des principaux problèmes de l'industrie de la robotique [7].
Ces problèmes de navigation globale et locale ont été résolus en utilisant plusieurs algorithmes qui peuvent être classés en trois catégories : déterministes (flou, réseau de neurones, neuro-flou), non déterministes ou stochastiques (GA, PSO, etc...) et algorithmes évolutionnaires. Cette dernière est l'hybridation de méthodes déterministes et non déterministes [8].
Actuellement Les techniques neuro-floues sont les plus utilisés dans la navigation des robots mobiles car les algorithmes neuro-flous sont des techniques déterministes intelligentes et basées sur la connaissance largement utilisées pour la navigation des robots mobiles. Ces techniques peuvent facilement modéliser le raisonnement, l'incertitude et la non-linéarité des environnements complexes. La navigation autonome sans collision est définie comme un ensemble de comportements abstraits comprenant la recherche d'objectifs, l'évitement d'obstacles, etc [8, 9].
Des études récentes ont montré que différents types d'algorithmes ont été développés pour résoudre des problèmes de navigation des robots mobiles. Un article [10] vise à étudier attentivement deux types d'approches avancées pour guider un robot mobile non holonome pour naviguer dans un environnement encombré d'obstacles statiques, tout d'abord un contrôleur à logique floue (FLC) a été conçu, deuxièmement un contrôleur de système d'inférence neuro floue adaptative (ANFIS) a été utilisé pour optimiser les résultats obtenus à partir du contrôleur flou, pour valider la faisabilité et l'efficacité des modèles proposés les logiciels V-REP et MATLAB sont utilisés, une évaluation comparative est alors faite et les résultats des simulations ont montré que le robot mobile pouvait naviguer avec succès dans l'environnement du labyrinthe avec les deux approches proposées, mais le contrôleur ANFIS a fourni de meilleurs résultats par rapport au contrôleur flou.
Dans certaines études [8], il est suggéré que les algorithmes de navigations et d'autres algorithmes d'optimisation soient combinés pour résoudre des problèmes, une étude de la littérature propose une méthode hybride GPS-ANFIS pour la navigation sans collision de robots mobiles autonomes. Le contrôleur basé sur GPS maintient la direction de navigation du robot vers la cible statique ou dynamique. Il utilise les coordonnées reçues des deux modules GPS sur les bords de l'axe longitudinal du robot ainsi que les coordonnées

de la cible pour la détourner de la trajectoire actuelle en faisant un certain angle vers la cible.
Dans une autre étude [11], une architecture de contrôle pour le suivi de trajectoire tout en évitant les obstacles et le réglage du contrôleur PID est proposée pour un robot mobile à entraînement différentiel (DMR), ils proposent un algorithme d'optimisation méta heuristique pour résoudre les problèmes de planification de chemin et le réglage du contrôleur en choisissant des fonctions objectives appropriées.
Dans une autre étude de la littérature [12], ils ont décrit la navigation d'un robot à roues automatisé Pioneer P-3dx entre des obstacles à l'aide d'un algorithme d'optimisation d'essaim de particules (PSO) réglé sur un réseau neuronal prédictif (FNN). D'après une étude comparative il a été constaté qu'un FNN à réglage PSO plus efficace devant FNN sans réglage pour la navigation automatisée du robot à roues Pioneer P-3dx. De plus, ils ont comparé le FNN optimisé par PSO à la technique de navigation Fuzzy Logic Controller optimisée par PSO précédemment développée pour montrer l'authenticité et la mise en œuvre en temps réel du FNN optimisé par PSO.
Une autre étude dans la littérature [13] propose un système intelligent pour la navigation de robots mobiles dans différents environnements, utilisant ANFIS et ACOr (Optimisation de colonie de fourmis pour les domaines continus). Dans un premier temps, ils ont utilisé le contrôleur ANFIS (Adaptive network-based fuzzy inference system),dans la deuxième étape, la méthode de colonie de fourmis en environnement continu ACOr est greffée dans la deuxième couche du réseau ANFIS pour l'hybridation, les simulations des mouvements du robot et des interfaces graphiques sont réalisées sous le langage C++.
Notre problème consiste à réaliser une navigation virtuelle d'un robot mobile pour ce faire en a choisit la plateforme virtuelle V-REP (Virtual Robot Expérimentation Platform) ; et les commandes sont calculer sur le logiciel Matlab par deux stratégies ,la premières par un controleur ANFIS (système d'inférence neuro-floue adaptative) ,et la deuxièmes par un controleur hybride ANFIS-PSO (particule swarm optimisation) pour montrer l'efficacité de l'algorithme hybride.

Contribution et objectifs

Notre travail consiste à réaliser une navigation virtuelle d'un robot mobile en utilisant les techniques de la réalité virtuelle simulées dans la plateforme virtuelle V-REP (Virtual Robot Expérimentation Platform) qui prend en considération les paramètres de l'environnement réel. Les deux stratégies de commande synthétisées sont réalisées grâce au logiciel MATLAB. La première stratégie utilise le contrôleur Neuro flou (ANFIS) (système d'inférence neuro-floue adaptative) de type Sugeno où les fonctions et les règles d'appartenance sont générées automatiquement par les techniques adaptatives. Quant à la deuxième stratégie propose un contrôleur hybride ANFIS-PSO où l'algorithme d'op-

timisation PSO permet de garantir que notre robot reste dans ses limitations physiques (limitation des vitesses des moteurs des roues) qui se fassent sur une plage de vitesse afin qu'ils soient bornées entre deux vitesses v_{min}, v_{max}. Par la suite, une étude comparative entre les deux stratégies de commande proposées pour évaluer les performances.
La communication entre V-REP et Matlab est effectuée en mode synchrone de sorte que V-REP reçoit les commandes depuis Matlab pour la navigation du robot mobile dans l'environnement virtuel conçu dans le but de chercher la cible et suivre une trajectoire la plus courte tout en évitant des obstacles sans collision. En retour transmit au logiciel Matlab les données mesurées à savoir : vitesses, positions, orientation, distances vers les obstacles, pour faire une boucle de contrôle.

Organisation de la thèse

Cette thèse est structurée comme suit : Dans le premier chapitre nous présentons les généralités sur les différents robots mobiles les plus utilisés en Robotique du point de vue structure et composants, ainsi que la navigation autonome d'un robot mobile. La classification et la structure des robots mobiles sont d'abord abordées. Ensuite nous étudions les différentes techniques adaptées à la navigation autonome d'un robot mobile, en présentant les trois grandes architectures de contrôle en mentionnant leurs avantages et inconvénients. Dans le deuxième chapitre nous synthétisons les modèles cinématique et dynamique du robot à entrainement différentiel, pour les valider par la suite grâce à des contrôleurs cinématique et dynamique (déduit à partir de la linéarisation par bouclage non linéaire). Le troisième chapitre porte sur la synthèse de deux stratégies de commande intelligente. La première stratégie utilise les concepts de la logique floue et les réseaux de neurones pour synthétiser le contrôleur Neuro-flou de type ANFIS. Quant à la deuxième stratégie fait appel à une hybridation de la première stratégie qui s'inspire de l'intelligence artificielle, et optimisation par l'algorithme essaim de particule (PSO) sur Matlab qui calcul les commandes qui sont les vitesses des roues du robot mobile Pioneer P3-dx de V-REP. Le quatrième chapitre porte sur la description de La réalité virtuelle et la plateforme virtuelle V-REP ainsi que la co-simulation entre Matlab et V-REP. Dans le dernier chapitre nous présentons les résultats de simulation obtenus par applications de ces stratégies synthétisées à la navigation d'un robot mobile dans un environnement parsemés d'obstacles. Les performances de ces deux structures de commande sont évaluées et compares aux autres commandes. Enfin, nous terminons notre travail par une conclusion générale et quelques perspectives.

Publication

Un article du rang A : A.Brahami, R. Mellah and S.Guermah. “Virtual navigation of mobile robot in V-REP using hybrid ANFIS-PSO controller”. CEAI, Vol.26, No.1, pp. 25-35, 2024.doi : 10.61416/ceai.v26i1.8797.

Chapitre 1

Navigation autonome d'un robot mobile

1.1 Introduction

L'une des premières étapes dans l'étude des robots consiste à définir la classe du robot selon les différentes actions à mener, le type de robot, ainsi que les différents éléments qui le constituent. Une fois ceci fait, il sera possible de développer des algorithmes permettant de le commander. Dans une première partie de ce chapitre, nous présentons les généralités sur les robots mobiles, les différentes classes des robots ensuite ces différents éléments constitutifs (les capteurs, les actionneurs et l'unité de traitement). En deuxième partie, nous présentons la navigation autonome d'un robot mobile, les architectures de contrôle qui permettent d'orchestrer les relations entre la perception, la décision (planification) et l'action. On conclura par la planification de chemin qui commencera par la modélisation de l'environnement de navigation par plusieurs cartes (topologique, métrique, hybride), on enchainera avec une présentation des méthodes de navigation classique et celles dites du Soft-Computing qui ont permis l'introduction des systèmes intelligents.

1.2 Généralités sur les robots mobiles

La robotique est un domaine de recherche important qui fait appel aux connaissances croisées de plusieurs disciplines. L'objectif est l'automatisation des systèmes mécaniques, en les dotant par des capacités de perception, de décision et d'action permettant au robot d'interagir rationnellement avec son environnement sans intervention humaine [14].La robotique mobile est un domaine dans lequel l'expérience pratique est primordiale. Au début, les robots font leur apparition dans l'industrie grâce aux capacités des manipulateurs qui imitent le bras humain tout en utilisant différents outils pour accomplir diverses tâches. Au fur et à mesure, ces bras gagnent de nouveaux degrés de liberté grâce aux plates formes

mobiles qui engendrent l'apparition des robots mobiles à roues [15].Les premiers robots autonomes utilisaient la roue mais pour certaines applications où la géométrie de l'environnement se diffèrent (terrain accidenté, endroits difficiles à atteindre), les recherches sont orientées vers d'autres moyens de locomotion, inspirés du monde des animaux. Ainsi, les robots à pattes sont apparus et on connaît déjà des robots hexapodes ou des robots bipèdes. Des problèmes tels que la conception mécanique, la perception de l'environnement, la modélisation de l'espace de travail, la planification de trajectoires sans collision et la synthèse des lois de contrôle non linéaires sont des exemples de différents sujets de recherche dans la robotique mobile [16]. L'intérêt indéniable de cette discipline est d'avoir permis d'augmenter considérablement nos connaissances sur la localisation et la navigation des systèmes autonomes. Cet axe de recherche étend le domaine d'application de la navigation autonome à toutes les sortes d'environnement non structurés, dû principalement à la nécessité d'aider ou de remplacer l'intervention humaine [17]. Lorsqu'il s'agit de faire une tâche à risque comme les applications nucléaires, spatiales ou militaires [18]. Historiquement, la navigation robotique a tout d'abord été considérée comme un problème de planification de trajectoires. Le modèle 3D de la scène est supposé connu et des stratégies globales ou locales permettent de définir la séquence de configurations que doit prendre le robot pour atteindre son objectif. Depuis le début des années 80, les avancées technologiques et algorithmiques permettent de contrôler directement les mouvements d'un système robotique à partir des informations perçues par ses capteurs. En particulier, les informations visuelles fournies par une caméra peuvent être utilisées pour définir une loi de commande du robot sans pour autant nécessiter le modèle de la scène. Les techniques dites d'asservissement visuel sont basées sur ces principes. Plus récemment, les recherches en robotique mobile ont considéré des environnements de navigation très vastes, à tel point que les capteurs (caméra ou même laser) ne peuvent pas les décrire que localement à un instant donné. Une représentation interne de l'environnement de navigation devient alors nécessaire. Elle permet dans ce cas d'étendre le champ d'action des capteurs du système robotique, mais aussi de définir un ensemble de stratégies de navigation adaptées à l'environnement traversé [19].

1.3 Classification des robots mobiles

La caractéristique la plus remarquable d'un robot mobile est évidement son moyen de locomotion. Celui-ci dépend directement du type d'application visé ainsi que de type du terrain dans lequel le robot mobile doit évoluer (environnement d'intérieur, extérieur, libre ou encombré d'obstacles, ...). Les robots mobiles sont classés généralement selon le type de locomotion utilisé en quatre groupes distincts ; qu'il soit : à roues, à chenilles, à pattes ou avec d'autres moyens de locomotions [20, 21].En effet, le type de locomotion définit deux types de contraintes :

- Les contraintes cinématiques, qui portent sur la géométrie des déplacements possibles du robot dans l'environnement de navigation.

- Les contraintes dynamiques liées aux effets du mouvement (accélérations, vitesses bornées, présence de forces d'inertie ou de frottement). Ces facteurs influent sur le mouvement exécuté. Selon la cinématique, un robot est dit holonome s'il peut se déplacer instantanément dans toutes les directions possibles. Il est dit non holonome si ses déplacements autorisés sont des courbes dont la courbure est bornée. Dans ce qui suit, on décrit brièvement les grandes classes des robots mobiles [17, 22].

1.3.1 Robot mobiles à roues

Dans le cadre de notre étude, nous nous intéresserons aux robots mobiles à roues de type unicycle. D'après [23], les robots mobiles peuvent être classés suivant la locomotion et la cinématique du robot. On distingue plusieurs types de robots mobiles à roues, classés principalement par la position et le nombre de roues utilisées. Nous citerons les quatre types de robots à roues couramment rencontrés [24]. Compte tenu de la simplicité du mécanisme de locomotion utilisé, ce type de robot est le plus répondu actuellement. La plupart des robots mobiles à roues opèrent dans des sites aménagés, des sites industriels ou des environnements intérieurs ; mais il existe également des applications en environnements extérieurs, comme l'exploration spatiale [15]. La grande majorité des robots de ce type présente des contraintes de non holonomie qui limitent le mouvement instantané que le robot peut réaliser, car il existe pour toutes ces roues un point unique (centre instantané de rotation (CIR)) de vitesse nulle autour duquel le robot tourne de façon instantanée. La commande se fait par la motorisation des roues installées. Ces contraintes augmentent la complexité du problème de planification de trajectoire et son contrôle [14, 25]. Les robots mobiles à roues peuvent être classés en plusieurs types avec des propriétés intéressantes : unicycle, différentielle, tricycle, de type voiture, omnidirectionnelle et à traction synchrone [26].Les modèles des robots mobiles à roues existant sont illustrés sur (Figure 1.1). Il existe plusieurs classes de robots mobiles à roues déterminées, principalement, par la position et le nombre de roues utilisées. Nous citerons ici les quatre classes principales de robots à roues.

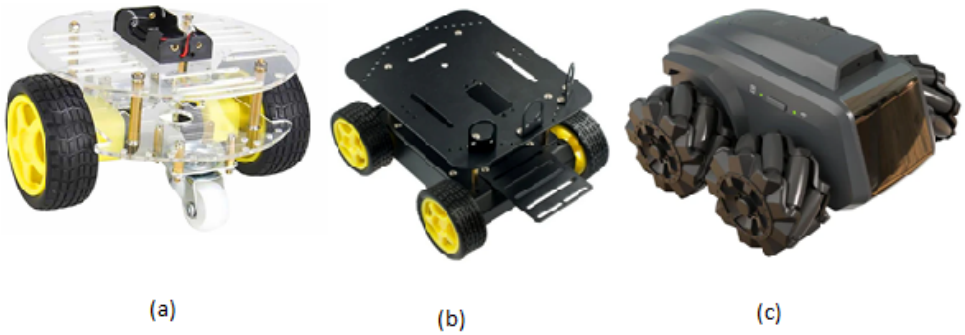


FIGURE 1.1 – Les robots mobiles à roues : (a) [27], (b) [28], (c) [29].

Robot mobile de type unicycle

Un robot mobile de type unicycle (Figure 1.2 et Figure 1.3) est actionné par deux roues indépendantes, il possède éventuellement des roues folles pour assurer sa stabilité. Son centre de rotation est situé sur l'axe reliant les deux roues motrices. C'est un robot non-holonome, en effet il est impossible de le déplacer dans une direction perpendiculaire aux roues de locomotion. Sa commande peut être très simple, il est en effet assez facile de le déplacer d'un point à un autre par une suite de rotations simples et de lignes droites. [30, 31, 32].

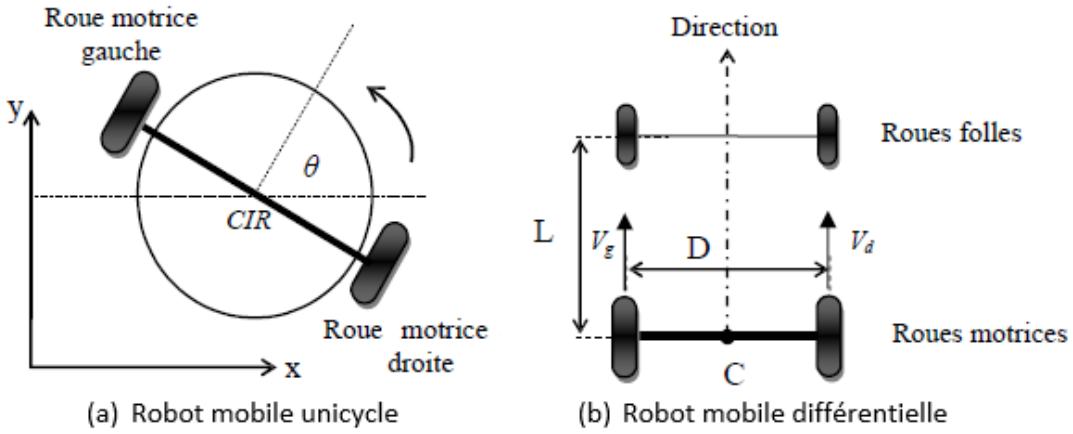
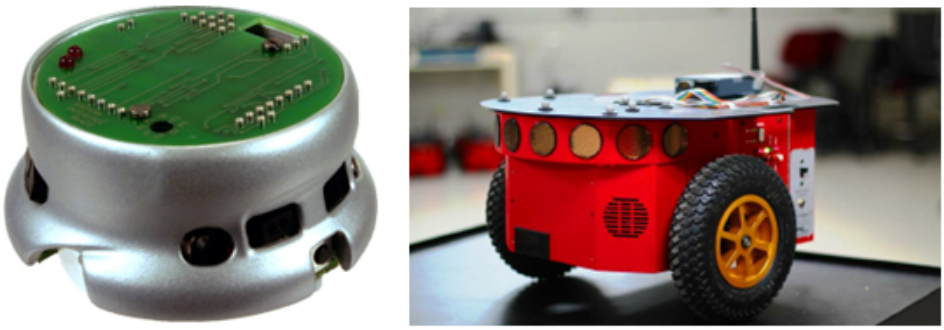


FIGURE 1.2 – Robots mobile de type unicycle.



(a) Khepera II (b) Pioneer P3

FIGURE 1.3 – Les robots mobiles unicycles : (a) Khepera II [33], (b) Pioneer P3 [34].

Robot mobile de type tricycle

Un robot mobile de type tricycle (Figure 1.4) est constitué de deux roues fixes placées sur un même axe et d'une roue centrée orientable placée sur l'axe longitudinal. Le mouvement du robot est donné par la vitesse des deux roues fixes et par l'orientation de la roue orientable. Son centre de rotation est situé à l'intersection de l'axe contenant les roues fixes et de l'axe de La roue orientable. C'est un robot non-holonome. En effet, il est impossible de le déplacer dans une direction perpendiculaire aux roues fixes. Sa commande est plus compliquée. Il est en général impossible d'effectuer des rotations simples à cause d'un rayon de braquage limité de la roue orientable. [30, 31, 32].

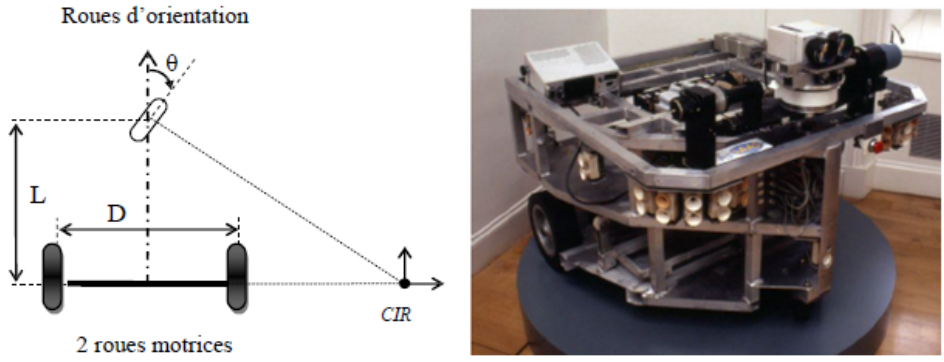


FIGURE 1.4 – Robot mobile de type tricycle (b) Robot mobile de type tricycle "HILLARE"[35].

Robot mobile de type voiture

Un robot mobile de type voiture (Figure 1.5) est semblable au tricycle, il est constitué de deux roues fixes placées sur un même axe et de deux roues centrées orientables placées elles aussi sur un même axe. Le robot mobile de type voiture est cependant plus stable puisqu'il possède un point d'appui supplémentaire. Toutes les autres propriétés du robot

voiture sont identiques au robot tricycle, le deuxième pouvant être ramené au premier en remplaçant les deux roues avant par une seule placée au centre de l'axe, et ceci de manière à laisser le centre de rotation inchangé. [30, 31, 32].

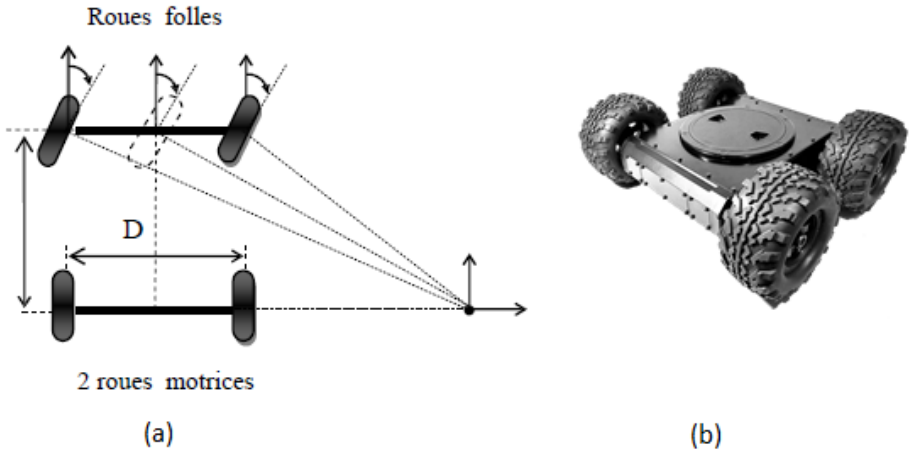


FIGURE 1.5 – Robot mobile de type voiture [36].

Robot mobile de type omnidirectionnel

Un robot mobile omnidirectionnel (Figure 1.6) est un robot qui peut se déplacer librement dans toutes les directions. Il est en général constitué de trois roues décentrées orientables placées en triangle équilatéral. L'énorme avantage du robot omnidirectionnel est qu'il est holonome puisqu'il peut se déplacer dans toutes les directions. Mais ceci se fait au dépend d'une complexité mécanique bien plus grande. [30, 31, 32].

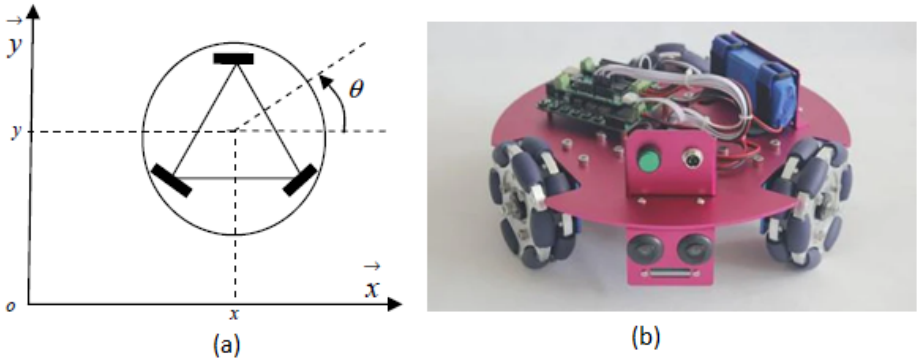


FIGURE 1.6 – Robot mobile de type omnidirectionnelle [37].

Comparaison des différents types

Nous pouvons observer dans le tableau ci-dessous un récapitulatif des avantages et des inconvénients des différents types de robots à roues[30, 31].

TABLEAU 1.1 – Avantages et inconvénients des différents types de robots à roues.

Type du robot	Avantages	Inconvénients
Unicycle	- Stable - Rotation sur soi-même - Complexité mécanique faible	- Non-holonome
Tricycle	- Complexité mécanique modérée	- Non-holonome - Peu stable - Pas de rotation sur soi-même
Voiture	- Stable - Complexité mécanique modérée	- Non-holonome - Pas de rotation sur soi-même
Omnidirectionnel	- Holonome - Stable - rotation sur soi-même	- Complexité mécanique - Importante

1.3.2 Robot mobile utilisant la chenille

Lorsque le terrain est accidenté, les roues perdent leur efficacité de locomotion. Ceci limite la capacité de mouvement du robot mobile équipé de ce type de système de locomotion. Dans ces conditions, les chenilles (Figure 1.7) sont plus intéressantes car elles permettent d'augmenter l'adhérence au sol et de franchir des obstacles plus importants [15].

FIGURE 1.7 – Robot mobile à chenille [38].

1.3.3 Robot mobile à pattes

Dans la situation ou le terrain est encore plus incertain, avec de grandes différences de hauteur comme par exemple un escalier ou un terrain très accidenté, les deux types précédents ne sont plus efficaces, et on fait recours aux robots mobiles à pattes (Figure 1.8). Ils ont des points d'appui discrets sur le terrain et sont donc la solution à ce problème de mouvement. Par contre, la conception et le contrôle d'un engin à pattes sont très

complexes. En plus, la vitesse d'évolution est généralement très réduite. La commande est très difficile, dépond de la multiplicité des actionneurs utilisés. Aibo de Sony est un exemple d'un robot mobile à pattes [15, 26].

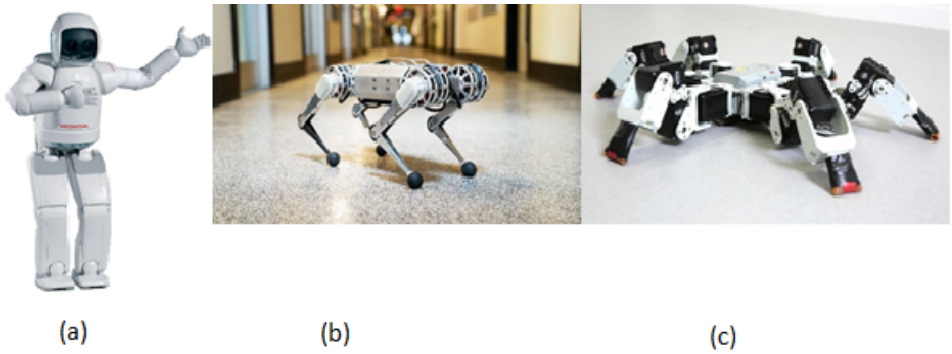


FIGURE 1.8 – Robot mobile à pattes : (a) [39] (b) [40] (c) [41].

1.3.4 Autres moyens de locomotion

Cette catégorie englobe les robots mobiles qui utilisent un moyen de locomotion différent des trois précédents. Par exemple, les robots mobiles qui se déplacent par reptation, les robots sous-marins, les robots d'exploration spatiale et les robots volants, ... etc. Les applications et la commande de ces robots sont très spécialisées, l'architecture est en général spécifique à l'application visée [17, 26].

1.4 Eléments constitutifs d'un robot mobile

En général, un robot est un assemblage complexe de pièces mécaniques et de pièces électroniques (Figure 1.9 et Figure 1.10). Les robots mobiles possédant une source d'énergie embarquée (batterie d'accumulateurs électriques) sont dit autonomes. Le plus souvent les pièces mécaniques et électroniques sont agencées de la manière suivante :

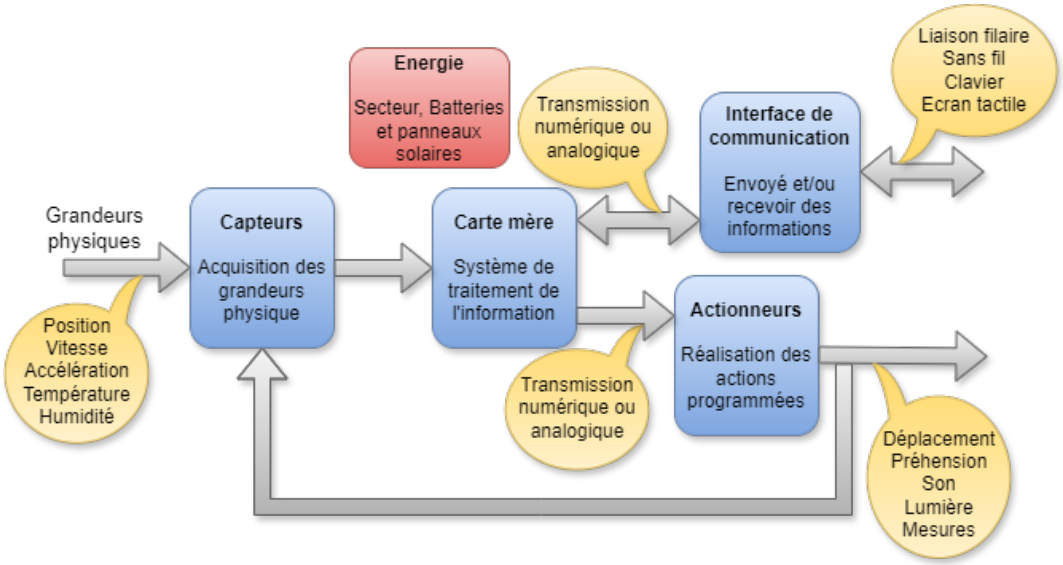


FIGURE 1.9 – Schéma d'illustration des éléments constituant d'un robot.

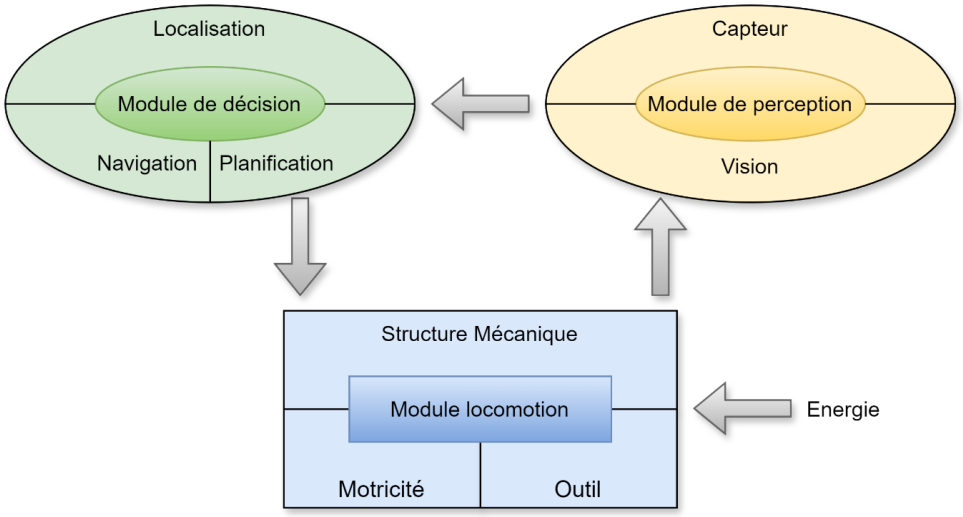


FIGURE 1.10 – Architecture modulaire d'un robot mobile.

1.4.1 Capteurs comme sources d'informations

Un capteur est un organe de prélèvement d'information qui élabore à partir d'une grandeur physique, une autre grandeur physique de nature différente (très souvent électrique). Cette grandeur représentative de la grandeur prélevée est utilisable à des fins de mesure ou de commande [24]. La commande des robots mobiles est basée sur deux types d'informations importantes ; les informations proprioceptives et les informations extéroceptives [20, 16]. Le système de perception est très important pour la sécurité du robot si l'environnement est encombré d'obstacles fixes ou bien mobiles (autres robot). Pour se focaliser sur le problème de navigation, nous allons nous restreindre dans ce chapitre aux capteurs utiles pour la tâche de navigation [17].

Capteurs intéroceptifs

Fournissent des données sur l'état interne du robot (vitesse, position, orientation, ...). Ces informations renseignent le robot en cas de mouvement, sur son déplacement dans l'espace (la localisation). Ce sont des capteurs que l'on peut utiliser directement, mais ils souffrent d'une dérive au cours du temps qui rend leur utilisation seule inefficace ou avec limitation. Nous citons par exemple : l'odomètre, radar doppler, systèmes inertiels, [20, 15].

Capteurs extéroceptifs

Ont pour objectif d'acquérir des informations sur l'environnement proche du véhicule. Ils fournissent des mesures caractéristiques de la position que le robot peut acquérir dans son environnement par la détection des objets qui contourne. Ces informations peuvent être de natures très variée. Nous citons comme exemple les télémètres à ultrason, infra-rouge, laser, les caméras,... etc.
Pour la navigation autonome d'un robot mobile et selon la mission visée (à accomplir), on peut utiliser aussi les capteurs suivants [15, 17, 20, 42] :

- Les capteurs tactiles : qui sont le plus souvent utilisés pour des arrêts d'urgence lorsqu'ils rencontrent un obstacle qui n'avait pas été détecté par le reste du système de perception.
- Les boussoles : permettant par la mesure du champ magnétique terrestre de déduire la direction du nord. Ces capteurs peuvent utiliser différentes technologies et ont l'avantage de fournir une direction de référence stable au cours du temps.
- Les balises : dans certaines applications, il est également possible d'utiliser des balises dont on connaît la position et qui pourront être facilement détectées par le robot afin de faciliter sa localisation. Le robot sera alors équipé d'une antenne directionnelle qui lui permettra de détecter la direction des différentes balises afin de déduire sa position.
- Le GPS (Global Positioning System) : est un système de balises universel dont les balises sont placées sur des satellites en orbite terrestre. Ce système permet donc d'avoir une mesure de la position dans un repère global couvrant la terre avec une précision variant de quelques dizaines de mètres à quelques centimètres suivant les équipements utilisés.

1.4.2 Actionneurs

Les actionneurs sont des organes qui transforment l'énergie qui leur est fournie en un phénomène physique utilisable comme des mouvements. Si on se limite aux actionneurs

pratiquement utilisables, il est possible de les classer suivant :

- Le type du mouvement généré : Dans l'état actuel de la technologie, on trouve les actionneurs linéaires qui développent une force et génèrent un mouvement de translation parallèlement à cette force et les actionneurs rotatifs qui développent un couple et génèrent un mouvement de rotation autour de l'axe du couple. Comme exemple, les moteurs électriques rotatifs sont fréquemment associés à des réducteurs mécaniques à engrenages.

- La nature de la source d'énergie : On dispose d'actionneurs pneumatiques qui utilisent l'air comprimé comme source d'énergie, d'actionneurs hydrauliques sous pression et d'actionneurs électriques qui utilisent l'énergie électrique. La puissance massique et le pouvoir d'accélération sont des critères importants qui permettent une comparaison objective de ces différents types d'actionneurs. Comme exemple, les vérins hydrauliques reliés par une tuyauterie à des pompes fournissent des pressions élevées[24].

1.4.3 Unité de traitement

Plus connue sous le nom de carte mère, elle gère l'ensemble des tâches. Il admet trois rôles essentiels :

- Le rôle de l'information qui consiste à collecter l'information venant des capteurs.

- Le rôle de la décision : en partant d'une tâche définie et en tenant compte des données du système et de l'environnement, il établit les actions adéquates.

- Le rôle de la communication

1.5 Navigation autonome d'un robot mobile

L'autonomie peut se définir par l'aptitude du robot à réaliser seul sa mission. Le robot autonome peut réagir à n'importe quel environnement (statique ou dynamique) qu'il doit percevoir à l'aide de capteurs et agir à l'aide d'un ensemble d'actionneurs (Figure 1.11). Depuis le début des années 1960, les recherches de la navigation sur les robots mobiles ont progressivement augmenté. Aujourd'hui, les robots mobiles autonomes effectuent des tâches diverses avec une vitesse élevée et une précision avec sécurité. L'article [43] résume les différentes techniques utilisées pour la navigation des robots mobiles autonomes. Parmi ces techniques, nous trouvons les méthodes de commande avec la logique floue qui peuvent être utilisées pour faire face à l'incertitude de l'environnement lorsqu'il est dynamique. Le robot est ainsi réactif pour agir rapidement face aux changements de l'environnement. Il est également capable de prédire l'état futur de son environnement [44]. Il y a aussi les

méthodes de navigation par radio dont l'efficacité a été prouvée pour la navigation autonome en utilisant le système RFID [45].Chaque robot autonome a une certaine capacité d'adaptation à un environnement inconnu et pour la réalisation de tâches prédéfinies. Ses capacités ont permis de s'intégrer dans le monde industriel, où l'être humain est remplacé dans les tâches difficiles, répétitives ou à risque. L'autonomie d'un robot peut être résumée par la connaissance du robot de la commande envoyée aux actionneurs à chaque pas de temps. Cette connaissance est dépendante de la mission à accomplir et les informations données par les différents capteurs. Il s'agit donc de construire une architecture de contrôle qui permet de définir un lien entre la perception et l'action connaissant l'objectif final [18]. L'objectif de la navigation d'un robot mobile est de permettre, à partir de la perception de l'environnement, de rechercher une trajectoire dans un espace libre entre un point initial et un point final (point cible), puis de se déplacer sur cette trajectoire. Si la perception se fait au niveau du robot, la décision de mouvement peut se prendre de façon partagée entre l'opérateur et le robot. L'autonomie peut se définir comme la capacité d'un système à réagir par lui-même à ses propres observations de l'environnement, sans intervention humaine. Les trois actions majeures de la navigation autonome sont : la perception, la décision et l'action.

- La perception représente l'aptitude du système à extraire, étudier et mettre en œuvre des données acquises par des capteurs pour permettre au robot de prendre les décisions et agir en fonction de l'environnement qui l'entoure [18]. Ces données permettent de mettre à jour un modèle de l'environnement (architectures hiérarchiques ou délibératives) ou peuvent être directement utilisées comme entrées de comportement de bas niveau (architecture purement réactive) [17]. Il existe deux types de perception : une globale et une locale. La perception locale désigne l'utilisation d'une information centrée sur le robot. Par exemple pour une perception visuelle, une seule image peut être prise. La perception globale désigne l'utilisation d'une séquence d'informations (une séquence d'images) et un repère de référence extérieur au robot [18].

- La décision permet au robot de planifier la séquence d'actions pour réaliser sa mission (planification de trajectoire libre d'obstacles par exemple). Cela se fait à partir de l'analyse de la perception [18].

- L'action consiste à mettre en œuvre des commandes ou des consignes aux actionneurs. Cette étape se fait généralement par des asservissements de bas niveau [18].

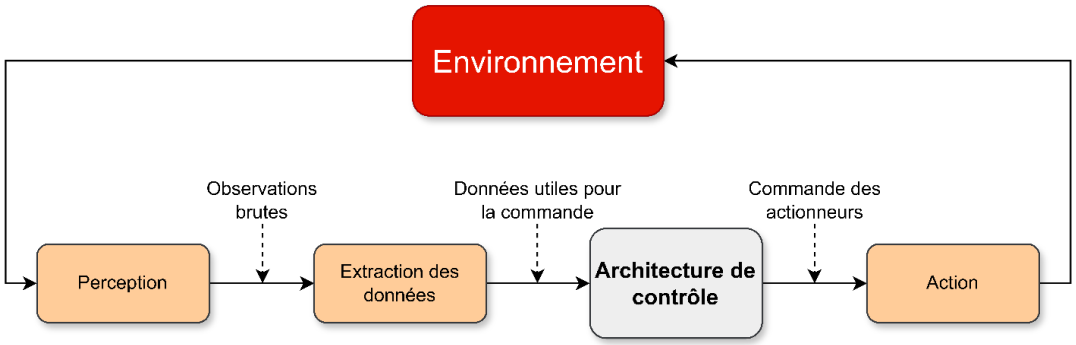


FIGURE 1.11 – Le schéma général pour la commande d'un robot autonome.

1.6 Architectures de contrôle

Une fois que les trois actions majeures à réaliser (perception, décision et action) sont définies, il faut établir les relations qui existent entre elles. C'est le rôle de l'architecture de contrôle. Elle permet de faire la coordination de différents processus internes d'un système robotisé. Les architectures de contrôle sont classées en trois catégories : délibérative, réactive et hybride. La Figure 1.12 schématise ces différentes approches. Chacune présente de nouveaux concepts et solutions pour résoudre le problème de navigation. Les sections qui suivent présentent chaque architecture. Pour plus de détails sur les architectures de commande, une synthèse complète a été réalisée dans [18, 46].

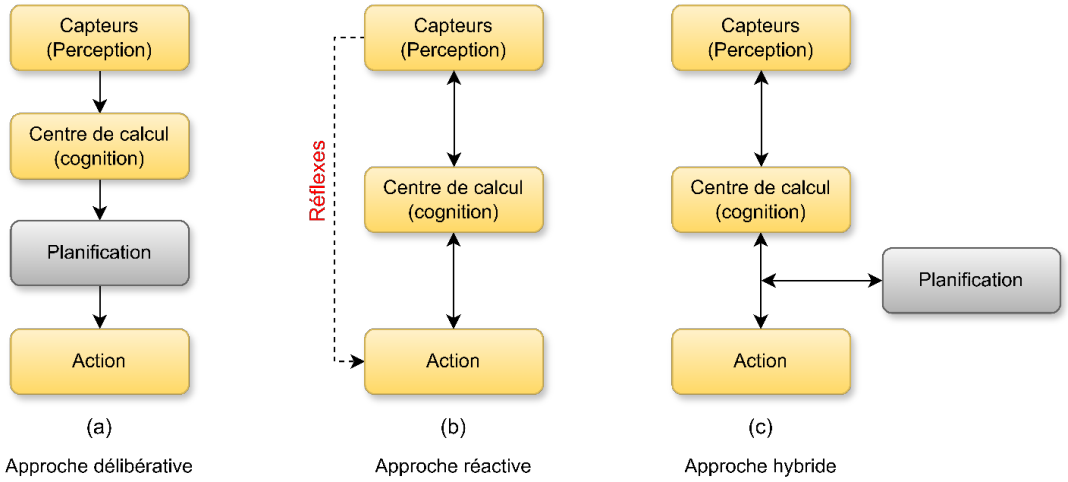


FIGURE 1.12 – Les différentes architectures de contrôle.

1.6.1 Contrôleur hiérarchique

L'approche délibérative (Figure 1.12(a)) dite aussi "hiérarchique" consiste à relier d'une manière séquentielle les trois actions : percevoir, planifier et agir [47].Après avoir perçu son environnement local, le système robotique doit toujours mettre à jour sa perception globale (une carte de l'environnement par exemple) et puis passer à la planification. La

planification pour la navigation est une étape très importante pour les robots autonomes. Elle permet à un robot mobile de calculer une trajectoire pour atteindre une cible tout en respectant des contraintes qui peuvent être liées à l'environnement (évitement de collision) ou la dynamique du robot. La planification délibérative recherche un chemin optimal et génère une trajectoire pour atteindre une cible, cette étape sera détaillée dans la section 1.7. Ensuite, le système d'exécution est appliqué pour effectuer une action dans le contexte du modèle statique de l'environnement et du système de planification pour accomplir une tâche donnée [48]. Cette architecture nécessite un modèle de l'environnement précis [18].

1.6.2 Contrôleur réactif

L'approche réactive (Figure 1.12(b)) a été développée à partir des années 1980 pour faire face aux limites de l'approche délibérative dans des environnements dynamiques et inconnus [49]. C'est une approche généralement utilisée en temps réel. Elle ne nécessite pas de modéliser l'environnement ni de planifier une trajectoire à l'avance. Il s'agit d'agir directement après la perception [50]. Cette architecture génère des commandes de contrôle basées sur l'environnement actuellement perçu.
Le contrôle robotique basé sur une approche réactive produit des performances robustes dans des environnements complexes et dynamiques. Les hypothèses prises pour déployer les systèmes purement réactifs sont que [18] :

- l'environnement est caractérisé par un manque de cohérence temporelle et de stabilité
- il est difficile de localiser un robot par rapport à un repère fixe tout au long de la mission
- la connaissance symbolique du monde représentatif a peu ou pas de valeur

1.6.3 Contrôleur hybride

L'architecture hybride (Figure 1.12(c)) a été initialement développée par [51] et [52]. Cette approche combine l'aspect de temps réel et la rapidité des réponses retrouvées dans l'approche réactive dans des environnements dynamiques ou inconnus, et l'efficacité de la planification de l'approche délibérative. L'architecture de contrôle hybride a été développée pour présenter de nouvelles approches en vue d'obtenir des systèmes de contrôle de supervision qui utilisent des architectures de contrôle réactives et délibératives [53, 54]. Ces travaux ont utilisé l'architecture hybride avec les trois étapes (modélisation, planification et réaction). Il s'agit d'un système de commande basé capteur pour la navigation des robots mobiles sans collision dans des environnements inconnus, encombrés et dynamiques [18].

1.7 Planification de chemin

La planification de chemin se définit par la recherche d'un mouvement libre de toute collision entre un robot possédant un nombre variable de degrés de liberté et son environnement. Ce chemin se définit par un point de départ et un point d'arrivée [55]. La planification de chemin dépend en grande partie de l'environnement dans lequel le robot va se déplacer. L'environnement pour un robot mobile est constitué d'un espace libre (navigable) et un espace occupé (non navigable) par des obstacles. Le robot doit détecter avec sa perception ces deux espaces pour pouvoir bien naviguer. Avant d'entamer la recherche d'un chemin approprié pour le robot il faut modéliser son environnement pour distinguer les espaces navigable et non navigable. C'est-à-dire qu'il faut construire une carte de l'environnement. Il existe plusieurs types de cartes (topologique, métrique, hybride...). Une fois les cartes acquises, elles permettent diverses fonctions clés nécessaires à la navigation de robot mobile, telles que la localisation, la planification des chemins, l'évitement des collisions et la détermination de cibles d'intérêt [56, 18]. Dans cette partie il faut distinguer ces trois termes : mouvement, chemin et trajectoire.

- **Un mouvement** est la manière de relier un point A à un instant initial à un point B à un instant final.
- **Un chemin** est le support de mouvement, il peut par exemple être représenté comme une courbe paramétrée en fonction d'une abscisse curviligne.
- **Une trajectoire** est le chemin défini en fonction de temps, c'est-à-dire l'évolution du paramètre de l'abscisse curviligne en fonction du temps.

L'étape de planification nécessite une modélisation de l'environnement. Différentes représentations existent pour modéliser un environnement et seront présentés dans la section suivante. Plusieurs méthodes de planification de trajectoires seront détaillées dans la partie 1.7.2.

1.7.1 Représentations de l'environnement

Pour permettre à un robot mobile de naviguer dans un environnement inconnu à priori, il faut construire une représentation de l'environnement à partir de la perception. Il est aussi très important dans une navigation autonome que le robot se localise par rapport à son environnement. La modélisation de ce dernier est donc indispensable.

Cartes topologiques

Une carte topologique mémorise un ensemble de régions (bureaux, couloirs, intersections, rues, champs...) ainsi que les liens qui existent entre ces régions (Figure 1.13). Cette

carte est représentée comme un graphe. Ce dernier est constitué de noeuds qui sont les régions (lieux) que le robot peut atteindre et d'arêtes qui permettent de lier les noeuds (chemins) [57, 58]. L'avantage de cette représentation est qu'elle ne requière pas un modèle métrique issu de la fusion de données de capteurs proprioceptifs.

Cartes métriques

La carte métrique représente les caractéristiques d'objets perçus (position, longueur, distance à un autre objet) dans un espace métrique (Figure 1.13). Ce dernier est généralement exprimé en trois dimensions et définit un référentiel unique pour toutes les informations perçues. Les objets détectés dans cette carte correspondent aux obstacles que le robot pourrait rencontrer dans son environnement et représentent l'espace occupé. Le complémentaire correspond alors à l'espace navigable, c'est-à-dire, à l'espace dans lequel le robot peut se déplacer. Cette vision objective de l'environnement, plutôt indépendante de tout robot donné, facilite également la réutilisation de ces cartes par différents robots. Pour l'être humain, ces cartes sont plus faciles à comprendre car elles correspondent à sa propre perception [59]. Les cartes métriques sont plus faciles à construire que les cartes topologiques en raison de la définition non ambigüe des lieux fournie par leurs coordonnées [60].

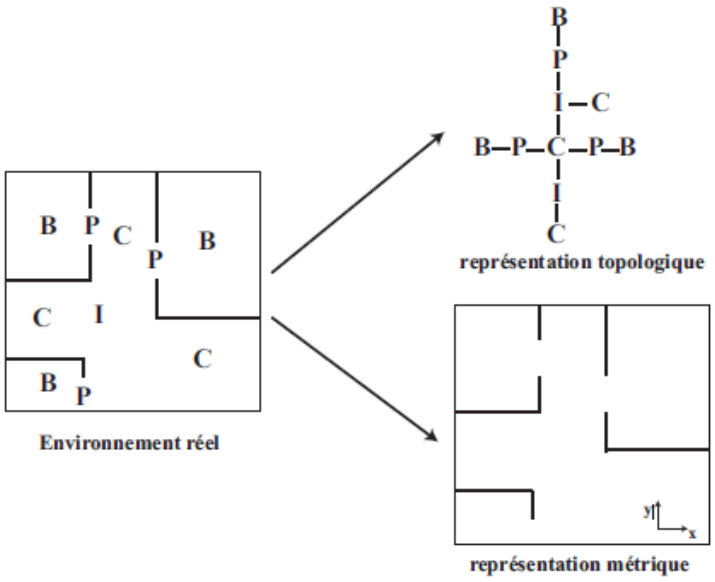


FIGURE 1.13 – Les représentations de l'environnement : la représentation topologique mémorise des lieux et des liens de déplacement. Dans cet exemple B=Bureau, P=Porte, C=Couloir, I=Intersection. La représentation métrique mémorise des objets (murs) avec une position dans un repère global [18].

Grilles d'occupation

Les cartes basées sur les grilles d'occupation sont des représentations métriques discrétisées de l'environnement du robot. Elles représentent des environnements par des cellules métriques qui contiennent des informations relatives à l'occupation de l'environnement. Chaque cellule contient un indice qui indique si l'espace correspondant est plutôt libre ou occupé. Cette représentation requiert très peu de temps de calcul, ce qui revient à une mise à jour d'une carte rapide et facile [59]. Le principe d'inférence bayésienne [61] et les fonctions de croyance [62] peuvent être utilisé pour estimer la probabilité d'occupation de chaque cellule.

Cartes hybrides

La représentation basée sur les grilles produit des cartes métriques précises et peut être utilisée pour une localisation plus précise et un évitement d'obstacles. Cependant, si une représentation métrique est utilisée dans des environnements de grandes tailles, la complexité de l'espace et du calcul deviendra non négligeable. La représentation topologique peut être utilisée beaucoup plus efficacement pour la planification de chemins. La précision et la cohérence des données sont difficiles pour la carte topologique dans un environnement à grande échelle. Ainsi, la carte hybride qui combine les deux approches devient plus appropriée pour représenter l'environnement. Cela va permettre de surpasser les limitations de chaque représentation et d'exploiter les avantages de chacune. Les cartes hybrides sont des cartes topologiques qui contiennent des informations métriques sur les arêtes et les noeuds du graphe [63, 64].

1.7.2 Méthodes de planification de trajectoire

Pour résoudre les problèmes de navigation plusieurs méthodes ont été développées et sont entrain d'être améliorées, elles peuvent se regrouper en deux grandes familles. La figure 1.15 résume les différentes méthodes de navigation [18, 65].

Méthodes classiques

Après avoir modélisé l'environnement à partir de la perception du robot, il faut se déplacer en évitant les obstacles. La détermination de ce déplacement se traduit par la recherche d'un chemin permettant au robot d'être le plus loin possible des collisions [18]. Ce sont des méthodes de planifications qui assurent cette tâche.
En outre, la planification consiste à calculer à partir du modèle de l'environnement les chemins joignants l'emplacement de départ du robot à son emplacement final, tout en respectant ses contraintes cinématiques, c'est-à-dire quelles sont réalisables par le robot [66], en évitant les obstacles fixes [67] ou mobiles [68]. Les approches les plus connues

pour définir des trajectoires acceptables sont : la décomposition en cellules, le champ de potentiel, la bande élastique, la fenêtre dynamique, le RoadMap, etc.

- **Graphe de Voronoï**

Le graphe de Voronoï (GV) est une méthode très utilisée pour la représentation d'une configuration de l'espace navigable. L'idée de base est de générer une ligne, appelée arête de Voronoï, équidistante de tous les points (Figure 1.14(a)). Le point où plusieurs arêtes de Voronoï se rencontrent est connu comme un sommet de Voronoï. Ce dernier représente généralement une correspondance physique aux configurations qui peuvent être détectées dans l'environnement. Cela rend la tâche de suivi de chemin beaucoup plus facile. S'il existe une stratégie implicite de contrôle local pour rester équidistant de tous les obstacles (le robot suit l'arête de Voronoï), alors il n'entrera pas en collision avec les obstacles car il restera au "milieu" de l'espace libre. Lorsque les obstacles sont des polygones, le diagramme de Voronoï se compose de segments droits et paraboliques [52, 69, 70]. Cette méthode est plus détaillée dans la section 1.4.3.1 de la thèse [18].

- **Graphe de Visibilité**

La méthode du graphe de visibilité est l'une des premières méthodes de planification de chemin. Elle utilise les extrémités des obstacles que le robot doit éviter. Elle s'applique principalement aux espaces de configuration bidimensionnels avec une région polygonale Cobstacle. Le graphe de visibilité est un graphe G dont les noeuds sont les configurations initiale et finale ainsi que tous les sommets des obstacles. Les liens de G sont les lignes qui relient deux noeuds qui ne coupent pas l'intérieur de la région Cobstacle [71] (Figure 1.14(b)).

- **Décomposition cellulaire**

La méthode de décomposition cellulaire consiste à décomposer l'espace des configurations libres du robot en un certain nombre d'ensembles disjoints appelés cellules (Figure 1.14(c)). Le graphe de connectivité est un élément important pour cette décomposition, il capture la structure de l'espace de configuration. Chaque cellule est représentée comme un noeud dans ce graphique. Deux noeuds sont connectés par un bord si et seulement si les deux cellules correspondantes sont adjacentes. La planification du chemin entre deux configurations est réalisée en deux étapes :

1. Recherche du graphe de connectivité et du chemin qui relie les deux configurations choisies.

2. Détermination de la résolution du problème de planification en utilisant les cellules adjacentes trouvées dans la première étape.

Cette approche a été introduite par [72] puis utilisée dans plusieurs travaux [73, 74]. Elle peut être utilisée pour la planification de plusieurs trajectoires de plusieurs robots comme il a été présenté dans [75, 18].

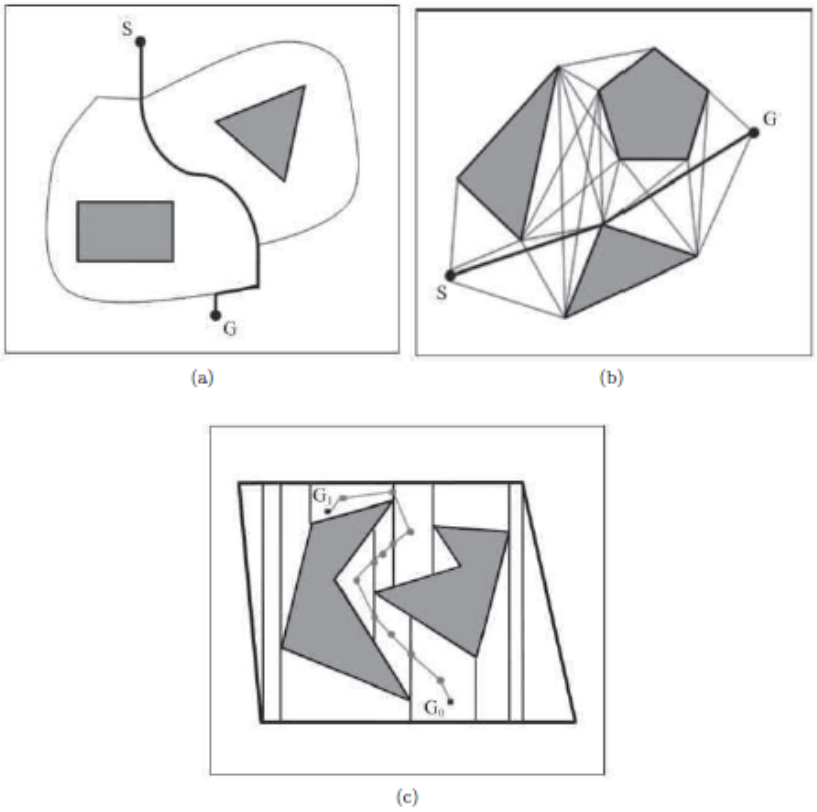


FIGURE 1.14 – Les différents graphes pour la planification de trajectoire extraits de [18] (a) Graphe de Vononoi (b) Graphe de visibilité et (c) décomposition cellulaire.

- **Champs de potentiel**

La méthode de planification par les champs potentiels a été introduite par [76]. Il s'agit de définir une fonction potentielle dans l'espace libre qui présente un minimum global à la configuration de l'objectif et de suivre sa plus forte pente. La fonction de potentiel peut être définie sur l'espace libre comme la somme d'un potentiel attractif tirant le robot vers la configuration d'une cible et d'un potentiel répulsif éloignant le robot des obstacles. Pour réaliser cela, l'espace dans lequel le robot doit naviguer est traité comme un champ de potentiel. La position cible pour le robot est considérée comme un minimum global et les obstacles sont considérés comme des maxima globaux [77].

1.7.3 Méthodes de Soft Computing

Le soft computing a été introduit par L.A. Zadeh comme un moyen pour le développement des systèmes intelligents répondant à des critères d'efficacité, de robustesse, de facilité d'implémentation et d'optimisation des temps de calcul, etc., mais aussi prenant

en considération l'aspect humain fréquemment présent dans les systèmes. Ainsi, les composantes essentielles du soft computing sont la logique floue [78], les réseaux neuronaux [79], le raisonnement probabiliste et les méthodes d'optimisation telles que les algorithmes génétiques [80]. Leur point de départ est la gestion d'incertitude et d'imprécision intrinsèque à la majorité des problèmes. L'avantage du soft computing est l'hybridation de plusieurs de ces composantes afin de bénéficier des avantages de chacune et compenser les inconvénients avec l'emploi d'une autre dont les propriétés sont complémentaires. Par exemple, la capacité d'apprentissage des réseaux neuronaux correspond à une carence en termes d'expressivité des résultats qu'on compense par l'utilisation de la logique floue qui utilise les descriptions linguistiques, ce qui a donné naissance aux réseaux neuro-flous. La figure suivante illustre les méthodes de navigation.

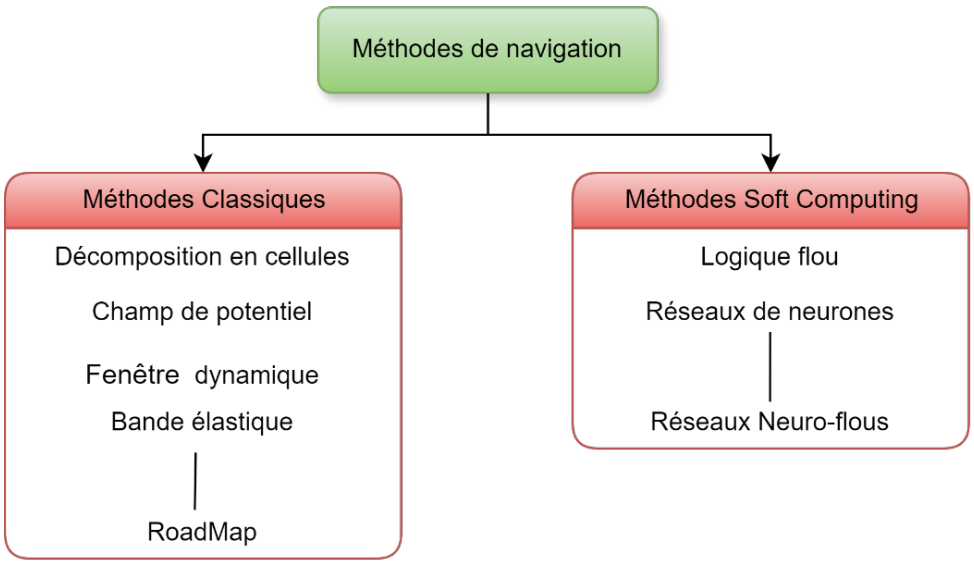


FIGURE 1.15 – Les méthodes de navigation.

1.8 Conclusion

La navigation autonome des robots mobiles est un domaine de recherche actif depuis les quatre dernières décennies. La navigation permet au robot mobile de se déplacer dans son environnement d'un point A à un point B avec évitement des obstacles. Une navigation sûre et autonome nécessite une bonne perception de l'environnement. Pour cela, un robot mobile doit être équipé d'un ou plusieurs capteurs. Une fois son environnement perçu, il peut extraire des informations nécessaires pour calculer son déplacement.
Ce chapitre a présenté la littérature et les étapes fondamentales à la navigation autonome ont été présentées (la perception, la planification et l'action), et la présentation des trois grandes architectures de contrôle des robots mobiles en mentionnant leurs avantages et inconvénients, l'architecture réactive a été retenue parce qu'elle est la mieux adaptée pour les environnements inconnus et incertains.

Ensuite nous sommes passés aux méthodes de navigation qui se divisent en méthodes classiques comme la méthode du champ de potentiel ... etc, et les méthodes dites soft computing comme les réseaux neuro-flous que nous avons utilisés également dans notre travail, on a utilisé cette dernière méthode vu que nous travaillons sur des environnements inconnus par le robot et on n'a pas besoin d'avoir une représentation de l'environnement ce qui augmente considérablement la réactivité du robot et le temps de calcul.

Chapitre 2

Modélisation d'un robot mobile à entrainement différentiel et validation des modèles par des contrôleurs cinématique et dynamique

2.1 Introduction

La modélisation d'un robot mobile muni de ses actionneurs et capteurs et des divers phénomènes physiques qui apparaissent lors de son fonctionnement est un principe ou une technique qui permet d'établir un modèle explicite en recensant les variables ou les facteurs explicatifs et l'importance relative de chacune de ces variables. Dans ce chapitre, nous utilisons un robot mobile à entrainement différentiel (DDWMR) ; Dans une première partie de ce chapitre, nous présentons deux types de modélisation à savoir : la modélisation cinématique et la modélisation dynamique, pour exprimer le mouvement du robot mobile par des équations de vitesse et d'accélération ; En deuxième partie, nous présentons une commande par deux contrôleurs imbriqués dont le premier c'est un contrôleur cinématique qui assure la poursuite de trajectoire et le deuxième c'est un contrôleur dynamique qui assure l'asservissement des vitesses linéaire et angulaire du robot mobile. On terminera par une présentation des résultats de simulation et une conclusion.

2.2 Modélisation du robot mobile à entrainement différentiel de type unicycle

La modélisation mathématique est une étape très importante pour la commande des robots ; Deux types de modèles sont généralement utilisés lors de la commande, à savoir : le modèle cinématique et le modèle dynamique.

2.2.1 Présentation des systèmes de coordonnée du robot

Dans le cadre de notre travail, nous utiliserons un robot à entrainement différentiel (DDMR) qui est montré dans la Figure 2.1 :

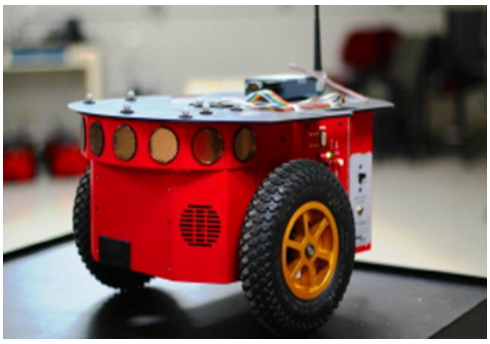


FIGURE 2.1 – robot mobile à entrainement différentiel (DDMR)[81].

Pour la modélisation du robot mobile au premier lieu il faut définir les systèmes de cordonnées tel que $[X_I, Y_I]$ représentent les deux axes du repère initial fixe quelconque, dont l'axe $\vec{z}$ est vertical et $[X_r, Y_r]$ est un repère mobile lié au robot mobile. On choisit généralement pour A un point remarquable de la plate-forme, typiquement le centre de l'axe des roues motrices s'il existe. Par conséquent la répartition des repères appropriés à un robot mobile est illustré dans la Figure (2.2).

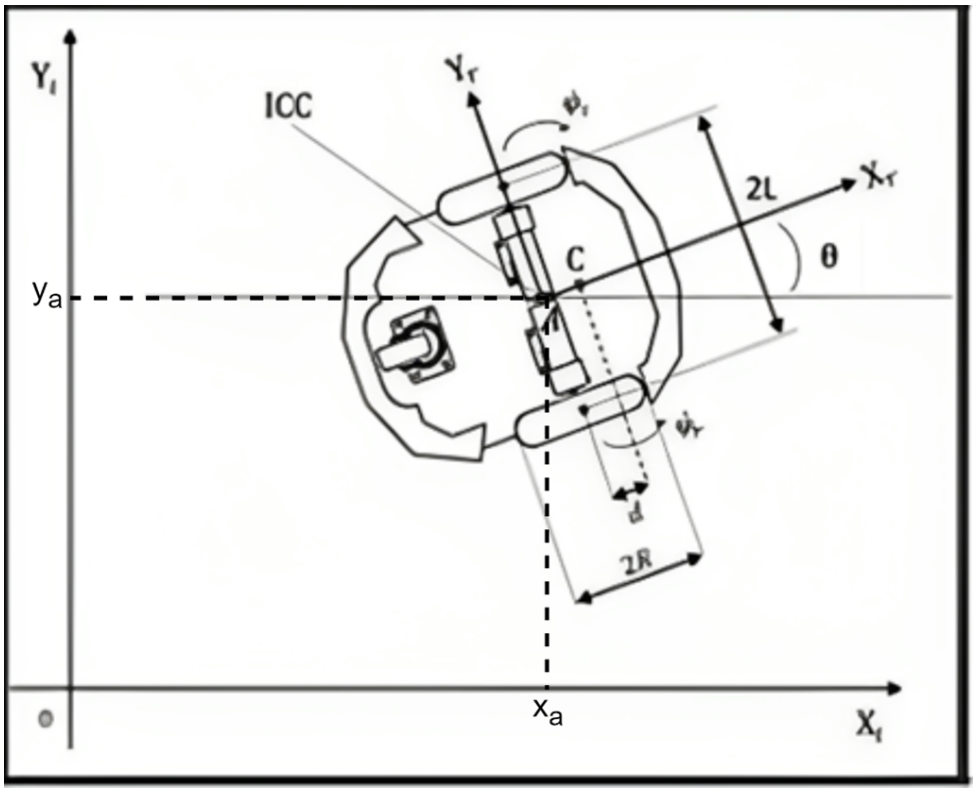


FIGURE 2.2 – Système de repérage [81].

Comme le montre la Figure (2.2), la position et l'orientation du robot dans le repère initial peuvent être définies comme suit :

$$q^I = \begin{bmatrix} x_a \\ y_a \\ \theta \end{bmatrix} \tag{2.1}$$

Les deux repères initial et mobile sont lié entre eux. La position de n'importe quel point sur le robot mobile peut être définie dans le repère de ce dernier et le repère initial comme suit : Soit : $X^r = \begin{bmatrix} x^r \\ y^r \\ \theta^r \end{bmatrix}$ et $X^I = \begin{bmatrix} x^I \\ y^I \\ \theta^I \end{bmatrix}$ sont les coordonnées du point donné dans le repère du robot mobile et le repère initial, respectivement. Alors, les deux coordonnées sont liées par la transformation suivante :

$$X^I = R(\theta)X^r \tag{2.2}$$

Tel que $R(\theta)$ est la matrice de rotation orthogonale donnée par cette expression :

$$R\,(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Cette transformation permettra également la gestion du mouvement entre les repères.

$$\dot{X}^I = R(\theta)\dot{X}^r \tag{2.4}$$

On verra dans la section suivante que l'équation (2.4) est très importante dans la dérivation des modèles cinématiques et dynamiques DDMR tels qu'ils décrivent la relation entre les vitesses dans les repères initial et mobile.
La position du robot est définie par un vecteur de cordonnées généralisées $q = (x; y; \theta; \varphi_R; \varphi_L)$
Tel que :
A : l'origine du repère mobile qui est le centre des deux roues sur l'axe des roues.
C : centre de gravité du robot (centre de masse).
L : distance entre A et le centre de la roue.
d : distance entre C et A.
θ : L'angle entre X_I et X_r.
R : rayon de chaque roue.
$\dot{\varphi}_R \; et \dot{\varphi}_L$:vitesses de rotation des roues droite et gauche respectivement.
CIR : Centre instantané de rotation.
ρ : rayon de courbure de la trajectoire du robot (distance du CIR au centre A).
ω : vitesse de rotation autour du CIR.
Les deux roues motrices ayant le même axe de rotation, le centre instantané de rotation (CIR) du robot est un point de cet axe, voir la Figure (2.3).

$$\begin{cases} v_R = r\dot{\varphi}_R = (\rho + L)\,\omega & (2.5) \\ v_L = r\dot{\varphi}_L = (\rho - L)\,\omega & (2.6) \end{cases}$$

En faisant (2.5) $+$ (2.6) et (2.5) $-$ (2.6), on obtient :

$$\begin{cases} (2.5) + (2.6) \Leftrightarrow r\,(\dot{\varphi}_R + \dot{\varphi}_L) = 2\rho\omega & (2.7) \\ (2.5) - (2.6) \Leftrightarrow r\,(\dot{\varphi}_R - \dot{\varphi}_L) = 2L\omega & (2.8) \end{cases}$$

(2.8) donne $\omega = \frac{r(\dot{\varphi}_R - \dot{\varphi}_L)}{2L}$ est la vitesse de rotation du robot autour du CIR.
Si $\dot{\varphi}_R = \dot{\varphi}_L$ le robot se déplace en ligne droite.
Si $\dot{\varphi}_R = -\dot{\varphi}_L$ le robot effectue une rotation sur lui-même.
Remplacer ω dans (2.7)

$$\Rightarrow \rho = L\frac{\dot{\varphi}_R + \dot{\varphi}_L}{\dot{\varphi}_R - \dot{\varphi}_L} \tag{2.9}$$

ρ Qui permet de situer le CIR sur l'axe des roues FIgure (2.3).
Dans l'étude de la cinématique, seules les vitesses sont prises en compte. Le mouvement d'un robot mobile est caractérisé par deux contraintes cinématiques non holonome à savoir : Aucun glissement latéral, roulement sans glissement.

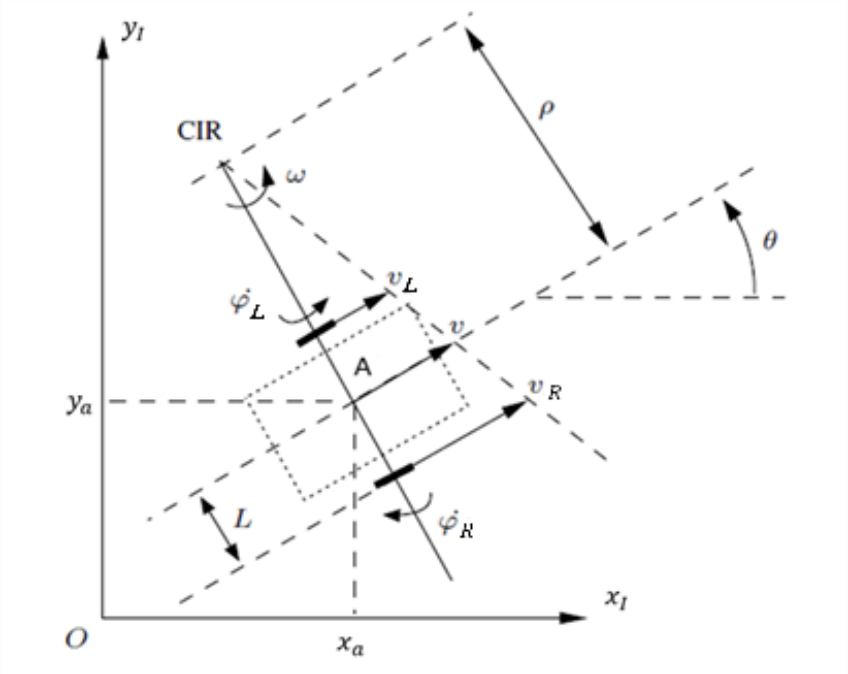


FIGURE 2.3 – Centre instantané de rotation d'un robot de type unicycle [81].

2.2.2 Hypothèses

Rappels sur la notion de non-holonomie [83] :

Définition 1 : : Étant donné un système mécanique dont l'espace de configuration est une variété différentielle Q de dimension n, on appelle contrainte cinématique une contrainte sur les vitesses du type :

$$\langle a(q), \dot{q} \rangle = 0, \forall\ q \in \mathcal{U}(q_0)$$

Avec $\mathcal{U}(q_0)$ un voisinage du point $q_0 \in Q$, et $a(\bullet)$ une forme différentielle (ou champ de covecteurs) de Q dans R^n.

Parmi les contraintes cinématiques du système, il faut distinguer celles qui sont en fait intégrables et qui peuvent, de ce fait, se ramener à des contraintes sur l'état seulement.

Définition 2 : Une contrainte cinématique $\langle a(q), \dot{q} \rangle = 0$ est dite contrainte intégrable s'il existe une fonction régulière $h : Q \longrightarrow R$ telle que $h = a$. Dans ce cas la contrainte est équivalente à la relation statique $h(q) = cste$.

Théorème 1.1.1 (Théorème d'intégrabilité de Frobenius) : Soient un système mécanique sur une variété Q de dimension, soumis à $k \leq n$ contraintes cinématiques indépendantes.

$$\langle a(q), \dot{q} \rangle = 0, \forall\ q \in \mathcal{U}(q_0), \forall\ i = 1, \ldots,\ k$$

Et $X_1, \ldots, X_{n-k}$ une famille de champs de vecteurs sur $\mathcal{U}(q_0)$ orthogonaux aux a_i . Si $Lie(X_i)(q)$ est de dimension $n - p$ pour tout$q \in \mathcal{U}(q_0)$ alors il existe p contraintes

intégrables (au sens où l'on peut trouver des fonctions scalaires $h_1, \ldots, h_p$ indépendantes et des fonctions $\lambda_{1,1}, \ldots, \lambda_{p,n}$ telles que $dh_i(q) = \sum_{j=1}^{n} \lambda_{ij}(q) a_j(q)$.
Nous sommes à présent prêts à définir un système non holonome.
Définition 3 : Un système non holonome est un système mécanique soumis à un ensemble de contraintes cinématiques suivantes :

$$\langle a(q), \dot{q} \rangle = 0, \forall\ q \in \mathcal{U}(q_0), \forall\ i = 1, \ldots,\ k$$

Ces contraintes doivent êtres indépendantes et non intégrables (au sens où il n'existe pas de contrainte intégrable).
En particulier, on appelle modèle cinématique du système le modèle décrit par cette relation :

$$\dot{q} = \bar{X}(q)\, v := \sum_{i=1}^{n-k} X_i(q)\, v_i$$

Avec $\{X_1, \ldots, X_{n-k}\}$ une famille de champs de vecteurs telle que $\langle a(q), X_j(q) \rangle = 0, \forall\ q \in \mathcal{U}(q_0), \forall\ i = 1, \ldots,\ k$. La variable $v = (v_i, \ldots, v_{n-k}) = (v_i, \ldots, v_m) \in R^m$ est assimilable à une variable de commande en vitesse. On appelle m (la dimension de l'espace des vitesses instantanées possibles) le nombre de degrés de liberté du système.
Le mouvement d'un robot mobile à entraînement différentiel est caractérisé par deux équations de contraintes non holonomes, obtenues par deux principales hypothèses (Aucun glissement latéral, roulement sans glissement).

Pas de mouvement de glissement latéral

Cette contrainte signifie simplement que le robot ne peut se déplacer que selon un mouvement décrivant une courbe (avant et arrière) mais pas latéralement [84, 82]. Cela permet de déduire que la vitesse du robot mobile associée au point A (centre) est nulle le long de l'axe latéral dans le repère mobile, soit :

$$\dot{y}_a^r = 0 \tag{2.10}$$

En utilisant la matrice de rotation orthogonale $R(\theta)$, l'expression de la vitesse du robot mobile associée au point A dans le repère initial (fixe) est donnée comme suit :

$$\begin{bmatrix} \dot{x}_a^I \\ \dot{y}_a^I \\ \dot{\theta}_a^I \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_a^r \\ 0 \\ \dot{\theta}_a^r \end{bmatrix} \Rightarrow \begin{cases} \dot{x}_a^I = \dot{x}_a^r.cos\theta \\ \dot{y}_a^I = \dot{x}_a^r.sin\theta \end{cases} \tag{2.11}$$

Ainsi on obtient :

$$-\dot{x}_a^I sin\theta + \dot{y}_a^I cos\theta = 0 \tag{2.12}$$

La relation entre les vitesses du centre de gravité C et le centre de l'axe de la roue A est la suivante :

$$\begin{cases} y_c = y_a + dsin\theta \\ x_c = x_a + dcos\theta \end{cases} \tag{2.13}$$

$$\begin{cases} \dot{x}_c = \dot{x}_a - d\dot{\theta}sin\theta \\ \dot{y}_c = \dot{y}_a + d\dot{\theta}cos\theta \end{cases} \tag{2.14}$$

En utilisant la matrice de rotation orthogonale $R(\theta)$, la vitesse du robot mobile dans le référentiel inertiel peut être donnée par :

$$\begin{bmatrix} \dot{x}_c^I \\ \dot{y}_c^I \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_a^r \\ \dot{Y}_a^r = 0 \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} -d\dot{\theta}sin\theta \\ d\dot{\theta}cos\theta \\ 0 \end{bmatrix} \tag{2.15}$$

Avec l'équation de non-holonomie :

$$-\dot{x}_c sin\theta + \dot{y}_c cos\theta - d\dot{\theta} = 0 \tag{2.16}$$

Roulement sans glissement

La contrainte de roulement pure (sans glissement) représente le fait que chaque roue maintient un seul point de contact P avec le sol comme le montre la Figure (2.4). Il n'y a aucun glissement de la roue dans son axe longitudinal (X_r) et aucun dérapage dans son axe orthogonal (Y_r). Les vitesses appropriées aux points de contact dans le repère du robot mobile sont liées aux vitesses des roues par ces relations :

$$\begin{cases} v_{pR} = R\dot{\varphi}_R \\ v_{pL} = R\dot{\varphi}_L \end{cases} \tag{2.17}$$

Avec R représente le rayon de la roue.

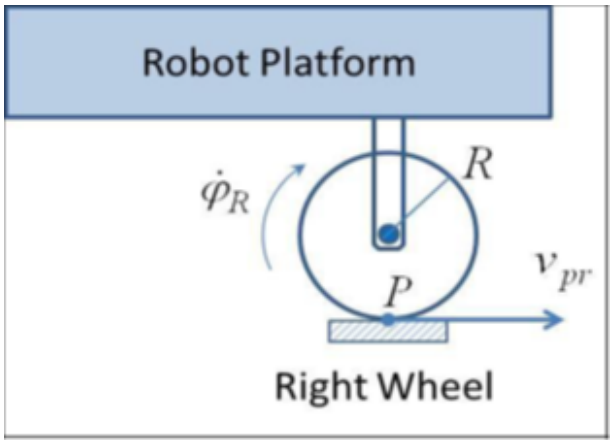


FIGURE 2.4 – Contact de la roue avec le sol [82].

Dans le repère initial, les coordonnées des roues peuvent être exprimées en fonction des coordonnées du centre du robot A $[x_a\ y_a]$ comme suit :
Roue droite :

$$\begin{cases} x_{pR} = x_a + Lsin\theta \\ y_{pR=} y_a - Lcos\theta \end{cases} \tag{2.18}$$

Roue gauche :

$$\begin{cases} x_{pL} = x_a - Lsin\theta \\ y_{pL=} y_a + Lcos\theta \end{cases} \tag{2.19}$$

Considérons une roue qui roule sans glisser sur un sol plan, comme le montre la Figure (2.5).
Considérons une roue qui roule sans glisser sur un sol plan, comme le montre la Figure (2.5). $R_I\left(O, \vec{i}_I, \vec{j}_I, \vec{k}_I\right)$ est un repère fixe associé au sol. $R_r\left(O_r, \vec{i}_r, \vec{j}_r, \vec{k}_r\right)$ est un repère associé à la roue tel que son origine coïncide avec le centre de la roue. L'axe $\vec{k}_r$ reste parallèle à celui du repère fixe $\vec{k}_I$, l'axe $\vec{j}_r$ coïncide avec l'axe de roulement de la roue et, finalement, l'axe $\vec{i}_r$ est donné par le produit $\vec{j}_r \wedge \vec{k}_r$. L'angle compris entre $\vec{i}_I$ et $\vec{i}_r$ est noté θ tandis que φ désigne l'angle entre $\vec{i}_r$ et $\vec{O_rc}$, avec c un point appartenant au périmètre de la roue. La vitesse angulaire de la roue autour de son axe central est notée $\dot{\varphi}$ et celle autour de l'axe vertical du vecteur directeur $\vec{k}_r$ est notée $\dot{\theta}$. On note (x, y) les deux premières coordonnées du centre O_r de la roue dans R_I, et r le rayon de la roue. La configuration de la roue est donnée par : $q = [x, y, \theta, \varphi]^T$[32],[83].

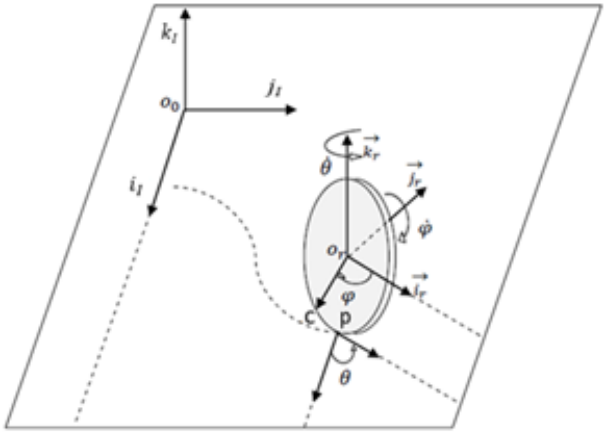


FIGURE 2.5 – Roue verticale qui roule sans glissement.

Finalement, P désigne le point de la roue en contact avec le sol [20] [24]. La contrainte de roulement sans glissement implique que la vitesse du point P par rapport au repère fixe R_I est nulle $\vec{V}_{p/R_I} = 0$. Soit $\vec{\omega}_{r/R_I} = \dot{\theta}\vec{k}_r + \dot{\varphi}\vec{j}_r$ le vecteur de vitesse angulaire de la roue par rapport au repère fixe R_I. Par application du théorème classique de composition de vitesses, on a :

$$\vec{V}_{p/R_I} = \vec{V}_{O_r/R_I} + \vec{\omega}_{r/R_I} \wedge \vec{O_rP} = \vec{0}$$

$$= \dot{x}\vec{i}_I + \dot{y}\vec{j}_I + \left(\dot{\theta}\vec{k}_r + \dot{\varphi}\left(-sin\theta\vec{i}_I + cos\theta\vec{j}_I\right)\right) \wedge \left(+r\vec{k}_I\right)$$

$$\vec{V}_{p/R_I} = \left(\dot{x} - r\dot{\varphi}cos\theta\right)\vec{i}_I + \left(\dot{y} - r\dot{\varphi}sin\theta\right)\vec{j}_I = 0$$

Il en résulte les deux contraintes cinématiques indépendantes :

$$\begin{cases} \dot{x} - r\dot{\varphi}cos\theta = 0 \\ \dot{y} - r\dot{\varphi}sin\theta = 0 \end{cases} \tag{2.20}$$

Ces deux contraintes mènent à l'équation suivant :

$$-\dot{x}sin\theta + \dot{y}cos\theta = \begin{bmatrix} -sin\theta & cos\theta & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = 0$$

Ce qui constitue une contrainte non holonome.
L'intégrabilité des deux contraintes (2.20) est testée en appliquant le théorème (1.1.1) [32, 83].

Avec : $[a_1(q), a_2(q)] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ -rcos\theta & -rsin\theta \end{bmatrix}$ $[X_1(q), X_2(q)] = \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \\ \frac{1}{r} & 0 \end{bmatrix}$

Après avoir vérifié que $lie\left(X_i(q)\right) = \mathbb{R}^4 \ \forall \ q\epsilon Q$ il en résulte que ces contraintes ne sont pas intégrables. On déduit également qu'un modèle cinématique de la roue (vertical) est donnée par :

$$\dot{q} = \overline{X}(q)\, v, \ \overline{X}(q) = \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \\ \frac{1}{r} & 0 \end{bmatrix}$$

Avec $v = (v_1, v_2)^T$, $v_1 = r\dot{\varphi}$ la vitesse de roulement et $v_2 = \dot{\theta}$ la vitesse angulaire de la roue autour de l'axe vertical.

2.2.3 Modélisation cinématique

On considère dans le cadre de ce travail un robot mobile à trois roues, dont deux roues motrices actionnées séparément et une roue libre pour assurer sa stabilité, ce qui traduit une conduite différentielle. Par conséquent le comportement de ce robot mobile peut être modélisé par les équations cinématiques pour obtenir un modèle unicycle. Il s'agit de l'un des modèles les plus simples de robot mobile à roues. En contrôlant les vitesses de rotation

des deux roues et notamment en jouant sur la différence de vitesse entre la roue droite et la roue gauche, il est possible de déplacer le robot mobile dans différentes directions à savoir : vers l'avant, vers l'arrière, à droite, et à gauche. Il est même possible de réaliser un demi-tour sur place. Un schéma illustratif des mouvements du robot et des vitesses associées est donné à la Figure (2.6) [85].

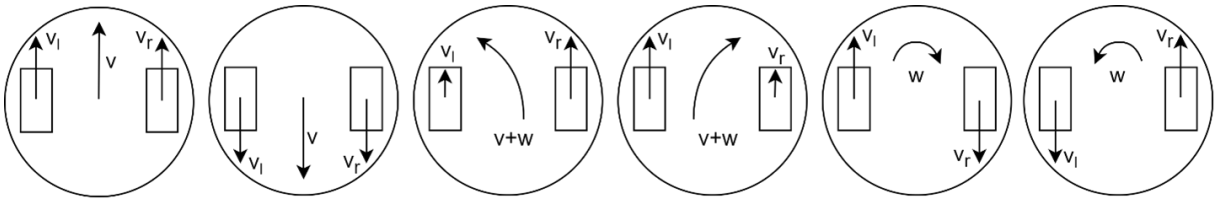


FIGURE 2.6 – schéma de différents déplacements possibles avec un robot mobile à roues de type DDMR.

On désigne par unicycle un robot mobile actionné par deux roues indépendantes et possédant éventuellement un certain nombre de roues folles assurant sa stabilité. Ces roues folles n'interviennent pas dans la cinématique du robot.
Ce type de robot mobile est très répondu en raison de sa simplicité de construction et de propriétés cinématiques intéressantes.

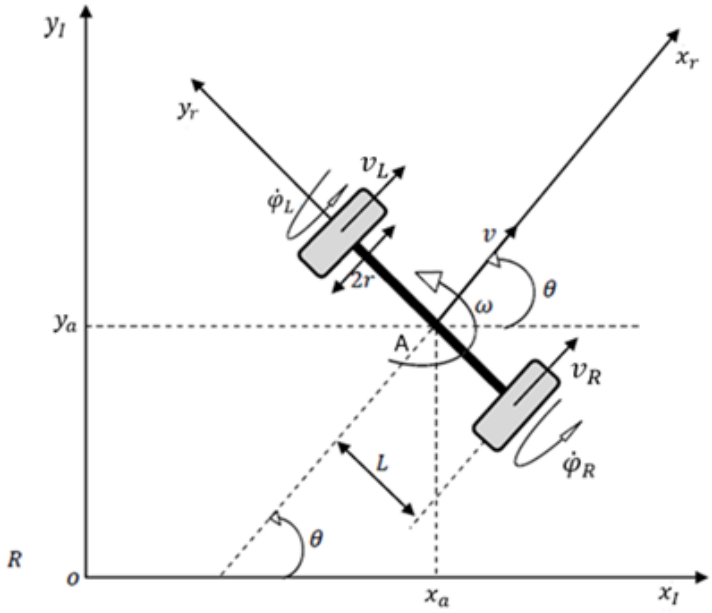


FIGURE 2.7 – Représentation cinématique d'un robot mobile de type DDMR.

Considérons le robot mobile de type unicycle schématise sur la Figure (2.7). Ce robot mobile est constitué de trois degrés de liberté, la position par les coordonnes $(x,\ y)$ et l'orientation par l'angle (θ). Par simplicité, le robot mobile est représenté par sa projection sur un plan parallèle au sol. L'unicycle est équipé de deux roues actionnées indépendamment [32]. Pour modéliser ce système nous considérons le point A situe au milieu des

deux roues motrices. Un repère $R_r$est associé à ce point, comme illustré sur la Figure (2.6). Un repère inertiel R_I est fixé au sol, dont l'axe $\vec{z}$ est vertical. Ainsi, un vecteur de configuration de l'unicycle incluant le train moteur est donné par :

$$q = (x_a, y_a, \theta, \varphi_R, \varphi_L) \in Q = \mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{R}^2$$

(x_a, y_a) représente les coordonnées du point A situé au milieu de l'axe des roues arrières.
θ : L'angle $\left(\vec{X}_I, \vec{X}_r\right)$ (l'angle d'orientation du robot).
φ_R : L'angle de rotation de la roue droite.
φ_L : L'angle de rotation de la roue gauche.
Soit r le rayon des roues et L la distance entre le point A et le point milieu de chaque roue. L'application des contraintes (2.20) à chacune des roues donne :
Roue gauche :

$$\begin{cases} \overline{\dot{x_a} - \dot{L}\sin\theta} - r\dot{\varphi}_L \cos\theta = 0 \\ \overline{\dot{y_a} + \dot{L}\cos\theta} - r\dot{\varphi}_L \sin\theta = 0 \end{cases} \tag{2.21}$$

$$\begin{cases} \dot{x}_a - L\dot{\theta}\cos\theta - r\dot{\varphi}_L \cos\theta = 0 \\ \dot{y}_a - L\dot{\theta}\sin\theta - r\dot{\varphi}_L \sin\theta = 0 \end{cases} \tag{2.22}$$

Roue droite :

$$\begin{cases} \overline{\dot{x_a} + \dot{L}\sin\theta} - r\dot{\varphi}_R \cos\theta = 0 \\ \overline{\dot{y_a} - \dot{L}\cos\theta} - r\dot{\varphi}_R \sin\theta = 0 \end{cases} \tag{2.23}$$

$$\begin{cases} \dot{x}_a + L\dot{\theta}\cos\theta - r\dot{\varphi}_R \cos\theta = 0 \\ \dot{y}_a + L\dot{\theta}\sin\theta - r\dot{\varphi}_R \sin\theta = 0 \end{cases} \tag{2.24}$$

Soit encore : (2.22) et (2.24) :

$$\begin{cases} (c_1)\dot{x}_a - L\dot{\theta}\cos\theta - r\dot{\varphi}_L \cos\theta = 0 \\ (c_2)\dot{y}_a - L\dot{\theta}\sin\theta - r\dot{\varphi}_L \sin\theta = 0 \\ (c_3)\dot{x}_a + L\dot{\theta}\cos\theta - r\dot{\varphi}_R \cos\theta = 0 \\ (c_4)\dot{y}_a + L\dot{\theta}\sin\theta - r\dot{\varphi}_R \sin\theta = 0 \end{cases} \tag{2.25}$$

A partir de ces quatre contraintes on peut les réduire à trois contraintes comme suit :

$$c_3(\cos\theta) + c_4(\sin\theta) =$$
$$\dot{x}_a\cos\theta + L\dot{\theta}\left(\cos\{\theta\}\right)^2 - r\dot{\varphi}_R\left(\cos\{\theta\}\right)^2 + \dot{y}_a\sin\theta + L\dot{\theta}\left(\sin\{\theta\}\right)^2 - r\dot{\varphi}_R\left(\sin\{\theta\}\right)^2 = 0$$

$$\Rightarrow \dot{x}_a\cos\theta + \dot{y}_a\sin\theta + L\dot{\theta} = r\dot{\varphi}_R$$

$$c_1(\cos\theta) + c_2(\sin\theta) =$$
$$\dot{x}_a\cos\theta - L\dot{\theta}\left(\cos\{\theta\}\right)^2 - r\dot{\varphi}_L\left(\cos\{\theta\}\right)^2 + \dot{y}_a\sin\theta - L\dot{\theta}\left(\sin\{\theta\}\right)^2 - r\dot{\varphi}_L\left(\sin\{\theta\}\right)^2 = 0$$

$$\Rightarrow \dot{x}_a \cos\theta + \dot{y}_a \sin\theta - L\dot{\theta} = r\dot{\varphi}_L$$

Les hypothèses 1 et 2 donnent :

$$\begin{cases} -\dot{x}_a \sin\theta + \dot{y}_a \cos\theta = 0 \\ \dot{x}_a \cos\theta + \dot{y}_a \sin\theta + L\dot{\theta} = r\dot{\varphi}_R \\ \dot{x}_a \cos\theta + \dot{y}_a \sin\theta - L\dot{\theta} = r\dot{\varphi}_L \end{cases} \tag{2.26}$$

Ces dernières contraintes peuvent s'écrire sous la forme matricielle suivante :

$$A(q)\,\dot{q} = 0 \tag{2.27}$$

Tel que le teste d'intégrabilité de ces contraintes sont testé par le théorème de Frobenius ce qui montre que la matrice $A(q)$ est une matrice de contraintes non-holonomes, donnée par :

$$A(q) = \begin{bmatrix} -\sin\theta & \cos\theta & 0 & 0 & 0 \\ \cos\theta & \sin\theta & L & -R & 0 \\ \cos\theta & \sin\theta & -L & 0 & -R \end{bmatrix} \tag{2.28}$$

Et $\dot{q}$ représente le dérivé de la coordonnée généralisée q, donnée par :

$$\dot{q} = \begin{bmatrix} \dot{x}_a \\ \dot{y}_a \\ \dot{\theta} \\ \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \tag{2.29}$$

L'équation (2.26) nous permet d'obtenir l'expression des vitesses linéaires des roues droite et gauche au point de contact$P$.

$$\begin{cases} v_{PR} = v_A + L\dot{\theta} \\ v_{PL} = v_A - L\dot{\theta} \end{cases} \tag{2.30}$$

Avec v_A la vitesse du point A, v_{PR} est la vitesse de la roue droite au point P et v_{PL} est la vitesse de la roue gauche au point P.

Posant : $\begin{cases} v = v_A \\ \dot{\theta} = \omega \end{cases}$ et $\begin{cases} v_{PR} = v_R \\ v_{PL} = v_L \end{cases}$

Ceci permet d'obtenir l'expression de la vitesse linéaire v et la vitesse angulaire ω du robot mobile en fonction des vitesses de rotation de la roue gauche $\dot{\varphi}_L$ et de la roue droite $\dot{\varphi}_R$.

$$\begin{cases} v = \frac{v_R + v_L}{2} = \frac{R(\dot{\varphi}_R + \dot{\varphi}_L)}{2} \\ \omega = \frac{v_R - v_L}{2L} = \frac{R(\dot{\varphi}_R - \dot{\varphi}_L)}{2L} \end{cases} \tag{2.31}$$

Dans le repère mobile les coordonnées du point A sont décrites par ces relations :

$$\begin{cases} \dot{x}_a^r = v = \frac{R(\dot{\varphi}_R + \dot{\varphi}_L)}{2} \\ \dot{y}_a^r = 0 \\ \dot{\theta}_a^r = w = \frac{R(\dot{\varphi}_R - \dot{\varphi}_L)}{2L} \end{cases} \tag{2.32}$$

En se servant de l'équation (2.3) (matrice de transformation), on peut écrire :

$$\begin{bmatrix} \dot{x}_a^I \\ \dot{y}_a^I \\ \dot{\theta}_a^I \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_a^r \\ \dot{y}_a^r \\ \dot{\theta}_a^r \end{bmatrix} \tag{2.33}$$

En remplaçant l'équation (2.32) dans (2.33), on obtient :

$$\begin{bmatrix} \dot{x}_a^I \\ \dot{y}_a^I \\ \dot{\theta}_a^I \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{R(\dot{\varphi}_R + \dot{\varphi}_L)}{2} \\ 0 \\ \frac{R(\dot{\varphi}_R - \dot{\varphi}_L)}{2L} \end{bmatrix} \tag{2.34}$$

$$\begin{bmatrix} \dot{x}_a^I \\ \dot{y}_a^I \\ \dot{\theta}_a^I \end{bmatrix} = \begin{bmatrix} \frac{R}{2}cos\theta & \frac{R}{2}cos\theta \\ \frac{R}{2}sin\theta & \frac{R}{2}sin\theta \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \tag{2.35}$$

En combinant (2.31) et (2.34) on obtient le modèle cinématique du robot comme suit :

$$\dot{q}_A^I = \begin{bmatrix} \dot{x}_a^I \\ \dot{y}_a^I \\ \dot{\theta}_a^I \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{2.36}$$

Tel que : $\dot{\theta}_a^I = \theta$.

En outre, on peut élaborer le modèle cinématique en considérant le centre de masse du robot mobile est le point C. Pour ce faire, considérons le vecteur de coordonnées généralisées suivant :

$$q = (x_c, y_c, \theta, \varphi_R, \varphi_L) \in \ Q = \mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^2$$

Avec (x_c, y_c) représente les coordonnées du point C correspondant au centre de gravité du robot (centre de masse) dans le repère initial.

En remplaçant l'équation (2.14) dans l'équation (2.26), les équations appropriées aux contraintes de roulement sont formulées comme suit :

$$\begin{cases} -\dot{x}_c \sin\theta + \dot{y}_c \cos\theta - d\dot{\theta} = 0 \\ \dot{x}_c \cos\theta + \dot{y}_c \sin\theta + L\dot{\theta} = r\dot{\varphi}_R \\ \dot{x}_c \cos\theta + \dot{y}_c \sin\theta - L\dot{\theta} = r\dot{\varphi}_L \end{cases} \tag{2.37}$$

Ces trois équations de contrainte peuvent être écrites sous la forme matricielle suivante :

$$A(q)\,\dot{q} = 0$$

Tel que le matrice de contraintes cinématiques est :

$$A(q) = \begin{bmatrix} -\sin\theta & \cos\theta & -d & 0 & 0 \\ \cos\theta & \sin\theta & L & -R & 0 \\ \cos\theta & \sin\theta & -L & 0 & -R \end{bmatrix} \tag{2.38}$$

Et le vecteur de coordonnées généralisées est :

$$\dot{q} = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta} \\ \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \tag{2.39}$$

En remplaçant l'équation (2.14) dans les équations (2.34) et (2.35), on obtient :

$$\dot{q}_c^I = \begin{bmatrix} \dot{x}_c^I \\ \dot{y}_c^I \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta & -d\ sin\theta \\ sin\theta & d\ cos\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{2.40}$$

Et la vitesse du robot mobile par rapport au centre de masse C du système de coordonnées inertielles est donnée par :

$$\begin{bmatrix} \dot{x}_c^I \\ \dot{y}_c^I \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{R}{2}\left(cos\theta - \frac{d}{L}sin\theta\right) & \frac{R}{2}\left(cos\theta + \frac{d}{L}sin\theta\right) \\ \frac{R}{2}\left(sin\theta + \frac{d}{L}cos\theta\right) & \frac{R}{2}\left(sin\theta - \frac{d}{L}cos\theta\right) \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \tag{2.41}$$

L'équation (2.40) représente le modèle cinématique du DDWMR.

2.2.4 Modélisation dynamique

Le modèle dynamique est nécessaire pour la simulation, l'analyse du mouvement du robot et la conception des variétés d'algorithmes de commande. Plusieurs formalismes tels que : le formalisme d'Euler-Lagrange, le formalisme de Newton-Euler et le principe

de D'Alembert permettent de faire la modélisation dynamique du robot. Dans notre cas, on s'intéressera uniquement au formalisme d'Euler-Lagrange [24, 82, 86].
L'approche dynamique de Lagrange est une méthode très puissante pour formuler les équations du mouvement des systèmes mécaniques. Cette méthode, introduite par Lagrange, est utilisée pour dériver les équations du mouvement en considérant les énergies cinétiques et le potentiel du système donné [82, 84]. Considérons un robot non-holonome avec n coordonnées généralisées $(q_1, q_2....., q_n)$ et soumis à m contraintes.
L'équation de Lagrange peut s'écrire sous la forme suivante :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \left(\frac{\partial L}{\partial q_i}\right) = F - A^T(q).\lambda_k \tag{2.42}$$

Avec :$L(q, \dot{q}) = T - V$est le lagrangien.
T : L'énergie cinétique du système ;
V : L'énergie potentielle du système ;
F : Le vecteur de force généralisée ;
A^T : matrice de contraintes ;
λ_k : Le vecteur des multiplicateurs de Lagrange associé aux contraintes ;
q_i :la coordonnée généralisée et $q = [x_c; y_c; \theta; \varphi_R; \varphi_L]$ de dimension $n = 5$;
A : centre des roues origine du repère mobile.
L'énergie cinétique T du système est donnée par :

$$\Longrightarrow\ T = T_c + T_{w_R} + T_{w_L} \tag{2.43}$$

Tel que :
T_c est l'énergie cinétique de la plateforme :

$$T_c = \frac{1}{2} m_c v_c^2 + \frac{1}{2} I_c \dot{\theta}^2 \tag{2.44}$$

T_{w_R} est l'énergie cinétique de la roue droite :

$$T_{w_R} = \frac{1}{2} m_w v_{wR}^2 + \frac{1}{2} I_m \dot{\theta}^2 + \frac{1}{2} I_w \dot{\varphi}_R \tag{2.45}$$

T_{w_L} est l'énergie cinétique de la roue gauche :

$$T_{w_L} = \frac{1}{2} m_w v_{wL}^2 + \frac{1}{2} I_m \dot{\theta}^2 + \frac{1}{2} I_w \dot{\varphi}_L \tag{2.46}$$

Avec :
m_c : masse de la plate-forme.
m_w : masse de chaque roue plus la masse du moteur.
v_{wR} : vitesse linéaire de la roue droite.

v_{wL} : vitesse linéaire de la roue gauche.
I_m : Moment d'inertie de chaque roue avec le moteur par rapport au diamètre de la roue.
I_w : Moment d'inertie de chaque roue avec le moteur par rapport à l'axe de la roue.
I_c : Moment d'inertie de la plate-forme du robot sans les roues, les moteurs, autour de l'axe vertical qui passe par le point C (centre de gravité).
Toutes les vitesses seront d'abord exprimées en fonction des coordonnées généralisées en utilisant l'équation générale de la vitesse dans le repère inertiel, donnée par cette relation :

$$v_i^2 = \dot{x}_i^2 + \dot{y}_i^2 \tag{2.47}$$

Les composantes x_i et y_i du centre de gravité et des roues peuvent être obtenu en termes de coordonnées généralisées comme suit :
Le point C dans le repère fixe a pour coordonnées :

$$\begin{cases} x_{PR} = x_{wR} \\ y_{PR} = y_{wR} \end{cases} et \begin{cases} x_{PL} = x_{wL} \\ y_{PL} = y_{wL} \end{cases} \tag{2.48}$$

$$\begin{cases} x_c = x_a + dcos\theta \\ y_c = y_a + dsin\theta \end{cases} \tag{2.49}$$

$$\begin{cases} x_{wR} = x_a + Lsin\theta \\ y_{wR} = y_a - Lcos\theta \end{cases} \tag{2.50}$$

$$\begin{cases} x_{wL} = x_a - Lsin\theta \\ y_{wL} = y_a + Lcos\theta \end{cases} \tag{2.51}$$

Pour le DDWMR, les coordonnées généralisées sont choisies comme suit : $q = [x_c; y_c; \theta; \varphi_R; \varphi_L]$ tel que $[x_c; y_c]$ représente les coordonnées du centre de masse du robot mobile dans le repère fixe. En utilisant les équations (2.43) à (2.45) ainsi que les équations (2.46) à (2.50), L'énergie cinétique totale du DDMR est :

$$T = \frac{1}{2}\left(m_c + 2m_w\right)\ \left(\dot{x}_a^2 + \dot{y}_a^2\right) + m_c d\dot{\theta}\left(\dot{y}_a cos\theta - \dot{x}_a sin\theta\right) + \frac{1}{2} I_w(\dot{\varphi}_R^2 + \dot{\varphi}_L^2) + \frac{1}{2} I_1 \dot{\theta}^2 \tag{2.52}$$

$$I_1 = I_c + m_c d^2 + 2m_w L^2 + 2I_m$$

L'énergie potentielle étant nulle car le robot se déplace sur un plan horizontal. Alors on a donc le Lagrangien :

$$L = T$$

L'équation (2.28) donne :

$$\begin{cases} x_a = x_c - dcos\theta \\ y_a = y_c - dsin\theta \end{cases} \text{ et } \begin{cases} \dot{x}_a = \dot{x}_c + d\dot{\theta}sin\theta \\ \dot{y}_a = \dot{y}_c - d\dot{\theta}cos\theta \end{cases}$$

Pour faire apparaitre les coordonnées du vecteur généralisé on remplace dans (2.51) et on obtient :

$$T = \frac{1}{2}m\left(\dot{x}_c^2 + \dot{y}_c^2\right) + (m - m_c)d\dot{\theta}\left(\dot{x}_c sin\theta \dot{-} \dot{y}_c cos\theta\right) + \frac{1}{2}I_w(\dot{\varphi}_R^2 + \dot{\varphi}_L^2) + \frac{1}{2}I\dot{\theta}^2 \qquad (2.53)$$

Tel que :

$$m = m_c + 2m_w \text{ et } \; I = I_c + 2m_w(L^2 + d^2) + 2I_m$$

$$A(q) = \begin{bmatrix} -\sin\theta & \cos\theta & -d & 0 & 0 \\ \cos\theta & \sin\theta & L & -R & 0 \\ \cos\theta & \sin\theta & -L & 0 & -R \end{bmatrix}$$

$$A^T(q) = \begin{bmatrix} -\sin\theta & \cos\theta & \cos\theta \\ \cos\theta & \sin\theta & \sin\theta \\ -d & L & -L \\ 0 & -R & 0 \\ 0 & 0 & -R \end{bmatrix}$$

En utilisant l'équation (2.41) avec le Lagrangienne $L = T$ les équations du mouvement du DDMR sont données par :

$$3\begin{cases} m\ddot{x}_c + 2dm_w\left(\ddot{\theta}sin\theta + \dot{\theta}^2cos\theta\right) = \lambda_1 sin\theta - \lambda_2 cos\theta - \lambda_3 cos\theta \\ m\ddot{y}_c - 2dm_w\left(\ddot{\theta}cos\theta - \dot{\theta}^2 sin\theta\right) = -\lambda_1 cos\theta - \lambda_2 sin\theta - \lambda_3 sin\theta \\ 2dm_w(\ddot{x}_c sin\theta - \ddot{y}_c cos\theta) + I\ddot{\theta} = \lambda_1 d - \lambda_2 L + \lambda_3 L \\ I_w\ddot{\varphi}_R = \tau_R + \lambda_2 R \\ I_w\ddot{\varphi}_L = \tau_L + \lambda_3 R \end{cases} \qquad (2.54)$$

L'équation (2.41) donne :

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} = B(q)\tau - A^T(q)\lambda \qquad (2.55)$$

Tel que :

$$M(q) = \begin{bmatrix} m & 0 & 2dm_w sin\theta & 0 & 0 \\ 0 & m & -2dm_w cos\theta & 0 & 0 \\ 2dm_w sin\theta & -2dm_w cos\theta & I & 0 & 0 \\ 0 & 0 & 0 & I_w & 0 \\ 0 & 0 & 0 & 0 & I_w \end{bmatrix},$$

$$V(q,\dot{q}) = \begin{bmatrix} 0 & 0 & 2dm_w\dot{\theta}cos\theta & 0 & 0 \\ 0 & 0 & 2dm_w\dot{\theta}sin\theta & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; B(q) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ et}$$

$$A^T(q)\lambda = \begin{bmatrix} -sin\theta & cos\theta & cos\theta \\ cos\theta & sin\theta & sin\theta \\ -d & L & -L \\ 0 & -R & 0 \\ 0 & 0 & -R \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix}$$

$M(q)$: est la matrice d'inertie symétrique définie positive de taille $n \; x \; n$;
$V(q,\dot{q})$: est la matrice des forces centrifuges et des forces de Coriolis ;
$B(q)$: est la matrice de transformation d'entrée ;
$A^T(q)$: est la matrice des contraintes non-holonomes ;
u : est le vecteur d'entrée, $u = \begin{bmatrix} \tau_R \\ \tau_L \end{bmatrix}$;
Ensuite, le système décrit par l'équation (2.54) est transformé en une forme plus pratique aux contrôle et simulation. L'objectif principal pour obtenir le modèle dynamique du robot est d'éliminer le terme $A^T(q)\lambda$ qui correspond aux forces de contraintes liées aux contraintes cinématiques, puisque les multiplicateurs de Lagrange λ_i sont inconnu. Cela se fait d'abord en définissant le vecteur réduit η.
Soit $\eta = \begin{pmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{pmatrix}$ un vecteur de vitesse auxiliaire. Ensuite, en exprimant les vitesses des coordonnées généralisées à l'aide du modèle cinématique (2.40). (Modèle cinématique directe) Ensuite nous avons :

$$\begin{bmatrix} \dot{x}_c^I \\ \dot{y}_c^I \\ \dot{\theta} \\ \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} = \begin{bmatrix} \frac{R}{2}(cos\theta - \frac{d}{L}sin\theta) & \frac{R}{2}(cos\theta + \frac{d}{L}sin\theta) \\ \frac{R}{2}(sin\theta + \frac{d}{L}cos\theta) & \frac{R}{2}(sin\theta - \frac{d}{L}cos\theta) \\ \frac{R}{2L} & -\frac{R}{2L} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \tag{2.56}$$

Ce qu'on écrit sous la forme condensée :

$$\dot{q} = S(q)\eta \tag{2.57}$$

Par conséquent :

$$\ddot{q} = \dot{S}(q)\,\eta + S(q)\dot{\eta} \tag{2.58}$$

$S(q)$ est une matrice de rang complet qui vérifie la condition suivante :

$$S^T(q)\,A^T(q) = 0 \tag{2.59}$$

Alors l'équation (2.54) peut s'écrire sous la forme :

$$M(q)\,(\dot{S}(q)\,\eta + S(q)\dot{\eta}) + V(q,\dot{q})\,S(q)\;\eta = B(q)\,\tau - A^T(q)\lambda \tag{2.60}$$

En multipliant les deux côtés de l'équation (2.59) par $S^T(q)$ on obtient :

$$S^T(q)\,M(q)\,S(q)\dot{\eta} + S^T(q)\,(M(q)\,\dot{S}(q) + V(q,\dot{q})\,S(q))\eta = S^T(q)\,B(q)\,\tau - S^T(q)\,A^T(q)\lambda \tag{2.61}$$

En posant :

$$\begin{cases} \bar{M}(q) = S^T(q)\,M(q)\,S(q) \\ \bar{V}(q,\dot{q}) = S^T(q)\,(M(q)\,\dot{S}(q) + V(q,\dot{q})\,S(q)) \\ \bar{B}(q) = S^T(q)\,B(q) \end{cases}$$

Ceci permet d'obtenir la forme compacte du modèle dynamique du robot mobile :

$$\bar{M}(q)\dot{\eta} + \bar{V}(q,\dot{q})\,\eta = \bar{B}(q)\,\tau \tag{2.62}$$

Tel que :

$$\bar{M}(q) = \begin{bmatrix} \frac{R^2}{4L^2}\left(m\left(d^2+L^2\right) - 4d^2m_w + I\right) + I_w & \frac{R^2}{4L^2}\left(m\left(L^2-d^2\right) + 4d^2m_w - I\right) \\ \frac{R^2}{4L^2}\left(m\left(L^2-d^2\right) + 4d^2m_w - I\right) & \frac{R^2}{4L^2}\left(m\left(d^2+L^2\right) - 4d^2m_w + I\right) + I_w \end{bmatrix}$$

$$\bar{V}(q,\dot{q}) = \begin{bmatrix} 0 & \frac{R^2}{2L^2}(dm_c\dot{\theta}) \\ -\frac{R^2}{2L^2}(dm_c\dot{\theta}) & 0 \end{bmatrix} \text{ et } \bar{B}(q) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Ce modèle dynamique est exprimé en fonction des vitesses angulaires des roues droite et gauche ($\varphi_{R,}\varphi_L$), la vitesse angulaire $\dot{\theta}$ et les couples moteurs (τ_R, τ_L) qui entraînent les deux roues droite et gauche respectivement. En outre, on peut représenter ce modèle par les vitesses linéaire et angulaire en insérant l'équation (2.31) dans l'équation (2.61) ce qui donne la représentation du modèle dynamique suivante :

$$\begin{cases} (m + \frac{2I_w}{R^2})\dot{v} - m_c d w^2 = \frac{1}{R}(\tau_R + \tau_L) \\ \left(d^2\left(m_c - 2m_w\right) + I + \frac{2L^2}{R^2}I_w\right)\dot{w} + m_c d w v = \frac{L}{R}(\tau_R - \tau_L) \end{cases} \tag{2.63}$$

Avec : $u = \begin{cases} u_1 = \tau_R + \tau_L \\ u_2 = \tau_R - \tau_L \end{cases}$

En remplaçant $(m + \frac{2I_w}{R^2})$ par m_0 et $d^2\left(m_c - 2m_w\right) + I + \frac{2L^2}{R^2}I_w$ par I_0 l'expression du

modèle dynamique non linéaire est le suivant :

$$\begin{cases} \dot{v}(t) = \frac{m_c d}{m_0}\omega^2 + \frac{1}{m_0 R}u_1(t) \\ \dot{\omega}(t) = -\frac{m_c d}{I_0}v.\omega + \frac{L}{I_0 R}u_2(t) \end{cases} \tag{2.64}$$

2.3 Contrôleur cinématique

La Figure (2.8) suivante donne schéma bloc de la stratégie de commande cinématique d'un robot mobile.

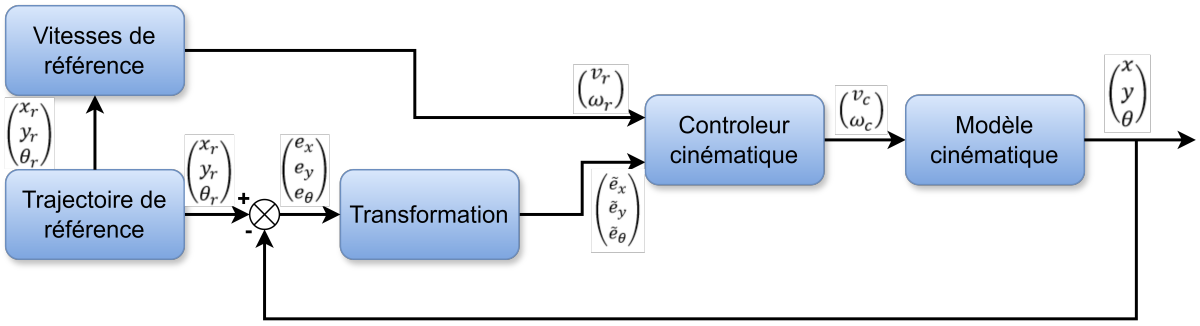


FIGURE 2.8 – Schéma de commande en boucle cinématique

En robotique mobile, comparativement au suivi de chemin, le suivi de trajectoire est considéré comme un problème très important car les robots mobiles sont appelés à se déplacer d'un point à un autre en un temps bien précis. Ainsi la forte utilisation des robots mobiles à réaliser certaines tâches, nécessite la mise en œuvre d'une loi de commande. Dans cette section, nous allons présenter et étudier la stabilité d'une loi de commande pour le contrôleur cinématique qui a pour objectif de minimiser les erreurs en position dans une boucle fermée extérieure. La Figure (2.9) illustre l'erreur de poursuite e_p entre un robot réel dont le centre de masse est le point A et un robot de référence dont r est son point matériel.

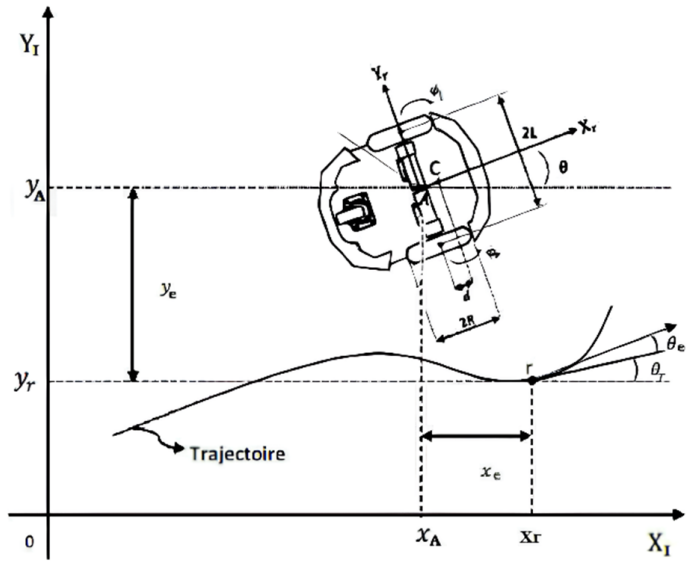


FIGURE 2.9 – Représentation de l'erreur de posture du robot

Le contrôleur cinématique utilisé dans notre travail est un contrôleur qui a été proposé dans [18, 86]. Il fut appliqué sur le modèle cinématique d'un robot mobile non holonome. Ce contrôleur permet d'assurer le suivi de trajectoire. Il ne tient pas compte de la dynamique du robot mobile. Afin de commander le robot, on définit deux postures du robot mobile à savoir :

- La posture de référence $P_r = (x_r, y_r, \theta_r)$ considérée comme la posture de but.
- La posture courante $P_a = (x_a, y_a, \theta_a)$ considérée comme la posture réelle du robot.

Ainsi l'erreur de posture dans le repère fixe est :

$$e_p = \begin{pmatrix} x_e \\ y_e \\ \theta_e \end{pmatrix} = R^{-1}(\theta)(P_r - P_a) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} (P_r - P_a) \tag{2.65}$$

Ou $R^{-1}(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$ est l'inverse de la matrice orthogonale de rotation entre les deux repères $\{X_i, Y_i\}$ et $\{X_r, Y_r\}$. L'erreur de posture est donnée par :

$$e_p = \begin{cases} (x_r - x_a)\cos\theta + (y_r - y_a)\sin\theta \\ -(x_r - x_a)\sin\theta + (y_r - y_a)\cos\theta \\ \theta_e = \theta_r - \theta_a \end{cases} \tag{2.66}$$

A partir de (2.35), on a :
$\dot{x}_a = v\cos\theta$ et $\dot{y}_a = v\sin\theta$.

$$\dot{x}_a \cos\theta + \dot{y}_a \sin\theta = v_a = v$$
$$\dot{x}_r \cos\theta + \dot{y}_r \sin\theta = v_r$$

A partir de la contrainte cinématique (2.12) ; on a :

$$\dot{x}_a \sin\theta - \dot{y}_a \cos\theta = 0 \text{ et } \dot{x}_r \sin\theta - \dot{y}_r \cos\theta = 0$$

En dérivant l'erreur de posture, on obtient :

$$\begin{aligned}\dot{x}_e &= (\dot{x}_r - \dot{x}_a)\cos\theta + (\dot{y}_r - \dot{y}_a)\sin\theta - (x_r - x_a)\,\dot{\theta}\sin\theta + (y_r - y_a)\,\dot{\theta}\cos\theta \\ &= y_e\omega - v_a + \dot{x}_r\cos\theta + \dot{y}_r\sin\theta \\ &= y_e\omega - v_a + \dot{x}_r\cos(\theta_r - \theta_a) + \dot{y}_r\sin(\theta_r - \theta_a) \\ &= y_e\omega - v_a + v_r\cos\theta_e\end{aligned}$$

$$\begin{aligned}\dot{y}_e &= -(\dot{x}_r - \dot{x}_a)\sin\theta + (\dot{y}_r - \dot{y}_a)\cos\theta - (x_r - x_a)\,\dot{\theta}\cos\theta - (y_r - y_a)\,\dot{\theta}\cos\theta \\ &= -x_e\omega + \dot{x}_a\sin\theta + \dot{y}_a\cos\theta + \dot{x}_r\sin\theta + \dot{y}_r\cos\theta \\ &= -x_e\omega - \dot{x}_r\sin(\theta_r - \theta_a) + \dot{y}_r\cos(\theta_r - \theta_a) \\ &= -x_e\omega + \dot{v}_r\sin\theta_e\end{aligned}$$

$$\dot{\theta}_e = \dot{\theta}_r - \dot{\theta} = \omega_r - \omega.$$

Ainsi, on peut écrire :

$$\begin{bmatrix}\dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e\end{bmatrix} = \begin{bmatrix}y_e\omega - v_a + v_r\cos\theta_e \\ -x_e\omega + \dot{v}_r\sin\theta_e \\ \omega_r - \omega\end{bmatrix} \tag{2.67}$$

v_r et ω_r étant, respectivement, la vitesse linéaire de référence et la vitesse angulaire de référence.
v_a est la vitesse linéaire du robot au point A ;
ω est la vitesse angulaire du robot au point A ;
$\theta_e = (\theta_r - \theta)$ Est l'erreur de l'angle d'orientation du robot.
A des fins de conformité, appelons v, au lieu de v_A, la vitesse linéaire du robot au point A. Ainsi,le contrôleur cinématique que nous avons utilisé est celui proposé dans [86], et est donné par :

Les commandes proposées dans [18] qui servent à faire converger l'erreur ($e_p = \begin{pmatrix}x_e \\ y_e \\ \theta_e\end{pmatrix}$) vers 0 :

$$\begin{bmatrix}v \\ \omega\end{bmatrix} = \begin{bmatrix}v_r\cos\theta_e + K_x x_e \\ \omega_r + v_r(K_y y_e + K_\theta\sin\theta_e)\end{bmatrix} \tag{2.68}$$

Où K_x, K_y et K_θ sont des constantes positives. Dans notre simulation on a choisi $K_x = 90.8$, $K_y = 90.7$ et $K_\theta = 0.5$

Cette loi de commande propose un suivi stable pour les robots mobiles non-holonomes. Elle est efficace si la trajectoire de référence est paramétrée pour bien calculer les paramètres v_r et ω_r . Elle nécessite également l'estimation précise de la position du robot mobile. Cependant la trajectoire de référence parfois ne peut pas être paramétrée et la position du robot ne donne pas une valeur précise suite au glissement des roues.
Pour démontrer la stabilité de la boucle cinématique, soit la fonction de Lyapunov définie positive :

$$L_0 = \frac{1}{2}\left(x_e^2 + y_e^2\right) + \frac{1}{K_y}(1 - cos\theta_e) \tag{2.69}$$

$$L_0 \geq 0 \text{ si } e_p = 0 \text{ , alors } L_0 = 0$$
$$\text{et si } e_p \neq 0 \text{ , alors } L_0 > 0$$

La dérivée temporelle de L_0 le long de la trajectoire est :

$$\dot{L}_0 = \dot{x}_e x_e + \dot{y}_e y_e + \frac{1}{K_y}(\dot{\theta}_e sin\theta_e) \tag{2.70}$$

En remplaçant $\dot{x}_e, \dot{y}_e$ et $\dot{\theta}_e$ par leurs expressions trouvées dans (2.66), on obtient :

$$\dot{L}_0 = -K_x x_e^2 - \frac{v_r K_\theta sin^2\theta_e}{K_y} \tag{2.71}$$

Si $v_r \geq 0$ alors $\dot{L}_0 \leq 0$. L'origine $(x_e,\ y_e,\ \theta_e) = 0$ est alors asymptotiquement stable.

2.4 Contrôleur dynamique :Linéarisation par bouclage non linéaire

Le contrôleur dynamique a deux fonctions. Premièrement, il minimise les erreurs de vitesse dans une boucle interne, (Figure 2.12), où la dynamique du robot mobile est considérée comme un sous-système. Deuxièmement, il calcule les commandes $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$ qui sont en fonctions des couples moteurs (τ_R,τ_L) qui agissent sur le robot mobile de telle manière à assurer les corrections en position et en vitesse.
Dans cette section de ce chapitre on va utiliser un contrôleur dynamique conçu à partir de la commande non linéaire synthétisée à base de la technique de linéarisation par bouclage non linéaire appliquée au modèle d'état du robot mobile à entrainement différentiel décrit par les modèles cinématique et dynamique de ce dernier.
L'objectif de cette section est de trouver un système entrées/sorties découplée à l'aide d'un contrôleur de linéarisation par bouclage à retour d'état non linéaire (Figure (2.11)). Cette technique élabore la loi de commande u(t) qui prend en entrée les vitesses linéaire $v(t)$ et angulaire $\omega(t)$.

On commence par la détermination du degré relatif de chaque sortie pour établir la matrice de découplage puis l'élaboration de la consigne externe qui découple les deux sorties, en tenant compte de la dynamique des erreurs de poursuite des trajectoires de références, et enfin la conception de la commande physique du système découplé et linéarisé. Une simulation est prévue pour valider l'intérêt de l'approche.
Les concepts de la théorie de la commande non linéaire et les principes de la linéarisation entrée-sortie sont présentées dans plusieurs travaux de la recherche en citant [87, 88].

2.4.1 Variables à contrôler

Le modèle d'état non linéaire du robot mobile a entrainement différentiel DDRM peut être présenté en regroupant le modèle dynamique et le modèle cinématique par le système suivant :

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} vcos\theta - d\omega sin\theta \\ vsin\theta + d\omega cos\theta \\ \omega \\ \frac{m_c d}{m_0}\omega^2 \\ -\frac{m_c d}{m_0} v.\omega \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m_0 R} \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{L}{I_0 R} \end{bmatrix} u_1 \tag{2.72}$$

Ce système est sous la forme :

$$\begin{cases} \dot{X} = F(X) + G(x)\, u \\ y = H(X) \end{cases} \tag{2.73}$$

Avec :
$F(X)$ est un vecteur d'ordre $(n = 5)$ et G est une matrice $[5 * 2]$.
$H(X)$ est le vecteur de sortie.
f,g et h Sont des fonctions lisses linéaires.
Les variables à contrôler sont les vitesses linéaire(v) et angulaire (w), regroupées en un seul vecteur : $\begin{bmatrix} v \\ \omega \end{bmatrix}$.

$$y = \begin{bmatrix} y_1(x) \\ y_2(x) \end{bmatrix} = \begin{bmatrix} h_1(x) \\ h_2(x) \end{bmatrix} = \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.74}$$

2.4.2 Condition de linéarisation

La condition de linéarisation permettant de vérifier si un système non linéaire admet une linéarisation entrée-sortie est l'ordre du degré relatif du système r.

2.4.3 Degré relatif

Le degré relatif d'une sortie est le nombre de fois qu'il faut dériver la sortie y pour faire apparaitre l'entrée u. Pour chercher le degré relatif du système en utilisant les crochets de lie [87].

Régle :

$$L_f = \sum_{i=1}^{n} f_i(x) \frac{\partial}{\partial x_i}(x)$$
$$L_f h(x) = \sum_{i=1}^{n} f_i(x) \frac{\partial h}{\partial x_i}(x)$$
$$y^{(i)} = L_f^i h(x) + L_g L_f^{i-1} h(x) u$$

- **Degré relatif du v :**
  Pour la sortie :

$$h_1(x) = v$$

  Pour

$$i = 1 \Rightarrow \dot{y}_1 = \frac{m_c d}{m_0}\omega^2 + \begin{bmatrix} \frac{1}{m_0 R} & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\Rightarrow r_1 = 1 \tag{2.75}$$

  Alors le degré relatif de v est $r_1 = 1$.

- **Degré relatif du ω :**
  Pour la sortie :

$$h_2(x) = \omega$$

  Pour

$$i = 1 \Rightarrow \dot{y}_2 = -\frac{m_c d}{I_0} v.\omega + \begin{bmatrix} 0 & \frac{L}{I_0 R} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\Rightarrow r_2 = 1 \tag{2.76}$$

  Alors le degré relatif deω est $r_2 = 1$.

Par conséquent, le degré relatif global (ou vectoriel) est $r = r_1 + r_2 = 2$.

2.4.4 Matrice de découplage

D'après la dérivée de lie précédente on obtient l'équation suivante :

$$y^r = L_f^r h\left(x\right) + L_g L_f^{r-1} h(x) u$$
$$\Rightarrow y^r = A\left(x\right) + D(x) u$$

Avec :

$$\begin{bmatrix} \dot{h}_1(x) \\ \dot{h}_2(x) \end{bmatrix} = \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{m_c d}{m_0}\omega^2 \\ -\frac{m_c d}{I_0} v.\omega \end{bmatrix} + \begin{bmatrix} \frac{1}{m_0 R} & 0 \\ 0 & \frac{L}{I_0 R} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.77}$$

$$\Rightarrow A\left(x\right) = \begin{bmatrix} \frac{m_c d}{m_0}\omega^2 \\ -\frac{m_c d}{I_0} v.\omega \end{bmatrix} \text{et } D\left(x\right) = \begin{bmatrix} \frac{1}{m_0 R} & 0 \\ 0 & \frac{L}{I_0 R} \end{bmatrix}$$

Pour que le retour d'état puisse exister, il faut que la matrice $D\left(x\right)$ soit non singulière (inversible).
Par conséquent : $\det(D\left(x\right)) \neq 0$ et $D\left(x\right)$ est inversible.
En conclusion, la non singularité de la matrice de découplage et le degré vectoriel du système nous permet de réaliser une linéarisation entrée-sortie par retour d'état non linéaire.

2.4.5 Linéarisation entrée-sortie par bouclage non linéaire

Pour linéariser le système, on applique le retour d'état non linéaire suivant :

$$u = D^{-1}(x)\left[-A\left(x\right) + v\right]$$

Où $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$ est une consigne externe ce qui aboutit à deux sous-systèmes mono variable, découplé et linéaire Figure (2.10).

$$\begin{bmatrix} \dot{h}_1(x) \\ \dot{h}_2(x) \end{bmatrix} = \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{2.78}$$

Le système (2.77) est résolu par une chaine d'intégrateur.

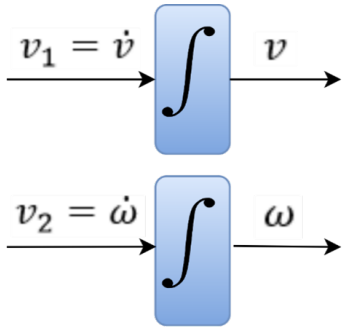


FIGURE 2.10 – sous-systèmes découplés et linéarisés.

Cette stratégie de commande se ramène à la linéarisation du système en chaine d'intégrateurs découplés, suivie d'un placement de pôles, c'est à dire une poursuite asymptotique de trajectoires avec convergence exponentielle des erreurs vers 0.

2.4.6 Elaboration de la consigne externe

A

Le choix de v (une variable qui représente une consigne externe) est :

$$v\left(t\right)=y_{ref}^{(r)}\left(t\right)+\sum_{i=1}^{r}c_{i}(y_{ref}^{(r-i)}\left(t\right)-y^{(r-i)}(t))$$

Conduit à la dynamique suivante :

$$e^{r}\left(t\right)+\sum_{i=1}^{r}c_{i}e^{(r-i)}\left(t\right)$$

Tel que :
$e=y_{ref}-y$: l'erreur de poursuite.
c_i : coefficient du polynôme de Hurwitz, la convergence à 0 est garantie.

Alors pour notre cas la consigne externe sera générée comme suit :

$$\begin{cases} v_1=\dot{v}_{ref}+k_v(v_{ref}-v) \\ v_2=\dot{\omega}_{ref}+k_\omega(\omega_{ref}-\omega) \end{cases} \tag{2.79}$$

Dans notre simulation les paramètres du controleur dynamique utilisé sont : $k_v=82.5$ et $k_\omega=81.5$

2.4.7 Elaboration de la loi de commande physique

La loi de commande physique adaptée au contrôle dynamique du robot mobile est donnée par l'expression :

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}=\begin{bmatrix} \tau_R+\tau_L \\ \tau_R-\tau_L \end{bmatrix}=D^{-1}(x)\left[-A\left(x\right)+\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}\right] \tag{2.80}$$

2.4.8 Schéma bloc du système linéarisé

La Figure (2.11) illustre la traduction sous forme de schéma bloc de la loi de commande de découplage et de linéarisation du système à commander.

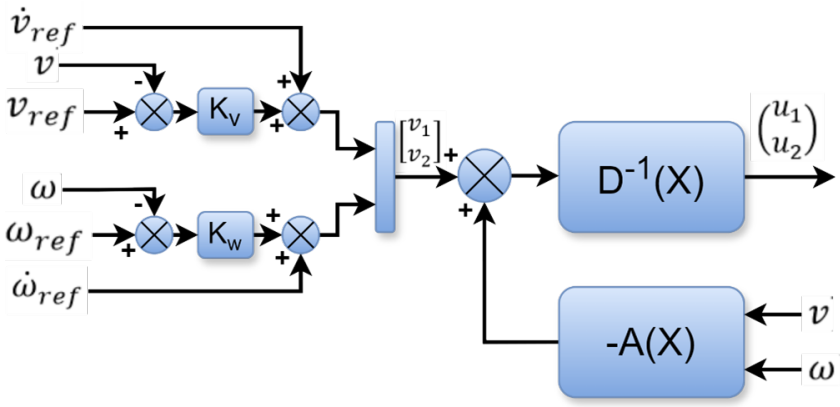


FIGURE 2.11 – Bloc de découplage et de linéarisation.

2.5 Validation des contrôleurs cinématique et dynamique par une simulation

La structure de commande adoptée est schématisée par la Figure (2.12) suivante :

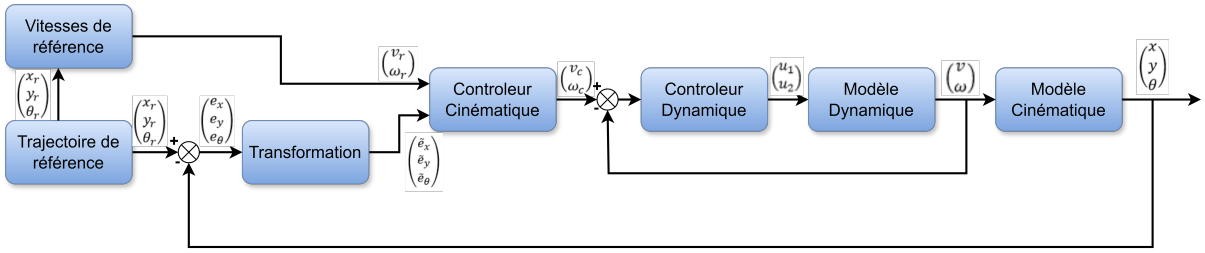


FIGURE 2.12 – Schéma de commande en boucles cinématique et dynamique.

Les simulations ont été réalisées avec le logiciel Matlab. Deux schémas de commande ont été utilisés pour réaliser les simulations. La Figure (2.8) représente le schéma de commande en boucle cinématique du robot mobile, tandis que la Figure (2.12) représente le schéma de commande en boucles cinématique et dynamique (deux boucles imbriquées). La stabilité est assurée pour chaque boucle fermée de manière individuelle. Si la stabilité asymptotique de toutes les boucles est assurée, cela signifie que le système est asymptotiquement stable.

Résultats de simulation de la boucle cinématique.

Les Figures (2.13, 2.14 et 2.15) représentent respectivement l'évolution de suivi de la position en x ,y du robot en fonction du temps et les erreurs de poursuite en position selon les deux axes x et y et la trajectoire parcourue par le robot par rapport à la trajectoire de réference circulaire.

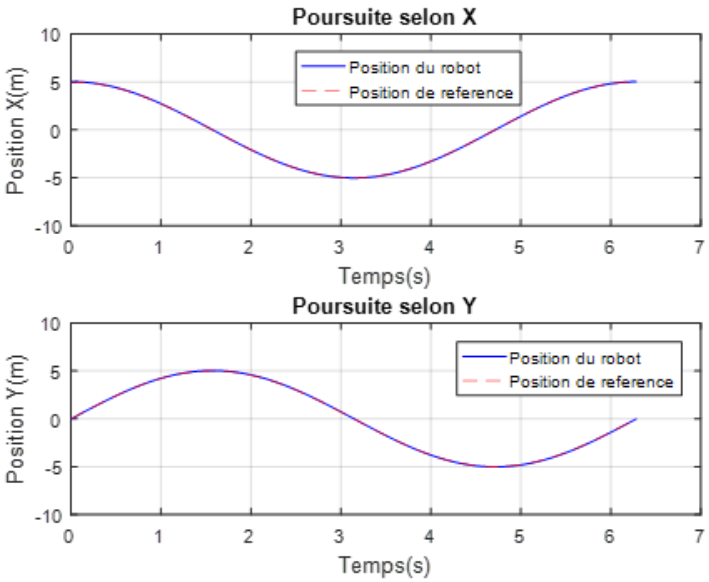


FIGURE 2.13 – Résultats de simulation de poursuite selon X et Y d'un robot mobile (DDMR) dans une boucle de commande cinématique (pour une trajectoire circulaire.

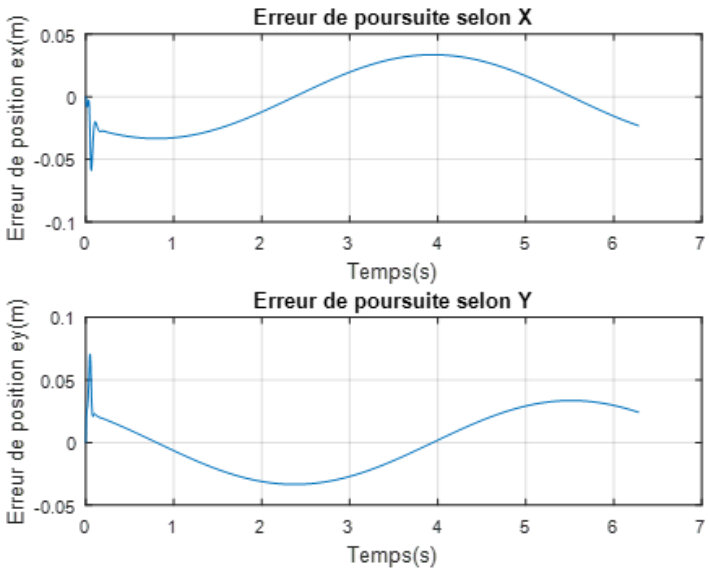


FIGURE 2.14 – Résultats de simulation des erreurs de poursuite d'un robot mobile (DDMR) dans une boucle de commande cinématique (pour une trajectoire circulaire.

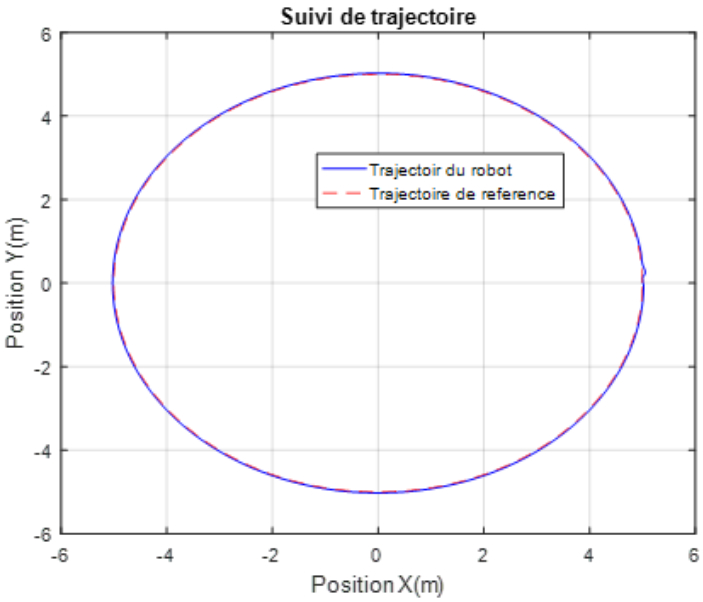


FIGURE 2.15 – Résultats de simulation de suivi de trajectoire circulaire dans une boucle de commande cinématique.

Les Figures (2.16, 2.17 et 2.18) représentent respectivement l'évolution de suivi de la position en x ,y du robot en fonction du temps et les erreurs de poursuite en position selon les deux axes x et y et la trajectoire parcourue par le robot par rapport à la trajectoire de réference en forme huit.

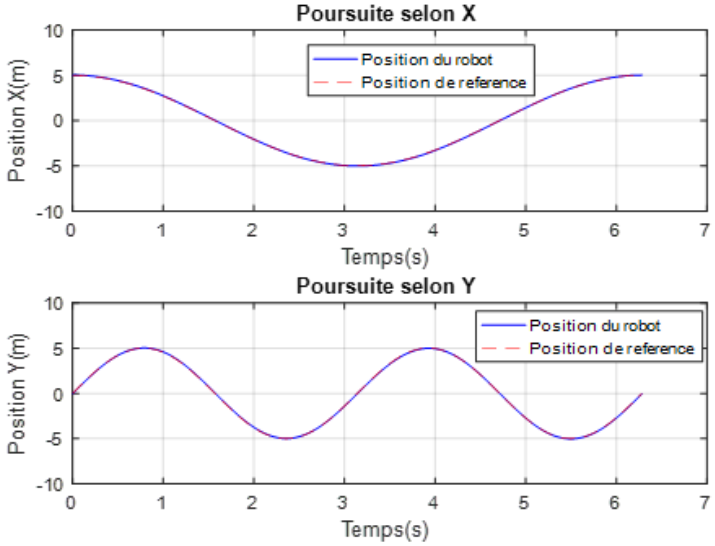


FIGURE 2.16 – Résultats de simulation de poursuite selon X et Y d'un robot mobile (DDMR) dans une boucle de commande cinématique (pour une trajectoire en forme de huit).

.

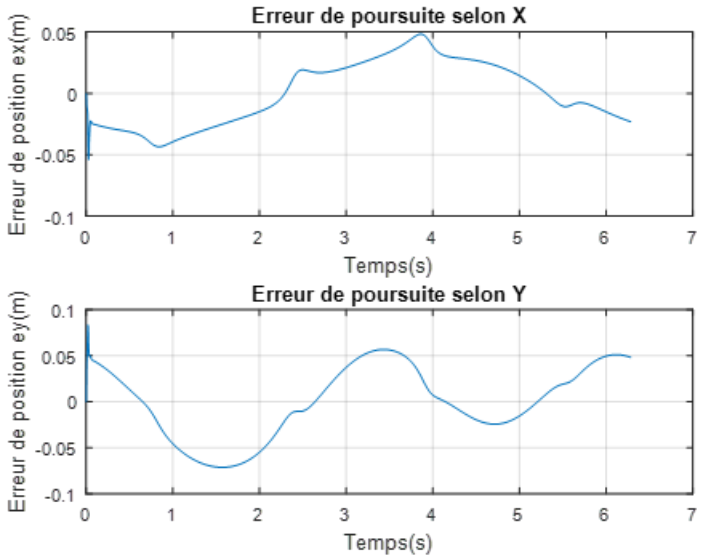


FIGURE 2.17 – Résultats de simulation des erreurs de poursuite d'un robot mobile (DDMR) dans une boucle de commande cinématique (pour une trajectoire en forme de huit).

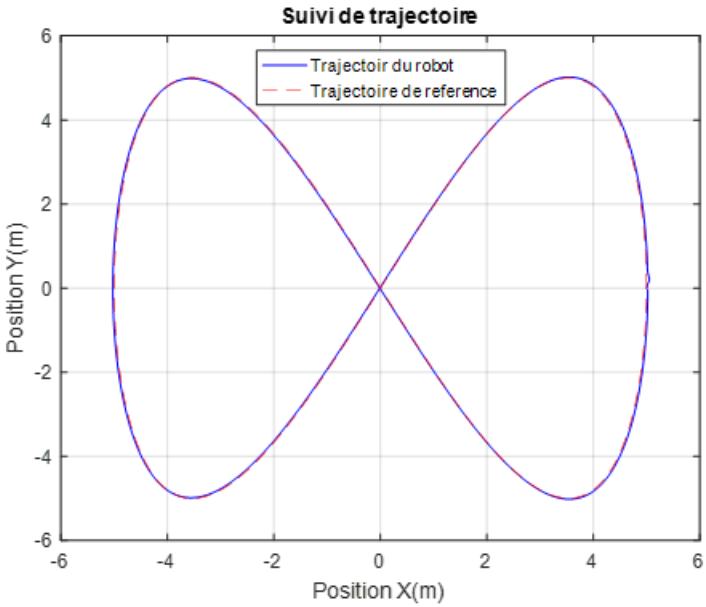


FIGURE 2.18 – Résultats de simulation de suivi de trajectoire de référence en forme de huit dans une boucle de commande cinématique.

On peut conclure d'une part que les erreurs de poursuite sont bornées par de faibles valeurs. D'autre part, l'évolution de la trajectoire montre que le robot suit effectivement les trajectoires de référence respectivement en forme de circule et en forme de huit. Ceci est obtenu grâce au choix convenable des paramètres de la stratégie de commande.
Par ailleurs, l'utilisation du contrôleur cinématique néglige la dynamique du système telle que la masse du robot mobile, le moment d'inertie, le coriolis et les vitesses centrifuges. La

prochaine étape pour améliorer l'architecture de contrôle de ce système consiste à ajouter un contrôleur de linéarisation par bouclage à retour d'état non linéaire au contrôleur cinématique. Cette méthode linéarisera la dynamique du système non linéaire et améliorera la réponse du suivi de trajectoire. Les résultats de simulation de la combinaison du contrôleur cinématique et du controleur par linéarisation par bouclage à retour d'état non linéaire sera présenté ci-dessous .

Résultats de simulation de la boucle cinématique et dynamique

Les Figures (2.19, 2.20, 2.21, 2.22 et 2.23) présentent respectivement l'évolution de poursuite en position selon les deux axes x ,y et l'évolution de la poursuite des vitesses linéaire et angulaire, ainsi les profils des erreurs de poursuite en position selon les deux axes$x$ et$y$.

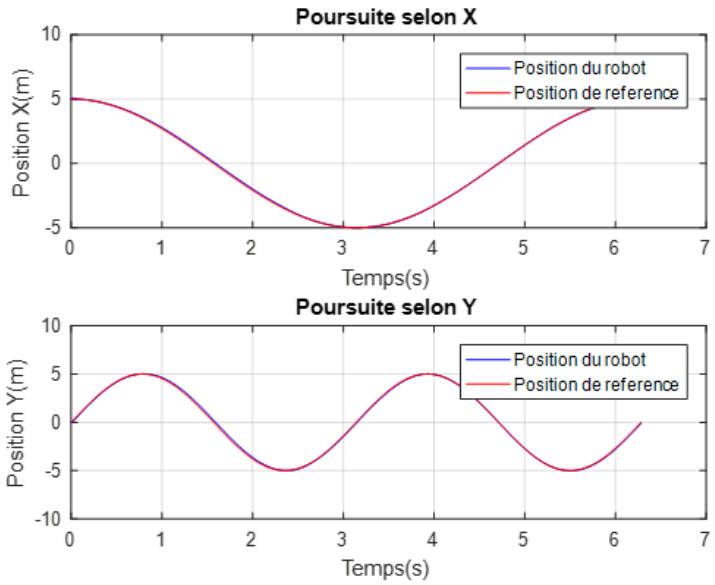


FIGURE 2.19 – Résultats de simulation de poursuite selon X et Y d'un robot mobile (DDMR) dans une boucles cinématiques et dynamique.

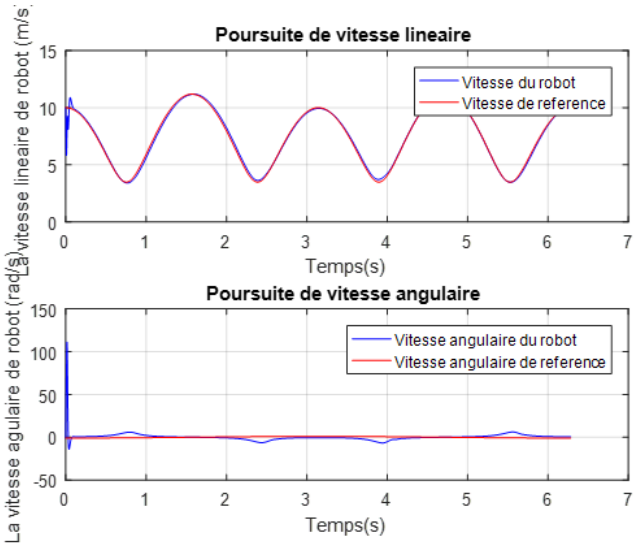


FIGURE 2.20 – Résultats de simulation des poursuites des vitesses d'un robot mobile (DDMR) dans une boucles cinématiques et dynamique.

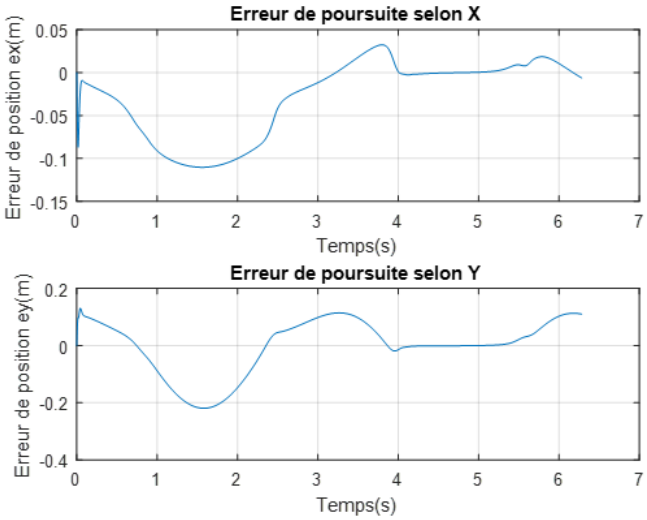


FIGURE 2.21 – Résultats de simulation des erreurs de poursuite selon X et Y d'un robot mobile (DDMR) dans une boucles cinématiques et dynamique.

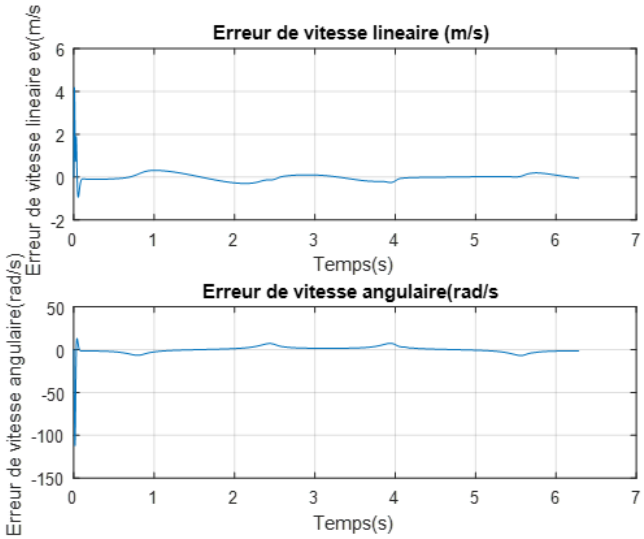


FIGURE 2.22 – Résultats de simulation des erreurs des vitesses linéaire et angulaire d'un robot mobile (DDMR) dans une boucles cinématiques et dynamique.

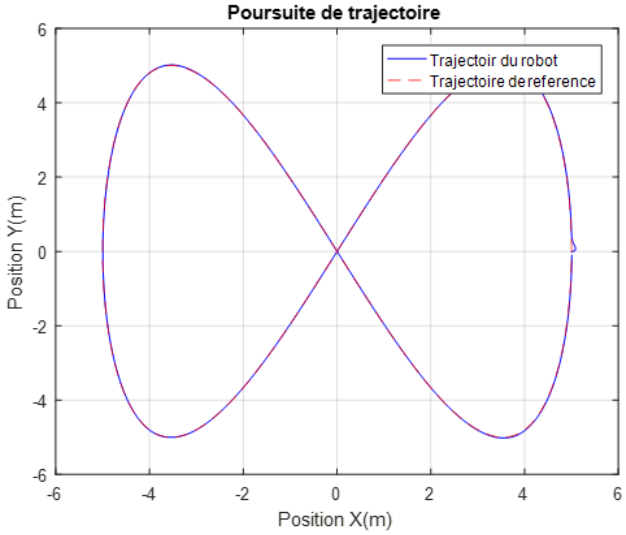


FIGURE 2.23 – Résultats de simulation de suivi d'une trajectoire en forme de huit répéter d'un robot mobile (DDMR) dans une boucles cinématiques et dynamique.

Ces résultats montrent des qualités de poursuite satisfaisantes. En effet, nous constatons d'une part que les erreurs de poursuite sont bornées par de faibles valeurs et d'autre part l'évolution de la trajectoire montre que le robot mobile suit effectivement la trajectoire de référence de forme huit répété.

2.6 Conclusion

Les travaux présentés dans ce chapitre ont porté sur la modélisation du robot mobile de type entrainement différentiel (DDMR). Premièrement le modèle cinématique, décrit

les vitesses du système est présenté par un ensemble d'équations différentielles du premier ordre. Deuxièmement, le modèle dynamique décrit les accélérations du robot présenté par un ensemble d'équations différentielles d'ordre deux synthétisées grâce l'application du formalisme de Lagrange pour fournir le modèle DDWMR non holonome sans dynamique de glissement.

Enfin, on a validé les deux modèles par une simulation sur Matlab en utilisant deux contrôleurs cinématique et dynamique qui assure respectivement le suivi de trajectoire et asservissement en vitesses.

Chapitre 3

Commande du robot mobile à entrainement différentiel par les techniques de l'intelligence artificielle ANFIS et optimisation par PSO

3.1 Introduction

Les outils d'intelligence artificielle telles que la logique floue, les réseaux de neurones et les réseaux neuro-flous sont des techniques utilisées pour résoudre plusieurs problèmes dans le domaine de l'industrie et de la robotique. Leurs champs d'application sont très vastes, afin d'atteindre les buts d'automatisation (coût, émission, pertes de puissance...), la modélisation et commande des systèmes industriels, et bien d'autres domaines.
La plupart des systèmes robotiques, présentent des modèles dynamiques non linéaires et complexes, ou parfois inconnus, ce qui rend leur commande très délicate. L'utilisation de l'intelligence artificielle est alors menée pour remédier à ces contraintes de contrôle, notamment la nécessité de disposer d'un modèle mathématique précis, sachant que les erreurs de modélisation affectent négativement les lois de commande, et contribuent à dégrader les performances des contrôleurs.
La logique floue, qui est une approche sans modèle, peut être utilisée pour contrôler un système non linéaire à plusieurs entrées-sorties, en utilisant des règles floues basées sur un raisonnement humain, défini par différentes fonctions d'appartenance [89]. Cette stratégie de contrôle peut offrir de bonnes performances, mais, il est difficile de déterminer les fonctions d'appartenance pour chaque entrée-sortie [90]. L'utilisation des réseaux de neurones artificiels est alors une solution adéquate. Ces réseaux permettent d'ajuster automatiquement les paramètres des entrées et des sorties du système flou [91], et ceci, en utilisant leur grande puissance de calcul grâce à leur capacité d'apprentissage, à l'aide

d'un algorithme d'optimisation non linéaire tel que la rétro-propagation [90, 92]. La combinaison de la logique floue et des réseaux de neurones artificiels permet de tirer profit des capacités d'inférence de la logique floue avec la capacité d'apprentissage des réseaux de neurones [65, 93], afin de former des réseaux neuro-flous puissants. L'utilisation des réseaux neuro-flous pour le contrôle des systèmes robotiques est alors appropriée pour compenser de la dynamique non linéaire de ces systèmes et les incertitudes dues aux retards de transmission et au contact avec l'environnement inconnu.
Dans la première partie de ce chapitre, nous allons présenter les principes de la logique floue et la commande floue. La deuxième partie sera consacrée à la description des réseaux de neurones artificiels et leurs différentes architectures. Par la suite, dans la troisième partie du chapitre, nous allons aborder les processus d'apprentissage. La quatrième partie sera consacrée à la description des techniques neuro-floues ainsi que les différentes combinaisons les plus utilisées dans le contrôle des systèmes. Dans la cinquième partie nous allons présenter des généralités sur l'optimisation méta-heuristique par essaim particulaire (PSO).
Enfin, la dernière partie sera consacrée à la synthèse de deux contrôleurs de type ANFIS pour commander le robot mobile à entraînement différentiel, dont le premier contrôleur permet de chercher la cible et le deuxième contrôleur permet d'éviter les obstacles. Par la suite nous allons développer une méthode d'optimisation méta-heuristique de type essaim de particule (PSO) en vue d'atteindre la navigation optimale du robot mobile qui est souvent l'un des défis majeurs le plus rencontrés dans l'implémentation en temps réel. Notre technique repose sur une hybridation entre neuro floue et le principe de l'optimisation approchée par l'essaim de particule (PSO) pour la navigation du robot mobile à entrainement différentielle de telle manière à chercher la cible tout en évitant les obstacles.

3.2 Généralité sur logique floue

3.2.1 Logique floue

Le professeur L.A. Zadeh [94] a établi les fondements théoriques de la logique floue au début des années soixante au juste 1965. Cette technique combine deux notions, les sous-ensembles flous et la théorie des possibilités qui permet la prise en compte de l'imprécision, en se basant sur le raisonnement humain plutôt que sur des calculs rigoureux [65, 95].
En outre, la logique floue est une méthodologie mathématique permettant de prendre en compte toutes les sortes de connaissances qualitatives de concepteurs et d'opérateurs dans l'automatisation des systèmes. Par conséquent les systèmes flous représentent un champ d'application intéressant pour les systèmes complexes non linéaires, le traitement d'images pour améliorer les caractéristiques d'images numériques couleur (luminosité,

teinte, brillance), la robotique mobile.
Généralement la logique floue s'y développe car il s'agit d'une approche efficace et générique [96].
La logique floue est bien adaptée aux informations imprécises, incertaines et vagues, par exemple en parlant de température de l'eau elle peut être dite froide, chaude, tiède ou en parlant de la taille d'une personne, petite, grande, etc.
La logique floue permet d'appréhender les phénomènes naturels difficilement modélisable, et cela, en utilisant les ensembles flous (fonctions d'appartenances) avec une définition des règles floues. Cette approche n'a besoin ni d'une identification ni d'un modèle mathématique du processus, en se basant uniquement sur le comportement du processus, on peut l'utiliser pour réaliser un contrôleur flou. La logique floue est utilisée dans toutes les applications où il y a un raisonnement humain [65].
Par ailleurs, la théorie des ensembles flous fournit une méthode appropriée, facilement et réalisable dans des applications temps réel. Ces caractéristiques rendent la logique floue abordable par les chercheurs avec une continuité d'applications.
Nous présentons les principaux travaux et la diffusion de cette méthode.

- En 1965 le concept introduit par Lotfi Zadeh "Fuzzy set theory". Définition des ensembles flous et opérateurs associés.
- En 1970 l'application des théories de la logique floue en médecine, commerce, etc.
- En 1974 Mamdani a validé d'une manière expérimentale la théorie énoncée par Zadeh sur la régulation floue d'une chaudière à vapeur.
- En 1985 les Japonais introduisent des produits grand public "Fuzzy Logic Inside".
- En 1990 la généralisation d'utilisation de cette technique.

Comparativement aux systèmes logiques basés sur des variables booléennes qui ne peuvent prendre que deux valeurs (0 ou 1) pour mettre des relations mathématiques, les systèmes flous peuvent être décrits par des règles linguistiques. Dans ce contexte les concepts de base de la logique floue sont les ensembles flou, variables flous, la décision à partir d'une base de règles SI... ALORS... et l'inférence floue. Nous avons cité trois classe, les systèmes flous linguistiques ou systèmes de Mamdani et systèmes flous de type Takagi-Sugeno-Kang (TSK) [97, 98].

3.2.2 Concepts fondamentaux flous

Le concept d'ensemble flou a été introduit pour éviter le passage brusque d'une classe à une autre (par exemple, de la classe noire à la classe blanche) et autoriser des éléments qui n'appartiennent pas complètement ni à l'une ni à l'autre (à être gris, par exemple),

ou encore appartiennent partiellement à chacune (avec un fort degré à la classe noire et un faible degré à la classe blanche, dans le cas du gris foncé). La définition d'un ensemble flou répond au besoin de représenter des connaissances imprécises, soit parce qu'elles sont exprimées en langage naturel par un observateur qui n'éprouve pas le besoin de fournir plus de précision ou n'en est pas capable, soit parce qu'elles sont obtenues avec des instruments d'observation qui produisent des erreurs de mesure [99, 100, 101]. La notion des ensembles flous nous permet de traiter :

- Des catégories aux limites mal définies.
- Des situations intermédiaires entre le tout et le rien.
- Le passage progressif d'une propriété à une autre.
- Des valeurs approximatives.

Donc, le concept des ensembles flous constitue un assouplissement de celui d'un ensemble donné.

3.2.3 Ensemble flou

La logique floue repose sur la théorie des ensembles flous, comme déjà mentionné elle est bien adaptée pour la description des phénomènes imprécis (exprimées soit par un langage humain, ou ils sont obtenus par des capteurs de mesure et d'observation imprécises). Ces phénomènes sont modélisés sous forme d'ensembles flous qui sont représentés par des variables linguistiques. Pour traiter numériquement ces variables linguistiques, il faut les soumettre à une définition mathématique basée sur les fonctions d'appartenance.
Un ensemble flou A d'un univers de discours U, est défini par une fonction d'appartenance, notée μ_A , on associe à chaque élément x de U un degré d'appartenance $\mu_A(x)$ indiquant le niveau d'appartenance de x à A. $\mu_A(x) = 1$ et $\mu_A(x) = 0$ correspond respectivement à l'appartenance et à la non-appartenance.
La Figure (3.1) montre la variable linguistique « AGE » représentée par trois sous-ensembles exprimés par les termes linguistiques « Jeune, Moyen et Vieux », sur un univers de discours compris entre 0 et 100.

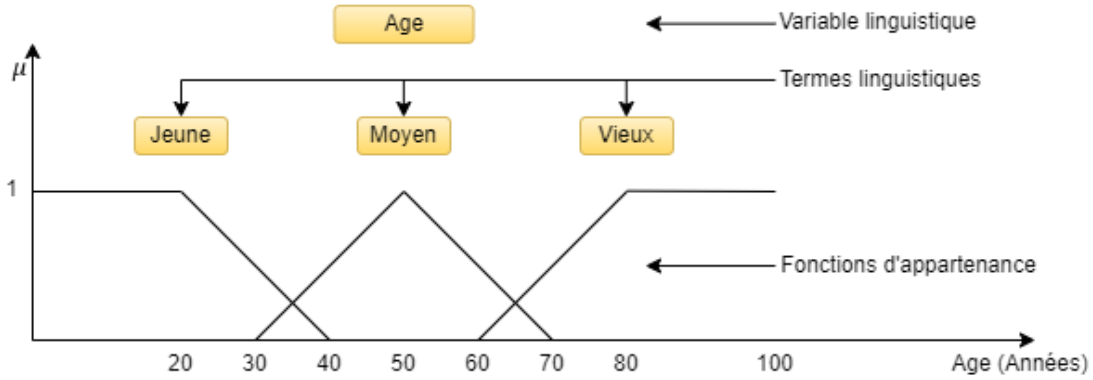


FIGURE 3.1 – les termes linguistiques des fonctions d'appartenance.

3.2.4 Variable linguistique

Une variable linguistique décrit une situation imprécise par des mots ou des expressions tels que : (négative, zéro, positive) ou (petit, très petit, moyen, grand, très grand). Une variable linguistique prend une valeur linguistique qui est un ensemble flou défini sur l'univers de discours [102].

3.2.5 Fonction d'appartenance

Les variables linguistiques sont représentées par des fonctions d'appartenance, de telle manière à ce que chaque variable floue x d'un ensemble flou A soit associée à une fonction d'appartenance qui représente le degré d'appartenance de x à A [103].
Les fonctions d'appartenance peuvent être représentées sous plusieurs formes selon l'application. Les formes les plus couramment utilisées dans la théorie du contrôle flou sont les fonctions triangulaires, trapézoïdales, sigmoïdes et gaussiennes [103].

- **Fonction triangulaire :**Elle est caractérisée par trois paramètres (a,b et c) qui définissent les coordonnées des trois sommets (Figure 3.2).

$$\mu_A(x) = max(min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0) \tag{3.1}$$

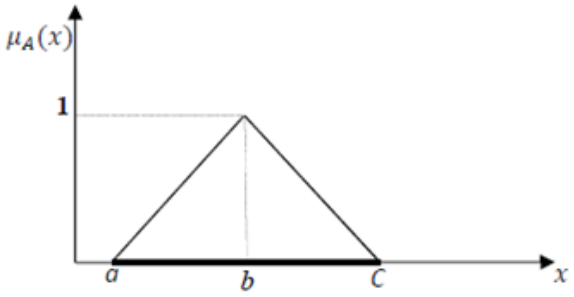


FIGURE 3.2 – Fonction triangulaire.

- **Fonction trapézoïdale :**Cette fonction est définie par quatre paramètres (a,b,c et$d$) qui déterminent les coordonnées des quatre sommets (Figure 3.3)

$$\mu_A(x) = max\left(min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right) \tag{3.2}$$

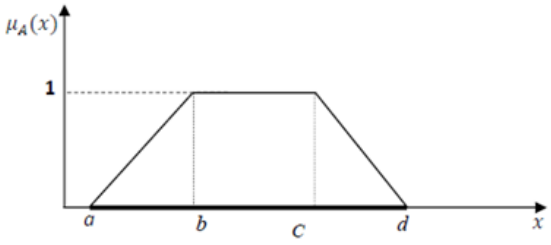


FIGURE 3.3 – Fonction trapézoïdale.

- **Fonction gaussienne :**Elle est caractérisée par deux paramètres (σ et$m$) (Figure 3.4).

$$\mu_A(x) = exp\left(-\frac{(x-m)^2}{2\sigma^2}\right) \tag{3.3}$$

Où, m etσ représentent respectivement, le centre et l'écart type de la fonction $\mu_A(x)$.

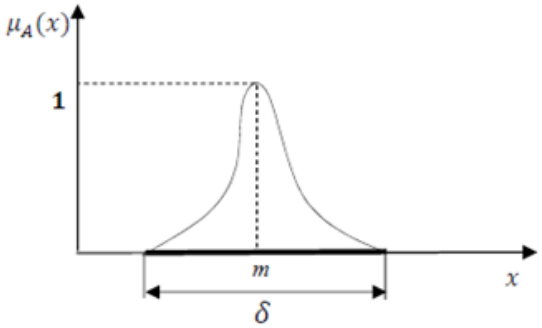


FIGURE 3.4 – Fonction gaussienne.

- **Fonction sigmoïde :**La fonction sigmoïde est définie par deux paramètres (a et b) (Figure 3.5).

$$\mu_A(x) = \left(\frac{1}{1 + \exp(-a(x-c))} \right) \tag{3.4}$$

Où a permet de contrôler la pente au point d'inflexion $x = c$.

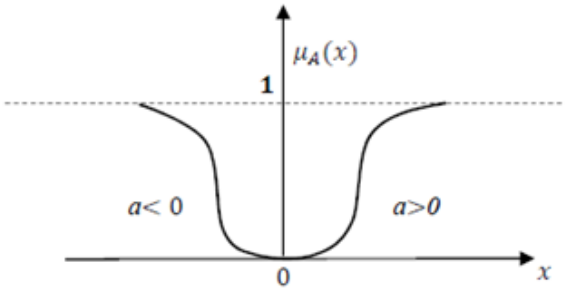


FIGURE 3.5 – Fonction sigmoïde.

3.2.6 Univers de discours

L'univers de discours représente le domaine de variation de la variable linguistique. Pratiquement, il représente le domaine du fonctionnement du processus à commander qu'il faut prendre en considération lors du réglage du régulateur [95].

3.2.7 Règle d'inférence

Les règles d'inférence sont l'ensemble des règles floues qui permettent de relier les variables floues d'entrée aux variables floues de sortie d'un système, au moyen de différents opérateurs flous [104]. Ces règles sont décrites sous la forme [105] comme suit :

Si condition 1 ET/OU condition 2 (ET/OU. . .) alors action sur les sorties OU
Si condition 3 ET/OU condition 4 (ET/OU. . .) alors action sur les sorties OU
...
Si condition n ET/OU condition n + 1 (ET/OU. . .) alors action sur les sorties. . .

3.2.8 Opérations sur les ensembles flous

La sortie d'un système flou dédié par l'utilisation des opérateurs logique tels que : "ET", "OU", et "NON" avec des théories qui sont interprétés par les opérations : "Minimum", "Maximum" et " complément à un" [106]. Conformément à la théorie du système flou on définit la réunion, l'intersection, le complément. . . d'ensembles flous.

- **La réunion :** L'ensemble flou de l'opération OU est un ensemble flou de fonction d'appartenance réalisé par :

$$\mu_{A\cup B}(x) = max(\mu_A(x)\ , \mu_B(x))\ \forall\ x\epsilon\ U \tag{3.5}$$

- **L'intersection :** L'ensemble flou de l'opération ET est un ensemble flou de fonction d'appartenance donné par :

$$\mu_{A\cap B}(x) = min(\mu_A(x)\ , \mu_B(x))\ \forall\ x\epsilon\ U \tag{3.6}$$

- **Le complément :** L'ensemble flou de l'opération NON est un ensemble flou de fonction d'appartenance représenté par l'équation :

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \forall\ x\epsilon\ U \tag{3.7}$$

3.2.9 Commande floue

Les techniques de contrôle classique présentent beaucoup de limitations. Il est souvent difficile de trouver le formalisme mathématique modélisant le comportement du système par un ensemble d'équations mathématiques lorsque le système est non linéaire, partiellement inconnu ou avec des incertitudes dynamiques imprévisibles. Par conséquent, la logique floue est utilisée, où tous ces comportements imprévisibles peuvent être modélisés à l'aide de l'approche linguistique de Zadeh, selon un modèle proche du raisonnement humain [105].

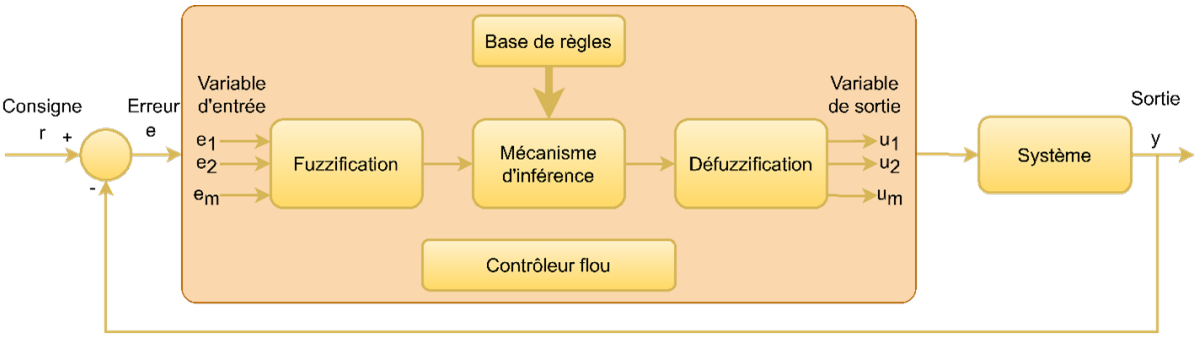


FIGURE 3.6 – schéma fonctionnelle d'un contrôleur flou.

L'utilisation de la logique floue dans le contrôle des systèmes représente un sujet de recherche d'actualité puisqu'elle permet d'obtenir une loi de commande considérablement efficace pour des systèmes complexes et fortement non linéaires sans avoir recours à un modèle mathématique bien défini, et cela en utilisant des inférences avec des règles floues basées sur des variables linguistiques. Dans ce qui suit, nous allons présenter les composants essentiels d'un régulateur flou (Figure 3.6) [105].

- **La base de connaissances (Base de règles) :**

Il s'agit d'un module contenant des indications relatives aux fonctions d'appartenance (formes et paramètres) associées aux variables d'entrée/sortie ainsi que l'ensemble des règles floues (les connaissances de l'expert humain) [96].
La base de règles d'un système flou décrit en termes qualitatifs le comportement d'une sortie lorsqu'elle est soumise à diverses entrées. Ce comportement est décrit par des règles d'inférence explicitées précédemment.
Dans ce cas, les entrées peuvent être une erreur, une variation de l'erreur ou une somme d'erreurs, et la sortie peut être l'action de contrôle ou l'action de variation de contrôle.

- **Fuzzification :**

La fuzzification consiste à transformer des grandeurs numériques générées par des capteurs à l'entrée en partie floue définie sur un espace de représentation lié à l'entrée. Elle caractérise le degré avec lequel ces grandeurs numériques appartiennent à un sous-ensemble flou, en utilisant différentes formes de fonctions d'appartenance dont le choix du nombre et de la forme est réalisé par l'expert selon le domaine d'application et la facilité de calculs [105].
Considérons une entrée e (erreur) définie par l'ensemble des variables linguistiques : $N = Négative$, $Z = Zéro$, $P = Positive$. L'univers de discours associé est $[-1 \quad 1]$ (Figure 3.7).

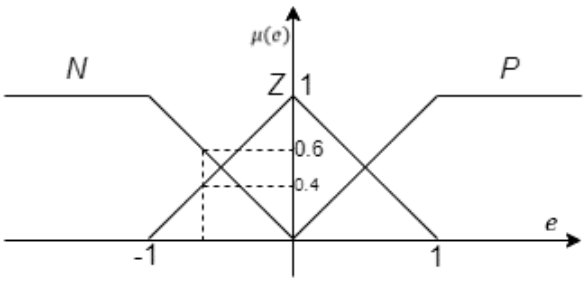


FIGURE 3.7 – Exemple de fuzzification.

Les degrés d'appartenance de l'entrée e aux sous-ensembles sont :

$$\mu_N(e) = 0.6$$

$$\mu_Z(e) = 1 - 0.6 = 0.4$$

$$\mu_P(e) = 0 \tag{3.8}$$

- **Mécanisme d'inférence :**

Cette étape consiste à transformer la partie floue obtenue par la fuzzification en une nouvelle partie floue [105]. Elle comprend trois composants : une base de règles contenant une sélection de règles floues, une base de données qui définit les fonctions d'appartenance utilisées dans les règles floues et un mécanisme de raisonnement qui exécute la procédure d'inférence sur les règles pour obtenir une conclusion raisonnable. Si la sortie de la conséquence est exprimée par une variable linguistique, le régulateur flou est de type Mamdani, et si la sortie est une valeur numérique ou une fonction mathématique, le régulateur est de type Takagi-Sugeno [105]. L'inférence floue nécessite l'utilisation d'un opérateur flou pour chaque règle. Plusieurs méthodes s'appliquent pour la réalisation de ces opérateurs, néanmoins les méthodes d'inférence floue MAX-MIN de Mamdani et somme-produit de Sugeno sont les plus utilisées [105].

- **Défuzzification :**

Cette étape représente l'inverse de la fuzzification. Elle consiste à transformer la partie floue provenant de l'inférence en une valeur numérique pour former un signal de contrôle dédié au système à commander [105, 107]. Plusieurs méthodes de défuzzification existent dans la littérature telles que la méthode du centre de gravité, la moyenne des maximas et le centre des maximas [105]. Néanmoins, la méthode la plus utilisée dans le contrôle flou est la défuzzification par centre de gravité, qui consiste à calculer l'abscisse du centre de gravité de la surface générée par la fonction d'appartenance résultante.

- **La technique du centre de gravité :**

Le centre de gravité est la méthode de défuzzification la plus connue et généralement la plus performante, mais elle possède l'inconvénient d'être couteuse en temps de calcul. En effet, elle ne consomme pas mal de ressources informatiques vu qu'elle transforme les variables linguistiques en données numériques. Elle décrit la sortie comme correspondant à l'abscisse du centre de gravité de la surface de la fonction d'appartenance définissant l'ensemble flou originaire de l'agrégation des conclusions. L'abscisse du centre de gravité peut être déterminé comme suit :

$$Z = \frac{\int_{z_1}^{z_0} z\mu\left(z\right) dz}{\int_{z_1}^{z_0} \mu\left(z\right) dz} \tag{3.9}$$

Au dénominateur l'intégral exprime la surface mais au numérateur il correspond au moment de la surface. En pratique le calcul du centre de gravité est estimé en calculant la moyenne d'un nombre de points échantillonnés sur la fonction (Figure 3.8) :

$$Z = \frac{\sum \mu_i z_i}{\sum \mu_i} \tag{3.10}$$

Plus le nombre de points retenus pour le calcul de la moyenne est élevé plus le temps nécessaire pour le calcul l'est aussi et inversement. Donc il faut faire un compromis entre la précision et le temps de calcul. Le choix de la méthode de défuzzification peut être très important vu le temps de calcul qui varie d'une méthode à une autre, par exemple dans un système embarqué le choix de la méthode est capital.

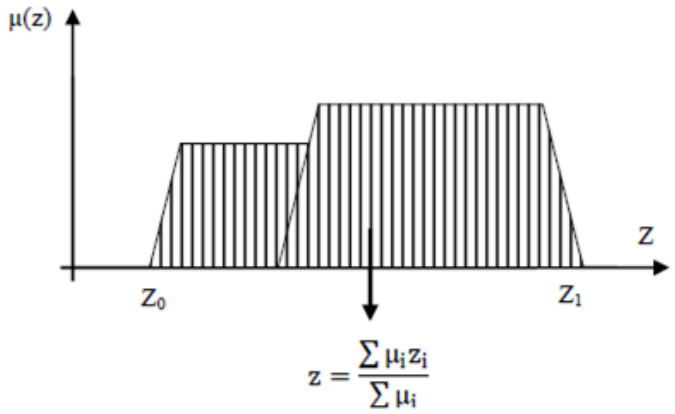


FIGURE 3.8 – Défuzzification par centre de gravité.

3.3 Généralité sur réseau de neurone artificiels

3.3.1 Réseaux de neurones

Les réseaux de neurones artificiels se révèlent être un outil très efficace. Différents exemples dans la littérature mettent en exergue l'utilité et l'efficacité des réseaux de neurones artificiels et surtout l'avenir prometteur quant à leur utilisation. Dans la dernière décennie, ils étaient presque exclusivement un domaine d'entreprises révolutionnaires avec des ressources pratiquement illimitées pour financer leurs recherches, comme Google, Facebook, Netflix, etc. Aujourd'hui, les méthodes d'intelligence artificielle semblent former une partie indivisible de toute entreprise réussie. Certains experts prédisent même que l'intelligence artificielle mènera à la prochaine révolution industrielle.
Les réseaux de neurones artificiels ont été proposés en 1943 par W. MC Culloch et W. Pitts. Ces réseaux sont capables de fournir un signal de sortie en fonction d'un certain nombre de signaux d'entrée diversement pondérés. Ils peuvent être connectés de diverses manières afin de constituer une technique de traitement de données bien comprise et maîtrisée, selon une architecture neuronale, désormais indispensable pour résoudre les problèmes qui intéressent différents aspects scientifiques. En effet, plusieurs travaux de recherche ont montré que les réseaux de neurones peuvent modéliser n'importe quel système non linéaire. En outre, ces travaux de recherche ont donné lieu à des applications très intéressantes dans plusieurs domaines [105].

3.3.2 Structure d'un neurone

Le neurone est une cellule composée d'un corps cellulaire et d'un noyau (Figure 3.9) [108, 109]. Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites. C'est par les dendrites que l'information est acheminée de l'extérieur vers le soma, corps du neurone. L'information traitée par le neurone s'oriente ensuite le long de l'axone pour être transmise aux autres neurones. La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire entre l'axone du neurone afférent et les dendrites du neurone efférent. La jonction entre deux neurones est appelée la synapse [101].

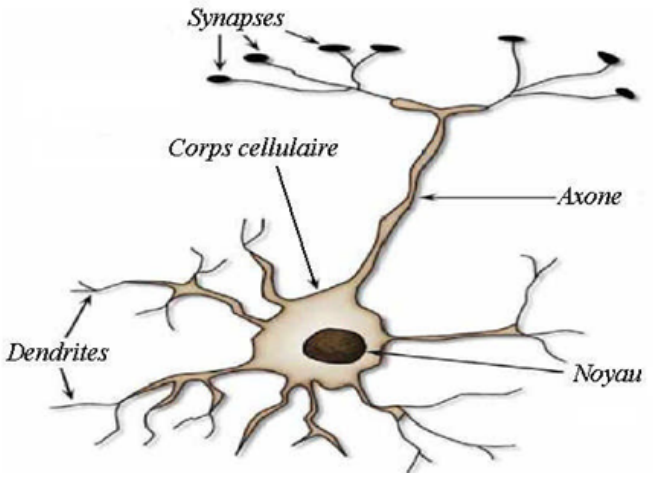


FIGURE 3.9 – Schéma simplifié d'un neurone biologique.

Par analogie avec le neurone biologique, le comportement du neurone artificiel est modélisé par deux phases comme présenté sur la (Figure 3.10) :

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

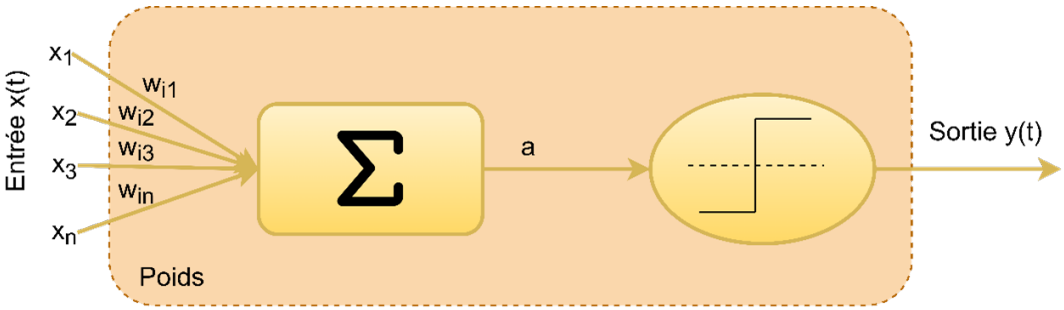


FIGURE 3.10 – Modèle d'un neurone artificiel.

Chaque neurone artificiel est un processeur élémentaire de traitement. Il reçoit un nombre variable d'entrées en provenance de neurones amont. A chacune de ces entrées est associé un poids w_{ij} représentant la force de la connexion [108, 110]. Le neurone artificiel modélisé par Mc Culloch et Pitts est représenté par la (Figure 3.10). x_i : Entrée du neurone i (ou sortie de neurone amont j). w_{ij} : La valeur du poids synaptique de la connexion dirigée du neurone j vers le neurone i,

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

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

$$\alpha_i = \sum_{j=1}^{n} w_{ij} x_j \tag{3.11}$$

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

$$y_i = f\left(\alpha_i - \theta_i\right) \tag{3.12}$$

3.3.3 Fonctions d'activation

La fonction de transfert ou d'activation définit la valeur de sortie d'un neurone en termes des niveaux d'activité de ses entrées. Cette fonction peut prendre différentes formes possibles telles que : fonction linéaire à seuil, fonction seuil, fonction gaussienne, etc (Figure 3.11).

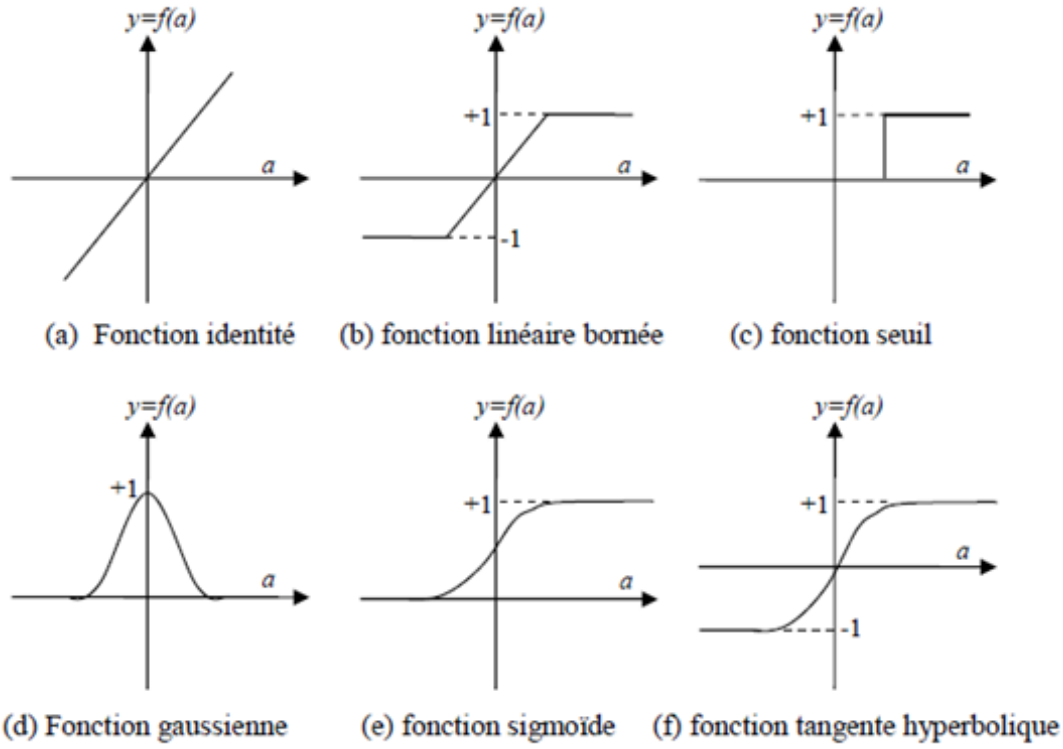


FIGURE 3.11 – Formes usuelles de la fonction d'activation.

3.3.4 Architecture des réseaux de neurones

Un réseau de neurones est un composant des organes non linéaires de traitement d'information du système physique, consacré à une méthode d'interconnexions entre ces organes et une loi d'apprentissage pour adapter les poids de connexions. La forme des réseaux de neurones est caractérisée par leurs partitions (Figure 3.12). Ces partitions sont les couches [111, 96].

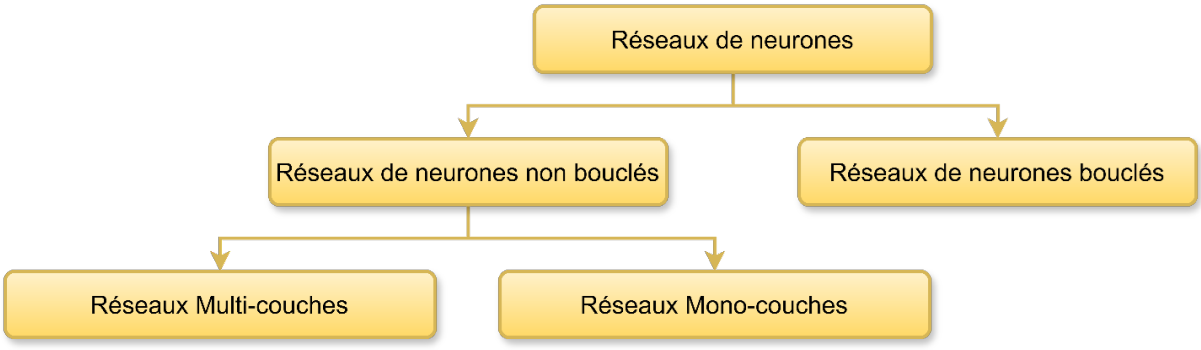


FIGURE 3.12 – Différentes types des réseaux de neurones.

Réseaux de neurones non bouclés (feed forward)

Un réseau de neurone non bouclé est composé d'un ensemble de neurones connectés entre eux, et l'opération des entrées vers les sorties s'effectue sans retour. Ce type de réseau

est devisé en deux architectures : les réseaux Mono-Couches et les réseaux Multi-Couches.

- **Réseaux Mono-Couches**

Ce type de réseau comme représenter dans la Figure (3.13) se forme par deux couches, une couche recevant les valeurs d'entrée qui applique un traitement de ces valeurs par l'intermédiaire des noeuds et une couche de sortie transmettant les résultats du traitement au milieu extérieur.

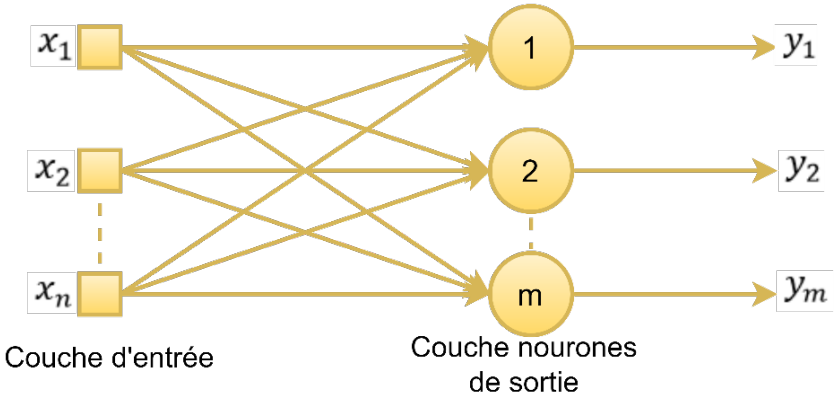


FIGURE 3.13 – Réseau de neurones artificiels non bouclé mono-couche.

- **Réseaux Multi-Couches**

Ces réseaux proactifs comprennent d'une ou de plusieurs couches cachées, dont les neurones de réseau sont des nœuds cachés ou unités cachées qui effectuent le calcul correspondant. Nous pouvons citer les plus utilisables, les réseaux perceptrons multicouches (Multi Layer Perceptron) MLP. Les réseaux neurones MLP sont basés sur l'algorithme de rétropropagation. Ce type est très répandu dans le réseau de neurone et peut être utilisé comme un système pratique pour effectuer une cartographie d'entrée / sortie non linéaire de nature générale, les étapes d'algorithme de rétro-propagation sera décrit ci-dessous.
Les réseaux MLP contient des couches. Chaque couche de connexion peut être assimilée à une fonction mathématique linéaire appliquée aux entrées. Celle-ci étant répartie sur l'ensemble des connexions. Ces réseaux se composent par trois couches :

- Couches d'entrée avec Ri des unités d'entrée.
- Couches cachées avec Rc des unités cachées.
- Couches de sorties avec Rs des unités de sortie.

La Figure 3.14 montre de façon schématique comment sont ordonnées les couches du réseau multicouches perceptrons de n entrées et m sorties.

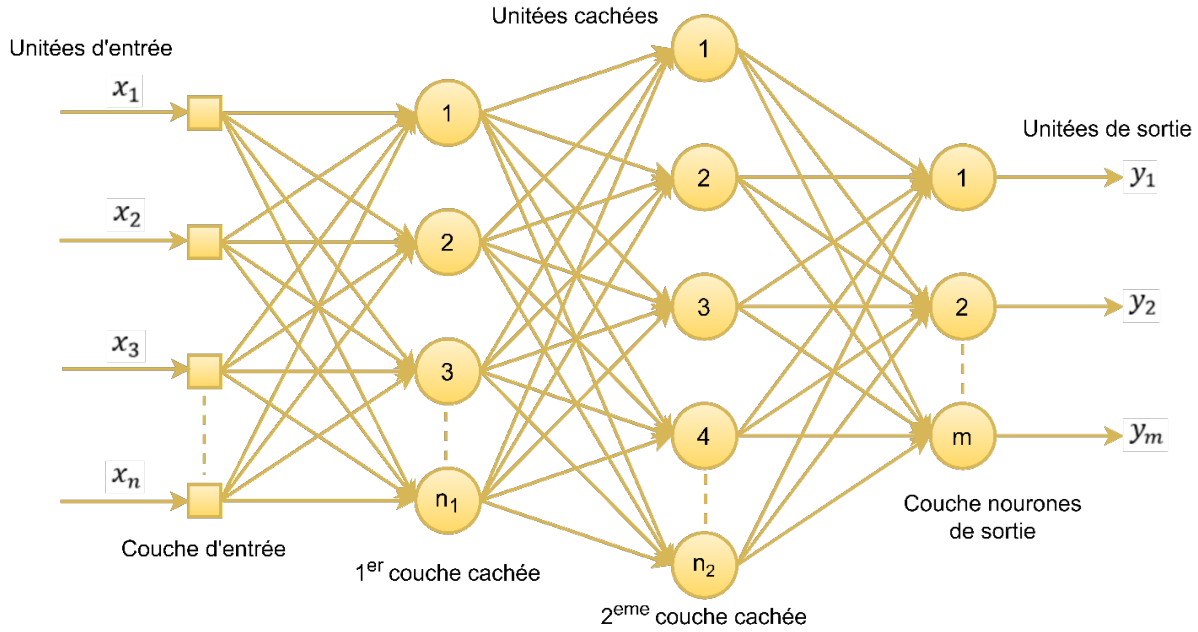


FIGURE 3.14 – Architecture générale d'un réseau multi-couche.

Réseaux de neurones bouclés (récurrents)

Un réseau de neurone bouclé réalise une ou plusieurs équations aux différences non-linaires, telle que la direction des fonctions réalisées vers la sortie avec retour [112]. Dans ce type d'architectures, les sorties des neurones (couches) sont utilisées comme des entrées de rétroaction pour d'autres neurones (couches), permettant ainsi aux signaux de circuler dans les deux sens (Figure 3.15).

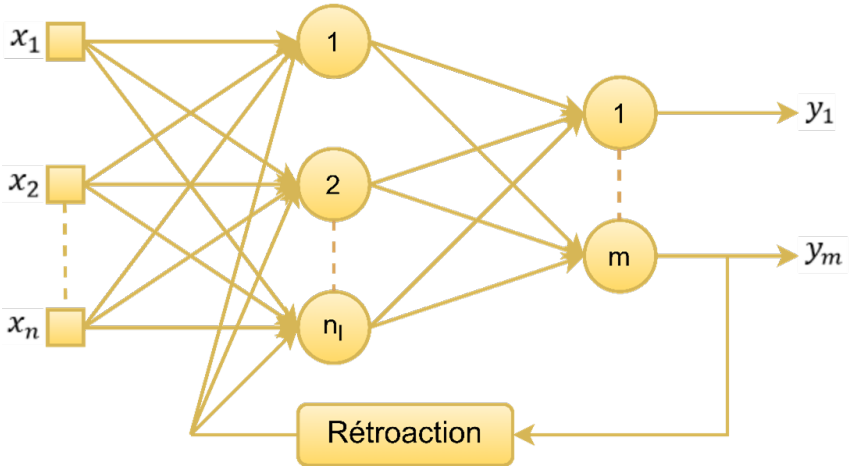


FIGURE 3.15 – Réseau de neurones artificiels bouclé.

Ces réseaux sont très puissants, leur dynamique leur permet de changer leurs états continuellement jusqu'à atteindre un point d'équilibre [113]. Ce type de réseaux est utilisé généralement pour la prédiction temporelle, l'identification, l'optimisation et le contrôle des processus. Parmi les principaux réseaux de neurones bouclés, on cite les réseaux de

Hopfield et de Perceptron [114].

3.4 Processus d'apprentissage

L'apprentissage du réseau de neurones permet la modification des poids entre les neurones ainsi que la valeur des biais de façon à améliorer les performances des réseaux de neurones en vue d'adapter le traitement effectué [115]. Nous pouvons citer deux principaux types d'apprentissage.

3.4.1 Apprentissage supervisé

Ce type consiste à un superviseur qui fournit une valeur de sortie (sortie désirée), que le réseau de neurones doit associer à une valeur d'entrée. Cet apprentissage permet d'ajuster les paramètres du réseau afin de minimiser l'erreur entre la sortie désirée et la sortie réelle du réseau, comme le montre la Figure 3.16.

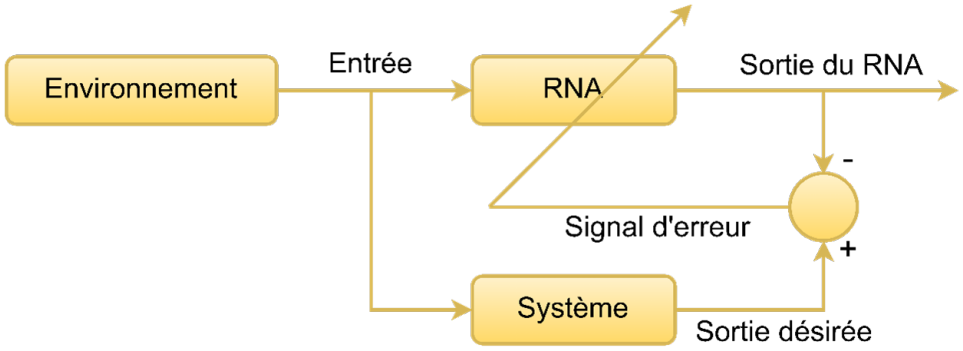


FIGURE 3.16 – Apprentissage supervisé.

- **Algorithme : Algorithme de rétro-propagation**

La méthode de rétro-propagation de l'erreur est une technique d'apprentissage supervisé. Elle permet d'ajuster les poids et les biais du RNA afin de minimiser l'erreur quadratique entre la sortie calculée du réseau et sa sortie réelle. Pour chaque couple entrée/sortie, une erreur est calculée. Si la réponse du réseau est différente de la réponse réelle, les poids et les biais sont modifiés en ligne sur le réseau de manière à tendre l'erreur vers zéro [95]. Considérons le réseau de neurone à trois couches (n, m, p) illustré dans la (Figure 3.17), et soit x^q le vecteur d'entrée du réseau et y^q le vecteur des sorties désirées.

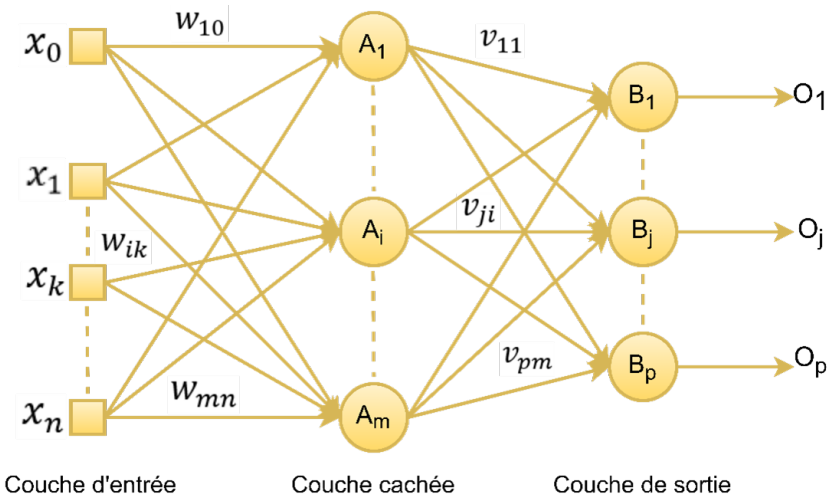


FIGURE 3.17 – Réseau de neurones à trois couches.

Considérons le problème de minimisation de la fonction objectif, définie par l'erreur quadratique E^q entre la sortie calculée du réseau O^q et la sortie désirée y^q.

$$E^q = \frac{1}{2}\sum_j \left(O_j^q - y_j^q\right)^2 \tag{3.13}$$

Soit v_{ji} les poids de connexion du neurone caché i au neurone de sortie j, et w_{ik} les poids de connexion du nœud d'entrée k au neurone caché i. L'objectif est calculé les poids v_{ji} et w_{ik} afin de minimiser E^q. Les équations des mises à jour des poids v_{ji} et w_{ik} sont définies par :

$$v_{ji} \rightarrow v_{ji} + \Delta v_{ji} \tag{3.14}$$

$$w_{ik} \rightarrow w_{ik} + \Delta wik \tag{3.15}$$

Premièrement, les poids v_{ji} sont ajustés en utilisant la méthode de la descente du gradient. Pour un neurone de sortie j et un neurone caché i, les poids v_{ji} sont mis à jour par [103] :

$$\Delta v_{ji} = \sum_{j=1}^{N} \Delta^q v_{ji} = -\eta \sum_{j=1}^{N} \frac{\partial E^q}{\partial v_{ji}} = \sum_{j=1}^{N} \left(-\eta \delta_j^q z_i^q\right) \tag{3.16}$$

Où z_i^q est l'entrée du neurone caché i, définie par :

$$z_i^q = f_i\left(\sum_{k=0}^{n} w_{ik} x_k^q\right) \tag{3.17}$$

$$\delta_j^q = \left(O_j^q - y_j^q\right) f_i'\left(\sum_{k=1}^{m} v_{ik} z_k^q\right) \tag{3.18}$$

Où f est la fonction d'activation. Pour un neurone caché i et un noeud d'entrée k, les poids w_{ik} sont mis à jour comme suit :

$$\Delta^q w_{ik} = -\eta \frac{\partial E^q}{\partial w_{ik}} \tag{3.19}$$

Avec :

$$\frac{\partial E^q}{\partial w_{ik}} = \frac{\partial E^q}{\partial O_i^q} \frac{\partial O_i^q}{\partial w_{ik}} \tag{3.20}$$

Où η est le taux d'apprentissage, et O_i^q est la sortie du neurone caché i, définie par :

$$O_i^q = f_i \left(\sum_{k=0}^{n} w_d x_i^q \right) = z_i^q \tag{3.21}$$

Nous avons :

$$\frac{\partial O_i^q}{\partial w_{ik}} = f_i' \left(\sum_{i=0}^{n} w_{ik} x_i^q \right) x_k^q \tag{3.22}$$

Soit $\delta_i^q = \frac{\partial E^q}{\partial O_i^q}$ dans la couche cachée, défini par :

$$\delta_i^q = \frac{\partial E^q}{\partial O_i^q} = \sum_{j=1}^{p} \frac{\partial E^q}{\partial O_j^q} \frac{\partial O_j^q}{\partial O_i^q} \tag{3.23}$$

Où j est dans la couche de sortie. le terme $\frac{\partial E^q}{\partial O_j^q}$ est calculé précédemment, donc :

$$O_i^q = f_j' \left(\sum_{l=1}^{m} v_{il} z_l^q \right) v_{ji} \tag{3.24}$$

Donc, δ_i^q pour le neurone caché i, est calculé à partir des valeurs déjà connues de δ_j^q pour tout j de la couche de sortie.
Cet algorithme est appelé algorithme de rétro-propagation parce qu'il commence à transmettre les modèles d'entrée x^q afin d'atteindre la couche de sortie, puis calcule les δ_j^q pour tout neurone de sortie j, ensuite propage les δ_j^q en arrière vers la couche inférieure (la couche cachée), afin de calculer les δ_i^q pour tous les neurones i de cette couche.

3.4.2 Apprentissage par renforcement

Dans ce type d'apprentissage, nous supposons qu'un comportement de référence n'est pas possible, mais en compensation, il est possible d'obtenir des indications qualitatives (vrai, faux, ...) sur les performances du réseau.

3.4.3 Apprentissage non supervisé

Cet apprentissage est caractérisé par l'absence de d'information sur la sortie désirée des données et l'absence de superviseur. La tâche du réseau consiste par exemple, à générer des regroupements de données selon des propriétés communes.

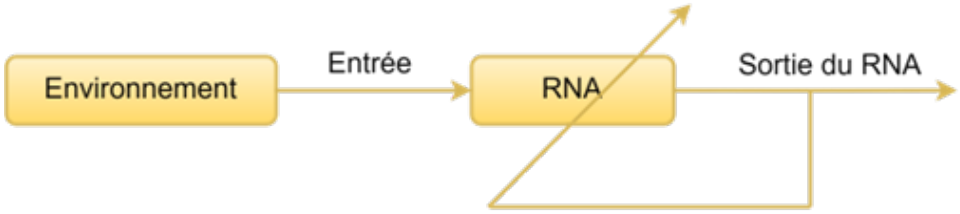


FIGURE 3.18 – Apprentissage non-supervisé.

3.5 Réseaux neuro-flous

3.5.1 Définition des systèmes neuro-flous

L'utilisation conjointe des réseaux de neurones et de la logique floue permet de tirer les avantages des deux méthodes ; les capacités d'apprentissage de la première et la lisibilité et la souplesse de la seconde. Diverses combinaisons de ces deux méthodes ont été développées depuis 1988. Elles ont donné naissance aux systèmes neuro-flous, qui sont le plus souvent orientés vers la commande de systèmes complexes et les problèmes de classification [101].
Selon la définition de George Lee un système neuro-flou est un réseau de neurones multicouches avec des paramètres flous, ou un système flou mis en application sous une forme distribuée parallèle. Par contre un système neuro-flou n'est pas un système expert, mais il peut être utilisé comme un approximateur universel [65].

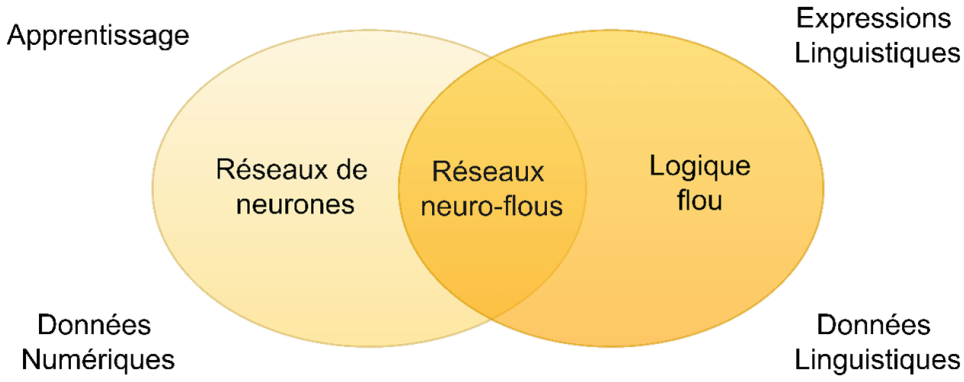


FIGURE 3.19 – Le principe de neuro-flou.

Le système neuro-floue est un système floue formé par un algorithme d'apprentissage inspiré de la théorie des réseaux de neurones et règles de la logique flou. La Figure (3.19)

expose l'intégration des réseaux de neurones et les systèmes d'inférence floue et présente le principe du système neuro-floue [116]. Le système neuro-floue hybride modifie sa structure interne pour refléter le rapport entre les entrées et les sorties dans l'ensemble de la formation. Aussi la précision d'un système neuro-floue est vérifiée après que le cycle de d'apprentissage soit complet en utilisant un ensemble séparé d'entrées et sorties appeler l'ensemble de la validation.

3.5.2 Motivations pour une approche hybride

Les dernières années ont vu se développer de nombreux travaux sur l'étude conjointe des réseaux de neurones et de la logique floue. Les raisons de cet engouement sont sans doute à rechercher dans les nombreuses similarités qui existent entre les deux approches. Tout d'abord les deux techniques présentent des propriétés d'approximateurs universels de fonctions. En effet il a été montré qu'il est possible d'approcher n'importe quelle fonction continue à l'aide d'un système flou ou d'un réseau neuronal classique à couches. On peut aussi trouver des similitudes au niveau de la structure. De même on peut trouver un point commun entre les fonctions réalisées par les neurones formels et les fonctions d'appartenance caractérisant les différentes variables d'un système flou. Enfin on peut établir une corrélation entre les opérations de multiplication et d'addition induites par la structure neuronale et celles de maximisation et de minimisation correspondant aux conjonctions et aux disjonctions des règles [117, 118].
Cependant la logique floue et les réseaux de neurones possèdent tous deux des points forts et des points faibles différents sont montré dans le Tableau 3.1 :

Tableau 3.1 – Avantages et inconvénients de la logique floue et des réseaux de neurones.

	La logique floue	**Les réseaux de neurone**
Avantages	- Intégration de la connaissance à priori. - Facilité de construction. - Possibilité d'interpréter la connaissance.	- Capacités d'apprentissage. - Parallélisme massif. - Résistance aux données bruitées.
Inconvénients	- Construction manuelle des règles. - Difficultés d'optimisation des différents paramètres.	- Aucune interprétation possible des résultats de l'apprentissage. - Difficulté d'estimer les paramètres l'apprentissage.

3.5.3 Types de combinaisons réseaux neuro-flous

Les types de réseau neuro-flous sont tirés des différentes combinaisons entre les réseaux de neurones et la logique floue. On peut les diviser en trois catégories Figure(3.20) selon [116] : coopératifs, concurrents et hybrides.

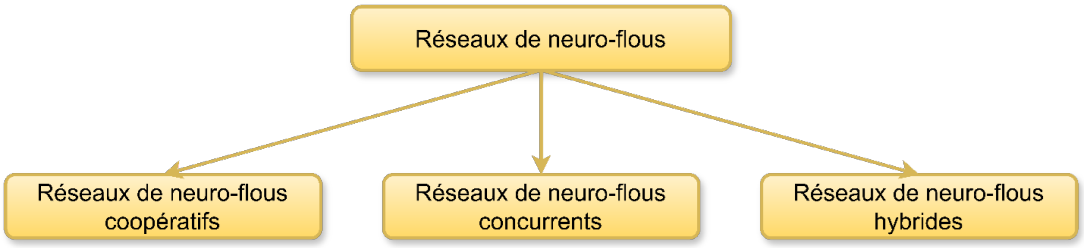


FIGURE 3.20 – les types de réseaux neuro-flous.

Les systèmes neuro-flous coopératifs

Le modèle coopératif Figure(3.21) est considéré comme un préprocesseur dans lequel le dispositif d'apprentissage du réseau de neurones détermine les règles floues ou le système d'inférence flou, depuis la base de données. Après cette étape le réseau de neurones reste en arrière-plan et le système flou est exécuté. Il existe plusieurs exemples de systèmes neuro-flous coopératifs : l'extraction des règles floues en utilisant les cartes auto-organisées par Pedrycz et al [119], les mémoires floues associatives « Fuzzy Associative Memories » (FAM) par Kosko [120], et les systèmes permettant l'apprentissage des paramètres des ensembles flous par Nomura et al [121].

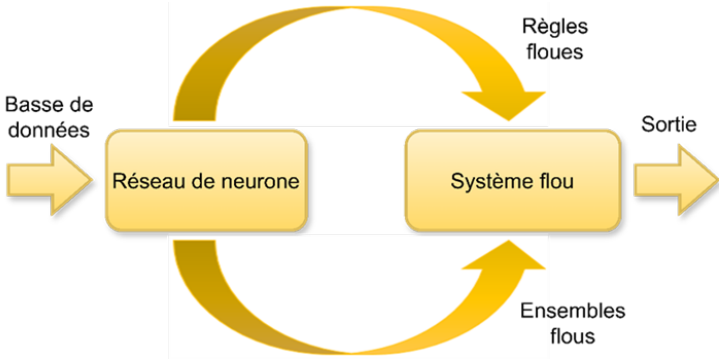


FIGURE 3.21 – Le modèle neuro-flou coopératif.

Les systèmes neuro-flous concurrents

Dans le modèle neuro-flou concurrent, le réseau de neurone accompagne le système flou en continu et inversement pour la détermination des paramètre. Cette méthode n'optimise pas le système flou mais aide à améliorer la performance de l'ensemble du système. L'apprentissage se déroule uniquement dans le réseau de neurone et le système flou reste

inchangé pendant cette phase. La Figure (3.22) représente un modèle neuro-flou concurrent où la base de données représente l'entrée du réseau de neurone, et sa sortie est traitée par le système flou.

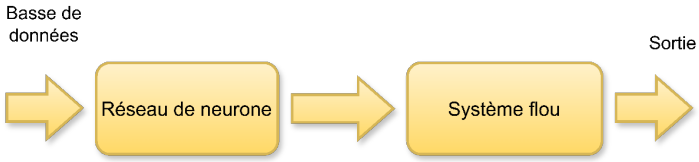


FIGURE 3.22 – Le modèle neuro flou concurrent.

Les systèmes neuro-flous hybrides

Nauck [116] a défini un système neuro-flou hybride comme étant un système flou qui utilise un algorithme d'apprentissage basé sur le gradient pour déterminer ses paramètres qui sont les fonctions d'appartenances et les règles floues, à partir d'une base de données constituées d'entrées et de sorties. Les approches neuro-floues modernes sont de type hybrides, le réseau de neurone et le système flou sont jumelés dans une architecture homogène. On parle alors d'un réseau de neurone avec des paramètres flous ou d'un système flou mis en application sous une forme parallèle. Dans le cadre de notre travail nous étudions les réseaux neuro-flous hybrides, dans ce qui suit nous présentons différents systèmes neuro-flous hybrides de type Mamdani et de Takagi-Sugeno (TS) [65].

- **Le système neuro-flou de type Mamdani**

Le système neuro-flou de type Mamdani utilise l'apprentissage supervisé (généralement l'apprentissage par rétro-propagation du gradient) pour adapter les paramètres des fonctions appartenances. Son architecture est illustrée à travers la (Figure 3.23). La fonction de chaque couche du réseau est décrite comme suit :

- **Couche1 (Couche d'entrée) :** Aucun calcul n'est fait dans cette couche. Chaque noeud, correspond à une variable d'entrée, elle transmet directement les valeurs d'entrées à la couche suivante.

- **Couche 2 (Couche de fuzzification) :** Les noeuds de cette couche correspondent à des termes linguistiques (excellent, bon, mauvais, etc.) des variables d'entrées de la première couche. La sortie représente le degré d'appartenance de la valeur d'entrée à un ensemble flou. La forme définitive des fonctions d'appartenance est affinée au cours de l'apprentissage.

- **Couche 3 (Couche des antécédents des règles) :** Chaque nœud de cette couche représente la partie antécédente d'une règl. Dans ce nœud on utilise généralement un opérateur T-norme. La sortie d'un nœud de la couche trois, représente le degré de vérité de la règle floue correspondante.

- **Couche 4 (Couche des conclusions des règles) :** Le nombre de nœuds dans cette couche est égal au nombre de règles. Ce nœud combine les antécédents des règles entrantes et détermine le degré auxquels ils appartiennent aux valeurs linguistiques de sortie.
- **Couches 5 (Couche de défuzzification) :** Cette couche calcule la sortie nette. Dans cette couche on utilise l'opérateur T-conorme, c'est la combinaison de toutes les conclusions des règles.

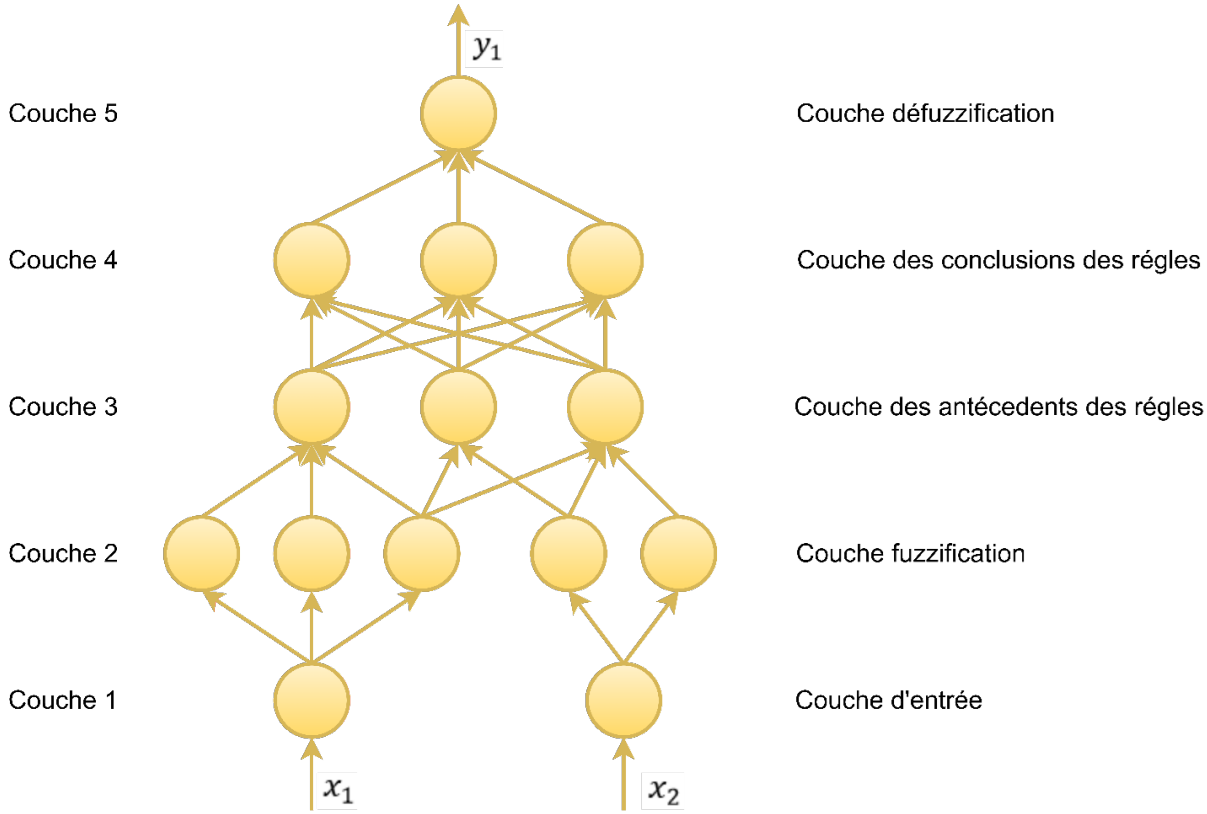


FIGURE 3.23 – L'architecture du réseau neuro-flou de type Mamdani.

- **Le système neuro-flou de type Takagi-Sugeno**

Le système neuro-flou de type TS représenté dans la (Figure 3.24) utilise une combinaison de deux algorithmes, l'algorithme de rétropropagation pour l'apprentissage des fonctions d'appartenance et l'algorithme d'estimation par moindres carrés pour la détermination de la combinaison linéaire des conclusions des règles. Par conséquent l'apprentissage se fait en deux étapes, Pour la première étape, les entrées sont propagées et les paramètres des conclusions sont déterminés par la méthode des moindres carrés, pendant que les fonctions d'appartenance d'entrées restent fixes. Pour la deuxième étape on utilise la rétropropagation du gradient pour ajuster les antécédents des règles, tandis que les paramètres des

conclusions demeurent fixes. Cette procédure est ensuite répétée plusieurs fois jusqu'à atteindre un but fixé par l'expert, comme par exemple un seuil auquel l'erreur quadratique doit aboutir. Les couches 1, 2 et 3 fonctionnent de la même manière que le système de type Mamdani [65].

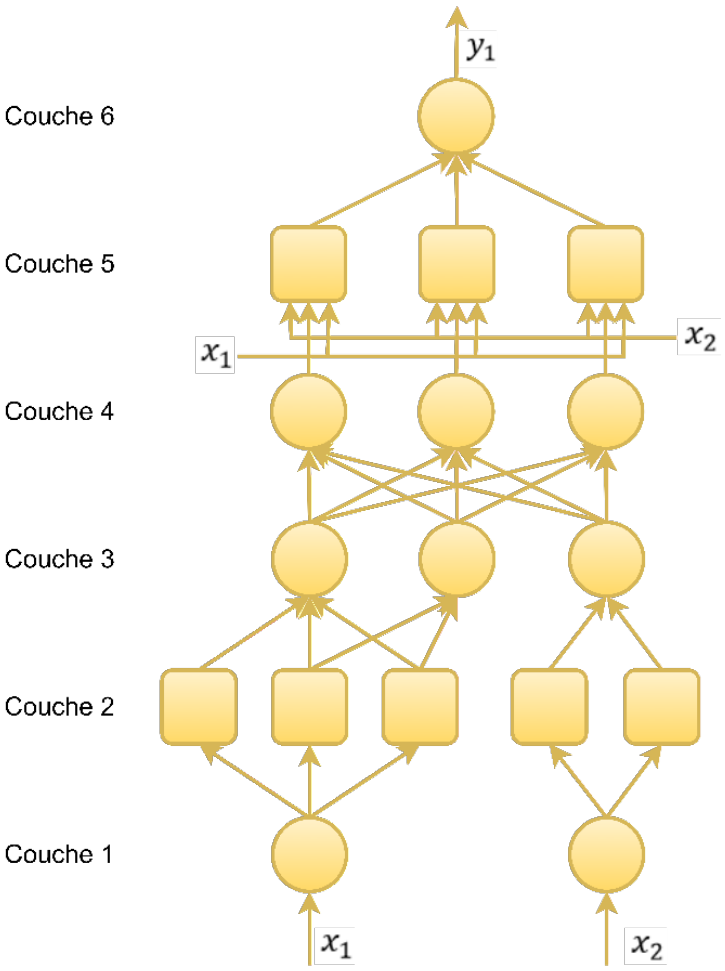


FIGURE 3.24 – L'architecture du réseau neuro-flou de type Takagi-Sugeno.

Quelques réseaux neuro-flous hybrides

Il existe plusieurs réseaux neuro-flous hybrides qui font usage de la complémentarité entre les réseaux de neurones et les systèmes flous en utilisant le système d'inférence de type Mamdani ou TS comme le ANFIS [122], le FALCON [123], le NEFCON [124], le NEFCLASS [125], le NEFPROX [126] et l'EFuNN [128].

3.5.4 Types d'implémentation des réseaux neuro-flous

Système d'inférence neuro-floue

Un système d'inférence neuro-floue est réalisé sous forme d'un réseau de neurone, où les paramètres de la logique floue sont représentés par les poids synaptiques du réseau

de neurone. Par conséquent, la structure de réseau de neurone permet d'ajuster automatiquement et optimiser les fonctions d'appartenance, et cela en modifiant les poids des connexions du RN via un algorithme d'apprentissage approprié.

Système d'inférence neuro-floue adaptatif (ANFIS)

Les systèmes d'inférence neuro-floue adaptatifs appartiennent à la famille des systèmes neuro-flous hybrides, qui combinent les avantages de la logique floue avec ceux des réseaux de neurones dans un seul réseau. La structure ANFIS a été proposée par Jang en 1993. Elle décrit le fonctionnement d'un système flou, avec des règles qui peuvent être de type Takagi-Sugeno ou Mamdani, sous forme d'un réseau de neurone, entraîné à l'aide d'un algorithme d'apprentissage de rétro-propagation, qui ajuste et optimise les paramètres de la prémisse et de la conséquence du réseau. Le système ANFIS comporte cinq couches, où les noeuds adaptatifs contenant des paramètres sont situés dans la première et la quatrième couche, alors que les autres noeuds sans paramètres sont fixes.
Les modèles ANFIS sont utilisés dans plusieurs applications de traitement de signal, de filtrage adaptatif et de contrôle des systèmes, et présentent de bonnes performances notamment dans les applications de contrôle.

3.5.5 Méthodes d'apprentissage des réseaux neuro-flous

Les algorithmes de calcul des réseaux de neurone artificiels et des réseaux neuro-flous, utilisés plus particulièrement en robotique mobile fournissent des systèmes intelligents capables de s'adapter aux environnements dynamiques inconnus, grâce à des mécanismes d'apprentissage en ligne ou hors ligne [108]. La méthode d'apprentissage en ligne consiste à ajuster les paramètres du réseau neuro-flou en temps réel au cours du processus, simultanément avec le fonctionnement du système à contrôler, en utilisant des algorithmes d'optimisations locale et globale. Dans ce cas, les données sont présentes de manière séquentielle par rapport au temps, et l'algorithme estime et prédit les paramètres du réseau en fonction des actions passées et finies. Par contre, dans le cas d'un apprentissage hors ligne, les paramètres du réseau sont mis à jour une seule fois, dans ce cas, ces paramètres sont alors fixes peu importe l'évolution du système à contrôler dans le temps.
Dans notre travail, nous allons utiliser deux contrôleurs de type ANFIS pour la navigation d'un robot mobile à entrainement différentiel (Pioneer P-3dx) dans le but de chercher les cibles tous en évitant des obstacles ; Néanmoins cette stratégie de commande ANFIS ne respecte pas les contraintes physique (qui seront décrites dans les sections ci-dessous) du robot pendant les simulations, pour cette raison nous avons développé un algorithme d'optimisation par essaim de particule PSO pour optimiser les commandes du contrôleur ANFIS.

3.6 Optimisation méta-heuristique

Les premières métaheuristiques datent des années 1980, et bien qu'elles soient d'origine discrète, on peut les adapter à des problèmes continus. Elles sont utilisées généralement quand les méthodes classiques ont échoué, et sont d'une efficacité non garantie. Le terme métaheuristique est utilisé par opposition aux heuristiques particulières pour un problème donné. Les métaheuristiques peuvent être utilisées pour plusieurs types de problèmes, tandis qu'une heuristique est adaptée à un problème donné. Les métaheuristiques ont également comme caractéristiques communes leur caractère stochastique, ainsi que leur inspiration, une analogie avec d'autres domaines de recherche (la biologie, la physique, etc.). Les métaheuristiques ne sont pas des méthodes figées ; il n'y a pas de relation d'ordre quant à l'efficacité d'un algorithme ou d'un autre, cela dépend plutôt des paramètres utilisés, de l'application elle-même ou du problème [129].

3.6.1 Classification

On peut distinguer les métaheuristiques qui font évoluer une seule solution sur l'espace de recherche à chaque itération et les métaheuristiques à base de population de solutions (Figure 3.25). En général, les métaheuristiques à base de solution unique sont plutôt axées sur l'exploitation de l'espace de recherche, on n'est donc jamais sûr d'obtenir l'optimum. Les métaheuristiques à base de population sont plutôt exploratoires et permettent une meilleure diversification et exploration de l'espace de recherche.

3.6.2 Les métaheuristiques à solution unique

Dans cette section, nous présentons les métaheuristiques à base de solution unique, appelées aussi méthodes de trajectoire. Contrairement aux métaheuristiques à base de population, les métaheuristiques à solution unique commencent avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche. Les méthodes de trajectoire englobent essentiellement la méthode de descente, la méthode du recuit simulé, la recherche taboue, la méthode GRASP, la recherche à voisinage variable, la recherche locale itérée, et leurs variantes.

3.6.3 Les métaheuristiques à population de solutions

Contrairement aux algorithmes partant d'une solution singulière, les métaheuristiques à population de solutions améliorent cette population, au fur et à mesure des itérations. On distingue dans cette catégorie, les algorithmes évolutionnaires, qui sont une famille d'algorithmes issus de la théorie de l'évolution par la sélection naturelle, énoncée par Charles Darwin [111] et les algorithmes d'intelligence en essaim qui, de la même ma-

nière que les algorithmes évolutionnaires, proviennent d'analogies avec des phénomènes biologiques naturels.

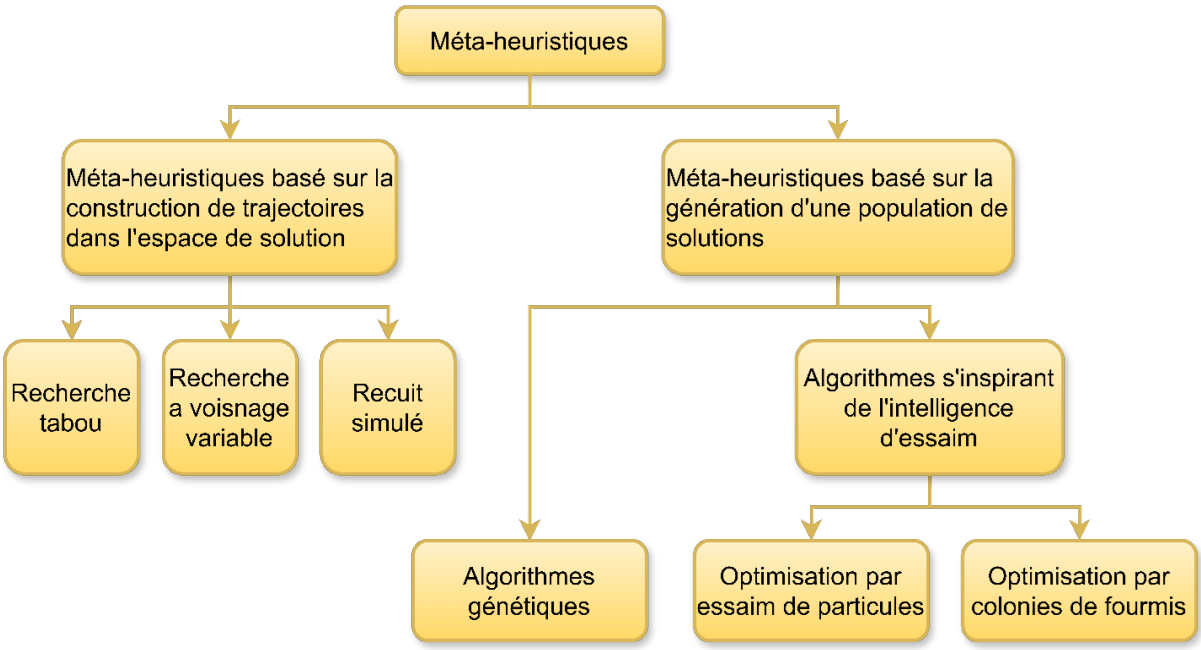


FIGURE 3.25 – Classification des méta-heuristiques.

3.7 Généralité sur l'optimisation méta-heuristique par essaim de particule (PSO)

L'optimisation par essaims de particule (PSO) est une méta-heuristique, inventée par Russel Eberhart (Ingénieur en électricité) et James Kennedy (socio psychologue) en 1995 [130, 81].

Cette méthode s'inspire à l'origine du monde du vivant ; elle s'appuie notamment sur un modèle développé par le biologiste Craig Reynolds à la fin des années quatre-vingt, permettant de simuler le déplacement d'un essaim (inspirer du comportement social des animaux évoluant en essaim). Une autre source d'inspiration, revendiquée par les auteurs est la socio-psychologie.

Cette méthode d'optimisation se base sur la collaboration des individus entre eux. Elle est d'ailleurs une méthode récente parmi les algorithmes évolutionnaires, qui s'appuient sur le concept d'auto-organisation. Cette idée veut qu'un groupe d'individus peu intelligent puisse posséder une organisation globale complexe.

L'échange d'information entre eux fait que, globalement, ils arrivent néanmoins à résoudre des problèmes difficiles, comme c'est le cas, par exemple, chez les abeilles vivant en essaim (exploitation de sources de nourriture, construction de rayon, etc.).

Ainsi grâce à des règles de déplacement très simple (dans l'espace de solution), les par-

ticules peuvent converger progressivement à un minimum global. Cette méta-heuristique semble cependant mieux fonctionner pour des espaces en variables continues.
L'optimisation par essaims de particule (PSO), dans sa version historique, est donc une méthode itérative collective anarchique au sens original du terme, mettant l'accent sur la coopération, partiellement aléatoire et sans sélection. Dans cette partie on détaillera ses caractéristiques et les formaliser afin d'obtenir un modèle exploitable.

3.7.1 Origine

Vers 1927, Karl Von Frish a découvert que les abeilles ramenaient à la ruche non seulement du nectar et du pollen, mais aussi de l'information. Il a patiemment décodé leur langage et l'observateur attentif peut maintenant les comprendre partiellement [131].
L'optimisation par essaim de particule est une méthode née aux Etats Unis sous le nom de Particle Swarm Optimization (PSO).
Initialement les deux concepteurs, Russel Eberhart et James Kennedy cherchaient à modéliser des interactions sociables entre des « agents » devant atteindre un objectif donné dans un espace de recherche commun, chaque agent ayant une certaine capacité de mémorisation et de traitement de l'information.
La règle de base et qu'il ne devait y avoir aucun chef d'orchestre, ni même aucune connaissance par les agents de l'ensemble des informations seulement des connaissances locales.
Dès les premières simulations, le comportement collectif de ces agents évoquait celui d'un essaim d'êtres vivants convergeant parfois en plusieurs sous-essaims vers des sites intéressants. Ce comportement se trouve dans bien d'autres modèles, explicitement inspirés des systèmes naturels. Ici la métaphore la plus pertinente est probablement celle de l'essaim d'abeilles, particulièrement, du fait qu'une abeille ayant trouvé un site prometteur sait en informer certaines de ces consœurs et que celles-ci vont tenir compte de cette information pour leur prochain déplacement, c'est-à-dire une abeille doit parfois combiner plusieurs informations, celle correspondant à sa propre connaissance du terrain et celle apportée par une butineuse, voire plusieurs presque simultanément. La manière dont elle le fait reste un mystère mais pour nous inspirer du comportement de nos abeilles, il nous faudra quand même la modéliser, donc, inventer une méthode de toute pièce. Le modèle s'est révélé être trop simple pour vraiment simuler un comportement social, mais par contre très efficace en tant qu'outil d'optimisation [131].

FIGURE 3.26 – Etude du comportement des abeilles dans une ruche [131].

Le fonctionnement de PSO fait qu'elle peut être classée dans les méthodes itératives (on approche peu à peu de la solution) et stochastiques (on fait appel au hasard). Sous ce terme un peu technique, on retrouve un comportement qui est aussi vieux que la vie elle-même : améliorer sa situation on se déplaçant partiellement au hasard et partiellement selon des règles prédéfinies.

3.7.2 Description et Principe général du PSO

La méthode méta-heuristique que nous avons étudiée dans le cadre de notre thèse est l'optimisation par essaims de particule. On dispose d'une fonction objective à optimiser dans un sens ou dans l'autre [132].
Un essaim est un ensemble de particules positionnées dans l'espace de définition de la fonction objective. Le principe de l'algorithme consiste à déplacer ces particules dans l'espace de définition afin de trouver la solution optimale. Une particule est caractérisée par plusieurs attributs :

- **Sa position actuelle :** c'est-à-dire ses coordonnées dans l'ensemble de définition et la valeur de la fonction objective lui correspond
- **Sa meilleure position :** c'est la valeur obtenue par la particule et ses coordonnées.
- **Sa vitesse :** cette donnée, recalculée à chaque itération de l'algorithme permet de déduire la position suivante de la particule. Elle est fonction de la meilleure position de la particule depuis le début de la recherche, du voisin le mieux positionné à l'instant actuel et de la vitesse précédente de la particule.
- **Ses voisins :** c'est un ensemble de particules qui influe sur ses déplacements, en particulier celui qui est le mieux positionné.

3.7.3 Formulation

En PSO, un « site intéressent » correspond à un optimum au moins local d'une certaine fonction définie dans un espace de recherche. Cette fonction peut être donnée par une formule mathématique ou, à défaut, par un algorithme, voir par le résultat du déroulement d'un processus, réel ou simulé. Le tout est que l'on sache calculer sa valeur en chaque point.
Pour sa version classique le PSO cherche les sites les plus intéressants c'est-à-dire l'optimum global de notre fonction fitness. Pour ce faire, le PSO s'inspire du comportement coopératif décrit dans notre métaphore : chaque particule est capable de communiquer à certaines autres particules la position et la qualité du meilleur site qu'elle connait, qualité que l'on peut interpréter comme étant sa valeur.

Taille de l'essaim

Tout d'abord, il faut définir un essaim dans l'espace de recherche, mais de quelle taille ? Les véritables essaims d'abeilles comptent typiquement 20000 individus, nous nous contenterons de taille de l'ordre de 20 à 40. Il s'avère en effet qu'en PSO ces tailles sont très souvent suffisantes. Ce qui compte plutôt c'est le nombre de fois où la fonction fitness doit être évaluée, car dans la plupart des problèmes réels, cette évaluation nécessite un temps non négligeable, et évidemment, pour une itération, ce nombre d'évaluations est égal au nombre de particules. Donc si nous voulons réduire le nombre total d'évaluations nécessaires pour trouver une solution, nous sommes au contraire tentés de diminuer la taille de l'essaim. Mais un essaim trop petit risque de mettre très longtemps pour trouver une solution ou même ne pas la trouver du tout. Donc il y a un compromis à trouver. Les expérimentateurs ont proposé des tailles de l'ordre de 20 à 30 particules qui, en effet, se révèlent tout à fait suffisante pour résoudre la quasi-totalité des problèmes de test classiques.

Liens d'information

Il faut définir, pour chaque particule, quelles sont ses informatrices. Toujours par analogie avec ce qui se passe dans une ruche, nous pouvons définir au hasard pour chaque particule son groupe d'informées, ce qui, automatiquement, détermine également les informatrices de chaque particule, puisque, formellement, nous établissons un graphe de relations entre les particules.
Combien d'informées, combien d'informatrices ? D'une part, si toutes les particules sont informées par chacune, toute l'information acquise à chaque instant est diffusée immédiatement, ce qui semble favorable. Mais d'autre part le risque est grand d'avoir un comportement trop uniforme : avec les mêmes informations, les particules vont agir de la même manière. Pour les recherches difficiles ce n'est pas efficace. Inversement, si chaque particule

n'a que trop peu d'informatrices, nous pourrons obtenir des comportements plus diversifiés, mais le risque est alors que l'information soit mal transmise. Or il est important que si une particule trouve un bon emplacement, toutes les autres, plus au moins directement, puissent avoir connaissance de cette information, pour en tirer parti.
Dans ces conditions, si le choix est fait au hasard à chaque pas de temps, prendre deux ou trois informées pour chaque particule semble un bon compromis. Il y a aussi d'autres versions de l'PSO qui ne font pas ce choix au hasard mais selon une règle tenant compte de certains critères, par exemple une sélection automatique permanente et judicieuse des informatrices.
La nature des informations transmises est évidemment importante, mais plus il y aura d'informations, plus il sera long et difficile à une particule de les traiter. Le plus délicat à modéliser est la manière dont une particule informée va calculer son prochain déplacement. Tout d'abord, notons qu'elle est en général déjà en train de se déplacer : elle possède donc une certaine vitesse. Ensuite, puisqu' elle est informatrice éventuelle d'autres particules, elle connait sa propre meilleure performance. Enfin, elle connait toutes les meilleures performances de ces informatrices et elle va garder la meilleure.
Il nous reste donc trois éléments à combiner : vitesse propre, meilleure performance propre et meilleure des meilleures performances des informatrices.
Imaginons trois cas extrêmes : premier cas, la particule est aventureuse et n'entend suivre que sa propre voie, alors elle va attribuer une confiance nulle aux informations reçues et même à sa propre expérience ; elle se contentera de suivre plus au moins la direction déjà suivie, c'est-à-dire que le prochain déplacement se fera en gros avec la même vitesse (intensité et direction) que le précédent. Deuxième cas, elle est très conservatrice ; elle va accorder une grande confiance à sa meilleure performance et tendra à y revenir sans cesse. Troisième cas elle ne s'accorde à elle-même aucune confiance ; elle se déplacera selon les indications de sa meilleure informatrice. La figure suivante (Figure 3.27) représente les trois éléments fondamentaux pour le calcul du prochain déplacement d'une particule [133].

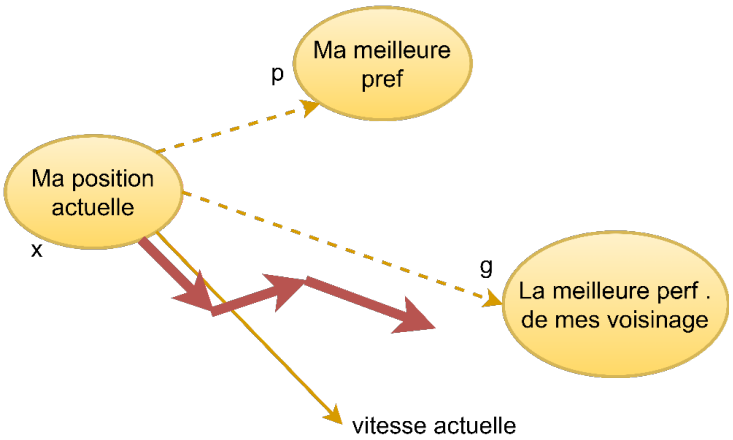


FIGURE 3.27 – Détermination de la nouvelle position d'une particule dans un processus PSO (les trois flèches grisées représentent les trois effets prise en compte) [131].

A partir des quelques informations dont elle dispose, une particule doit décider de son prochain mouvement, c'est-à-dire décider de sa nouvelle vitesse. Pour ce faire, elle combine linéairement trois informations :

- Sa vitesse actuelle ;
- Sa meilleure performance ;
- La meilleure performance de ses voisines (ses informatrices).

À l'aide de trois paramètres parfois appelés coefficients de confiance, qui pondèrent trois tendances :

- Tendance à suivre sa propre voie ;
- Tendance conservatrice (revenir sur ses pas) ;
- Tendance à suivre le meilleur voisin.

Nous avons donc trois éléments fondamentaux, schématisés par la figure précédente : selon sa vitesse actuelle, vers sa propre meilleure performance et vers celle de ses meilleures informatrices. Pour calculer le déplacement vrai à partir de ces trois vecteurs de base, le plus simple est d'en faire une pondération linéaire, grâce à des coefficients de confiance. Tout l'art des premières versions du PSO consiste en la définition judicieuse de ces coefficients.

Initialisation

L'initialisation consiste simplement à placer d'abord au hasard les particules selon une distribution uniforme dans l'espace de recherche. Ceci est une étape que l'on retrouve dans à peu près tous les algorithmes d'optimisation itérative stochastique. En pratique, il n'est pas souhaitable que trop de particules tendent à sortir de l'espace de recherche

dès le premier pas de temps, ni d'ailleurs plus tard. Pour les premières formulations nous nous contentons de tirer au hasard les valeurs des composantes de chaque vitesse, selon une distribution uniforme dans [134] :

$$\left[\frac{(x_{min} - x_{max})}{2}, \frac{(x_{max} - x_{min})}{2}\right] \tag{3.25}$$

Equations du mouvement

Une caractéristique très importante de le PSO du moins dans ses versions classiques est qu'elle ne pratique aucune sélection. L'idée est en effet que les faibles d'aujourd'hui sont peut-être les forts de demain. Les particules à piètre performance sont conservées, avec l'espoir que parmi elles se trouvent précisément les originales, les dissidentes qui permettront de découvrir le meilleur site existant dans l'espace de recherche. Les expérimentations ont d'ailleurs parfaitement justifié cet espoir.
L'espace de recherche est de dimension D. La position courante d'une particule dans cet espace à l'instant t est donc donnée par un vecteur $x(t)$ à D composantes. Sa vitesse courante est $v(t)$. La meilleure position trouvée jusqu'ici par cette particule est donnée par un vecteur $pbest(t)$. Enfin, la meilleure position trouvée par les informatrices de la particule est indiquée par un vecteur $gbest(t)$. On indique par l'indice i, la $i_{éme}$ particule de chacun des vecteurs $x_i(t)$, $v_i(t)$, $pbest_i(t)$ et $gbest_i(t)$. Avec ces notations, les équations de mouvement d'une particule sont :

$$\begin{cases} v_i(t\ +\ 1)\ =\ \omega\ v_i(t)\ +\ c_1[pbest_i(t)\ -\ x_i(t)] + c_2[gbest(t)\ -\ \ x_i(t)] \\ x_i\ \ (t+1) =\ x_i\,(t) + v_i(t+1) \end{cases} \tag{3.26}$$

Les coefficients de confiances sont définis de la manière suivante : ω est constante (confiance en son propre mouvement) ;
c_1 et c_2 (respectivement confiance en sa meilleure performance et en celle de sa meilleure informatrice) sont choisis au hasard à chaque pas de temps selon une distribution uniforme dans un intervalle $[0, c_{max}]$ donné. C'est pourquoi le système (3.26) peut être réécrit de manière plus explicite, en mettant en évidence les systèmes aléatoires les éléments aléatoires.

$$\begin{cases} v_i(t\ +\ 1)\ =\ \omega\ v_i(t)\ +\ c_{max}r_1[pbest_i(t)\ -\ x_i(t)] + c_{max}r_2[gbest(t)\ -\ \ x_i(t)] \\ x_i\ \ (t+1) =\ x_i\,(t) + v_i(t+1) \end{cases} \tag{3.27}$$

Tel que : $r_1 = alea\,[0,1]$ et $r_2 = alea\,[0,1]$ des nombres aléatoires entre 0 et 1.
Pour utiliser ce modèle, on doit définir les deux paramètres ω et c_{max}, dont ce dernier peut être vu comme la confiance maximale accordée par la particule à toute performance trans-

mise par une autre particule. Pour chaque problème les « bonnes » valeurs ne peuvent pas être trouvées qu'expérimentalement, grâce à deux règles, dégagées après de nombreux tests.
La première règle stipule que ω doit être de valeur absolue inférieure à 1. Elle se comprend intuitivement si l'on considère ce qui se passe lors de plusieurs pas de temps successifs, dans le cas très particulier où la particule est la meilleure informatrice d'elle-même. Nous avons alors en effet et à chaque pas de temps la vitesse est simplement multipliée par ω. S'il est de valeur absolue supérieure à 1, la vitesse augmente sans cesse et la convergence est impossible. Notons qu'en théorie rien n'interdit à ce coefficient d'être négatif ; le comportement obtenu étant alors fortement oscillatoire, mais ce n'est jamais le cas en PSO classique. Nous le supposerons donc positif. En pratique ce coefficient ne doit pas être ni trop petit, ce qui induit une convergence prématurée, ni trop grand, ce qui au contraire peut ralentir exagérément la convergence. Les auteurs des premiers travaux sur le PSO préconisaient de la prendre égale à $0,7$ ou $0,8$.
La deuxième règle indique simplement que le paramètre c_{max} ne doit pas être trop grand, une valeur de l'ordre de $1,5$ à $1,7$ étant considérée comme efficace dans la plupart des cas. A l'époque où elle a été énoncée, cette règle n'avait pas de justification, même intuitive. Elle était purement expérimentale.
En effet, les valeurs préconisées sont très proches de celles déduites plus tard à travers d'analyses mathématiques qui ont confirmé que pour une bonne convergence les valeurs de ω et c_{max} ne doivent pas être choisies indépendamment.
Par conséquent les couple de valeurs $(0,71,47)$ et $(0,81,62)$ sont effectivement corrects. Les premiers expérimentateurs sont les suivants : James Kennedy et Russel Eberhart, auxquels on peut ajouter Yuhui Shi, qui ont fait un bon travail [131].

Confinement d'intervalle

Lors des premières expérimentations du PSO les fonctions utilisées étaient définies pour toutes valeurs. Lors de l'évolution de l'essaim il pouvait arriver qu'une particule sorte de l'espace de recherche initialement défini, mais cela n'a pas d'importance puisque la valeur de sa position pouvait en fait encore être calculée, sans planter l'exécution.
En revanche ce cas se reproduit rarement, car dans la plupart des langages de programmation et avec la plupart des compilateurs, la fonction de l'évaluation est donnée comme suit :

$$f(x) = \sum_{i=1}^{D} \sqrt{x_i} \tag{3.28}$$

En effet, cette fonction nous renvoie un message d'erreur dès que l'une des coordonnées x_i est négative. Par conséquent, il a fallu très vite ajouter un mécanisme pour éviter qu'une particule sorte de l'espace de recherche. Le plus simple de tous est le confinement d'intervalle.

Supposons toujours, par simplicité, que l'espace de recherche soit $[x_{min},\ x_{max}]$ alors ce mécanisme stipule que si une coordonnée x_i calculée selon les équations de mouvement sort de l'intervalle $[x_{min},\ x_{max}]$ on lui attribue en fait la valeur du point frontière le plus proche[135]. En pratique cela revient à remplacer la deuxième ligne du système (3.27) par :

$$x_i(t+1) = min\left(max\left(x_i(t) + v_i(t+1), x_{min}\right), x_{max}\right) \tag{3.29}$$

Cette forme simple, quoique donnant des résultats corrects, néanmoins, elle a un inconvénient.
En effet, nous sommes dans un cas de figure où la vitesse propre de la particule tend à la faire sortir de l'espace de recherche. Le confinement ramène la particule à la frontière de l'espace de recherche sans changer sa vitesse. Celle-ci est donc en générale modifiée au prochain pas de temps, mais il n'est pas rare qu'elle reste plus ou moins orientée de la même manière, alors la particule tendra à nouveau à dépasser la frontière, puis elle sera ramenée par le confinement, etc. En pratique, tout se passera comme si elle était « collée » à cette frontière.

Structure de l'Algorithme

L'PSO est un algorithme à population. Il commence par une initialisation aléatoire de l'essaim dans l'espace de recherche. A chaque itération de l'algorithme, lui correspond un déplacement des particules conformément aux équations de mouvement données précédemment (3.27). Une fois le déplacement des particules effectué, les nouvelles positions sont évaluées. Les p_i ainsi que q_i sont alors mis à jour. Cette procédure est résumée par l'algorithme suivant. N est le nombre de particules de l'essaim.
Le critère d'arrêt peut être différent suivant le problème posé. Si l'optimisation globale est connue à priori, on peut définir une « erreur acceptable » comme critère d'arrêt. Sinon, il est commun de fixer un nombre maximum d'évaluations de la fonction objective ou un nombre maximum d'itérations comme critère d'arrêt. Cependant, au regard du problème posé et des exigences de l'utilisateur, d'autres critères d'arrêt peuvent être utilisés.

- **Organigramme d'optimisation par essaim de particule :**

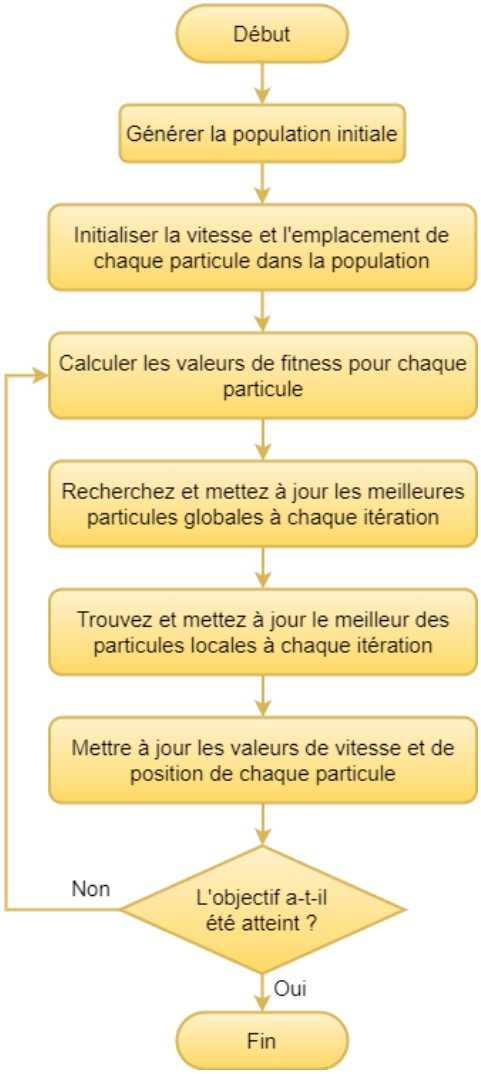


FIGURE 3.28 – Détermination de la nouvelle position d'une particule dans un processus PSO (Organigramme de l'algorithme PSO).

3.8 Commande hybride ANFIS-PSO appliquée au robot mobile

Dans le cadre de notre thèse un algorithme hybride ANFIS-PSO est développé afin d'assurer une navigation virtuelle d'un robot mobile à entrainement différentiel de type Pioneer P-3dx non-holonome dans des environnements encombré et inconnu ; ANFIS est utilisé pour que le robot cherche la cible tout en évitant les obstacles, et cela par deux contrôleurs de type ANFIS tel que le première assure la recherche de la cible est nommé par « ANFIS de suivi » et le deuxième assure l'évitement des obstacles est nommé « ANFIS d'évitement d'obstacles » ; PSO est utilisé pour l'optimisation des vitesses des roues avant de les envoyés au robot mobile.

3.8.1 Commande neuro-floue ANFIS du robot mobile afin de chercher la cible

Afin de minimiser les erreurs de poursuite, on a utilisé une commande cinématique à base de deux contrôleurs ANFIS, en vue de calculer le vecteur de commande composé de la vitesse linéaire v et de la vitesse angulaire ω. Dans notre travail, les entrées du premier contrôleur sont l'erreur de position selon l'axe x notée e_x et sa dérivée $\dot{e}_x$ nommée x_1 et x_2 dans l'architecture ci-dessous, pour calculer la commande v correspond à la sortie u_1 comme le montre l'architecture ci-dessous (Figure 3.30) et les entrées du deuxième contrôleur sont l'erreur de position selon l'axe y notée e_y et sa dérivée $\dot{e}_y$ pour calculer la commande ω correspond à la sortie u_2 comme le montre l'architecture ci-dessous (Figure 3.30). Le régulateur ANFIS de suivi est composé de deux régulateurs, comme le montre la (Figure 3.29). Il permet de calculer les commandes u_1 et u_2, qui sont la vitesse linéaire et la vitesse angulaire qui seront utilisées afin de rechercher les cibles. Chaque régulateur ANFIS reçoit deux entrées x_1, x_2 qui sont l'erreur de position et l'erreur de vitesse et donne une sortie u comme le montre la structure du réseau ANFIS de la (Figure 3.30). L'entrée x_1 est associée à trois ensembles flous A_1, A_2, A_3, et l'entrée x_2 est associée à trois ensembles flous B_1, B_2 ,B_3. La sortie u est modélisée par un système flou de type Sugeno, composé des neuf règles suivantes :

- **Règle 1 :** Si x_1 est A_1 et x_2 est B_1 alors :

$$u_1 = f_1(x_1,\ x_2) = a_1x_1 + b_1x_2 + c_1 \tag{3.30}$$

- **Règle 2 :** Si x_1 est A_1 et x_2 est B_2 alors :

$$u_2 = f_2(x_1,\ x_2) = a_2x_1 + b_2x_2 + c_2 \tag{3.31}$$

- **Règle 3 :** Si x_1 est A_1 et x_2 est B_3 alors :

$$u_3 = f_3(x_1,\ x_2) = a_3x_1 + b_3x_2 + c_3 \tag{3.32}$$

- **Règle 4 :** Si x_1 est A_2 et x_2 est B_1 alors :

$$u_4 = f_4(x_1,\ x_2) = a_4x_1 + b_4x_2 + c_4 \tag{3.33}$$

- **Règle 5 :** Si x_1 est A_2 et x_2 est B_2 alors :

$$u_5 = f_5(x_1,\ x_2) = a_5x_1 + b_5x_2 + c_5 \tag{3.34}$$

- **Règle 6 :** Si x_1 est A_2 et x_2 est B_3 alors :

$$u_6 = f_6(x_1,\ x_2) = a_6x_1 + b_6x_2 + c_6 \tag{3.35}$$

- **Règle 7 :** Si x_1 est A_3 et x_2 est B_1 alors :

$$u_7 = f_7(x_1,\ x_2) = a_7 x_1 + b_7 x_2 + c_7 \tag{3.36}$$

- **Règle 8 :** Si x_1 est A_3 et x_2 est B_2 alors :

$$u_8 = f_8(x_1,\ x_2) = a_8 x_1 + b_8 x_2 + c_8 \tag{3.37}$$

- **Règle 9 :** Si x_1 est A_3 et x_2 est B_3 alors :

$$u_9 = f_9(x_1,\ x_2) = a_9 x_1 + b_9 x_2 + c_9 \tag{3.38}$$

Tel que a_i, b_i,c_i,pour $i = 1 : 9$. Sont les paramètres de conséquence linéaires du système d'inférence floue.

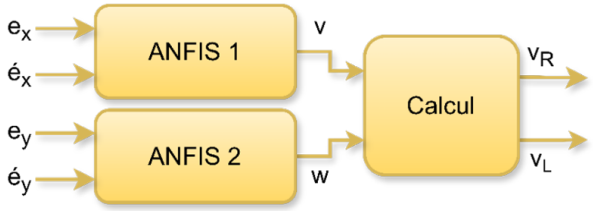


FIGURE 3.29 – ANFIS de suivi.

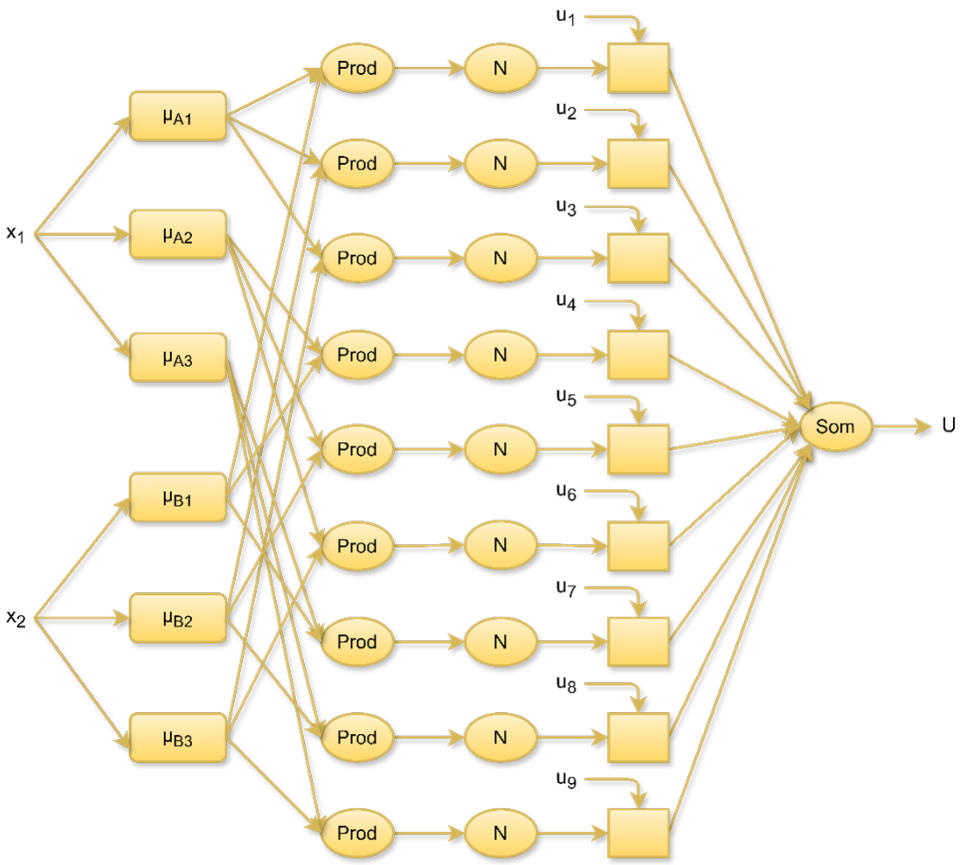


FIGURE 3.30 – Structure du ANFIS de suivi.

- **Couche 1 : Fuzzification**

Cette couche contient six nœuds qui transforment les entrées numériques mesurées par les capteurs en interprétations linguistiques. Chaque nœud i de cette couche est un nœud adaptatif doté d'une fonction d'appartenance à paramètres ajustables :

$$O_{1,i} = \mu_{A_i}(x_1) \ \ pour\ i = 1:3$$

$$O_{1,i} = \mu_{B_{i-3}}(x_2) \ \ pour\ i = 4:6$$

$O_{1,i}$ Calcule ses activations qui sont les degrés d'appartenance des variables x_1 et x_2 respectivement aux sous-ensembles flous A_i et B_i représentés par des fonctions gaussiennes décrites par :

$$\mu_{A_i}(x_1) = \exp\left(-\left(\frac{x_1 - c_i}{\sigma_i}\right)^2\right) \ pour\ i = 1:3 \tag{3.39}$$

$$\mu_{B_{i-3}}(x_2) = \exp\left(-\left(\frac{x_2 - c_i}{\sigma_i}\right)^2\right) \ pour\ i = 4:6 \tag{3.40}$$

Tel que c_i est le centre de la gaussienne et σ_i l'écart type.

- **Couche 2 : Dégrée d'activation**

Génère le degré d'activation approprié à la règle.

$$O_{2,i} = w_i = \mu_{A_j}(x_1).\mu_{B_k}(x_2) \ pour\ i = 1:9\ et\ j,k = 1:3 \tag{3.41}$$

- **Couche 3 : Normalisation**

Chaque nœud de cette couche est un nœud fixe appelé N. Sa sortie représente le degré d'activation normalisé de la $i_{éme}$ règle.

$$O_{3.i} = \bar{w_i} = \frac{w_i}{\sum_{k=1}^{9} w_k} \ for\ i = 1:9 \tag{3.42}$$

- **Couche 4 : Calcul des sorties des règles**

Chaque nœud de cette couche est un nœud adaptatif dont la fonction est :

$$O_{4,i} = \bar{w_i} f_i = \bar{w_i}(a_i x_1 + b_i x_2 + c_i) \tag{3.43}$$

Où $\bar{w_i}$ est la sortie de la couche 3 qui représente le degré d'activation normalisé de la règle et a_i,b_i,c_i sont les paramètres de sortie ajustables de la règle i. Ces paramètres sont appelés paramètres de la conséquence.

- **Couche 5 : Défuzzification**

Cette couche comprend un seul nœud fixe, qui calcule la sortie globale qui est la somme de tous les signaux provenant de la couche 4 :

$$O_{5,i} = u = \sum_{i=1}^{9} \bar{w}_i f_i = \frac{\sum_{i=1}^{9} w_i f_i}{\sum_{i=1}^{9} w_i} \tag{3.44}$$

Dans notre cas, nous considérons que les paramètres de la prémisse sont fixes, tandis que ceux de la conséquence sont ajustés à l'aide d'un algorithme d'apprentissage en ligne [65] en minimisant la fonction objectif $J = \frac{1}{2}e^2$, où e est l'erreur entre la position actuelle et la position souhaitée pour trouver les paramètres de conséquence. L'objectif est de trouver les paramètres a_i,b_i,c_i du vecteur des paramètres à ajuster en utilisant la méthode de la descente du gradient combinée avec l'approche du filtre de Kalman étendu [95].

3.8.2 Commande neuro-floue ANFIS du robot mobile pour éviter les obstacles

Le régulateur ANFIS d'évitement d'obstacles représenté sur la (Figure 3.31) permet de calculer les commandes u_1 et u_2 qui sont la vitesse linéaire et la vitesse angulaire respectivement, qui seront par la suite utilisées afin d'éviter les obstacles sans collision. Le réseau ANFIS d'évitement d'obstacles est représenté dans la structure de la (Figure 3.32). Il reçoit trois entrées x_1, x_2 et x_3 représentent les distances entre l'obstacle et le robot qui sont mesurées par des capteurs à ultrasons (Distance avant (FD), Distance gauche (LD) et Distance droite (RD)) et calcul deux sortiesu_1 et u_2.
L'entrée x_1 est associée à trois ensembles flous A_1,A_2,A_3 et l'entrée x_2 est associée à trois ensembles flous B_1,B_2,B_3, l'entrée x_3 est associée à trois ensembles flous C_1,C_2,C_3. Les sorties u_1 et u_2 sont modélisées par un système flou de type Sugeno, composé de vingt-sept règles, et les étapes suivantes s'effectuent de la même manière que le contrôleur précédent.

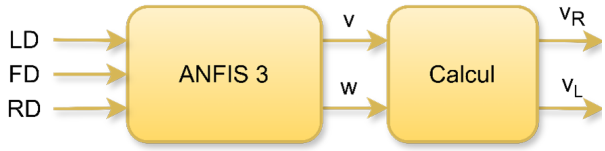


FIGURE 3.31 – ANFIS d'évitement d'obstacles.

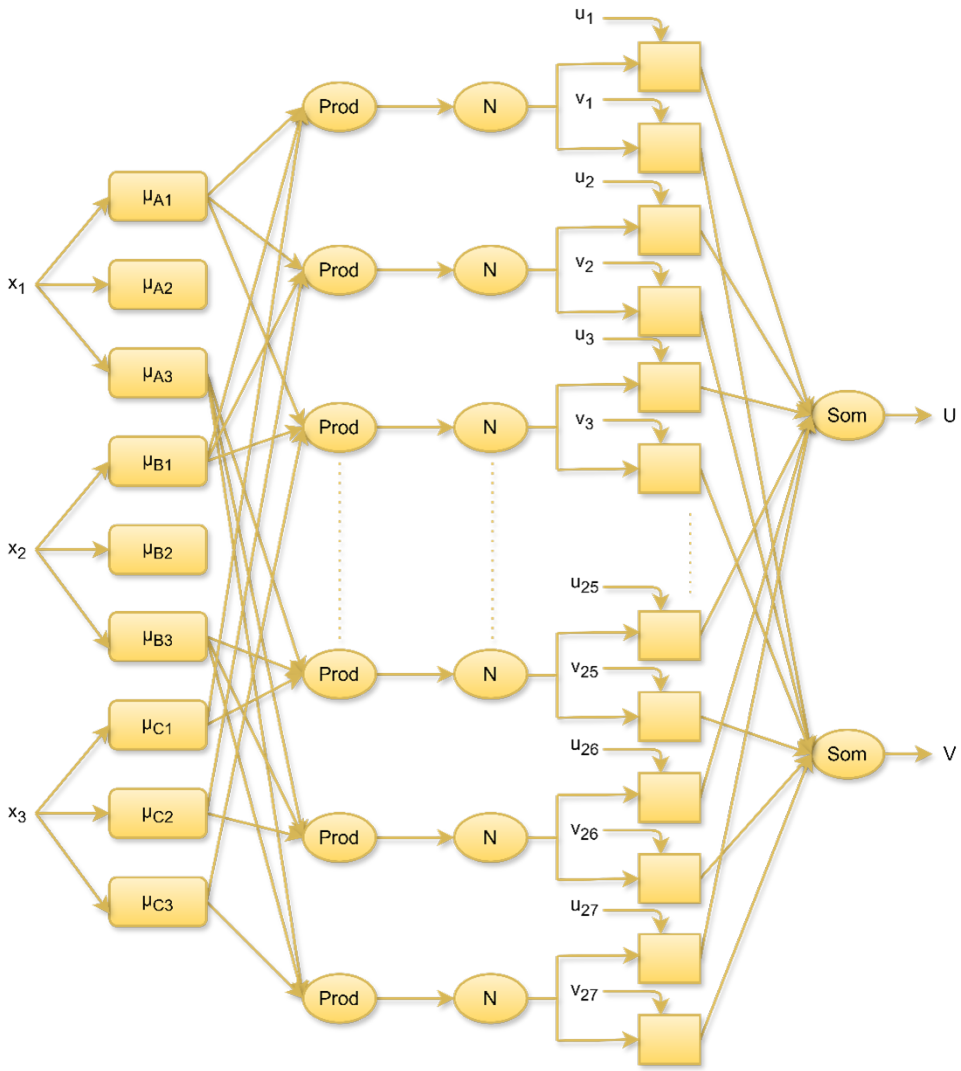


FIGURE 3.32 – Structure du ANFIS d'évitement d'obstacles.

3.8.3 Optimisation des commandes par l'algorithme PSO

Dans notre travail on a utilisé un algorithme d'optimisation par essaim de particule (PSO) qui est une méthode d'optimisation méta-heuristique dans le but de minimiser l'erreur quadratique moyenne (RMSE) entre la valeur actuelle et la valeur prédite des sorties du contrôleur ANFIS. La fonction objective (fitness) choisie dans notre travail est donnée comme suit :

$$f_{obj} = RMSE(\%) = \left[\sum_{1}^{k}\left(\frac{v_R - \hat{v}_R}{k}\right)^2 + \left(\frac{v_L - \hat{v}_L}{k}\right)^2\right] * 100 \tag{3.45}$$

tel que v_R et v_L sont les sorties du controleur ANFIS et $\hat{v}_R$ et $\hat{v}_L$ sont les populations. k varie entre $1:1:n$, n est le nombre de particiles dans la population dans notre travail égale à 100.

- **Choix des contraints d'optimisation :**

Après plusieurs simulations avec le contrôleur ANFIS dans différents environnements de navigation proposés dans notre travail, nous avons remarqué que les vitesses des roues droite notée v_R et gauche notée v_L envoyées au pionnier-P3DX dépassent les vitesses limites de ces roues. Pour résoudre ce problème et améliorer les mouvements de notre robot mobile nous avons introduit un algorithme d'optimisation par essaim de particule (PSO) en vue d'optimiser les sorties du contrôleur ANFIS de telle sorte que les vitesses des roues et la vitesse du robot soient limitées comme suit $v_{min} \leq v_R \leq v_{max}$ et $v_{min} \leq v_L \leq v_{max}$ et $v_{min-robot} \leq v_{robot} \leq v_{max-robot}$) respectivement, avant de les envoyer au pionnier-P3DX. À partir de ces conditions nous avons déduit les contraintes (de type ≤ 0) utilisées par l'algorithme PSO pour minimiser la fonction objective de l'équation (3.45) :

$$v_{min} \leq v_R \leq v_{max} \Rightarrow \begin{cases} v_R - v_{max} \leq 0 \\ -v_R + v_{min} \leq 0 \end{cases} \tag{3.46}$$

$$v_{min} \leq v_L \leq v_{max} \Rightarrow \begin{cases} v_L - v_{max} \leq 0 \\ -v_L + v_{min} \leq 0 \end{cases} \tag{3.47}$$

$$v_{min-robot} \leq \frac{v_R + v_L}{2} \leq v_{max-robot} \Rightarrow \begin{cases} v_R + v_L - 2v_{max-robot} \leq 0 \\ -v_R - v_L + 2v_{min-robot} \leq 0 \end{cases} \tag{3.48}$$

$$w_{min-robot} \leq \frac{v_R - v_L}{2L} \leq w_{max-robot} \Rightarrow \begin{cases} v_R - v_L - 2Lw_{max-robot} \leq 0 \\ -v_R + v_L + 2Lw_{min-robot} \leq 0 \end{cases} \tag{3.49}$$

Comme v_R et v_L sont des variables à optimiser qui représentent les particules, l'ensemble de ces particules présente une population. Les quatre premières contraintes sont trouvées à partir des limites réelles de vitesses des roues, tandis que les quatre contraintes suivantes ont été déduites par l'équation (2.31) qui représente la cinématique du robot afin d'introduire son comportement réel. L'algorithme PSO est le suivant :

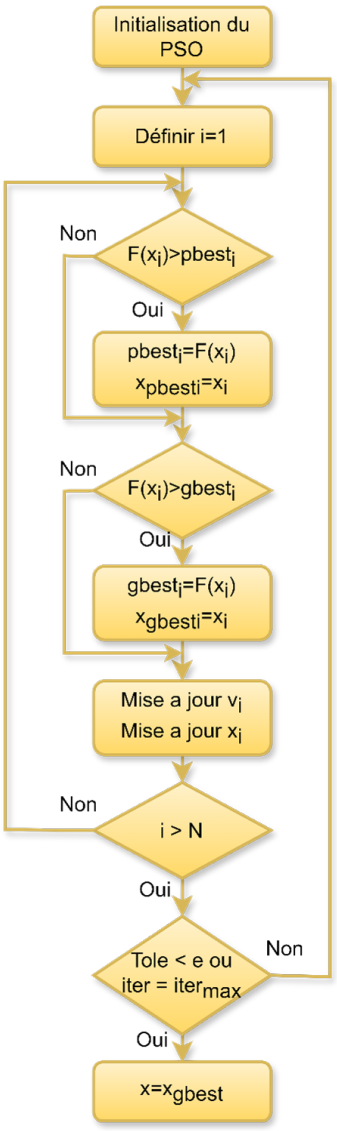


FIGURE 3.33 – Organigramme de l'algorithme d'optimisation par essaim de particules "PSO".

La mise à jour de la position et la vitesse des particules est effectué à travers les équations de mouvement suivant :

$$v_i(t+1) = \omega\, v_i(t) + c_1 r_1[pbest_i(t) - x_i(t)] + c_2 r_2[gbest(t) - x_i(t)] \tag{3.50}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3.51}$$

$v_i(t)$ la vitesse de la particule i au temps t, $x_i(t)$ la position de la particule i au temps t ; c_1 et c_2 ($0 \leq c_1 \leq 2$, et $0 \leq c_2 \leq 2$) sont des coefficients constants fixés par l'utilisateur sont choisi dans notre cas $c_1 = c_2 = 2$ et ω ($0 \leq \omega < 1$) est mis à jour a chaque itération par $w = wmax - (wmax - wmin) * ite/maxite$ tel que on a choisi $wmax = 0.9$, $wmin = 0.4$ et $maxite = 10$, et r_1 et r_2 sont des nombres aléatoires tirés à chaque itération ; $gbest(t)$ est la meilleure solution trouvée jusqu'au temps t et $pbest_i(t)$ est la meilleure solution trouvée

par la particule i. Afin d'assurer à travers plusieurs simulations un temps de réponse très rapide de l'algorithme PSO, on a fixé un nombre maximum d'itérations à 10 puis on a sélectionné le meilleur résultat. Ce nombre maximum d'itérations est choisi de telle sorte qu'il soit suffisant pour que l'algorithme converge vers les résultats optimaux en un temps très court. Le temps de convergence de l'algorithme PSO dans notre simulation est de 0,09s [81].

3.8.4 Structure de l'algorithme hybride ANFIS-PSO

Dans notre travail le robot mobile est contrôlé soit par ANFIS de suivi soit par ANFIS d'évitement d'obstacles. Cependant, la sélection d'un des contrôleurs se fait en fonction de la distance obtenue entre les capteurs ultrason du robot et les obstacles, si la distance est inférieure à 20 cm le robot est contrôlé par l'ANFIS d'évitement d'obstacles sinon par ANFIS de suivi comme le montre l'organigramme de la (Figure 3.34).

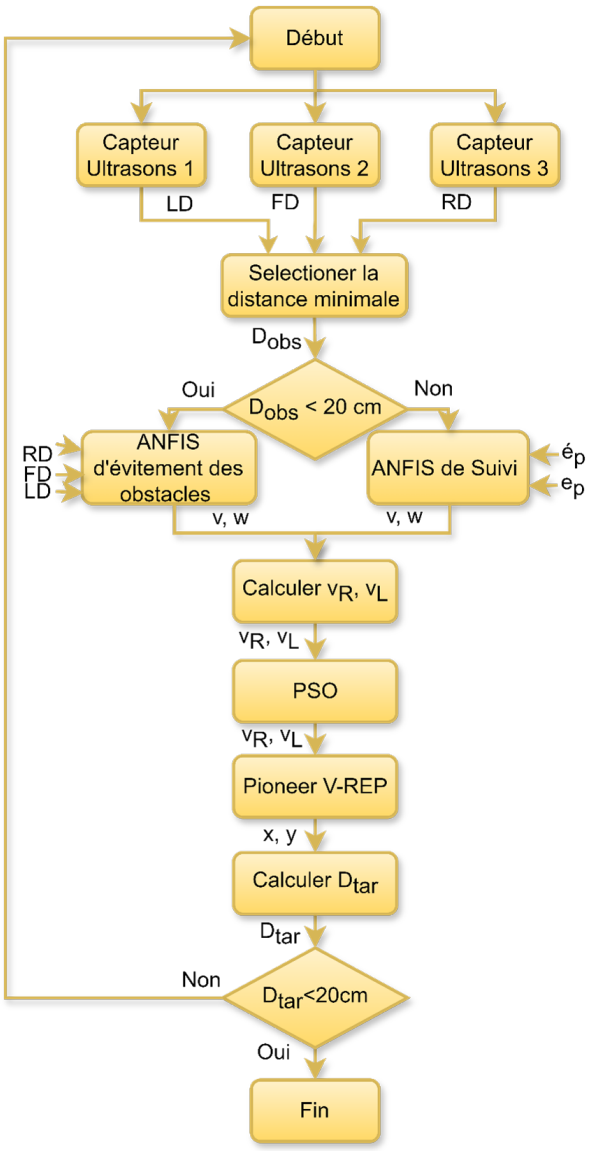


FIGURE 3.34 – Organigramme de navigation virtuel utilisant la méthode ANFIS-PSO.

3.9 Conclusion

Dans ce chapitre, nous avons présenté les techniques de l'intelligence artificielle à savoir : la logique floue, des réseaux de neurones et des réseaux neuro-flous. Nous avons d'abord décrit les notions de base de la logique floue ainsi que la structure de la commande floue. Ensuite, nous avons abordé le concept des réseaux de neurones artificiels et leur apprentissage par l'algorithme de rétro-propagation. Par la suite, nous avons présenté les structures de base des combinaisons de la logique floue avec les réseaux de neurones, plus particulièrement la structure ANFIS.

Dans la deuxième partie de ce chapitre nous avons présenté l'intérêt d'optimisation par la méthode des essaims de particules PSO. D'abord nous avons commencé par citer son origine ainsi que son principe, puis nous avons présenté son formalisme mathématique. Par

la suite, nous avons parlé d'un algorithme itératif adaptatif qui modifie son comportement en fonction de sa découverte progressive.
La dernière partie du chapitre a été consacrée pour le développement des contrôleurs ANFIS avec hybridation avec l'algorithme PSO proposés pour la navigation d'un robot mobile à entrainement différentielle de telle manière à chercher une cible tout en évitant les obstacles.
Dans le chapitre qui suit, nous allons faire une description de la réalité virtuelle dans le domaine de la robotique mobile et le logiciels virtuel V-REP afin de réaliser une navigation virtuelle d'un robot mobile avec le contrôleur hybride ANFIS-PSO dans un environnement virtuel en utilisant l'environnement MATLAB et V-REP et on conclura par une simulation et analyse des résultats.

Chapitre 4

La réalité virtuelle et la plateforme virtuel V-REP

4.1 Introduction

L'expression **réalité virtuelle (RV)** est souvent employée dans les médias, l'utilisation de la RV est populaire dans le monde du jeu vidéo, pour présenter des simulateurs servant à l'entraînement des pilotes d'avion. Parfois, l'expression servira à vanter le réalisme des effets spéciaux créés pour un film [136], mais elle a également un intérêt pour la recherche. Les scientifiques s'intéressant aux comportements humains utilisent la RV pour manipuler et contrôler l'ensemble de l'environnement expérimental (le monde artificiel), ce qui est impossible dans le monde physique. Tarr and Warren [137] montre l'intérêt de la RV dans les années 2000 car elle présente un bon compromis entre contrôle et similitude de l'environnement. Par exemple, lors de l'étude des interactions avec les piétons, il est possible de contrôler parfaitement toutes les actions des piétons dans le temps, leurs comportements, leurs attributs (par exemple, le sexe, la taille, la morphologie, les vêtements...) [138]. Nous pouvons donc constater que la réalité virtuelle possède un périmètre de définition très large et mouvant.
En outre, pour des raisons de coût, de complexité de mise en œuvre ou de danger, on exploite la réalité virtuelle dans le but de confronter un utilisateur à des évènements réels. C'est le cas par exemple pour les simulations de scènes de guerre (pour des militaires ou pour le jeu), de situations d'urgences, de simulations sportives et plus largement d'entraînement (conduite de voitures, de chars, d'avions). Par ailleurs, les robots avancés sont encore trop chers et il se peut que les utilisateurs ne puissent pas transcrire leurs connaissances et expériences sous forme de contrôleurs adéquats au bon fonctionnement, ce qui se traduit par des dommages pendant l'expérimentation dans les laboratoires.
C'est à partir de ces constatations que les chercheurs en robotique utilisent les techniques de la réalité virtuelle qui est une combinaison de technologies qui permet une interaction

de l'humain avec la simulation créée par un ordinateur [139]. Récemment ces techniques de la réalité virtuelle ont été développés par des chercheurs dans des laboratoires virtuels en utilisant des simulateurs pour la conception d'environnement virtuel afin de tester leurs expériences, théories, les idées avant la mise en œuvre réelle [1, 9] plusieurs simulateurs sont utilisés dans le domaines de la robotique comme : ARGoS [140] , Webots [141] et V-REP [142].

Dans notre travail on a utilisé V-REP qui est une Plateforme de simulation destinée pour le développement des algorithmes rapides, les simulations d'automatisation d'usine, le prototypage et la vérification rapide, l'éducation relative à la robotique, le contrôle à distance, le double contrôle de la sécurité, etc.

La première partie de ce chapitre sera consacrée à présentation des généralités sur la réalité virtuelle. Ensuite, nous synthétisons une méthodologie d'application de la réalité virtuelle à la Robotique.

Dans la deuxième partie de ce chapitre nous proposons des outils de simulation qui offrent davantage de réalisme. Notre approche consiste donc à immerger notre robot dans un environnement afin de chercher les cibles avec évitement des obstacles sans collision. Pour ce faire, nous utilisons la réalité virtuelle.

Dans la troisième partie de chapitre nous abordons d'abord la description de la plateforme de réalité virtuelle à base du Logiciel V-REP à savoir : ses concepts, sa terminologie ainsi que ses avantages. Ensuite nous étudions les étapes nécessaires pour connecter V-REP avec MATLAB, en vue de répondre au problème de navigation d'un robot mobile à travers l'utilisation des différentes fonction du Logiciel MATLAB permettant la transmission et la récupération des données.

4.2 Réalité virtuelle

4.2.1 Rappel historique

Avant de définir la réalité virtuelle, il nous semble intéressant de citer quelques travaux des précurseurs de la réalité virtuelle, tellement précurseur que le terme de réalité virtuelle n'existait pas encore... En 1962 Morton Heilig [143] crée le Sensorama (Figure 4.1) pour un aperçu du dispositif, considéré comme le premier système de réalité virtuelle, il permettait de simuler une balade en bicyclette dans le quartier de Brooklyn. Pour cela, Heilig a créé un dispositif proposant un écran grand angle permettant une vision stéréoscopique, une restitution sonore spatialisée, un siège vibrant, un ventilateur et un diffuseur d'odeur.

FIGURE 4.1 – Une affiche ventant le Sensorama [144].

En 1965 Sutherland [145] reprend des travaux d'Heilig et crée le premier visiocasque (Figure 4.2). Celui-ci est vu par Sutherland comme une fenêtre sur le virtuel. Le dispositif comporte deux tubes cathodiques monochromatiques mais n'offre pas encore une vision stéréoscopique (les images affichées sont les mêmes).L'histoire de ces inventeurs plus détaillée dans le livre de Rheingold [146].

FIGURE 4.2 – Une vue du premier visiocasque [145].

4.2.2 Différentes définitions de la réalité virtuelle

Différentes définitions présentent dans la littérature, se retrouvent sur un certain nombre de notions clés. Nous ne reprenons ici qu'un échantillon que nous espérons représentatif.

Burdea et Coiffet [147] et Fuchs [148] définissent la réalité virtuelle par le triptyque Immersion, Interaction et Imagination.

Ellis [149] préfère à l'expression de réalité virtuelle, celle sémantiquement plus cohérente d'environnement virtuel. Voici la définition qu'il en fait :

Les environnements virtuels sont des images virtuelles interactives améliorées par des processus particuliers et complétées par des restitutions provenant d'autres modalités telles que le son et l'haptique et ce pour convaincre les utilisateurs qu'ils sont immergés dans un espace synthétique.

En 1992, Zeltzer[150] propose un nouveau triptyque composé cette fois-ci de l'autonomie, de l'interaction et de la présence. L'autonomie désigne le comportement des entités virtuelles présentes dans l'environnement virtuel. L'interaction pointe la capacité d'interagir avec ces entités et l'environnement. La présence est ici associée à la stimulation sensorielle.

En 2000, Jaron Lanier revient sur sa vision de la réalité virtuelle dans une interview [151] :

La réalité virtuelle permet via l'informatique de créer l'illusion d'être dans un monde alternatif avec d'autres personnes. Il s'agit d'une sorte de rêve que vous faites de façon consciente et auquel d'autres personnes participent.

Enfin, la communauté scientifique française s'est accordée autour d'une finalité et deux définitions. Voici comment Fuchs [148] précise la finalité de la réalité virtuelle :

La finalité de la réalité virtuelle est de permettre à une personne (ou à plusieurs) une activité sensori-motrice et cognitive dans un monde artificiel, créé numériquement, qui peut être imaginaire, symbolique ou une simulation de certains aspects du monde réel.

Fuchs [148] propose une définition fonctionnelle de la réalité virtuelle :

La réalité virtuelle va permettre à l'homme de s'extraire de la réalité physique pour changer virtuellement de temps, de lieu et (ou) de type d'interaction : interaction avec un environnement simulant la réalité ou interaction avec un monde imaginaire ou symbolique.

Arnaldi et al [152] proposent une définition technique :

La réalité virtuelle est un domaine scientifique et technique exploitant l'informatique et des interfaces comportementales en vue de simuler dans un monde virtuel le comportement d'entités 3D, qui sont en interaction en temps réel entre elles et avec un ou des utilisateurs en immersion pseudo-naturelle par l'intermédiaire de canaux sensori-moteurs.

La définition technique permet de décrire avec efficacité les fonctionnalités d'un disposi-

tif de réalité virtuelle. L'énoncé de la finalité et la définition fonctionnelle sont-elles plus axées sur l'expérience dans le monde artificiel créé.

4.2.3 Réalité virtuelle et la robotique

La réalité virtuelle présente également un intérêt pour la robotique. En particulier, elle est utile lorsqu'un robot et un humain sont amenés à coopérer dans une tâche. Son pouvoir de contrôle de l'environnement en fait un outil idéal pour l'entraînement des personnes destinées à interagir avec des robots. Par exemple, Devigne et al. [153] présentent une plateforme d'immersion en réalité virtuelle (Figure 2.7) pour l'apprentissage à la conduite d'un fauteuil roulant électrique semi-autonome, c'est-à-dire équipé d'une assistance à la conduite anti-collision. La réalité virtuelle peut être vue également comme un organe de sécurité entre une personne et un robot collaboratif. Par exemple, Hernoux et al. [154] utilise la réalité virtuelle comme interface de sécurité entre un cobot (robot collaboratif) et un opérateur.
La réalité virtuelle a également été suggérée comme outil d'étude pour le domaine de l'interaction humain robot. Des études comparatives ont été menées, comme Li et al. [155], Wainer et al. [156] qui comparent l'interaction humain robot en réalité virtuelle par rapport à une interaction réelle. Il n'y a cependant pas d'interaction physique ni d'interactions entre trajectoires : l'humain et le robot sont fixes.
Dans notre travail on a utilisé la technologie de la réalité virtuelle pour créer avec précision un environnement de navigation d'un robot mobile (Pioneer P-3dx) adéquat avec celui de navigation réelle des robots, dans le but de tester nous algotithmes de commande avant la mise en oeuvre réelle pour éviter les dommage.

4.3 Outils de simulation

Afin de concevoir les environnements virtuels de navigation des robots mobiles en utilisant les techniques de la réalité virtuelle on va commencer par présenter les outils de simulation utilisés par les roboticiens ; tel que avant l'année 2011, Staranowicz and Mariottini [157], Castillo-Pizarro et al. [158, 159], Ivaldi et al. [5] ont fait des études comparatives sur les outils de simulation en robotique, ils ont conclu que les outils modulaire comme Gazebo et V-REP sont fortement actifs, alors que les outils spécifiques sont disparu comme le logiciel HumanS.

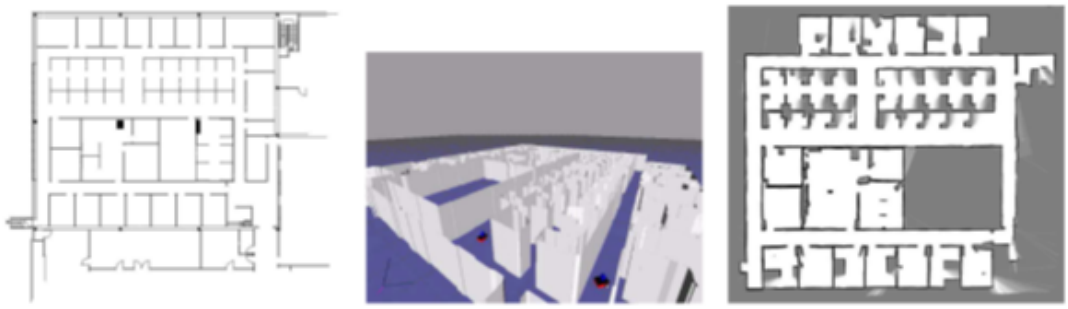


FIGURE 4.3 – Exemple d'utilisation du logiciel Gazebo [160] : une carte 2D dessinée à la main, puis l'extrusion 3D créée à partir de la carte 2D dans laquelle circule une simulation d'un robot mobile Pioneer, et enfin la carte de l'environnement générée par laser dans gazebo.

Aujourd'hui les outils disponibles sont nombreux ; Gerkey et al. [161] présentent un outil de simulation 2D pour la simulation multi-robot. Koenig and Howard [160] présente le logiciel Gazebo qui est un logiciel de simulation 3D conçu pour reproduire aussi fidèlement que possible l'environnement dynamique d'un (ou plusieurs) robot mobile, ce simulateur est développé en collaboration avec le logiciel ROS, il est très populaire dans la communauté robotique. Michel [162] présentent le logiciel libre Webots qui est comme Gazebo, un logiciel de simulation de robotique mobile offrant un environnement de prototypage rapide pour la modélisation, la programmation et la simulation de robots. C'est un logiciel libre depuis 2018 seulement, il est moins complexe que Gazebo. Rohmer et al. [163] présentent le logiciel V-REP, Ce simulateur a une plus large palette de techniques de programmation et des modèles de simulation et des contrôleurs plus facilement portables sur de vrais robot que ses concurrents Gazebo et Webots. V-REP serait donc plus facile à utiliser, mais demeure plus fermé que Gazebo et Webots.
ces dernière années les auteurs cherchaient à donner à leurs robots des environnements plus réalistes que ceux proposés par Gazebo, Webots, ou V-REP tel que, Echeverria et al. [164] présentent le logiciel MORSE, Konrad [165] présente le moteur de jeu vidéo Unity, Enfin Dosovitskiy et al. [136] présente le logiciel CARLA plus récent que les précédents simulateurs cités, CARLA a été conçu sur la base du moteur de jeu vidéo ultra réaliste Unreal Engine. CARLA a été conçu pour répondre à un besoin spécifique : créer un outil de simulation dédié au développement des véhicules autonomes. Le simulateur propose des environnements urbains photo-réalistes, et est conçu pour fonctionner facilement avec des réseaux de neurones.
Dans notre travail on a choisi d'utiliser le logiciel d'expérimentation des robots virtuels (V-REP)pour réaliser une co-simulation avec Matlab afin de tester nos algorithmes de navigation.

4.4 Logiciel virtuel V-REP

V-REP (Virtual Robot Experimentation Platform) est un simulateur de robot, développé et entretenu par « Coppelia Robotics », est créée en 2010 et s'est développé au cours des dernières années. De nos jours, c'est le logiciel le plus utilisés dans le domaine de la robotique (avec une licence gratuite pour un but éducatif). Comme le disent ses auteurs : "Le V-REP est le couteau suisse parmi les simulateurs de robots", l'environnement de simulation est modélisé à l'aide de l'éditeur de scène inclus. Il contient un grand nombre de modèles prédéfinis pour différents types de robots [142].
V-REP est un simulateur de robot 3D basé sur une architecture de contrôle distribuée : les programmes de contrôle (ou scripts) peuvent être directement attachés aux objets de la scène et s'exécuter simultanément de manière filetée ou non filetée. Cela rend le V-REP très polyvalent et idéal pour les applications multi robots, et permet aux utilisateurs de modéliser des systèmes robotiques de la même manière qu'en réalité où le contrôle est la plupart du temps également distribué. V-REP vous permet d'éditer et de simuler des systèmes robotiques complets ou des sous-systèmes (capteurs, mécanismes, etc.). V-REP contient un grand nombre d'exemples, de modèles de robots, de capteurs et d'actionneurs. Des nouveaux modèles peuvent également être conçus et ajoutés à VREP pour mettre en œuvre des expériences de simulation personnalisées [81]. Il offre une multitude de fonctionnalités qui peuvent être facilement intégrées et combinées grâce à une API et une fonctionnalité de script exhaustive, V-REP est également une multi plate-forme qui peut être utilisée sous Windows, Linux et Mac OS. Il propose quatre moteurs de physique différents, notamment Bullet Physics, Open Dynamics, Newton et Vortex Dynamics, qui permettent un calcul dynamique rapide et personnalisable permettant de simuler des interactions réelles entre la physique et les objets.

4.4.1 Avantages de V-REP

V-REP présente plusieurs avantages par rapport à d'autres simulateurs, tels qu'une licence gratuite et ne nécessitant pas de carte graphique ou de GPU spécifique. Excellente solution pour :

- Prototypage et vérification rapides.
- Simulation de systèmes d'automatisation industrielle.
- Développement rapide d'algorithmes.
- Enseignement lié à la robotique.
- Surveillance à distance.

- Contrôle du matériel.
- Surveillance de la sécurité.
- Présentation du produit.

La version utilisée dans notre travail (V-REP PRO EDU, Version 3.6.1. rev. 3) présente de nombreuses fonctionnalités intéressantes, on cite :

- **Contenu portatif et multiplateforme :** Il permet la création de contenu portable, évolutif et facile à entretenir ou modifiable : un seul fichier portable peut contenir un modèle ou une scène entièrement fonctionnelle, y compris un code de contrôle.
- **Détection de collision et calcul de distance :** Contrôle rapide des interférences et calcul de la distance minimale.
- **Simulation du capteur de vision.**
- **Simulation du capteur de proximité.**
- **Interfaces utilisateur personnalisées :** Un nombre illimité d'éléments d'interface utilisateur entièrement personnalisables. Style Windows ou style OpenGL.
- **Chemin / planification de mouvement :** Elle est prise en charge de manière très flexible via la bibliothèque OMPL enveloppée dans un plugin V-REP.
- **Remote API :** Plus de 100 fonctions V-REP incorporables : contrôlez une simulation ou le simulateur lui-même à distance (par exemple, à partir d'un robot réel ou d'un autre PC). Facile à utiliser, prend en charge le mode d'opération synchrone ou asynchrone, il est optimisé pour un transfert de données important et réduit le délai de communication. Six langages de programmation sont pris en charge : C / C ++, Python, Java, Urbi, Matlab et Octave. Cela rend V-REP très polyvalent et idéal pour les applications multirobots (Figure 4.4)(Figure 4.5).

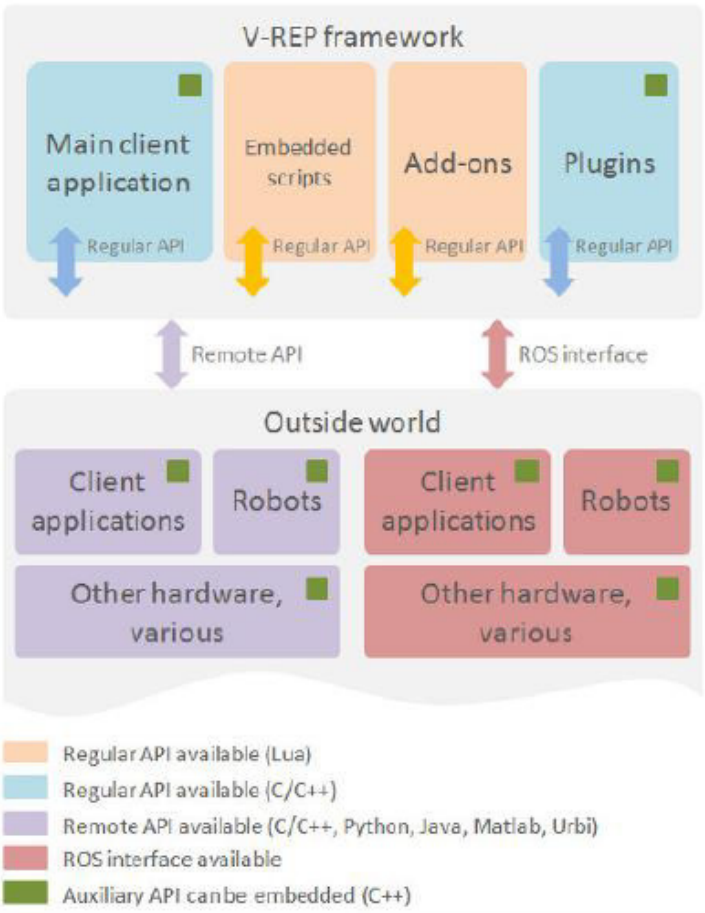


FIGURE 4.4 – Structure de Remote API pour V-REP (en violet)

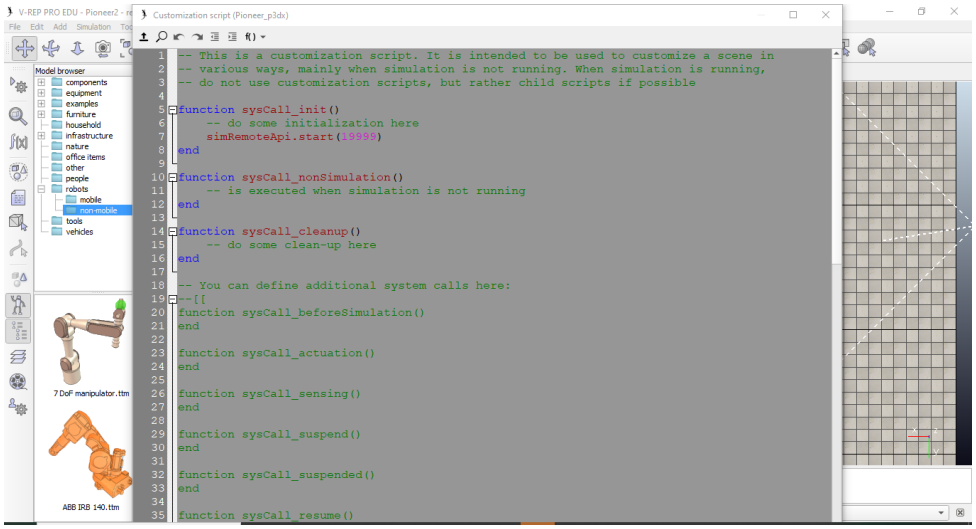


FIGURE 4.5 – Remote API

4.5 Configuration de l'environnement de simulation

Dans notre système, MATLAB et V-REP utilisent une API distante (Remote API) pour réaliser la navigation virtuelle d'un robot mobile dans un environnement encombré, MATLAB est utilisé comme client et V-REP est utilisé comme serveur, où MATLAB envoie des commandes à V- REP et reçoit les données de V-REP.

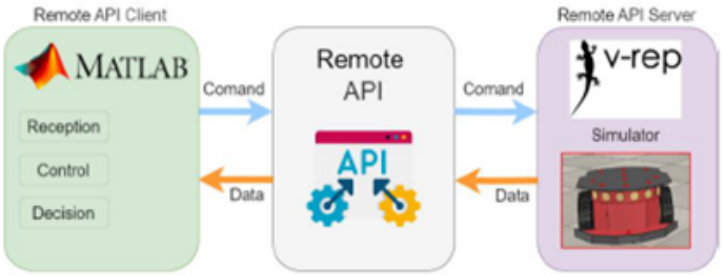


FIGURE 4.6 – Schéma de connexion entre MATLAB et V-REP.

4.5.1 Connexion Matlab et V-REP

Étapes nécessaires pour établir une connexion entre MATLAB et V-REP : **Etape 1 :** Créer un dossier qui va jouer le rôle du « `WorkSpace` » (espace de travail). **Etape 2 :** Copier les trois fichiers suivant du répertoire de V-REP : [`C:\Program Files \V-REP3\V-REP_ PRO_EDU\programming \remoteApiBindings \matlab\matlab`] vers le « `WorkSpace` » :

- `remApi.m`
- `remoteApiProto.m`
- `simpletest.m`

Etape 3 : Copier le fichier [`remoteApi.dll`] du répertoire V-REP : [`C:\Program Files \V-REP3\V-REP_ PRO_ EDU\programming \remoteApiBindings \lib\lib`], ici on doit choisir l'extension qui correspond au celle de MATLAB qui nous utilisons soit x64 bit ou x32 bit.
Etape 4 : Lancer V-REP, ensuite crée une nouvelle scène et ouvrir le « `Customization Script (Pioneer P-3dx)` » puis dans la condition de l'initialisation ajouté l'appelle de la fonction `simRemoteApi.Start(19999)` (Figure 4.7), qui va assurer la connexion avec MATLAB par « `RemoteAPI` » à chaque lancement de simulation sur le port numéro 19999, Enfin sauvegarder la scène dans le « `WorkSpace` » (crée à l'étape 1).

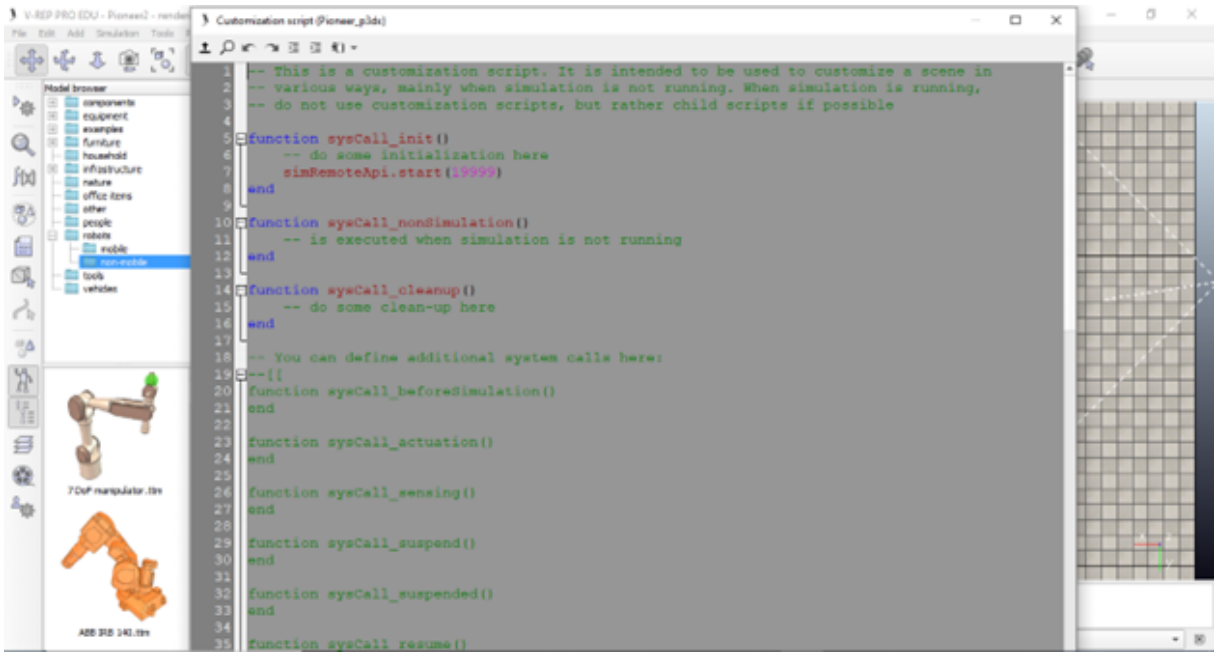


FIGURE 4.7 – Fichier de personnalisation.

Etape 5 : Maintenant que nous avons configuré la scène V-REP et copier les fichiers de la bibliothèque, on doit pointer le « `WorkSpace` » de MATLAB vers notre « `WorkSpace` » (le dossier crée à l'étape 1). Les trois fichiers, ou « `remoteApi.dll` », « `remApi.m` » et « `remoteApiProto.m` » sont les fichiers clés pour la co-simulation V-REP et Matlab. A noter que lorsque nous faisions de la co-simulation, ces trois fichiers doivent être collés et mis dans le même dossier avec le modèle de robot V-REP et le code Matlab correspondant au contenu expérimental. **Etape 6 :** Crée un nouveau script [`Matlab--> Home--> New--> Script`]. Dans ce script on va établir la connexion avec V-REP comme suit :

- Création d'un nouvel objet de type « `remApi` », nommé par exemple vrep.
- Fermer toutes les connexions ouvertes avec V-REP en utilisant la fonction suivante : « `vrep.simxFinish (-1)` » pour éviter la possibilité perturbé notre travail.
- Crée une connexion avec V-REP par la fonction :« `vrep.simxStart` » en utilisant l'adresse IP local « `127.0.0.1` » sur le port « `19999` ».

Une fois qu'on termine l'établissement de la connexion entre V-REP et MATLAB on peut commencer notre travail, Il est préférable d'appelé le destructeur : « `delete ()` » à la fin du travail. Nous allons expliquer ci-dessous les fonctions de la bibliothèque « `remApi` » que nous avons utilisé [166] afin d'établir la connexion entre matlab et VREP :

Simxstart :

La description	Démarre un thread de communication avec le serveur (c'est-à-dire VREP). Un même client peut démarrer plusieurs threads de communication (mais un seul thread de communication pour une adresse IP et un port).
Prototype	**[clientID] = vrep.simxStart (connectionAddress, connectionPort, waitUntilConnected, doNotReconnectOnceDisconnected, timeOutInMs, commThreadCycleInMs).**
Paramètres	**ConnectionAddress :** l'adresse IP où se trouve le serveur (c'est-à-dire V-REP). **ConnectionPort :** le numéro de port où se connecter. **WaitUntilConnected :** si vrai, alors la fonction se bloque jusqu'à ce qu'elle soit connectée (ou expiré). **DoNotReconnectOnceDisconnected :** s'il est vrai, le thread de communication ne tentera pas une seconde connexion si une connexion a été perdue. **TimeOutInMs :** **Si positif :** le délai de connexion en millisecondes pour la première tentative de connexion. Dans ce cas, le délai d'attente pour bloquer les appels de fonction est de 5000 millisecondes. **Si négatif :** sa valeur positive est le délai d'attente pour bloquer les appels de fonction. Dans ce cas, le délai de connexion pour la première tentative de connexion est de 5000 millisecondes. **CommThreadCycleInMs :** indique à quelle fréquence les paquets de données sont envoyés en va-et-vient. La réduction de ce nombre améliore la réactivité et une valeur par défaut de 5 est recommandée.
Valeurs de retour	**ClientID :** l'ID du client, ou -1 si la connexion au serveur n'était pas possible (c'est-à-dire un délai d'attente a été atteint). Un appel à simxStart doit toujours être suivi à la fin avec un appel à simxFinish si simxStart n'a pas retourné -1.

Simxfinish :

La description	Termine le fil de communication. Elle devrait être la dernière fonction remoteAPI appelée côté client. SimxFinish ne doit être appelé qu'après un appel réussi à simxStart. Il s'agit d'une fonction d'aide de remoteAPI.
Prototype	vrep.simxFinish (numéro ou clientID)
Paramètres	**ClientID :** l'ID du client. Se référer à simxStart. Peut-être -1 pour mettre fin à tous les fils de communication en cours d'exécution.

Delete :

La description	**Destructeur :** décharger la bibliothèque.

Pour assurer la connexion entre MATLAB et V-REP, l'API distante doit être dans le même répertoire que nous avons créé pour sauvegarder tous nos codes MATLAB (fonction PSO, fonction ANFIS, le programme principal...etc), la connexion entre MATALB et V-REP est assuré par des lignes de commande ajoutées au programme principal comme indiqué dans l'organigramme ci-dessous :

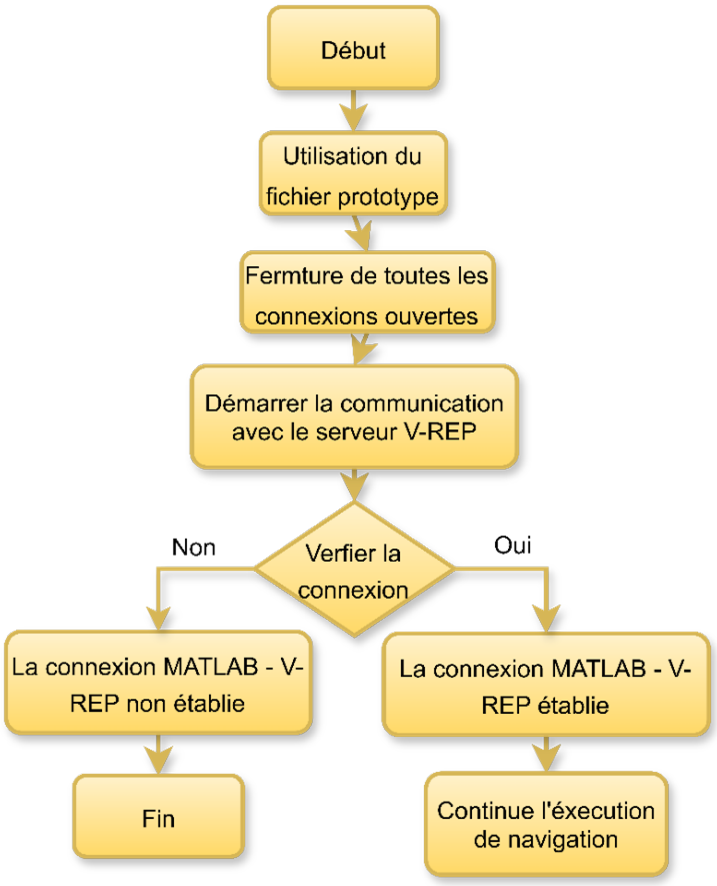


FIGURE 4.8 – Organigramme de connexion MATLAB et V-REP.

4.5.2 Mode de connexion Matlab-V-REP

Il existe deux modes de communication possibles entre V-REP et MATLAB [166] : le mode synchrone et le mode asynchrone, dans notre cas la connexion en mode synchrone a été choisi.

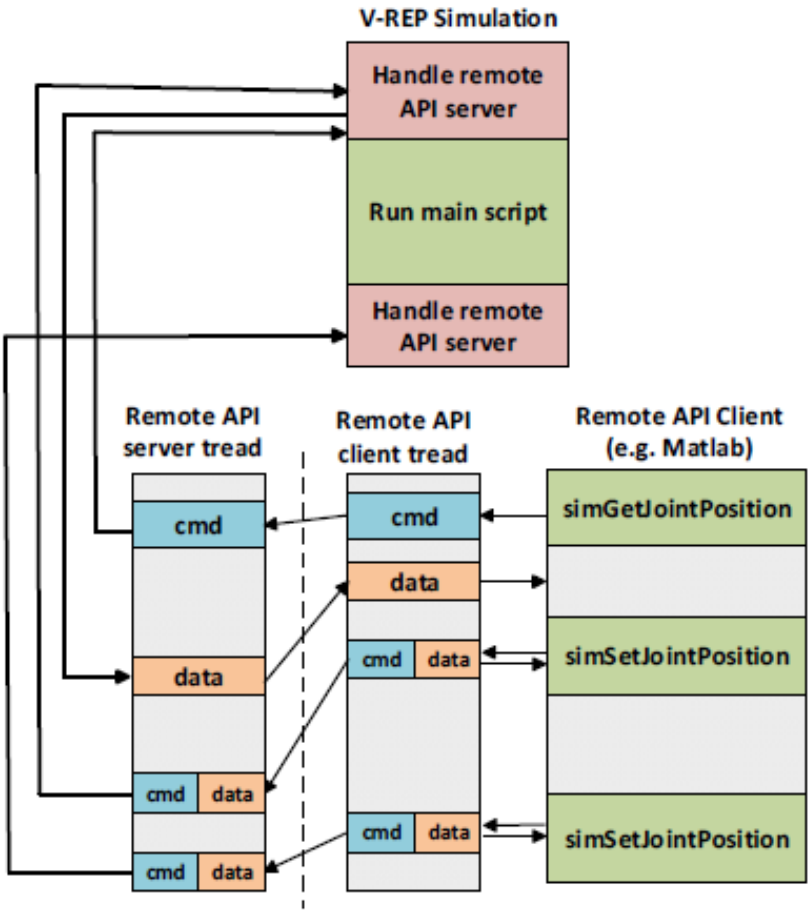


FIGURE 4.9 – V-REP MATLAB synchronous mode [166].

4.5.3 Robot mobile « Pioneer P-3dx »

Les robots de recherche Pioneer sont les robots mobiles intelligents les plus populaires au monde pour l'éducation et la recherche. La polyvalence, la fiabilité et la durabilité les rend la plate-forme préférée pour la robotique intelligente avancée. Pioneer P-3dx montrée dans la (Figure 4.10) est un petit robot d'entraînement différentiel à deux roues, léger, actionné et idéal pour l'utilisation à l'intérieur d'un laboratoire ou d'une salle d'étude. [167].

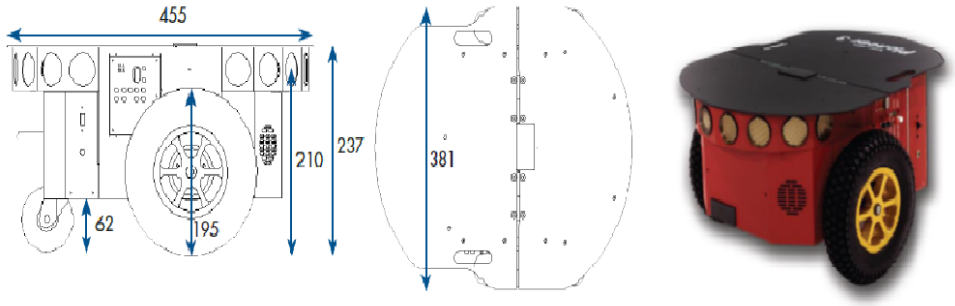


FIGURE 4.10 – Robot mobile Pioneer P-3dx.

TABLEAU 4.1 – Caractéristiques du robot mobile Pioneer P-3dx.

Caractéristiques	
Corps	1,6 mm en aluminium (revêtu de poudre)
Pneus	Caoutchouc rempli de mousse
Opération	
Poids du robot	9 Kg
Charge utile	17 Kg
Mouvement différentiel	
Rayon de chaque roue motrice	9.75 cm
Distance entre la roue et le centre des deux roues motrices	17.13 cm
Rayon de rotation	0 cm
Rayon oscillant	26.7 cm
Max. Vitesse avant / arrière	1.2 m/s
Vitesse de rotation	300 deg/s
Max. Traversable Step	2.5 cm
Max. Traversable Gap	5 cm
Max. Niveau traversable	25 %
Terrain traversable	Intérieur, accessible en fauteuil roulant.

4.5.4 Préparation de l'environnement virtuel sur V-REP

La préparation de l'environnement virtuel V-REP est une étape très importante afin de réaliser une co-simulation avec Matlab dans le but de tester les techniques de la commande intelligente développer dans notre travail dans des environnement reflété a la réalité avant d'implémenter ces techniques sur un robot réel et dans des environnements inconnu et

encombré , pour cela on a fait une conception de deux environnements virtuel le premier contient le robot pioneer P-3dx ,une cible et sans obstacles représenter par (Figure 4.11) et le deuxième contient le robot pioneer P-3dx, une cible , les obstacles statiques et les murs représenter par (Figure 4.12).

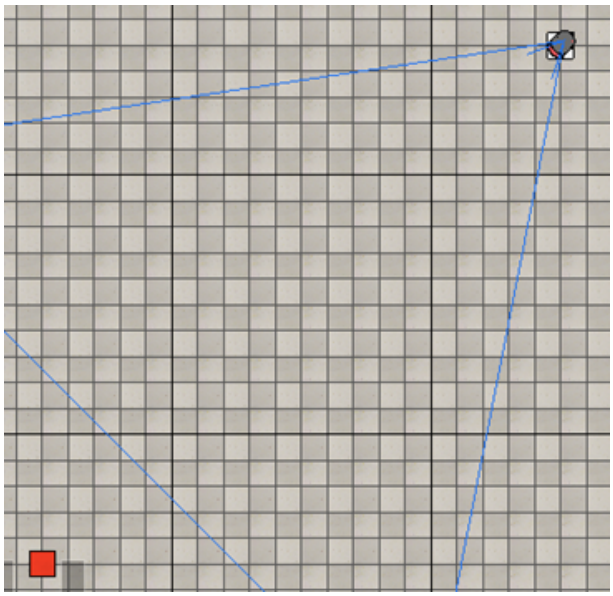


FIGURE 4.11 – Environnement 1 de V-REP de simulation sans obstacles.

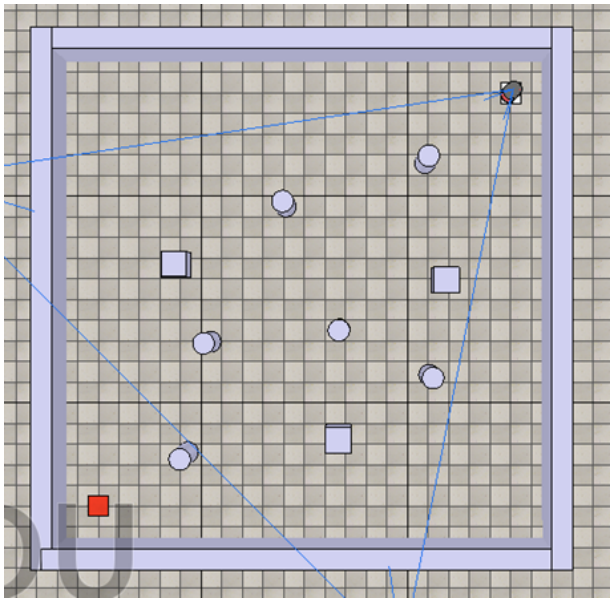


FIGURE 4.12 – Environnement 2 de V-REP de simulation avec obstacles.

4.5.5 Déclaration des objets V-REP dans Matlab

Afin de réaliser la navigation virtuelle d'un robot mobile (pioneer P-3dx) de type entrainement différentiel, il faut déclarer les objets conçus dans l'environnement virtuel V-REP ci-dessus dans l'environnement Matlab par la fonction (SimxGetObjectHandle)

pour qu'on puisse utiliser d'autres fonctions de récupération des données et transmissions des commandes dans le but de réaliser une Co-simulations.
Les objets à déclarer sont :

- Pioneer P-3dx ;
- Moteur de la roue gauche ;
- Moteur de la roue droite ;
- Cible ;
- Capteurs ultrasons.

SimxGetObjectHandle :

La description	Récupère un Handler d'objet en fonction de son nom.
Prototype	**[returnCode, handle] = vrep.simxGetObjectHandle (clientID, objectName , operationMode)**
Paramètres	**ClientID :** l'ID du client. Se référer à simxStart. **ObjectName :** nom de l'objet. Si possible spécifiez toujours le nom complet de l'objet, y compris le #. **OperationMode :** mode de fonctionnement de la fonction remoteAPI. Le mode de fonctionnement recommandé pour cette fonction est simx_opmode_blocking.

La description des fonctions simxstart ,simGetObjectHandle,simxsetJointTargetVelcity Dans le lien suivant [166].

4.5.6 Récupération des données de V-REP par Matlab

Dans le but de calculer la commande par le contrôleur hybride ANFIS-PSO utiliser pour la navigation virtuelle dans un environnement encombré et inconnu on a utilisé les fonctions (SimxGetObjectOrientation, SimxGetObjectPosition, simxGetObjectVelocity, simxReadProximitySensor) ci-dessous afin de récupérer les informations qui suit :

- L'orientation du robot.
- La position du robot.
- La vitesse linéaire et angulaire du robot.
- Les vitesses linéaires et angulaires des roues gauches et droite.
- Les distances entrent les obstacles et le robot.
- Position de la cible.

SimxGetObjectOrientation :

La description	Récupère l'orientation (angles d'Euler) d'un objet.
Prototype	[**returnCode, eulerAngles**] = **vrep.simxGetObjectOrientation (clientID, objectHandle, relativeToObjectHandle, operationMode)**
Paramètres	**ClientID :** l'ID du client. Se référer à simxStart. **ObjectHandle :** handle de l'objet. **RelativeToObjectHandle :** indique par rapport à quel cadre de référence nous voulons l'orientation. Spécifiez -1 pour récupérer l'orientation absolue, sim_handle_parent pour récupérer l'orientation par rapport au parent de l'objet, ou un ObjectHandle relatif à l'image de référence dont vous souhaitez. **orientationOperationMode :** mode de fonctionnement de la fonction remoteAPI. Les modes de fonctionnement recommandés sont simx_opmode_streaming (le premier appel) et simx_opmode_buffer (les appels suivants).
Valeurs de retour	**ReturnCode :** un code de retour de la fonction remoteAPI. **EulerAngles :** 3 valeurs représentant les angles d'Euler (alpha, bêta et gamma).

SimxGetObjectPosition :

La description	Récupère la position d'un objet.
Prototype	[**ReturnCode, position du tableau**] = **vrep.simxGetObjectPosition (clientID, objectHandle, relativeToObjectHandle, operationMode)**.
Paramètres	**ClientID :** l'ID du client. Se référer à simxStart. **ObjectHandle :** handle de l'objet. **RelativeToObjectHandle :** indique par rapport à quel cadre de référence nous voulons la position. Spécifiez -1 pour récupérer la position absolue, sim_handle_parent pour récupérer la position par rapport au parent de l'objet, ou un ObjectHandle relatif à la référence dont vous souhaitez la position. **OperationMode :** mode de fonctionnement de la fonction remoteAPI. Les modes de fonctionnement recommandés pour cette fonction sont simx_opmode_streaming (le premier appel) et simx_opmode_buffer (les appels suivants).
Valeurs de retour	**ReturnCode :** un code de retour de la fonction remoteAPI. **Position :** 3 valeurs représentant la position (x, y, z).

simxGetObjectVelocity

La description	Récupère la vitesse linéaire et angulaire d'un objet.
Prototype	[**Nombre returnCode,tableau linearVelocity,tableau angularVelocity**]=**simxGetObjectVelocity(nombre clientID,nombre objectHandle,nombre operationMode**).
Paramètres	**clientID :** l'ID client. Se référer à simxStart. **objectHandle :** handle de l'objet. **operationMode :** mode de fonctionnement d'une fonction remoteAPI. Les modes de fonctionnement recommandés pour cette fonction sont simx_opmode_streaming (le premier appel) et simx_opmode_buffer (les appels suivants).
Valeurs de retour	**returnCode :** un code de retour de fonction remoteAPI. **linearVelocity :** 3 valeurs représentant la vitesse linéaire (vx, vy, vz). **angularVelocity :** 3 valeurs représentant la vitesse angulaire (dAlpha, dBeta, dGamma).

Capteur de proximité simxRead

La description	Lit l'état d'un capteur de proximité.
Prototype	[**numéro returnCode,bool detectionState,array detectorPoint,numéro detectorObjectHandle,array detectorSurfaceNormalVector**]=**simxReadProximitySensor(numéro clientID,numéro sensorHandle,numéro operationMode**).
Paramètres	**clientID :** l'ID client. Se référer à simxStart. **sensorHandle :** poignée du capteur de proximité. **operationMode :** mode de fonctionnement d'une fonction remoteAPI. Les modes de fonctionnement recommandés pour cette fonction sont simx_opmode_streaming (le premier appel) et simx_opmode_buffer (les appels suivants).
Valeurs de retour	**returnCode :** un code de retour de fonction remoteAPI.

4.5.7 Transmission des commandes de Matlab vers V-REP

Une fois les commandes sont calculées dans Matlab ils seront transmis au robot pioneer P-3dx de VREP afin de réaliser les tâches de navigation en cherchant la cible et évitant de obstacles.
La fonction SimxSetJointTargetVelocity permet d'envoyer les commandes de Matlab vers V-REP qui sont :

- Vitesse linéaire de la roue droite du robot ;
- Vitesse linéaire de la roue gauche du robot.

SimxSetJointTargetVelocity :

La description	Définit la vitesse intrinsèque cible d'une articulation non sphérique.
Prototype	**[returnCode] = vrep.simxSetJointTargetVelocity (clientID,jointHandle, targetVelocity, operationMode)**
Paramètres	**ClientID :** l'ID du client. Se référer à simxStart. **JointHandle :** handle de l'articulation. **TargetVelocity :** vitesse cible de l'articulation (vitesse linéaire ou angulaire selon le type de joint). **OperationMode :** mode de fonctionnement de la fonction remoteAPI. Les modes de fonctionnement recommandés pour cette fonction sont simx_opmode_oneshot ou simx_opmode_streaming.
Valeurs de retour	**ReturnCode :** un code de retour de la fonction remoteAPI.

L'organigramme ci-dessous (Fiqure 4.13) résume les étapes de configuration de simulation afin de réaliser une navigation virtuelle d'un robot mobile.

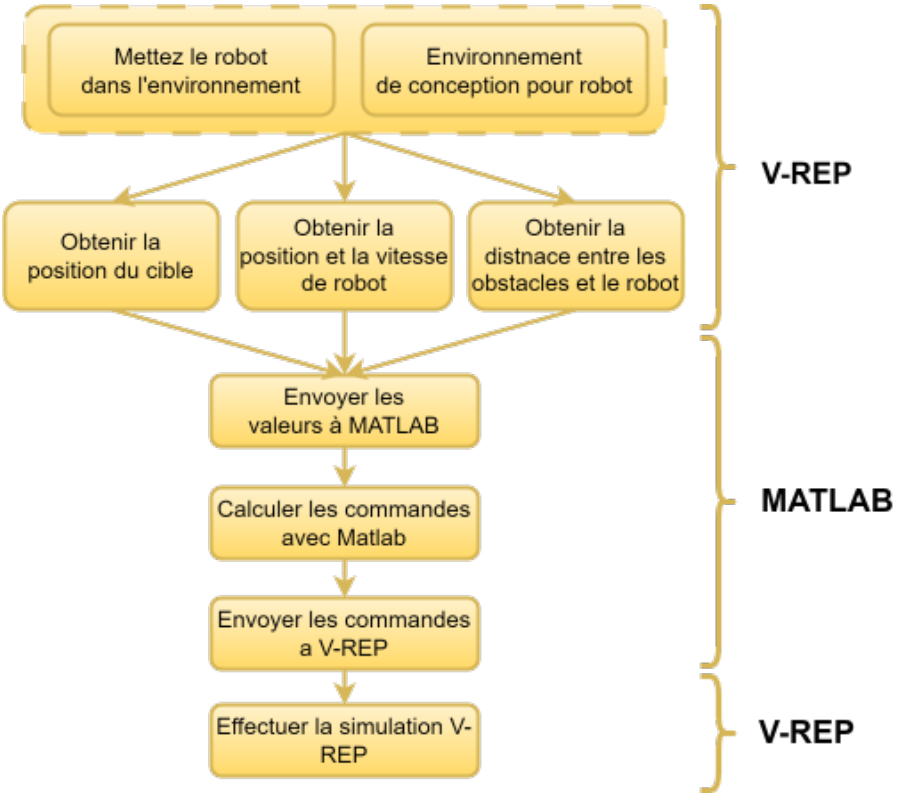


FIGURE 4.13 – Organigramme de de transfert de données MATLAB-V-REP.

4.6 Conclusion

Dans ce chapitre nous avons présenté l'histoire de la réalité virtuelle et ces différentes définitions présenter par ces précurseurs. Ensuite nous avons présenté la relation entre la réalité virtuelle et la robotique et cela par diverses technologies permettant l'immersion dans des environnements virtuels en utilisant différents simulateurs de conception en citant (Gazebo,vrep,unity. . .).
Dans notre thèse nous avons travaillé avec l'environnement de simulation V-REP, nous avons présenté ces avantages, par la suite nous avons présenté les étapes nécessaires pour connecter MATLAB et V-REP en utilisant « Remote API ». Enfin nous avons définie des différentes fonctions utilisées dans notre thèse afin de réaliser la navigation virtuelle d'un robot mobile pioneer P-3dx en utlisant Matlab et V-REP.
Dans le chapitre qui suit on va présenter les résultats de simulations virtuel et discussion des résultats.

Chapitre 5

Résultats de simulations et discussions

5.1 Introduction

Afin de réaliser la navigation virtuelle d'un robot mobile Pioneer P-3dx dans le but de chercher une cible ou plusieurs cibles tout en évitant des obstacles, nous avons utilisé la plateforme d'expérimentation de robots virtuels (V-REP) comme simulateur physique, qui fournit un environnement adéquat avec notre propre plateforme virtuelle. Le V-REP comprend de nombreux robots, objets, structures, actionneurs et capteurs.
Les simulations que nous présenterons dans ce chapitre visent à évaluer les performances des contrôleurs utilisés pour effectuer la navigation d'un robot mobile Pioneer P-3dx dans des environnements différents grâce à une Co-simulation entre V-REP et Matlab. Au premier lieu nous avons fait une comparaison entre le contrôleur cinématique dans un environnement de navigation sans obstacles pour chercher une cible puis plusieurs cibles et le contrôleur cinématique en hybride avec l'algorithme d'optimisation par essaim de particule (PSO) dans le même environnement. Deuxièmement, nous avons fait une comparaison entre le contrôleur ANFIS dans un environnement sans obstacles pour chercher une cible puis plusieurs cibles et le contrôleur ANFIS dans le même environnement sans obstacles avec l'algorithme PSO. Troisièmement, nous avons fait une comparaison entre le contrôleur ANFIS dans un environnement avec obstacles pour chercher une cible et le contrôleur ANFIS dans le même environnement avec l'algorithme PSO. Cette comparaison est faite en termes de précision, de robustesse et de temps de calcul.

5.2 Navigation virtuelle du robot mobile Pioneer P-3dx

Nous avons conçu sur V-REP trois environnements virtuels de simulation afin de tester nos algorithmes en tenant compte des contraintes physiques réelles du robot. Le premier consiste à naviguer le robot d'un point de départ vers une cible sans obstacles, le deuxième a pour objectif de naviguer le robot d'un point de départ vers plusieurs cibles sans obstacles, alors que le dernier est de naviguer d'un point de départ vers une cible en évitant plusieurs obstacles. Les paramètres du robot, des régulateurs sont déclarés dans le programme Matlab.
La méthode de navigation proposée qui utilise le contrôle hybride ANFIS-PSO est structurée comme suit :

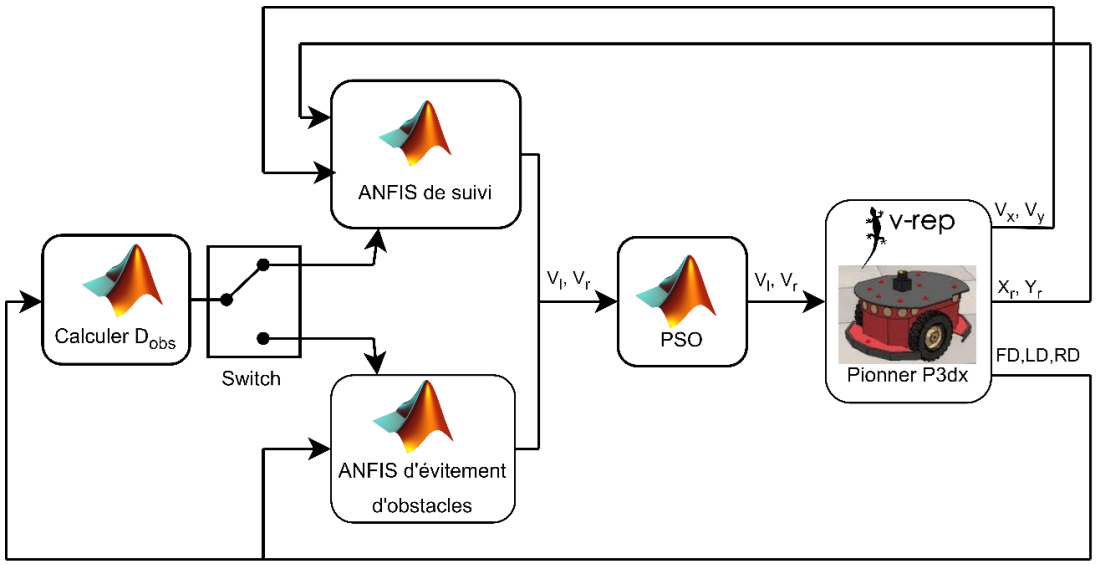


FIGURE 5.1 – Diagramme de simulation virtuelle globale.

L'organigramme de l'algorithme de navigation virtuel est représenté dans la section (3.8.4) du chapitre 3.

5.3 Simulation avec contrôleur cinématique sans obstacles

Pour bien se familiariser avec la plateforme virtuelle V-REP nous avons effectué des tests de simulation en rapport à la stratégie de commande cinématique développée au chapitre 2 (section 2.3), de telle sorte que le robot mobile doit naviguer vers une cible puis vers plusieurs cibles dans un environnement sans obstacles. Au premier lieu nous avons utilisé le contrôleur cinématique (Figure 5.2) et deuxièmement nous combiné le

contrôleur cinématique avec l'algorithme d'optimisation par essaim de particule (PSO) (Figure 5.3) développé au le chapitre 2 (section 3.8.3) pour montrer la réaction du robot mobile ainsi que le comportement des deux contrôleurs et l'efficacité de l'algorithme PSO.

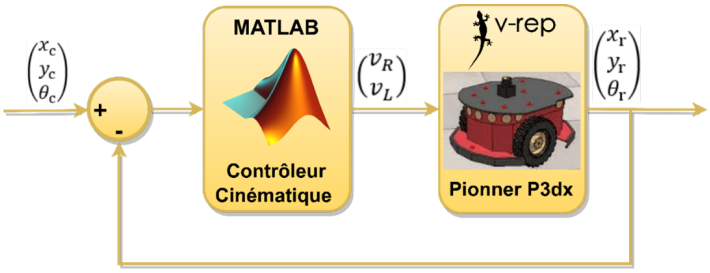


FIGURE 5.2 – Diagramme de simulation virtuelle par le contrôleur cinématique.

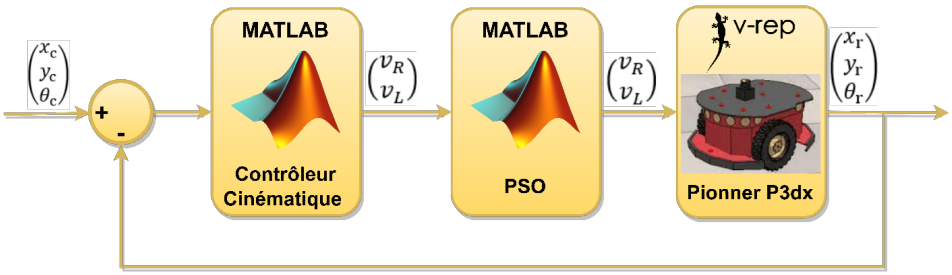


FIGURE 5.3 – Diagramme de simulation virtuelle par le contrôleur hybride cinématique-PSO.

5.3.1 Scénario 1

Dans cette simulation, le robot mobile Pioneer P-3dx doit naviguer vers une cible dans un environnement sans obstacles Figure (5.4) en utilisant d'abord le contrôleur cinématique, par la suite le contrôleur hybride Cinématique-PSO.

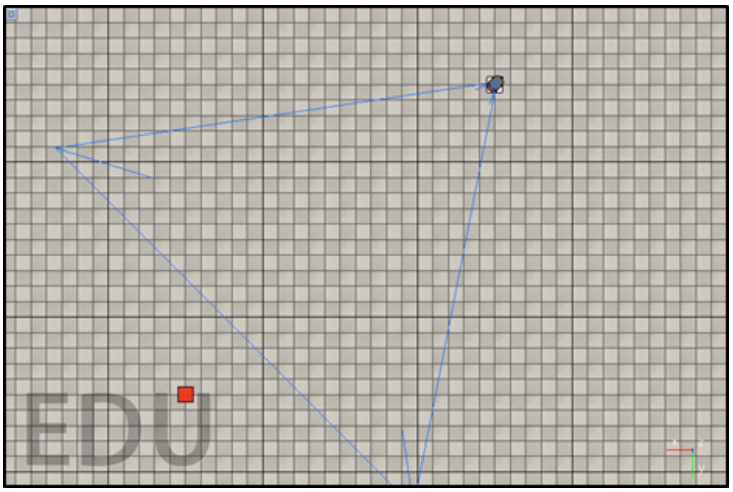


FIGURE 5.4 – Environnement V-REP de simulation virtuel vers une cible sans obstacles.

Les résultats de simulations sont représentés dans Les Figures ci-dessous :

Une seule cible sans PSO

La Figure (5.5) présente les comportements des erreurs de la trajectoire parcourue par le robot par rapport à une cible statique et des erreurs de vitesse par le contrôleur cinématique sans obstacles.

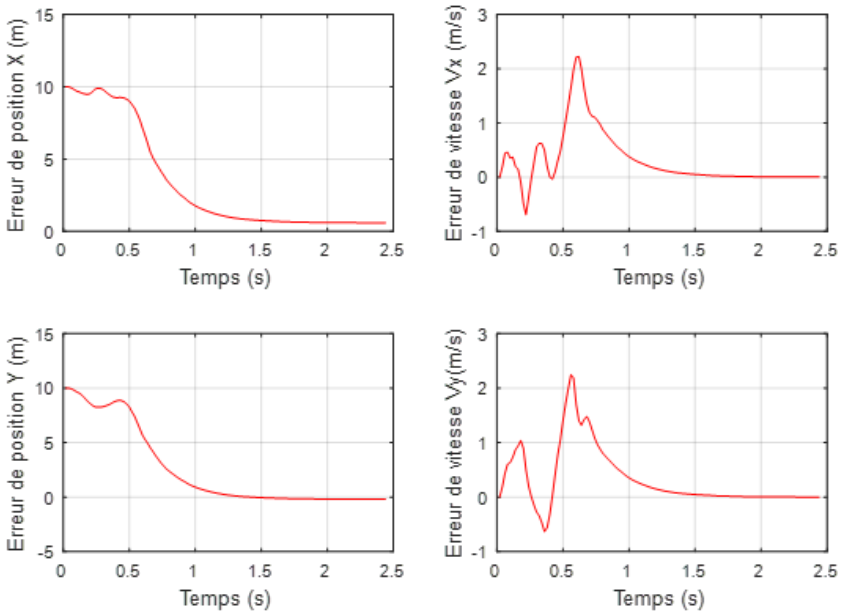


FIGURE 5.5 – Comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur cinématique sans obstacles.

La Figure (5.6) présente les comportements de poursuite et de vitesse par le contrôleur cinématique sans obstacles.

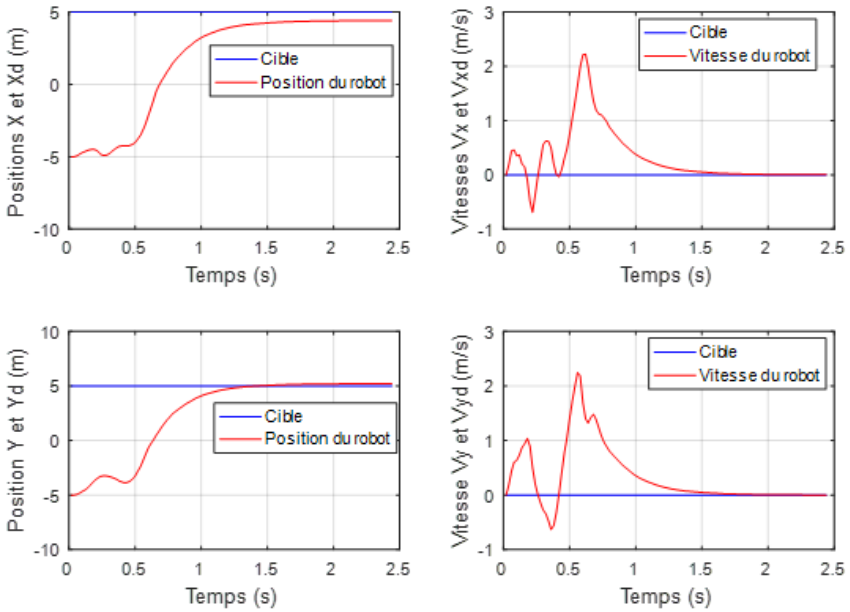


FIGURE 5.6 – Comportements de poursuite et de vitesse par le contrôleur cinématique sans obstacles.

La Figure (5.7) présente les comportements des vitesses des roues par le contrôleur cinématique sans obstacles.

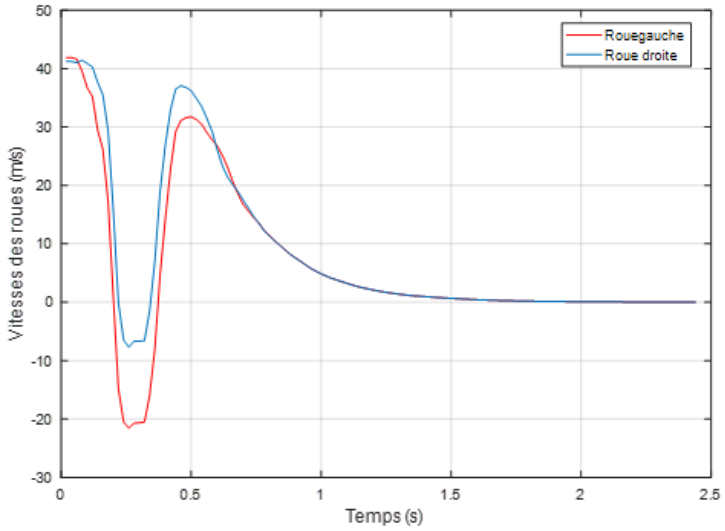


FIGURE 5.7 – Comportements des vitesses des roues par le contrôleur cinématique sans obstacles.

La Figure (5.8) présente le chemin parcouru par le robot par le contrôleur cinématique sans obstacles.

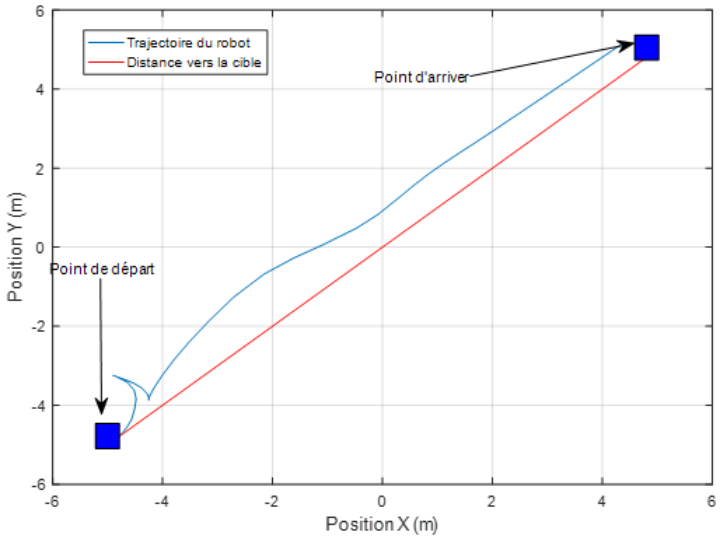


FIGURE 5.8 – Chemin parcouru par le robot par le contrôleur cinématique sans obstacles.

La Tableau (5.1) présente les statistiques des commandes envoyées au robot (Figure 5.8) par le contrôleur cinématique sans obstacles.

TABLEAU 5.1 – Statistiques des commandes envoyées au robot (Figure 5.8).

	Vitesse de roue gauche	**Vitesse de roue droite**
Min(m/s)	-21.55	-7.68
Max(m/s)	41.92	41.45
Moyenne(m/s)	6.406	8.335

Une seule cible avec PSO

La Figure (5.9) présente les comportements des erreurs de la trajectoire parcourue par le robot par rapport à une cible statique et des erreurs de vitesse par le contrôleur hybride cinématique-PSO sans obstacles.

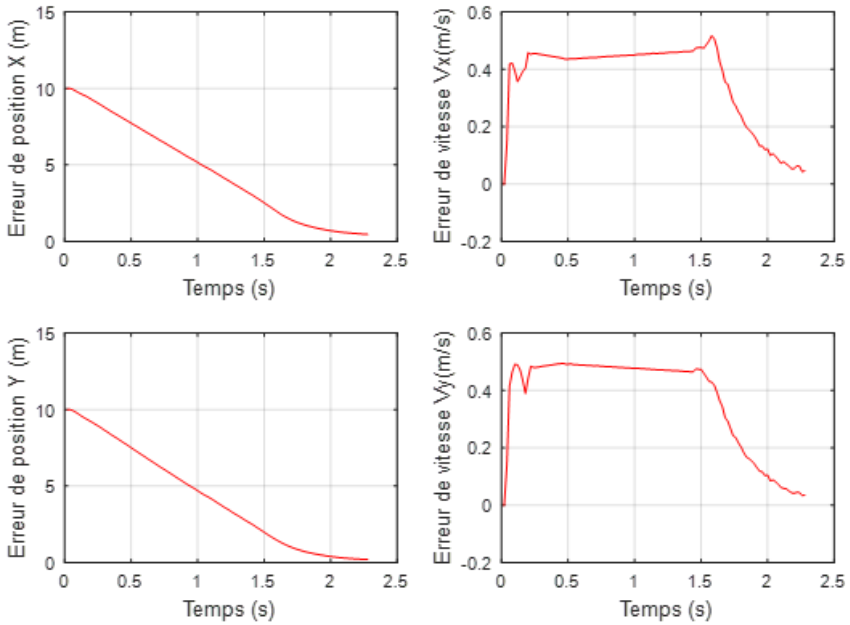


FIGURE 5.9 – Comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur hybride cinématique-PSO sans obstacles.

La Figure (5.10) présente les comportements de poursuite et de vitesse par le contrôleur hybride cinématique-PSO sans obstacles.

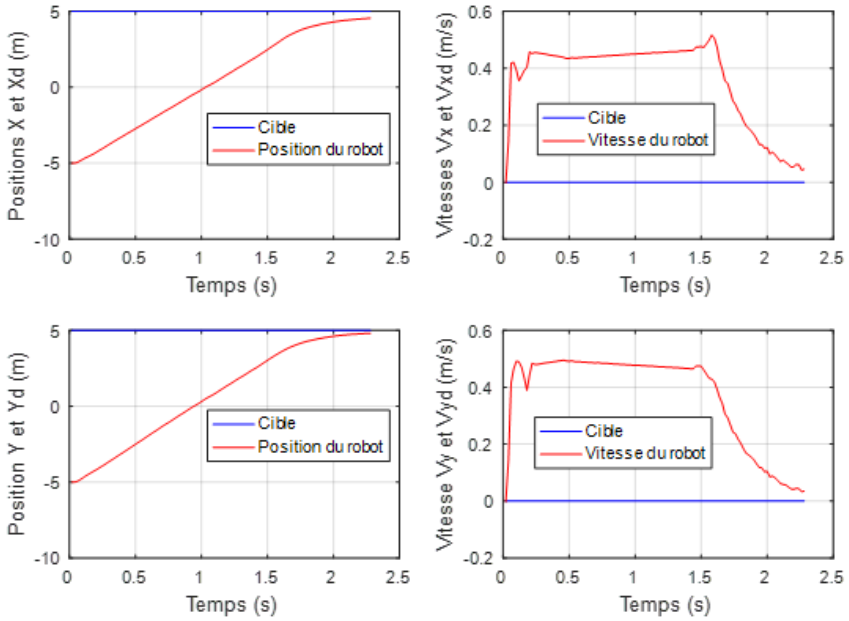


FIGURE 5.10 – Comportements de poursuite et de vitesse par le contrôleur hybride cinématique-PSO sans obstacles.

La Figure (5.11) présente les comportements des vitesses des roues par le contrôleur hybride cinématique-PSO sans obstacles.

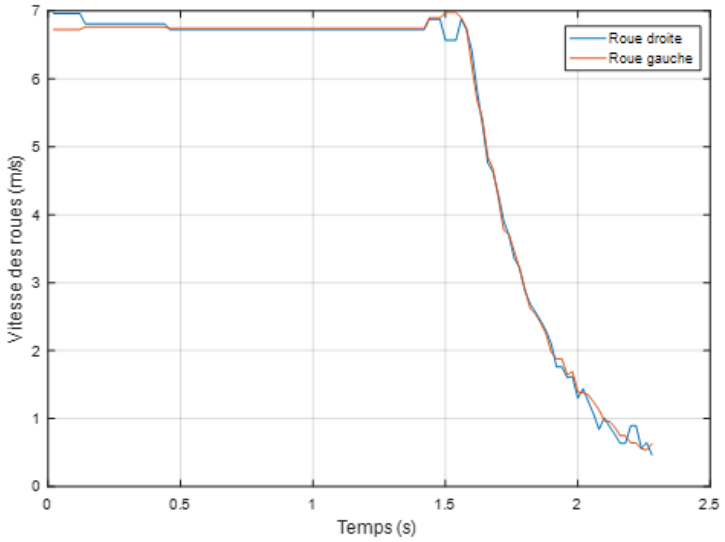


FIGURE 5.11 – Comportements des vitesses des roues par le contrôleur hybride cinématique-PSO sans obstacles.

La Figure (5.12) présente le Chemin parcouru par le robot par le contrôleur hybride cinématique-PSO sans obstacles.

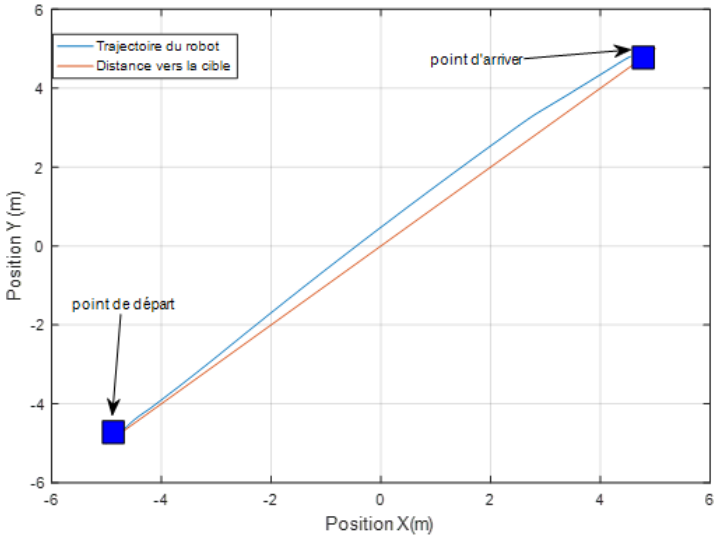


FIGURE 5.12 – Chemin parcouru par le robot par le contrôleur hybride cinématique-PSO sans obstacles.

La Tableau (5.2) présente les statistiques des commandes envoyées au robot (Figure 5.11).

TABLEAU 5.2 – Statistiques des commandes envoyées au robot (Figure 5.11).

	Vitesse de roue gauche	**Vitesse de roue droite**
Min(m/s)	0.5404	0.4673
Max(m/s)	6.975	6.961
Moyenne(m/s)	5.394	5.39

Ces résultats graphiques permettent de relever les caractéristiques dynamiques relatives aux erreurs en position et en vitesse entre les sorties et les entrées cartésiennes. On constate que les deux contrôleurs (cinématique et cinématique-PSO) naviguent avec succès vers la cible, malgré qu'avec le contrôleur cinématique la trajectoire du robot est affectée par une perturbation au démarrage puis il se stabilise pour atteindre la cible, alors que le contrôleur hybride cinématique-PSO nous a donné une très bonne trajectoire plus stable et plus rapide.
Les tableaux (5.1 et 5.2) nous permettent de comparer les valeurs minimale, maximale et moyenne des commandes appliquées au robot. On constate que les commandes développées par le contrôleur cinématique présentent des valeurs très élevés en valeur absolue ce qui provoquent des perturbations au démarrage, qui rendent le robot suit un chemin long. En revanche avec l'introduction de l'algorithme PSO les commandes respectent ces contraintes physiques décrites dans le chapitre 3 section (3.8.3) en terme de vitesse ce qui a aidé à améliorer le chemin en éliminant les perturbations du démarrage.

5.3.2 Scénario 2

Dans cette simulation, le robot mobile Pioneer P-3dx doit naviguer vers plusieurs cibles dans un environnement sans obstacles par exploitation du contrôleur cinématique puis le contrôleur hybride Cinématique-PSO. La Figure (5.13) présente l'nvironnement V-REP de simulation virtuel vers plusieurs cibles sans obstacles.

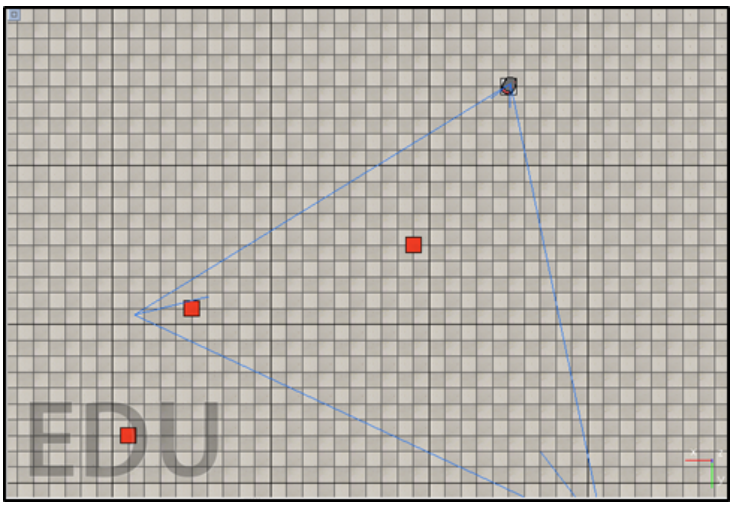


FIGURE 5.13 – Environnement V-REP de simulation virtuel vers plusieurs cibles sans obstacles.

Les résultats de simulations sont représentés dans Les figures ci-dessous :

Plusieurs cibles sans PSO

La Figure (5.14) présente les comportements des erreurs de la trajectoire parcourue par le robot par rapport à chaque cible statique et des erreurs de vitesse par le contrôleur cinématique sans obstacles.

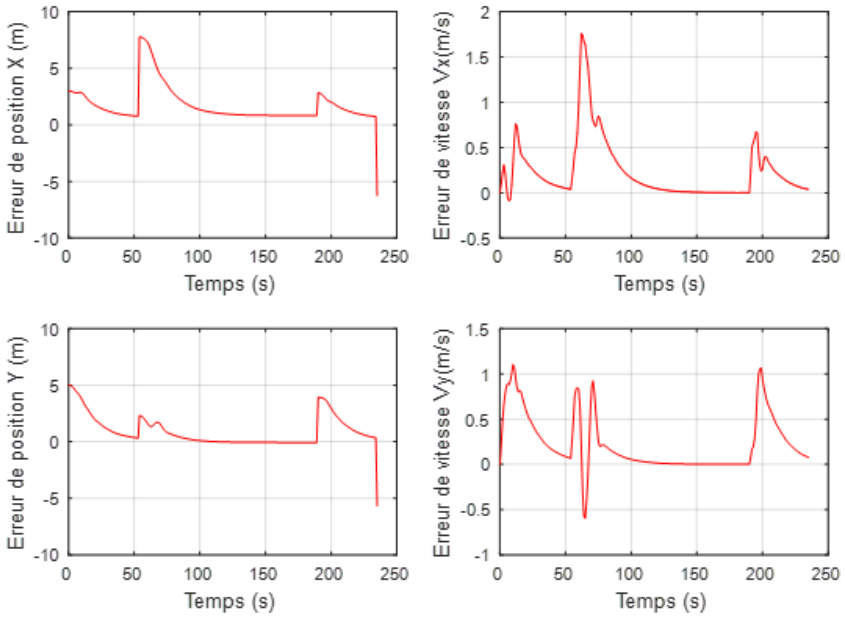


FIGURE 5.14 – Comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur cinématique sans obstacles.

La Figure (5.15) présente les comportements de poursuite et de vitesse par le contrôleur cinématique sans obstacles.

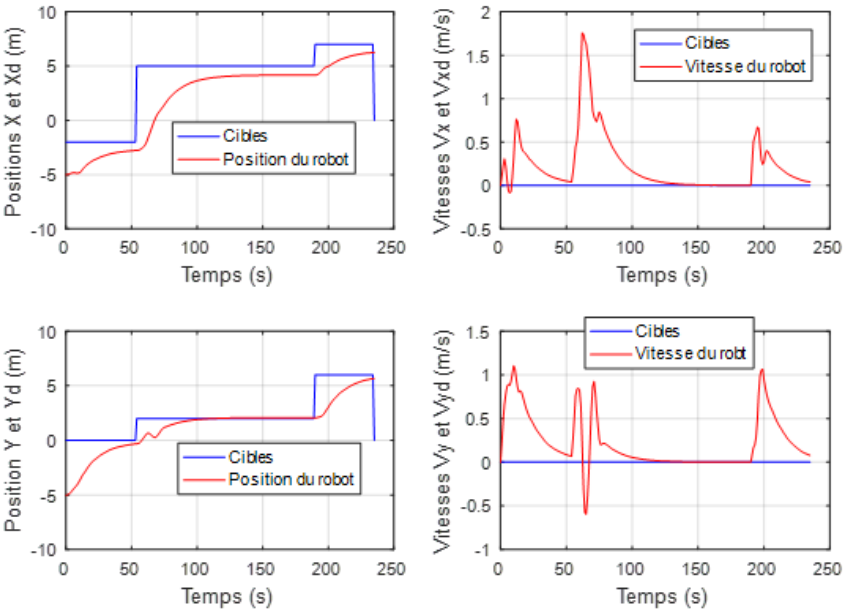


FIGURE 5.15 – Comportements de poursuite et de vitesse par le contrôleur cinématique sans obstacles.

La Figure (5.16) présente les comportements des vitesses des roues par le contrôleur cinématique sans obstacles.

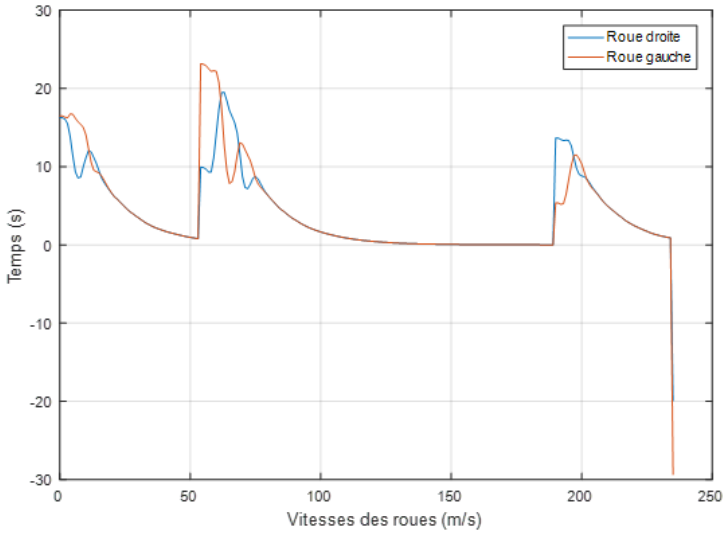


FIGURE 5.16 – Comportements des vitesses des roues par le contrôleur cinématique sans obstacles.

La Figure (5.17) présente les chemin parcouru par le robot par le contrôleur cinématique sans obstacles.

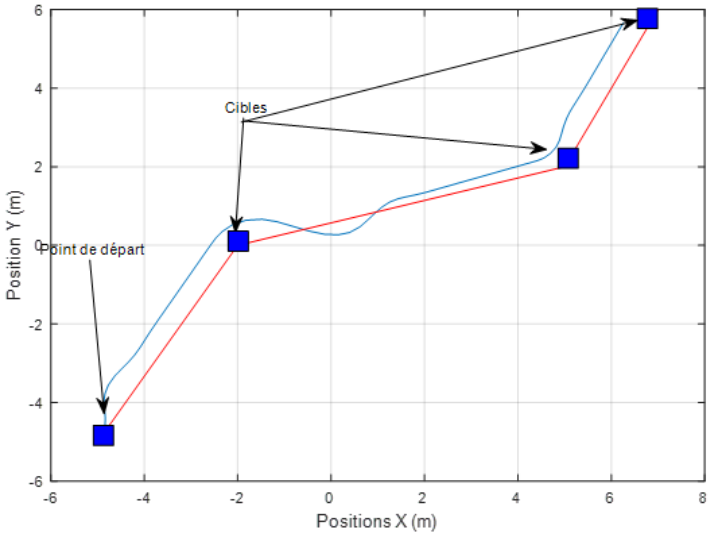


FIGURE 5.17 – Chemin parcouru par le robot par le contrôleur cinématique sans obstacles.

La Tableau (5.3) présente les statistiques des commandes envoyées au robot (Figure 5.16).

TABLEAU 5.3 – Statistiques des commandes envoyées au robot (Figure 5.16).

	Vitesse de roue gauche	**Vitesse de roue droite**
Min(m/s)	-29.36	-19.92
Max(m/s)	23.17	19.52
Moyenne(m/s)	4.228	4.039

Plusieurs cibles avec PSO

La Figure (5.18) présente les comportements des erreurs de la trajectoire parcourue par le robot par rapport à chaque cible statique et des erreurs de vitesse par le contrôleur hybride cinématique-PSO sans obstacles.

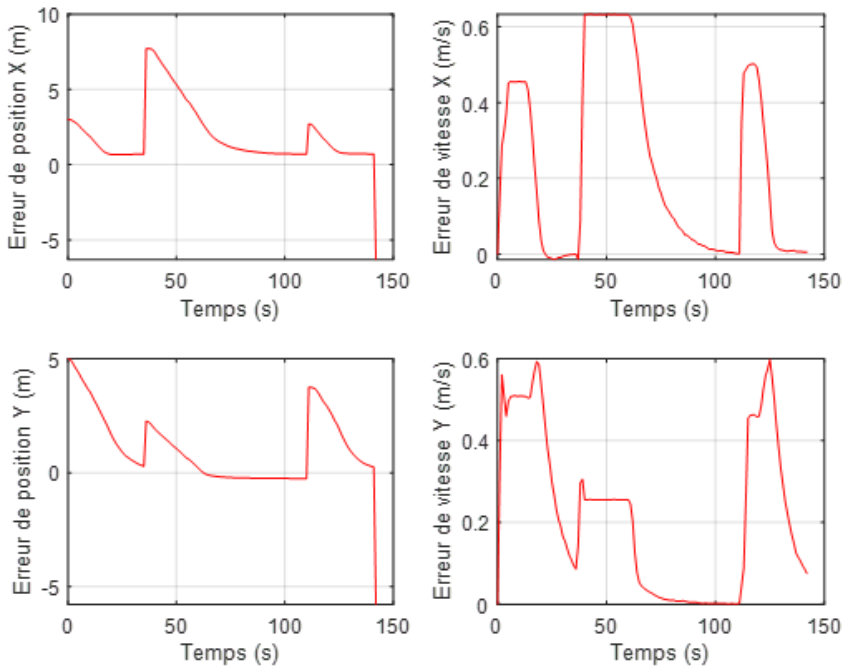


FIGURE 5.18 – Comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur hybride cinématique-PSO sans obstacles.

La Figure (5.19) présente les comportements de poursuite et de vitesse par le contrôleur hybride cinématique-PSO sans obstacles.

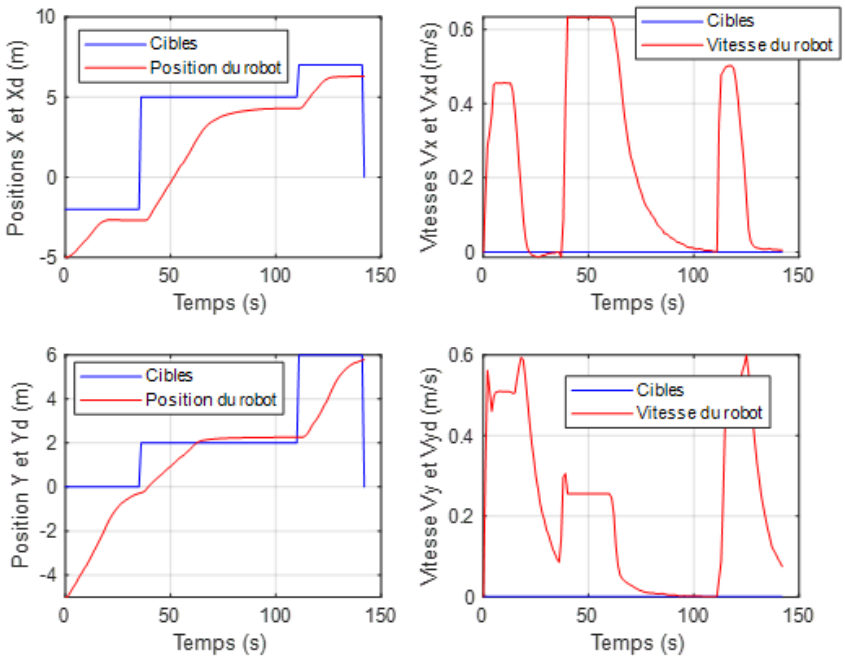


FIGURE 5.19 – Comportements de poursuite et de vitesse par le contrôleur hybride cinématique-PSO sans obstacles.

La Figure (5.20) présente les comportements des vitesses des roues par le contrôleur hybride cinématique-PSO sans obstacles.

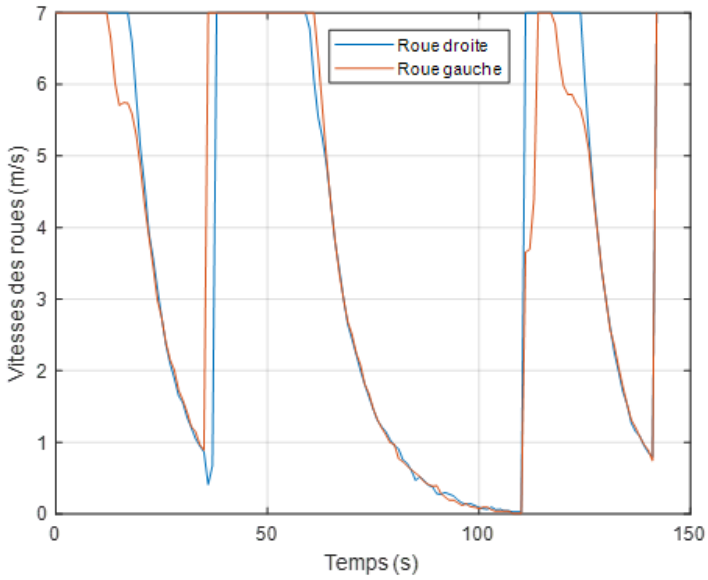


FIGURE 5.20 – Comportements des vitesses des roues par le contrôleur hybride cinématique-PSO sans obstacles.

La Figure (5.21) présente le chemin parcouru par le robot par le contrôleur hybride cinématique-PSO sans obstacles.

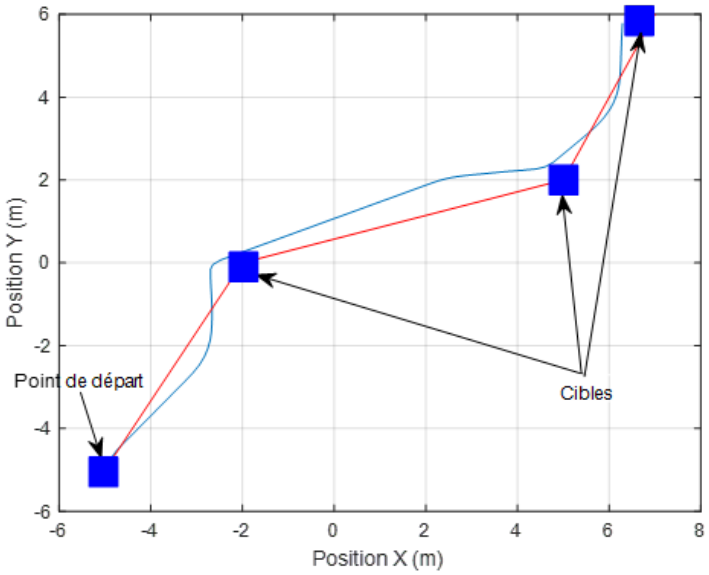


FIGURE 5.21 – Chemin parcouru par le robot par le contrôleur hybride cinématique-PSO sans obstacles.

La Tableau (5.4) présente les statistiques des commandes envoyées au robot (Figure 5.20).

TABLEAU 5.4 – Statistiques des commandes envoyées au robot (Figure 5.20).

	Vitesse de roue gauche	**Vitesse de roue droite**
Min(m/s)	0.2265	0.2096
Max(m/s)	7	7
Moyenne(m/s)	4.253	4.298

À partir des Figures ci-dessus, on peut déduire que les deux contrôleurs (cinématique et cinématique-PSO) naviguent avec succès vers plusieurs cibles, malgré qu'avec le contrôleur cinématique le profil de la trajectoire du robot présente des perturbations au démarrage de chaque point puis il se stabilise pour atteindre les cibles, alors que le contrôleur hybride cinématique-PSO nous a donné une très bonne trajectoire plus stable et plus rapide. Ceci s'explique physiquement par le fait que l'algorithme PSO optimise les commandes du contrôleur cinématique pour que le chemin parcouru par le robot soit optimal.
Les tableaux (5.3 et 5.4) nous permettent de comparer les valeurs minimale, maximale et moyenne des commandes appliquées au robot. On constate que les commandes du contrôleur cinématique présentent des pics très élevées en valeur absolue ce qui provoquent des perturbations au démarrage dans chaque point, qui rendent parfois le chemin parcouru par le robot est long. Par avec l'introduction de l'algorithme PSO, les commandes respectent ces contraintes physiques décrites dans le chapitre 3 section (3.8.3) en terme de vitesse ce qui a aidé à améliorer le chemin en éliminant les perturbations du démarrage.

5.3.3 Etude comparatif

La Tableau (5.5) présente des valeurs comparatives pour la simulation par les contrôleurs cinématique et cinématique-PSO sans obstacles.

Tableau 5.5 – Tableau comparatif pour la simulation par les contrôleurs cinématique et cinématique-PSO sans obstacles.

Nombre de Cibles	Paramètre de comparaison	Sans PSO	Avec PSO
Une cible	**Temps (s)**	2.44	2.28
	Erreur de position X (m)	0.5892	0.4504
	Erreur de position Y (m)	- 0.1946	0.1945
	Erreur de Vitesse X (m/s)	0.0006	0.0460
	Erreur de Vitesse Y (m/s)	0.0006	0.0350
	Distance a la cible (m)	0.6206	0.4907
	Distance parcourue (m)	15.7446	13.7051
Plusieurs Cibles	**Temps (s)**	235	142
	Erreur de position X (m)	0.7532	0.7099
	Erreur de position Y (m)	0.3587	0.2466
	Erreur de Vitesse X (m/s)	0.0390	0.0058
	Erreur de Vitesse Y (m/s)	0.0782	0.0746
	Distance a la cible (m)	0.8187	0.7411
	Distance parcourue (m)	17.4954	17.5534

Le (Tableau 5.5) ci-dessus permet de réaliser une étude comparative des performances en terme de temps, précision, rapidité et distance entre les deux contrôleurs cinématique et cinématique- PSO dans un environnement sans obstacles. D'après le (Tableau 5.5) on peut remarquer que l'utilisation du contrôleur hybride cinématique-PSO dans la stratégie de commande du robot mobile pour atteindre les cibles, permet de constater que la distance parcourue et le temps nécessaire pour le chemin à parcourir sont faibles, comparativement à l'utilisation du contrôleur cinématique seul dans la stratégie de commande. Par ailleurs les profils des erreurs de position et de vitesse montrent bien que le robot atteint les cibles avec une précision meilleure par le contrôleur hybride.

5.4 Simulation avec contrôleur ANFIS sans obstacles

L'objectif de cette simulation est de réaliser la navigation virtuelle d'un robot mobile Pioneer P-3dx en utilisant deux contrôleurs intelligents ANFIS (Figure 5.22) et le contrôleur hybride ANFIS–PSO (Figure 5.23), Et comparer le comportement des deux contrôleurs. Pour ce faire nous avons considéré deux scénarios.

Dans le premier, le robot mobile doit naviguer vers une cible dans un environnement sans obstacles (Figure 5.24), dans le deuxième le robot doit naviguer vers plusieurs cibles dans un environnement sans obstacles (Figure 5.33).

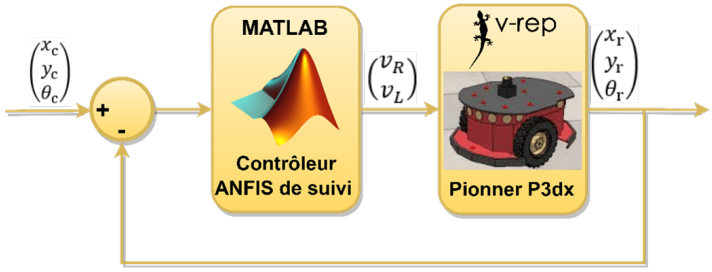


FIGURE 5.22 – Diagramme de simulation virtuelle par le contrôleur ANFIS.

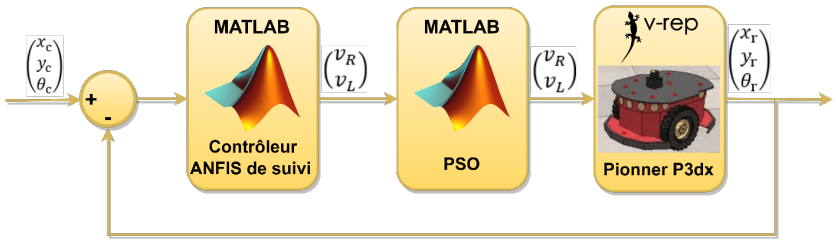


FIGURE 5.23 – Diagramme de simulation virtuelle par le contrôleur hybride ANFIS-PSO.

5.4.1 Scénario 1

Dans cette simulation, le robot mobile doit naviguer vers une cible dans un environnement sans obstacles par le contrôleur ANFIS puis par le contrôleur hybride ANFIS-PSO.

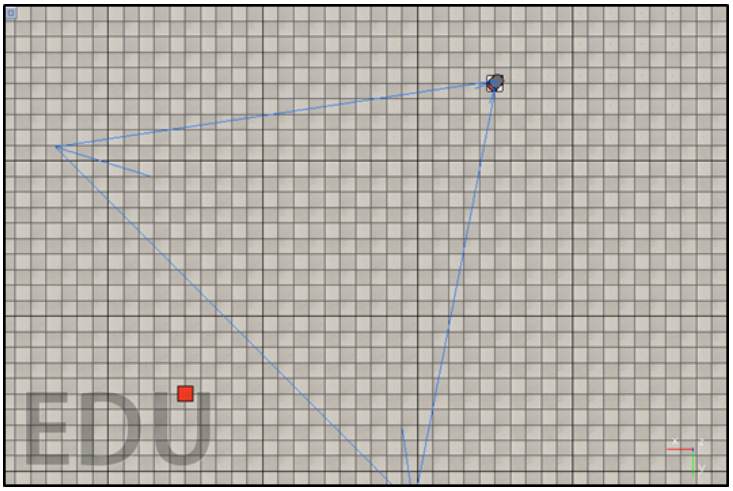


FIGURE 5.24 – Environnement V-REP de simulation virtuel vers une cible sans obstacles.

Les résultats de simulations sont représentés dans les figures ci-dessous :

Une seule cible sans PSO

La Figure (5.25) présente les comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur ANFIS sans obstacles.

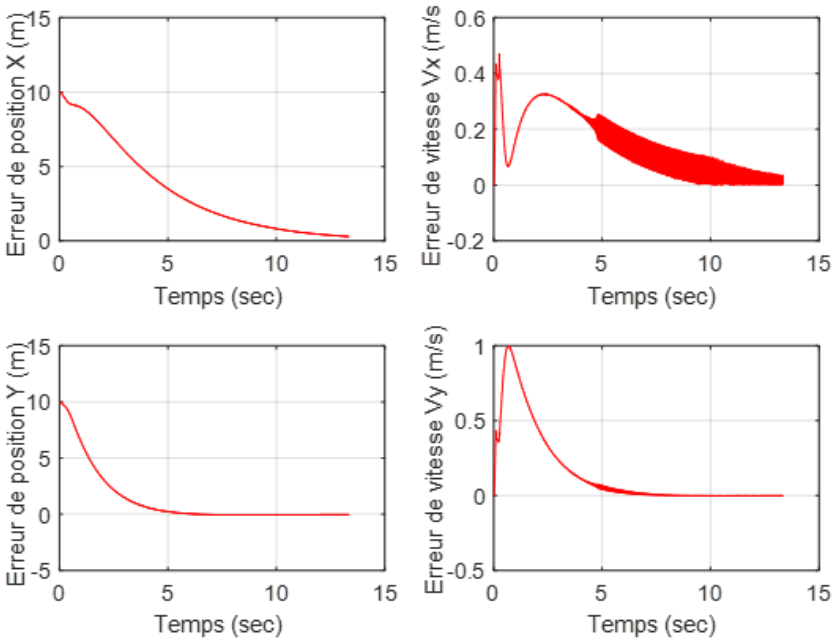


FIGURE 5.25 – Comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur ANFIS sans obstacles.

La Figure (5.26) présente les comportements de poursuite et de vitesse par le contrôleur ANFIS sans obstacles.

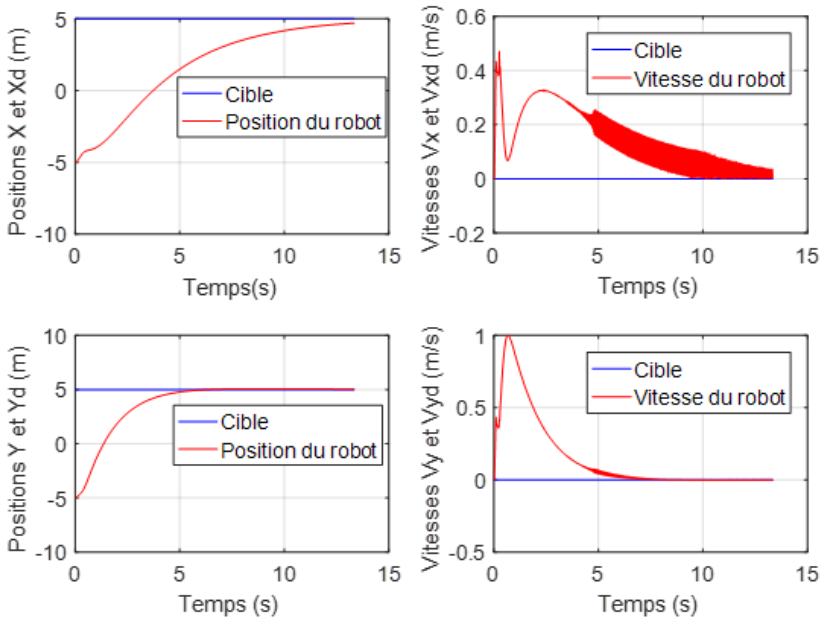


FIGURE 5.26 – Comportements de poursuite et de vitesse par le contrôleur ANFIS sans obstacles.

La Figure (5.27) présente les comportements des vitesses des roues par le contrôleur ANFIS sans obstacles.

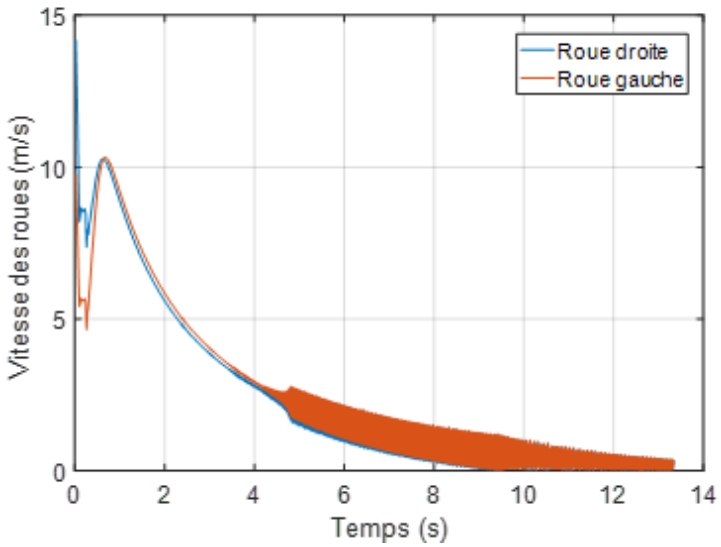


FIGURE 5.27 – Comportements des vitesses des roues par le contrôleur ANFIS sans obstacles.

La Figure (5.28) présente le chemin parcouru par le robot par le contrôleur ANFIS sans obstacles.

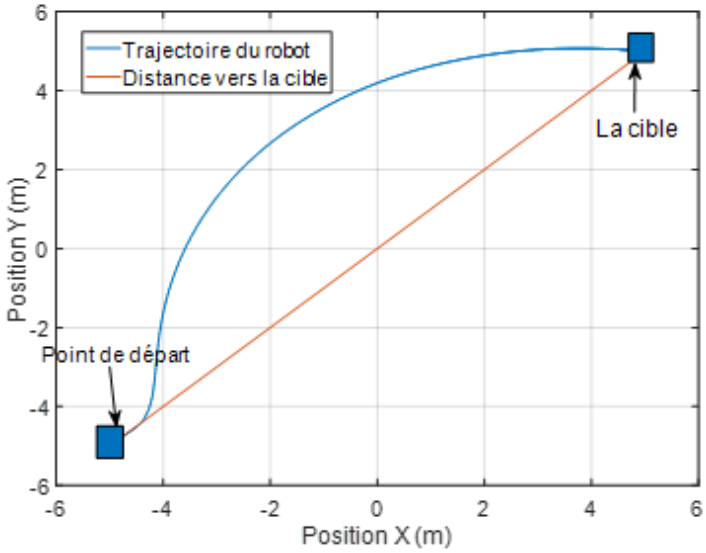


FIGURE 5.28 – Chemin parcouru par le robot par le contrôleur ANFIS sans obstacles.

La Tableau (5.6) présente les Statistiques des commandes envoyées au robot (Figure 5.27).

TABLEAU 5.6 – Statistiques des commandes envoyées au robot (Figure 5.27).

	Vitesse de roue gauche	**Vitesse de roue droite**
Min(m/s)	0.000441	0.001482
Max(m/s)	10.32	14.18
Moyenne(m/s)	2.481	2.492

Une seule cible avec PSO

La Figure (5.29) présente les comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur hybride ANFIS-PSO sans obstacles.

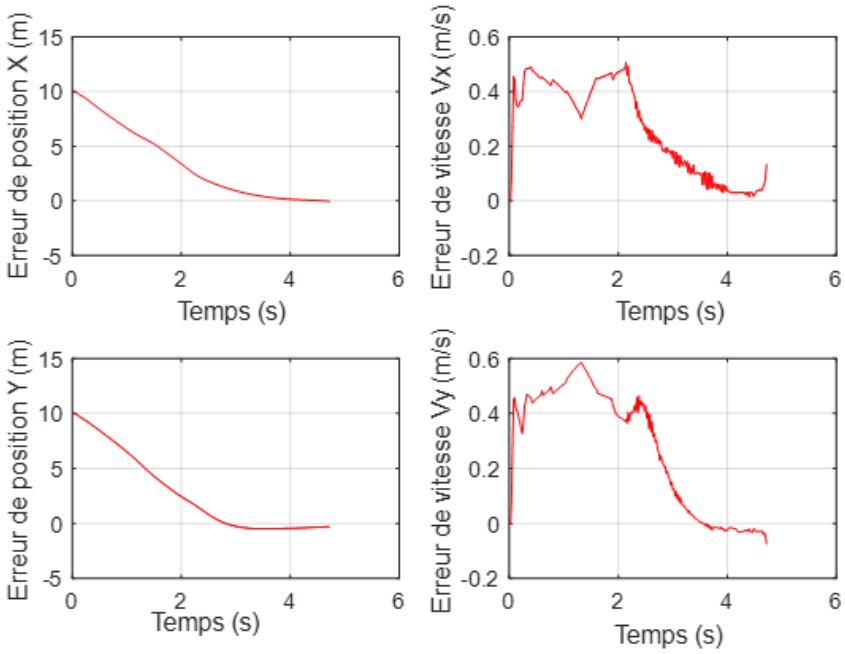


FIGURE 5.29 – Comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur hybride ANFIS-PSO sans obstacles.

La Figure (5.30) présente les comportements de poursuite et de vitesse par le contrôleur hybride ANFIS-PSO sans obstacles.

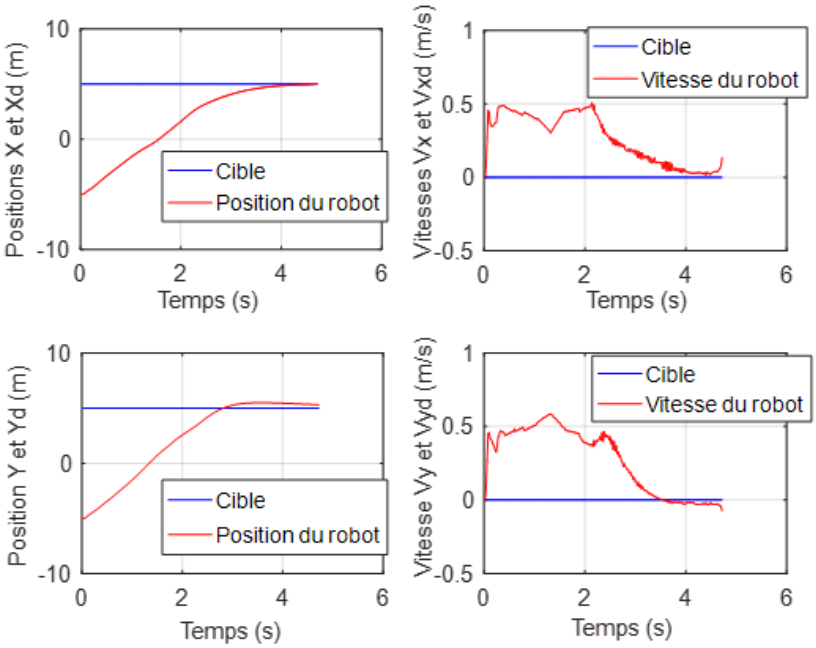


FIGURE 5.30 – Comportements de poursuite et de vitesse par le contrôleur hybride ANFIS-PSO sans obstacles.

La Figure (5.31) présente les comportements de poursuite et de vitesse par le contrôleur hybride ANFIS-PSO sans obstacles.

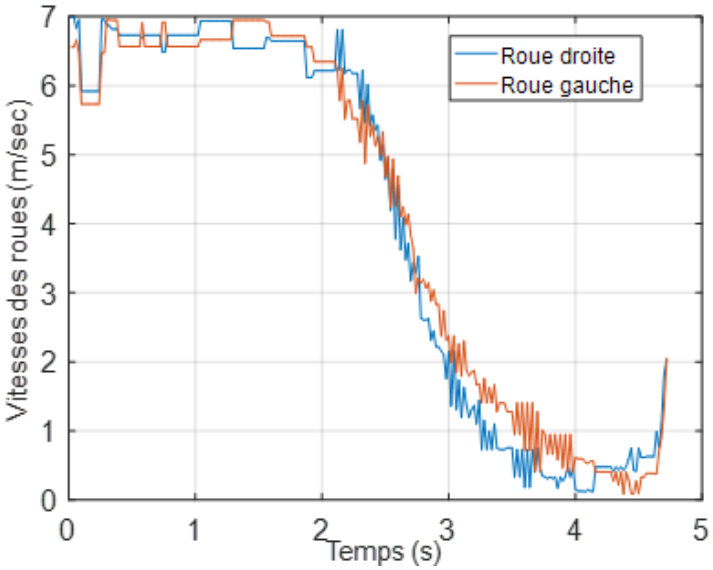


FIGURE 5.31 – Comportements de poursuite et de vitesse par le contrôleur hybride ANFIS-PSO sans obstacles.

La Figure (5.32) présente le chemin parcouru par le robot par le contrôleur hybride cinématique-PSO sans obstacles.

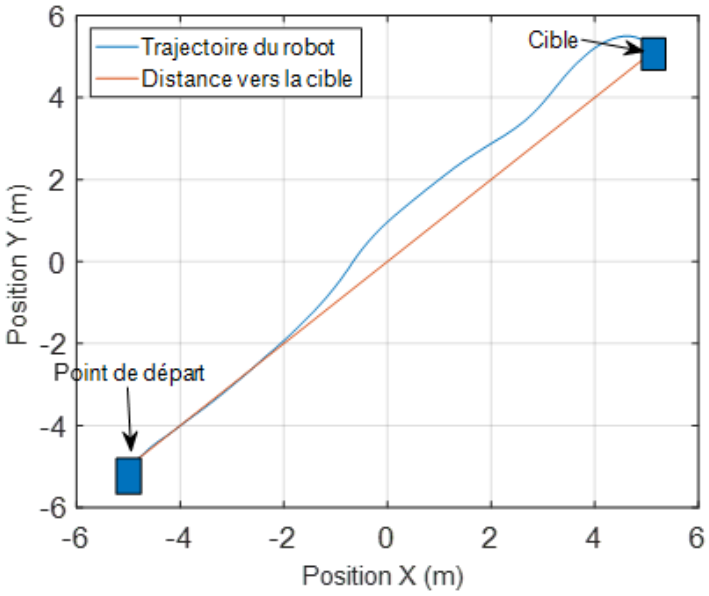


FIGURE 5.32 – Chemin parcouru par le robot par le contrôleur hybride cinématique-PSO sans obstacles.

La Tableau (5.7) présente les statistiques des commandes envoyées au robot (Figure 5.31).

TABLEAU 5.7 – Statistiques des commandes envoyées au robot (Figure 5.31).

	Vitesse de roue gauche	**Vitesse de roue droite**
Min(m/s)	0.08369	0.119
Max(m/s)	6.957	6.988
Moyenne(m/s)	4.11	4.015

À partir des Figures ci-dessus, on conclut après une période d'adaptation que : Les erreurs de suivi en position et en vitesse sont nettement inférieures à celles obtenues pour les commandes précédentes (cinématique et hybride cinématique-PSO). Ceci est dû au fait que les stratégies de commande à base du contrôleur ANFIS compensent les effets non linéaires. En parallèle, les deux contrôleurs (ANFIS et ANFIS-PSO) permettent de naviguer avec succès le robot mobile vers la cible dans un environnement sans obstacles, néanmoins avec le contrôleur ANFIS le robot suit une trajectoire longue pour atteindre la cible, alors que le contrôleur hybride ANFIS-PSO nous a donné une très bonne trajectoire plus stable et plus rapide. Ceci permet de confirmer la validité de l'algorithme proposé.
Les tableaux (5.6 et 5.7) nous permettent de comparer les valeurs minimale, maximale et moyenne des commandes appliquées au robot, nous avons remarqué que les commandes du contrôleur ANFIS atteint des pics élevés pour cette raison le robot suit un chemin long ; par contre avec l'introduction de l'algorithme PSO les commandes respectent ces contraintes physiques décrites dans le chapitre 3 section (3.8.3) en terme de vitesse qui

conduit le robot à suivre le chemin le plus court.

5.4.2 Scénario 2

Dans cette simulation, le robot mobile doit naviguer d'un point de départ vers plusieurs cibles dans un environnement sans obstacles par le contrôleur ANFIS puis par le contrôleur hybride ANFIS-PSO.

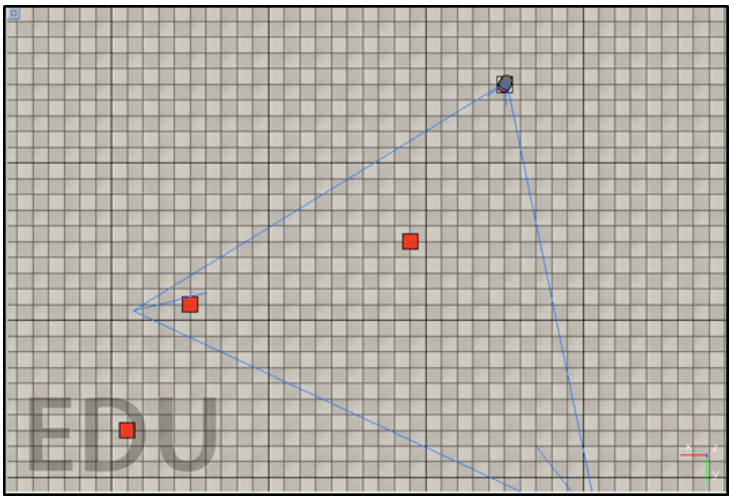


FIGURE 5.33 – Environnement V-REP de simulation virtuel vers plusieurs cibles sans obstacles.

Les résultats de simulations sont représentés dans Les figures ci-dessous :

Plusieurs cibles sans PSO

La Figure (5.34) présente les comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur ANFIS sans obstacles.

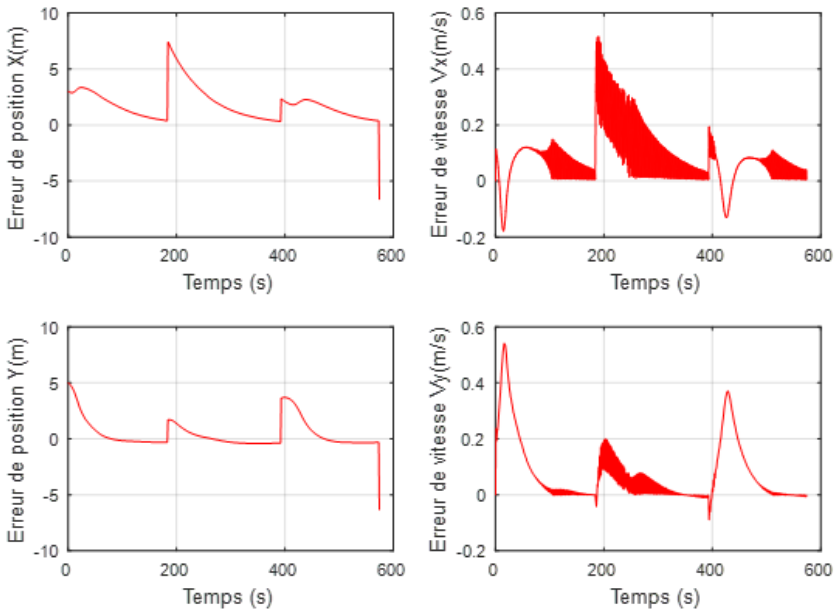


FIGURE 5.34 – Comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur ANFIS sans obstacles.

La Figure (5.35) présente les comportements de poursuite et de vitesse par le contrôleur ANFIS sans obstacles.

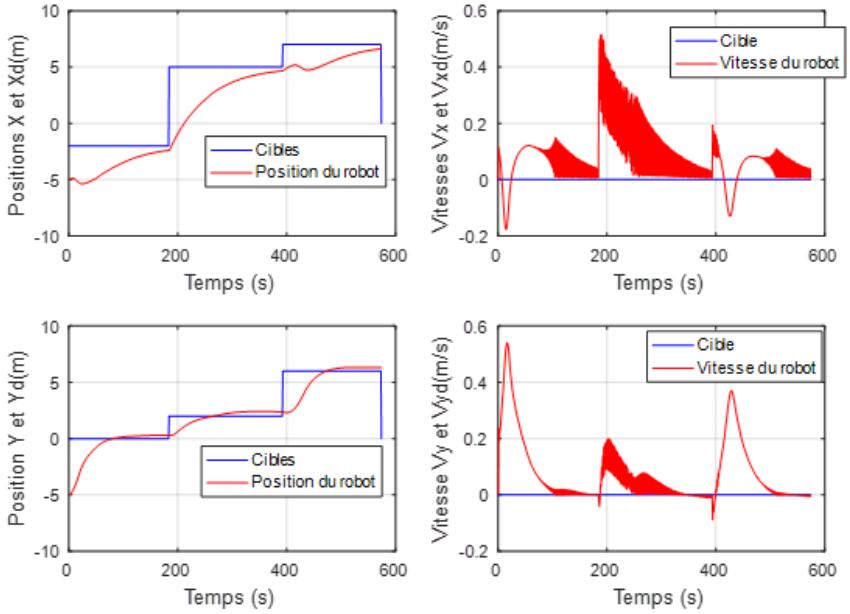


FIGURE 5.35 – Comportements de poursuite et de vitesse par le contrôleur ANFIS sans obstacles.

La Figure (5.36) présente les comportements des vitesses des roues par le contrôleur ANFIS sans obstacles.

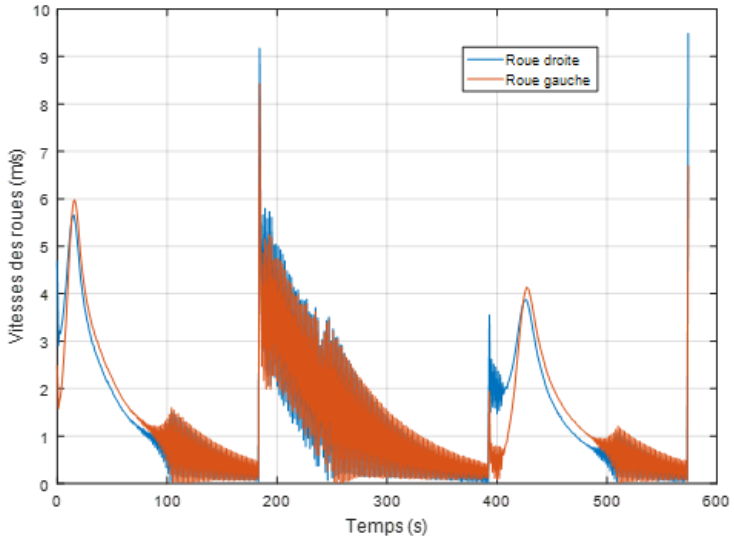


FIGURE 5.36 – Comportements des vitesses des roues par le contrôleur ANFIS sans obstacles.

La Figure (5.37) présente le chemin parcouru par le robot par le contrôleur ANFIS sans obstacles.

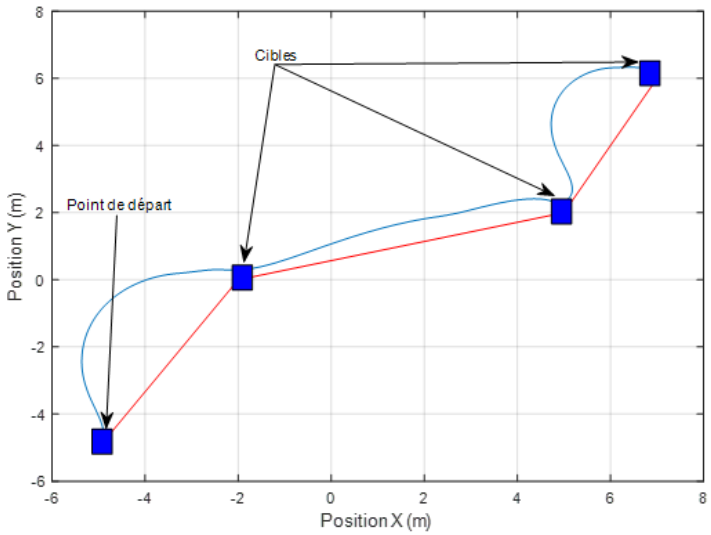


FIGURE 5.37 – Chemin parcouru par le robot par le contrôleur ANFIS sans obstacles.

La Tableau (5.8) présente les statistiques des commandes envoyées au robot (Figure 5.36).

TABLEAU 5.8 – Statistiques des commandes envoyées au robot (Figure 5.36).

	Vitesse de roue gauche	**Vitesse de roue droite**
Min(m/s)	0.006547	0.01784
Max(m/s)	8.428	9.488
Moyenne(m/s)	1.416	1.402

Plusieurs cibles avec PSO

La Figure (5.38) présente les comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur hybride ANFIS-PSO sans obstacles.

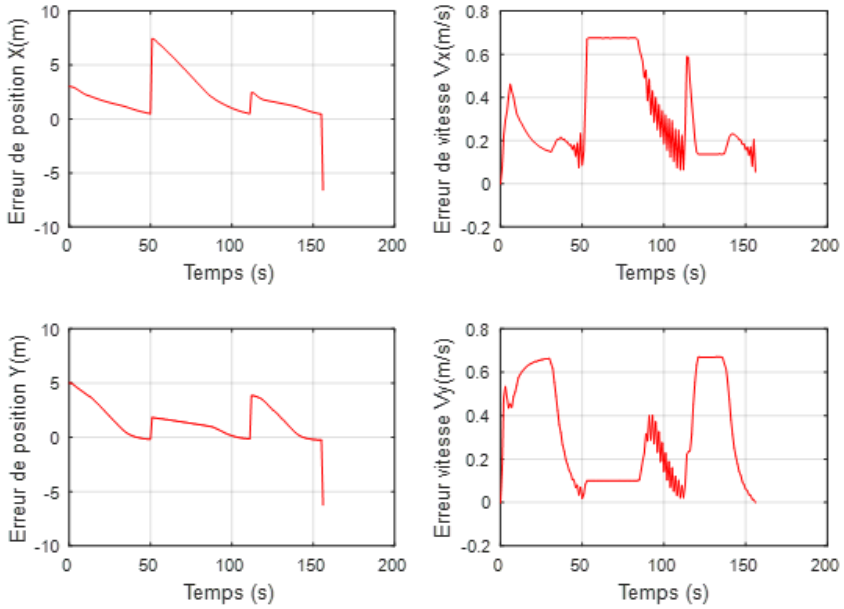


FIGURE 5.38 – Comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur hybride ANFIS-PSO sans obstacles.

La Figure (5.39) présente les comportements de poursuite et de vitesse par le contrôleur hybride ANFIS-PSO sans obstacles.

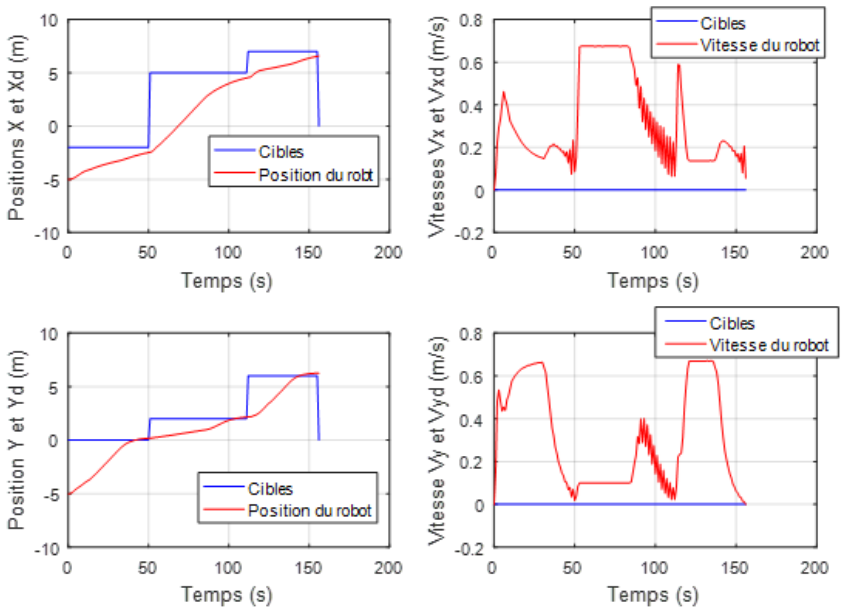


FIGURE 5.39 – Comportements de poursuite et de vitesse par le contrôleur hybride ANFIS-PSO sans obstacles.

La Figure (5.40) présente les comportements des vitesses des roues par le contrôleur hybride ANFIS-PSO sans obstacles.

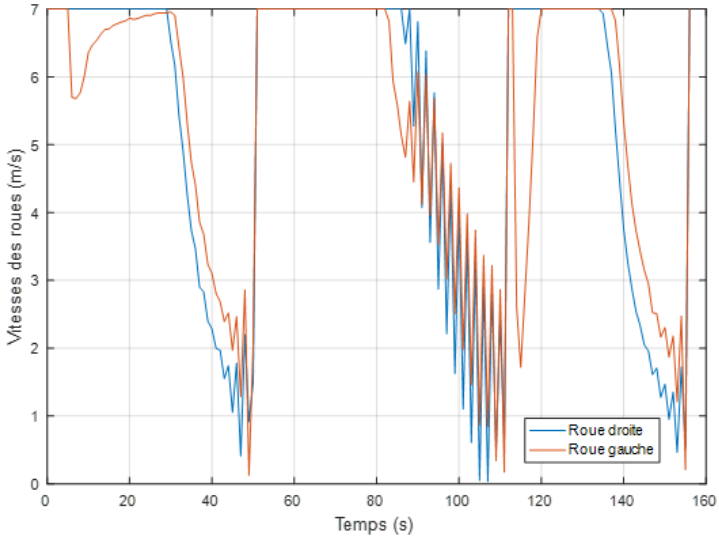


FIGURE 5.40 – Comportements des vitesses des roues par le contrôleur hybride ANFIS-PSO sans obstacles.

La Figure (5.41) présente le chemin parcouru par le robot par le contrôleur hybride ANFIS-PSO sans obstacles.

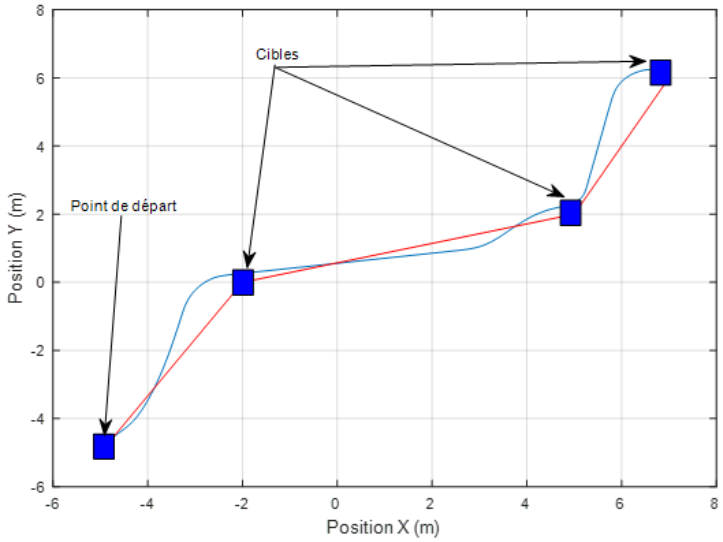


FIGURE 5.41 – Chemin parcouru par le robot par le contrôleur hybride ANFIS-PSO sans obstacles.

La Tableau (5.9) présente les statistiques des commandes envoyées au robot (Figure 5.40).

Tableau 5.9 – Statistiques des commandes envoyées au robot (Figure 5.40).

	Vitesse de roue gauche	**Vitesse de roue droite**
Min(m/s)	0.1275	0.03488
Max(m/s)	7	7
Moyenne(m/s)	5.341	5.304

À partir des Figures ci-dessus, les deux contrôleurs (ANFIS et ANFIS-PSO) naviguent avec succès vers différentes cibles dans un environnement sans obstacles, malgré qu'avec le contrôleur ANFIS le robot suit une trajectoire longue pour atteindre chaque cible, alors que le contrôleur hybride ANFIS-PSO nous a donné une très bonne trajectoire plus stable et plus rapide.
Les allures des vitesses présentent des oscillations aléatoires avec des amplitudes importantes à chaque fois que le robot se rapproche de la cible, pour la rejoindre avec une très bonne précession, par contre avec l'introduction du PSO les oscillations devient très faible.
Les tableaux (5.8 et 5.9) nous permettent de comparer les valeurs minimale, maximale et moyenne des commandes envoyer au robot, nous avons remarqué que les commandes du contrôleur ANFIS atteint des pics élevés pour cette raison le robot suit un chemin long ; par contre avec l'introduction de l'algorithme PSO les commandes respectent ces contraintes physiques décrites dans le chapitre 3 section (3.8.3) en terme de vitesse qui conduit le robot à suivre le chemin le plus court.

5.4.3 Etude comparatif

La Tableau (5.10) présente les valeurs comparatives pour la simulation par les contrôleurs ANFIS et ANFIS-PSO sans obstacles.

Tableau 5.10 – Tableau comparatif pour la simulation par les contrôleurs ANFIS et ANFIS-PSO sans obstacles.

Nombre de Cibles	Paramètre de comparaison	Sans PSO	Avec PSO
Une cible	**Temps (s)**	13.34	4.72
	Erreur de position X (m)	0.2967	- 0.0453
	Erreur de position Y (m)	- 0.0332	- 0.2907
	Erreur de Vitesse X (m/s)	0.0039	0.1340
	Erreur de Vitesse Y (m/s)	- 0.0002	- 0.0745
	Distance a la cible (m)	0.2987	0.2942
	Distance parcourue (m)	15.8394	14.8715
Plusieurs Cibles	**Temps (s)**	574	156
	Erreur de position X (m)	0.3875	0.4492
	Erreur de position Y (m)	-0.3193	- 0.2413
	Erreur de Vitesse X (m/s)	0.0060	0.0549
	Erreur de Vitesse Y (m/s)	-0.0014	- 0.0005
	Distance a la cible (m)	0.4952	0.4724
	Distance parcourue (m)	20.3744	18.3319

Le (Tableau 5.10) ci-dessus représente des résultats de simulation par deux contrôleurs ANFIS et ANFIS-PSO dans un environnement sans obstacles qui nous permet de réaliser une étude comparative en terme de temps, précision, rapidité et distance, pour montrer l'efficacité de chaque méthode.
D'après le (Tableau 5.10) nous avons remarqué que le temps nécessaire et la distance parcourue par le robot pour atteindre les cibles est moins avec le contrôleur hybride ANFIS-PSO, les erreurs de position et de vitesse montre bien que le robot atteint les cibles avec une précision meilleure par le contrôleur hybride.

5.5 Simulation avec contrôleur ANFIS Avec obstacles

Dans cette simulation, le robot mobile doit naviguer vers une cible dans un environnement avec obstacles, au premier lieu avec contrôleur ANFIS (Figure 5.22) et deuxièmement avec ANFIS en hybride avec l'algorithme optimisation par essaim de particule (PSO) (Figure 5.23) pour comparer le comportement des deux contrôleurs. La Figure (5.42) présente l'environnement V-REP de simulation virtuel vers une cible sans obstacles.

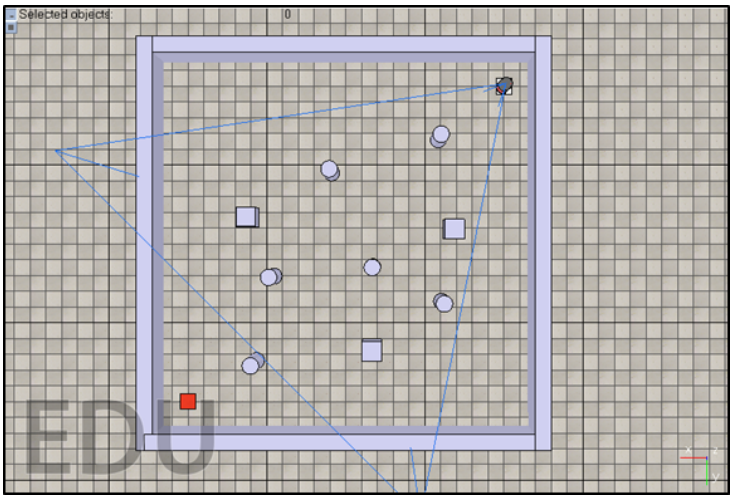


FIGURE 5.42 – Environnement V-REP de simulation virtuel vers une cible sans obstacles.

Les résultats de simulations sont représentés dans Les figures ci-dessous :

Une seule cible sans PSO

La Figure (5.43) présente les comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur ANFIS avec obstacles.

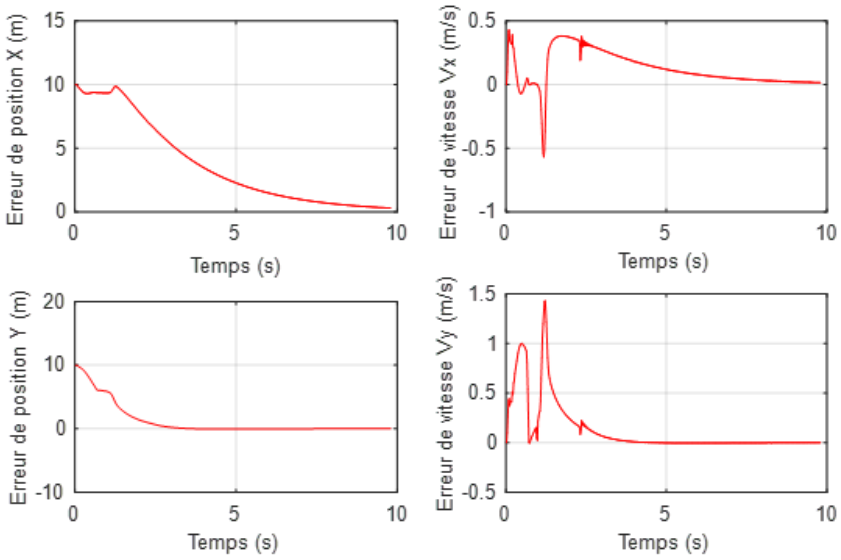


FIGURE 5.43 – Comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur ANFIS avec obstacles.

La Figure (5.44) présente les comportements de poursuite et de vitesse par le contrôleur ANFIS avec obstacles.

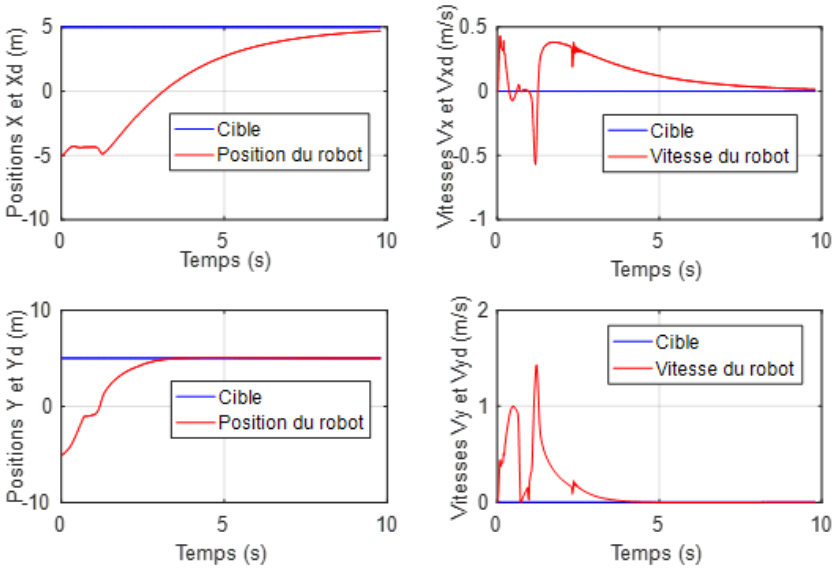


FIGURE 5.44 – Comportements de poursuite et de vitesse par le contrôleur ANFIS avec obstacles.

La Figure (5.45) présente les comportements des vitesses des roues par le contrôleur ANFIS avec obstacles.

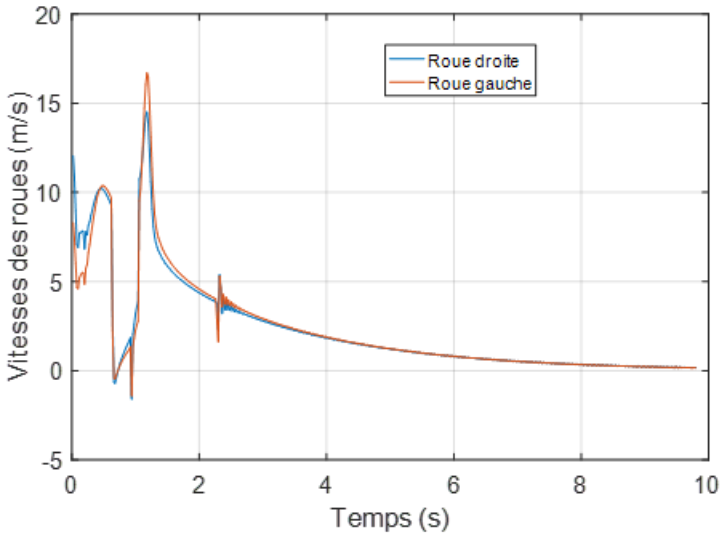


FIGURE 5.45 – Comportements des vitesses des roues par le contrôleur ANFIS avec obstacles.

La Figure (5.46) présente le chemin parcouru par le robot par le contrôleur ANFIS avec obstacles.

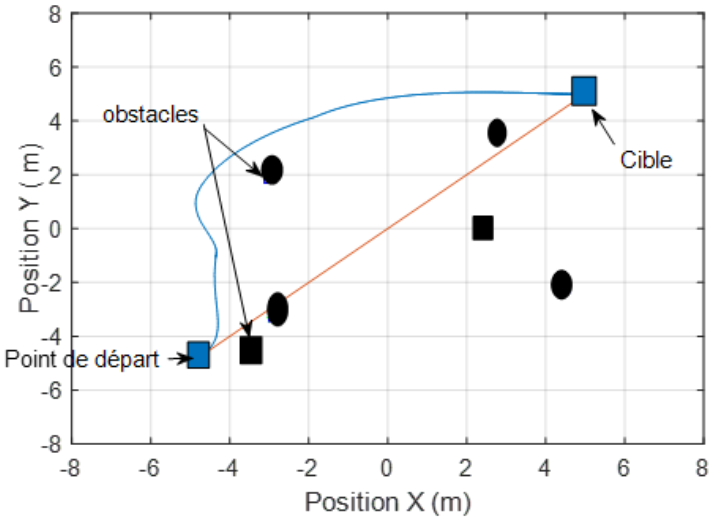


FIGURE 5.46 – Chemin parcouru par le robot par le contrôleur ANFIS avec obstacles.

La Tableau (5.11) présente les statistiques des commandes envoyées au robot (Figure 5.45).

TABLEAU 5.11 – Statistiques des commandes envoyées au robot (Figure 5.45).

	Vitesse de roue gauche	**Vitesse de roue droite**
Min(m/s)	-1.414	-1.586
Max(m/s)	16.69	14.5
Moyenne(m/s)	2.317	2.315

Une seule cible avec PSO

La Figure (5.47) présente les comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur hybride ANFIS-PSO avec obstacles.

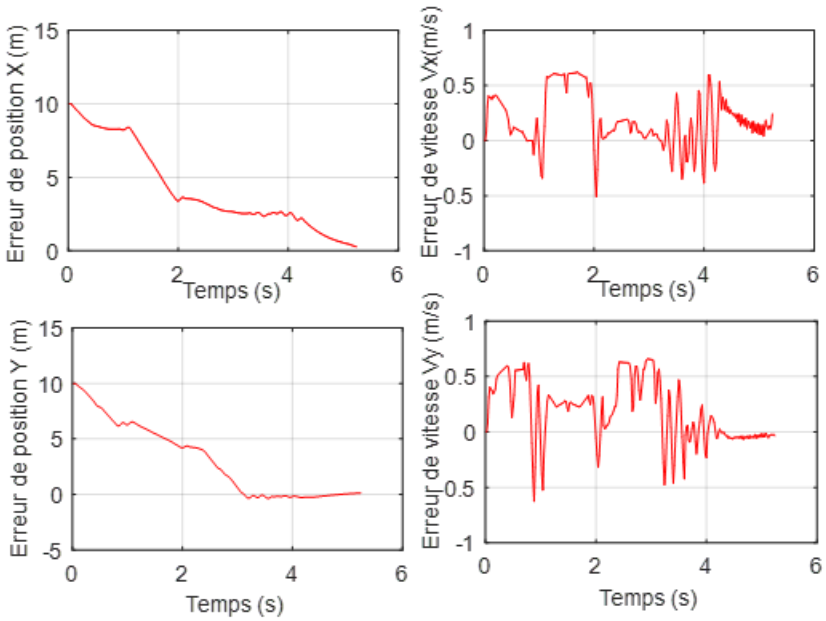


FIGURE 5.47 – Comportements des erreurs de poursuite et des erreurs de vitesse par le contrôleur hybride ANFIS-PSO avec obstacles.

La Figure (5.48) présente les comportements de poursuite et de vitesse par le contrôleur hybride ANFIS-PSO avec obstacles.

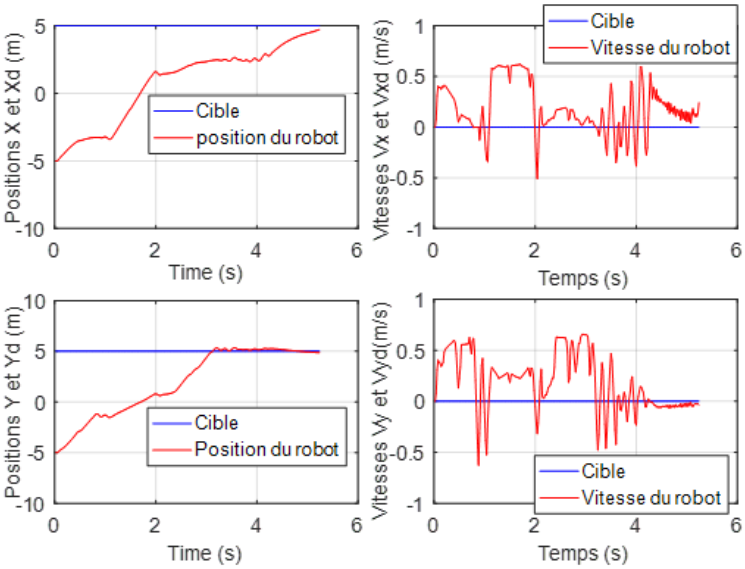


FIGURE 5.48 – Comportements de poursuite et de vitesse par le contrôleur hybride ANFIS-PSO avec obstacles.

La Figure (5.49) présente les comportements des vitesses des roues par le contrôleur hybride ANFIS-PSO avec obstacles.

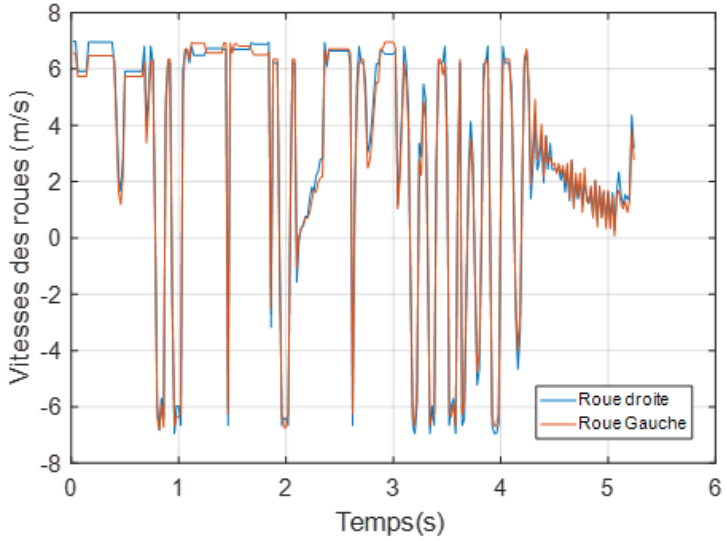


FIGURE 5.49 – Comportements des vitesses des roues par le contrôleur hybride ANFIS-PSO avec obstacles.

La Figure (5.50) présente le chemin parcouru par le robot par le contrôleur hybride ANFIS-PSO avec obstacles.

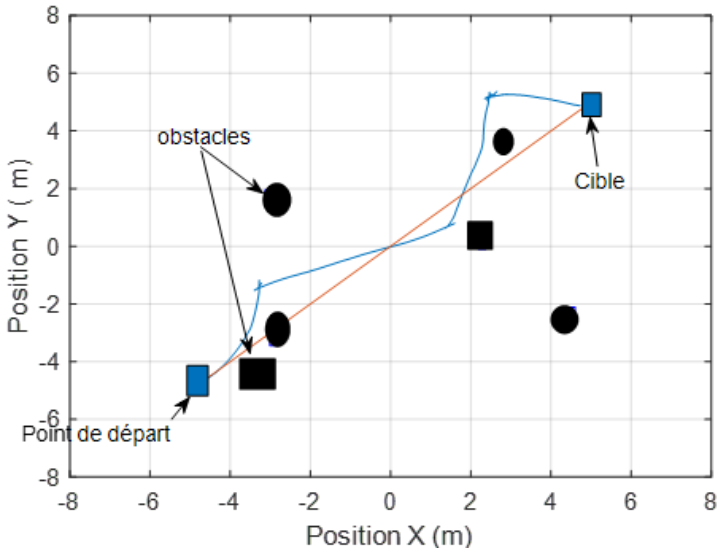


FIGURE 5.50 – Chemin parcouru par le robot par le contrôleur hybride ANFIS-PSO avec obstacles.

La Tableau (5.12) présente les statistiques des commandes envoyées au robot (Figure 5.49).

Tableau 5.12 – Statistiques des commandes envoyées au robot (Figure 5.49).

	Vitesse de roue gauche	**Vitesse de roue droite**
Min(m/s)	-6.831	-6.952
Max(m/s)	6.95	6.988
Moyenne(m/s)	2.951	3.029

Les simulations ci-dessus réaliser dans l'environnement avec obstacles (Figure 5.42) permet d'obtenir les performances montrées sur les (Figures 5.43-5.50)). Les (Figures 5.43-5.46) représentent les résultats obtenus à base de la stratégie de commande qui utilise des régulateurs de type ANFIS. Quant aux (Figures 5.47-5.50)) représentent les résultats obtenus par la stratégie de commande qui combine les régulateurs de type ANFIS avec l'algorithme PSO.
On conclut après une période d'adaptation que : Les deux méthodes assurent que le robot cherche la cible tout en évitant des obstacles ce qui confirme la robustesse de nos algorithmes. Par ailleurs, le robot mobile est plus rapide par l'application de la stratégie de commande qui combine la méthode ANFIS –PSO car le robot mobile réagit au évitement d'obstacles d'une façon optimale en suivant un chemin optimal comme montrer la (Figures 5.50) ainsi que le temps de réponse et l'erreur de poursuite sont meilleurs. Les (Tableaux 5.11 et 5.12) représentent les commandes du robot mobile représentées par les vitesses des roues gauche et droite qui montrent bien l'efficacité d'introduire l'algorithme PSO qui optimise les commandes en respectant toutes les contraintes physiques du robot mobile mentionnées dans le chapitre 3 section (3.8.3) ainsi l'optimisation d'évitement d'obstacle en choisissant un chemin le plus court ce qui signifie que l'algorithme ANFIS-PSO est le plus robuste. D'après cette analyse on constate que l'algorithme ANFIS-PSO donne des bons résultats en terme robustesse, temps de réponse, la distance parcouru, l'efficacité par rapport à l'algorithme ANFIS.

5.5.1 Etude comparatif

La Tableau (5.13) présente des valeurs comparatives pour la simulation par les contrôleurs ANFIS et ANFIS-PSO avec obstacles.

Tableau 5.13 – Tableau comparatif pour la simulation par les contrôleurs ANFIS et ANFIS-PSO avec obstacles.

Nombre de Cibles	Paramètre de comparaison	Sans PSO	Avec PSO
Une cible	**Temps (s)**	9.8	5.24
	Erreur de position X (m)	0.2998	0.2701
	Erreur de position Y (m)	0.0094	0.1256
	Erreur de Vitesse X (m/s)	0.0195	0.2417
	Erreur de Vitesse Y (m/s)	- 0.0001	- 0.0358
	Distance a la cible (m)	0.3001	0.2980
	Distance parcourue (m)	17.5699	20.7045

Le (Tableau 5.13) ci-dessus représente des performances obtenues à partir des résultats de simulation par deux contrôleurs ANFIS et ANFIS-PSO dans un environnement avec obstacles qui nous permet de réaliser une étude comparative en terme de temps, précision, rapidité et distance, pour montrer l'efficacité de chaque méthode.
D'après le (Tableau 5.13) nous avons remarqué que le temps nécessaire par le robot mobile pour atteindre la cible tout en évitant tous les obstacles est moins avec le contrôleur hybride ANFIS-PSO, les erreurs de position et de vitesse montrent bien que le robot atteint les cibles avec une précision meilleure par le contrôleur hybride.

5.6 Conclusion

Dans ce chapitre nous avons présenté les résultats de simulation de la navigation virtuelle d'un robot mobile en utilisant la plateforme d'expérimentation des robots virtuels V-REP qui nous permet la conception d'environnements virtuels par des commandes intelligentes basées sur des régulateurs neuro-floue écrits en code Matlab.
Les résultats de notre travail montrent bien l'efficacité des deux méthodes qui assurent la recherche d'une cible en évitant des obstacles statiques ; le contrôleur hybride ANFIS-PSO donne des meilleurs résultats par rapport à la méthode ANFIS.
V-REP nous a aidé à réaliser les environnements virtuels de simulation près de la réalité et qui prennent en considération les caractéristiques physiques du robot et l'environnement ; afin d'avoir une vision globale sur les problèmes et les résultats attendus avant l'implémentation de ces techniques de commande sur un robot réel dans un environnement réel.
Comme travail supplémentaire, nous suggérons de réaliser d'autre navigation dans des environnements plus complexe comme ajouter des obstacles dynamiques, environnement labyrinthe, et planification de trajectoire... etc.

Conclusion générale

La navigation d'un robot mobile peut être commandée suivant plusieurs stratégies qui correspondent à des manières différentes de générer les lois de commande. Le travail que nous avons présenté vise d'une part à évaluer l'intérêt des commandes intelligentes que ce soit dans un espace dépourvu d'obstacles ou parsemés d'obstacles et d'autre part à examiner l'apport de l'hybridation de ces commandes intelligentes et l'algorithme méta heuristique par l'essaim de particule (PSO) écrits en code Matlab à ce problème.
Tout d'abord, nous avons donné un aperçu de la robotique mobile en général. Nous avons présenté des généralités sur le robot mobile et ces éléments constitutifs, la navigation autonome et planification du chemin pour le robot mobile.
Nous avons choisi pour réaliser notre recherche le robot mobile à entrainement différentiel (DDWMR) qui a suscité un grand intérêt au cours de la dernière décennie ; cela est principalement dû aux progrès croissants des technologies d'instrumentation et de calcul et aux avantages qu'offre les robots mobile à roues (WMR) par rapport aux autres modèles de robots mobiles ; Nous sommes passés à la modélisation du robot mobile à entrainement différentiel (DDWMR) ; d'abord le modèle cinématique, puis le modèle dynamique complet de ce robot mobile ; puis on à valider les deux modèles par une simulation Matlab par deux contrôleur cinématique et dynamique (Linéarisation par bouclage non linéaire).
Nous développons une commande intelligente à base du Système d'Inférence Neuro Floue Adaptatif (ANFIS) et optimisation par l'algorithme méta heuristique de l'essaim de particule (PSO) écrits en code Matlab ; nous avons proposé un système hybride ANFIS- PSO afin de réaliser une navigation virtuelle du DDWMR dans le but de chercher une cible ou plusieurs tout en évitant des obstacles sans collision. En raison de l'efficacité de cette méthode par le contrôleur hybride ANFIS-PSO dans la navigation des robots mobiles, nous avons présenté la théorie de la logique floue, réseaux de neurones et le contrôleur ANFIS. Nous avons ensuite présenté la théorie de l'algorithme d'optimisation par essaim de particule.
Nous avons fait une description de la réalité virtuelle et son application dans le domaine de la robotique, ainsi une description de la plateforme virtuelle de simulation des robots virtuels qui prend en considérations les caractéristiques physiques du monde réel. Par la suite nous avons établi une connexion en mode synchrone entre V-REP et Matlab afin de réaliser une Co-simulation pour assurer la récupération des données depuis V-REP

et traitement d'information sur Matlab qui calcule les commandes qui seront envoyer au robot en temps réel ; et l'interface V-REP définit notre application dans notre travail qui nous permet de réaliser la conception des environnements virtuels dans lesquels nous avons réalisé notre navigation du robot mobile P-3dx.
Les tests réalisés en simulation montrent bien l'efficacité des deux méthodes qui assurent la recherche d'une cible tout en évitant des obstacles statiques. Toutefois, nous avons noté que le contrôleur hybride ANFIS-PSO donne des meilleurs résultats car le robot parcourt un chemin le plus optimale comparativement au contrôleur ANFIS.
Par ailleurs, V-REP nous a aidé à réaliser les environnements virtuels de simulation près de la réalité et qui prend en considération les caractéristiques physiques du robot mobile et l'environnement ; afin de prendre une vision globale sur les problèmes et les résultats attendu avant l'implémentation de ces techniques de commande sur un robot mobile réel dans un environnement réel.
Comme travail supplémentaire, nous suggérons de réaliser d'autres navigations dans des environnements plus complexes dotés d'obstacles dynamiques, environnement labyrinthe. Ceci peut compliquer la planification de trajectoire... etc. A titre de perspectives, nous proposons dans le cadre des travaux futurs, d'implémenter ces travaux sur un robot mobile réel.
En conclusion, nous avons essayé de dégager une méthodologie à large spectre qui ouvre le champ à une possible standardisation de la conception des lois de commande à base de structure neuro-floue adaptative (ANFIS) optimisée par l'algorithme méta heuristique par l'essaim de particule (PSO) écrits en code Matlab en vue de naviguer un robot mobile dans un environement parsemé d'obstacles.

Bibliographie

[1] F.Ernesto, F.Gonzalo, P.Emmanuel, V.Héctor, D.Sebastian. Teaching control in mobile robotics with V-REP and a Khepera IV library. 2016 IEEE Conference on Control Applications (CCA) Part of 2016 IEEE Multi-Conference on Systems and Control September 19-22, 2016. Buenos Aires, Argentina.

[2] T. Niemueller, G. Lakemeyer and A. Ferrein, "The RoboCup logistics league as a benchmark for planning in robotics," in In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)—WS on Planning and Robotics (PlanRob), Jerusalem, Israel, 7–11 June 2015 .

[3] V. Milanés and L. Bergasa, "Introduction to the special issue on," New trends towards automatic vehicle control and perception systems. Sensors 2013, vol. 13, pp. 5712-5719, 2013.

[4] N. Nilsson, "A Mobile Automaton : An Application of Artificial Intelligence Techniques.," in In Proceedings of the 1st International Joint Conference on Artificial intelligence (IJCAI'69), Morgan Kaufmann Publishers Inc. : San Francisco, CA, USA, 1969 ; pp. 509–520, Washington, DC, USA, 7–9 May 1969.

[5] S. Ivaldi, V. Padois, and F. Nori. Tools for dynamics simulation of robots : a survey based on user feedback. arXiv :1402.7050 [cs], Feb. 2014.

[6] Gharajeh MS and Jond HB.An intelligent approach for autonomous mobile robots path planning based on adaptive neuro-fuzzy inference system. Ain Shams Engineering Journal 13 (2022) 101491.

[7] B.Nursena, B.Mehmet, K.Mehmet. PSO Based Path Planning Approach for Multi Service Robots in Dynamic Environments. 2018 International Conference on Artificial Intelligence and Data Processing (IDAP).

[8] MS.Gharajeh, HB.Jond. Hybrid Global Positioning System-Adaptive Neuro-Fuzzy Inference System based autonomous mobile robot navigation. Robotics and Autonomous Systems 134 (2020) 103669.

[9] G.Faris, E.Fabregas, E.Peralta, E.Torres, S.Dormido. A Khepera IV library for robotic control education using V-REP. IFAC PapersOnLine 50-1 (2017) 9150–9155.

[10] H.Batti, C.Ben Jabeur, H.Seddik. Autonomous smart robot for path predicting and finding in maze based on fuzzy and neuro-Fuzzy approaches. Asian J Control. 2020, 1–10.

[11] UI.Mubashir, TK.Ameer, L.Shuai.Bio-inspired BAS : Run-time Path-planning And The Control of Differential Mobile Robot. EAI Endorsed Transactions on AI and Robotics Journal.2020, 1-10.

[12] P.Anish, SP.Vikas, HMd.Ehtesham, RP.Dayal.V-REP-based navigation of automated wheeled robot between obstacles using PSO-tuned feedforward neural network. Journal of Computational Design and Engineering, 2020, 7(0), 1–8.

[13] M.Lazreg, N.Benamrane. Intelligent System for Robotic Navigation Using ANFIS and ACOr. Applied Artificial Intelligence Journal.2019, 1-20.

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

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

[16] S. G. Shuzhi, F. L. Lewis, "Autonomous Mobile Robots, Sensing, Control, Decision,Making and Applications", Taylor and Francis Group, 2006.

[17] C. Lakhmissi, " Navigation Autonome d'un Robot Mobile par des Techniques Neuro-Floues ", Thèse de Doctorat de l'Université Mohamed Khider – Biskra, 2014.

[18] H. Ben Said, "Navigation autonome et commande référencée capteurs de robots d'assistance à la personne", Thèse de Doctorat de l'Université de Limoges 7252, 2018.

[19] A. Remazeilles, "Navigation à partir d'une mémoire d'images", Thèse de Doctorat de l'université de rennes 1, 2004.

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

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

[22] A. BENMACHICHE, "Approche de Navigation Coopérative et Autonome des Robots Mobiles (Application sur un chantier de construction)", Thèse de Doctorat de l'université l'Université Badji Mokhtar –Annaba, 2016.

[23] A. Dobra, "General classification of robots. Size criteria," in Robotics in Alpe-Adria-Danube Region (RAAD), 2014 23rd International Conference on, 20 14, pp. 1-6.

[24] B. BELOBO MEVO, “contribution à la commande adaptative et robuste d’un robot mobile de type unicycle avec modèle non-linéaire”, mémoire présenté comme exigence partielle de la maîtrise en ingénierie à UQAT de l’Université du Québec.

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

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

[27] https ://elektronicavoorjou.nl/fr/Les-produits/Robot-%C3%A0-imp%C3%A9riale-2-roues-motrices/

[28] https ://eu.robotshop.com/fr/products/dfrobot-4wd-arduino-mobile-platform

[29] https ://eu.robotshop.com/fr/products/moorebot-scout-ai-powered-autonomous-mobile-robot

[30] S. Lens, "Locomotion d’un robot mobile". Mémoire présenté en vue de l’obtention du grade d’Ingénieur Civil Informaticien Année académique 2007-2008. Université de liége.

[31] B. Khemisti, "commande d’un robot mobile par réseaux de neurones artificiels" Mémoire en vue de l’obtention du diplôme de magister en électronique. Option : Robotique. Université El Hadj Lakhdar Batna.

[32] B. DJEMAI, “ Commande Optimale Appliquée à un Robot Mobile”, thèse de magister de l’Université de Batna Faculté de Technologie, 2013.

[33] https ://www.researchgate.net/publication/273471749_Station_Keeping_through_Beacon-referenced_Cyclic_Pursuit....khepera II

[34] https ://www.researchgate.net/publication/273471749_Station_Keeping_through_Beacon-referenced_Cyclic_Pursuit.....pioneer

[35] "1977 – “HILARE” Autonomous Mobile Robot," 2011.

[36] https ://www.lynxmotion.com/a4wd3-rugged-rovers

[37] https ://eu.robotshop.com/fr/products/3wd-omni-directional-starter-mobile-robot-kit

[38] https ://www.cdiscount.com/juniors/plein-air/kit-de-chassis-de-voiture-a-chenilles-en-caoutchou/f-12004420404-aih1688255095684.html ?idOffre=3614749422

[39] http ://cf2016e23bbis.eklablog.com/la-mobilite-des-robots-a128131800

[40] https ://ici.radio-canada.ca/nouvelle/1156709/robot-mit-mini-cheetah-massachusetts-institute-technology

[41] https ://actu.epfl.ch/news/les-robots-a-six-pattes-plus-rapides-que-ce-qu-off

[42] E. Laurin, "Système Intelligent d'Assistance à la Perception dans la Conduite de Véhicule", Thèse de Doctorat, Université de Sherbrooke, 2000.

[43] R. S. Pol et M. Murugan (2015). A review on indoor human aware autonomous mobile robot navigation through a dynamic environment survey of different path planning algorithm and methods. In Industrial Instrumentation and Control (ICIC), 2015 International Conference on, pages 1339–1344. IEEE.

[44] E. Ayari, S. Hadouaj et K. Ghedira (2010). A fuzzy logic method for autonomous robot navigation in dynamic and uncertain environment composed with complex traps. In Computing in the Global Information Technology (ICCGI), 2010 Fifth International Multi-Conference on, pages 18–23. IEEE.

[45] S. Park, S. et Hashimoto (2009). Autonomous mobile robot navigation using passive rfid in indoor environment. IEEE Transactions on Industrial Electronics, 56(7) :2366–2373.

[46] D. Nakhaeinia, S. Tang, S. M. Noor et O. Motlagh (2011). A review of control architectures for autonomous navigation of mobile robots. International Journal of Physical Sciences, 6(2) :169–174.

[47] R. Chatila et J.-P. Laumond (1985). Position referencing and consistent world modeling for mobile robots. In Robotics and Automation. Proceedings. 1985 IEEE International Conference on, volume 2, pages 138–145. IEEE.

[48] R. Huq, G. K. Mann et R. G. Gosine (2008). Mobile robot navigation using motor schema and fuzzy context dependent behavior modulation. Applied soft computing, 8(1) :422–436.

[49] H. Seraji et A. Howard (2002). Behavior-based robot navigation on challenging terrain : A fuzzy logic approach. IEEE Transactions on Robotics and Automation, 18(3) :308–321.

[50] C. Ye et D. Wang (2000). A novel behavior fusion method for the navigation of mobile robots. In Systems, Man, and Cybernetics, 2000 IEEE International Conference on, volume 5, pages 3526–3531. IEEE.

[51] R. C. Arkin (1998). Behavior-Based Robotics, volume 1 of 1. The MIT Press, Cambridge, Massachusetts London, England, 1 edition. 1.

[52] R. R. Murphy (2000). Introduction to AI Robotics, volume 1 of 1. The MIT Press, Cambridge, Massachusetts London, England, 1 edition. 1.

[53] J. Minguez et L. Montano (2005). Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios. Robotics and Autonomous Systems, 52(4) :290–311.

[54] Z. Du, D. Qu, F. Xu, et D. Xu, (2007). A hybrid approach for mobile robot path planning in dynamic environments. 2007 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 1058–1063.

[55] D. Halperin (1994). Robot motion planning and the single cell problem in arrangements. Journal of Intelligent & Robotic Systems, 11(1) :45–65.

[56] S. Thrun (2003). Learning occupancy grid maps with forward sensor models. Autonomous robots, 15(2) :111–127.

[57] L. B. Marinho, J. S. Almeida, J. W. M. Souza, , V. H. C. Albuquerque et P. P. Rebouças Filho, (2017). A novel mobile robot localization approach based on topological maps using classification with reject option in omnidirectional images. Expert Systems with Applications, 72 :1–17.

[58] L. Gerstmayr-Hillen, F. Röben, M. Krzykawski, S. Kreft, D. Venjakob, et R. Möller (2013). Dense topological maps and partial pose estimation for visual control of an autonomous cleaning robot. Robotics and Autonomous Systems, 61(5) :497–516.

[59] H. Ghazouani (2012). Navigation visuelle de robots mobiles dans un environnement d'intérieur. PhD thesis, Université Montpellier II-Sciences et Techniques du Languedoc.

[60] D. Filliat et J.-A Meyer (2003). Map-based navigation in mobile robots : : I. a review of localization strategies. Cognitive Systems Research, 4(4) :243–282.

[61] A. Elfes (1989). A tesselated probabilistic representation for spatial robot perception and navigation. In Proceedings of the NASA Conference on Space Telerobotics, volume 2, pages 341–350.

[62] J. Moras, J. Dezert et B. Pannetier (2015). Grid occupancy estimation for environment perception based on belief functions and pcr6. In SPIE Defense+ Security, pages 94740P–94740P. International Society for Optics and Photonics.

[63] P. Schmuck, S. A. Scherer et A. Zell (2016). Hybrid metric-topological 3d occupancy grid maps for large-scale mapping. IFAC-PapersOnLine, 49(15) :230–235.

[64] S. Jia, H. Shen, X. Li, W. Cui et K. Wang (2012). Autonomous robot exploration based on hybrid environment model. In 2012 IEEE International Conférence on Information and Automation.

[65] B. Fayçal, “Navigation neuro-floue d’un robot mobile dans un environnement inconnu”, Thèse de Doctorat de l’Université des sciences et de la technologie d’Oran Mohamed Boudiaf (USTO), 2016.

[66] E. Dombre and W. Khalil, "Robot Manipulators : Modeling," Performance Analysis and Control, vol. 20, p. 24, 2007.

[67] M. Defoort, "Contributions à la planification et à la commande pour les robots mobiles coopératifs," Ecole Centrale de Lille, 2007.

[68] V. Delsart, "Navigation autonome en environnement dynamique : Une approche par déformation de trajectoire," Université de Grenoble, 2010.

[69] F. Aurenhammer, R. Klein et D.-T. Lee (2013). Voronoi diagrams and Delaunay triangulations. World Scientific Publishing Co Inc.

[70] T. Iwaszko (2012). Généralisation du diagramme de Voronoï et placement de formes géométriques complexes dans un nuage de points. PhD thesis, Université de Haute Alsace-Mulhouse.

[71] I. Rekleitis, V. Lee-Shue, A. P. New et H. Choset (2004). Limited communication, multi-robot team based coverage. In Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on, volume 4, pages 3462–3468. IEEE.

[72] J. M. Keil et J.-R Sack (1985). Minimum decompositions of polygonal objects. Machine Intelligence and Pattern Recognition, 2 :197–216.

[73] F. Lingelbach (2004). Path planning using probabilistic cell decomposition. In Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on, volume 1, pages 467–472. IEEE.

[74] C. Cai et S. Ferrari (2009). Information-driven sensor path planning by approximate cell decomposition. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(3) :672–689.

[75] A. Swingler et S. Ferrari (2010). A cell decomposition approach to cooperative path planning and collision avoidance via disjunctive programming. In Decision and Control (CDC), 2010 49th IEEE Conference on, pages 6329–6336. IEEE.

[76] O. Khatib (1986). Real-time obstacle avoidance for manipulators and mobile robots. The international journal of robotics research, 5(1) :90–98.

[77] H. M. Choset (2005). Principles of robot motion : theory, algorithms, and implementation. MIT press.

[78] I. da Silva, F. A. Gomide, and W. Do Amaral, "Navigation of mobile robots using fuzzy logic controllers," in Advanced Motion Control, 1998. AMC'98-Coimbra., 1998 5th International Workshop on, 1998, pp. 346-349.

[79] S. Yamaguchi and H. Itakura, "A modular neural network for control of mobile robots," in Neural Information Processing, 1999. Proceedings. ICONIP'99. 6th International Conference on, 1999, pp. 661-666.

[80] L. Qiang, C. Yin, L. Ming, and Y. Zheng, "Evolutionary algorithm for path planning of mobile robot," in Intelligent Control and Automation, 2000. Proceedings of the 3rd World Congress on, 2000, pp. 1206-1209.

[81] A. Brahami ,R. Mellah, S. Guermah (2024). Virtual navigation of mobile robot in V-REP using hybrid ANFIS-PSO controller. Control Engineering and Applied Informatics. Vol.26, No.1, pp. 25-35, 2024.

[82] Y. SAIDI (2021). Robust Mobile Robot Navigation using Fuzzy Logic Controllers with Robustness Analysis. Doctor in Automatic in National Polytechnic School Department of Automation Process Control Laboratory.

[83] M. Maya. Commande référencée capteur des robots non holonomes. Automatique / Robotique. École Nationale Supérieure des Mines de Paris, 2007. Français.

[84] R. Dhaouadi, AA. Hatab (2013). Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies : A Unified Framework. Adv Robot Autom 2 : 107. doi : 10.4172/2168-9695.1000107.

[85] L. Génevé (2017). Système de déploiement d'un robot mobile autonome basé sur des balises. Pour obtention. Docteur de l'université de Strasbourg.

[86] O. Mohareri , R. Dhaouadi , A.B.Rad (2012). Indirect adaptive tracking control of a nonholonomic mobile robot via neural networks. Journal Neurocomputing

[87] M. MECIFI (2013). Contribution à la commande non linéaire d'un robot manipulateur hyper dynamique. MAGISTER de l' Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF.

[88] Alaa Dib. Commande non linéaire et asservissement visuel de robots autonomes. Automatique /Robotique. Supélec, 2011. Français.

[89] R. Mellah, S. Guermah, R. Toumi, Adaptive control of bilateral teleoperation system with compensatory neural-fuzzy controllers, International Journal of Control, Automation and Systems, Vol. 15, pp. 1-11, 2017.

[90] R. Mellah, R. Toumi, Compensatory neuro-fuzzy control of bilateral teleoperation system, Proc. of 20th International Conference on Methods and Models in Automation and Robotic ; 2015 Aug 24-27 ; Miedzyzdroje, Poland, pp. 382-387, 2015.

[91] W. Po-Ngaen, Adaptive four-channel neuro-fuzzy control of a master-slave robot, International Journal of Advanced Robotic Systems, Vol. 10, pp. 1-8, 2013.

[92] R. Mellah, R. Toumi, Control bilateral teleoperation by compensatory ANFIS, Advanced Mechatronics Solutions, Vol. 393, pp. 167-172, 2016.

[93] W.E. Kelly, R. Challoo, R. Mclauchlan, S.I. Omar, Neuro-fuzzy control of a robotic arm, Proceedings of the Artificial Neural Networks In Engineering Conference, pp. 837-842, 1996.

[94] C.-T. Lin and C. G. Lee, "Neural-network-based fuzzy logic control and decision system," Computers, IEEE Transactions on, vol. 40, pp. 1320-1336, 1991.

[95] H. Khati (2019). Commande d'une Architecture de Téléopération par la Carte FPGA, thèse de doctorat de l'Université Mouloud Mammeri de Tizi-Ouzou.

[96] A.O Ghrieb (2020). Supervision d'un robot manipulateur virtuel par les réseaux de neurons, thèse de doctorat de l'Université Batna 2 – Mostefa Ben Boulaïd.

[97] E. Kim, M. Park, S. Ji & M. Park, "A new approach to fuzzy modeling", IEEE Transactions on Fuzzy Systems, v. 5(3), p. 328–337, 1997.

[98] L.Wang & R. Langari, "Complex systems modeling via fuzzy logic", IEEE Transactions On Systems, Man, And Cybernetics-Part B : Cybernetics, v. 26(1), p. 100–106, 1996.

[99] H. Abdelouahab, "Calcul de trajectoire et contrôle de déplacement d'un robot ". Thèse de doctorat, université Ferhat Abbas Setif, 2007.

[100] B. B. Meunier, "La logique floue et ses applications", Edition Addison Wesley France, SA, 1995.

[101] B.Kasmi (2022). Commande neuronale robuste pour la navigation d'un robot mobile. Thèse de doctorat, universite ferhat abbas setif-1.

[102] J. M.Keller, D. Liu, D. B. Fogel, Fundamentals of Computational Intelligence : Neural Networks, Fuzzy Systems, and Evolutionary Computation, Wiley-Blackwell, 2016.

[103] H.T. Nguyen, N.R. Prasad, C.L. Walker, E.A. Walker, A first course in fuzzy and neural control, Chapman and Hall/CRC, 2002.

[104] R. Mellah, Contribution à la commande adaptative neuro-floue. Application à la robotique,Thèse de Doctorat à l'Université des Sciences et de la Technologie Houari Boumediene, Alger, Mai 2006.

[105] N. Siddique, Intelligent Control : A Hybrid Approach Based on Fuzzy Logic, Neural Networks and Genetic Algorithms, Springer, 2014.

[106] G.J. Klir & B. Yuan, "Fuzzy sets and fuzzy logic : theory and applications", PrenticeHall. Inc. Upper Saddle River, NJ, USA, 1994.

[107] O. Castillo, P. Melin, J. Kacprzyk, Fuzzy Logic Augmentation of Neural and Optimization Algorithms : Theoretical Aspects and Real Applications, Springer, 2018.

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

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

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

[111] R.S. Burns, "Advanced Control Engineering", butterworth heinemann ed, 2001.

[112] P.S. Sastry, G. Santharam & K.P. Unnikrishnan, "Memory neuron networks for identification and control of dynamical systems", IEEE Transactions on Neural Networks v. 5(2), p. 306–319, 1994.

[113] S. Chakravertys, S. Mall, Artificial Neural Networks for Engineers and Scientists, Solving Ordinary Differential Equations, CRC Press, 2017.

[114] I. N. Da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, S. F. D. R. Alves, Artificial Neural Networks, A Practical Course, Springer, 2017.

[115] T. Xiang, K.F. Man, K.M. Luk & C.H. Chan, "Design of multiband miniature handset antenna by mom and hga", IEEE Antennas and wireless propagation letters 5, 179–182, 1997.

[116] D. Nauck, "Neuro-fuzzy systems : review and prospects", European congress on intelligent technique and sift computing (EUFIT'97), Aachen, p.1044-1053, 1997.

[117] E. Gauthier, "Gestion d'une flotte de véhicules autonomes à l'intérieur d'un parking haute densité", Rapport d'étude, Institut National Polytechnique de Grenoble DEA Informatique Option Robotique Vision Image, juin 1995.

[118] E. Gauthier, "Utilisation des réseaux de neurones artificiels pour la commande d'un véhicule autonome", Thèse de doctorat, Institut National Polytechnique de Grenoble, Janvier 1999.

[119] W. Pedrycz and H. Card, "Linguistic interpretation of self-organizing maps," in Fuzzy Systems, 1992., IEEE International Conference on, 1992, pp. 371-378.

[120] B. Kosko, "Neural networks and fuzzy systems : a dynamical systems approach to machine intelligence/book and disk," Vol. 1Prentice hall, 1992.

[121] H. Nomura, I. Hayashi, and N. Wakami, "A learning method of fuzzy inference rules by descent method," in Fuzzy Systems, 1992., IEEE International Conference on, 1992, pp. 203-210.

[122] J.-S. Jang, "Neuro-fuzzy modeling : Architectures, analyses, and applications", University of California, Berkeley, 1992.

[123] C.-T. Lin and C. G. Lee, "Neural-network-based fuzzy logic control and decision system," Computers, IEEE Transactions on, vol. 40, pp. 1320-1336, 1991.

[124] D. Nauck and R. Kruse, "NEFCON-I : An X-Window based simulator for neural fuzzy controllers," in Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on, 1994, pp. 1638-1643.

[125] D. Nauck and R. Kruse, "NEFCLASSmdash ; a neuro-fuzzy approach for the classification of data," in Proceedings of the 1995 ACM symposium on applied computing, 1995, pp. 461-465.

[126] D. Nauck and R. Kruse, "Neuro-fuzzy systems for function approximation," Fuzzy Sets and Systems, vol. 101, pp. 261-271, 1999.

[127] Mohamed Boukens. Contribution à la commande des robots manipulateurs mobiles. Automatique / Robotique. Université de Jijel (Algérie) ; Université de Picardie Jules Verne, 2017. Français.

[128] N. K. Kasabov and Q. Song, Dynamic Evolving Fuzzy Neural Networks with'm-out of-n'Activation Nodes for On-line Adaptive Systems : Department of Information Science, University of Otago, 1999.

[129] J. Dréo, A. Pétrowski, P. Siarry and E. Taillard. "Métaheuristiques pour l'optimisation difficile ». Eyrolles. Paris, 2003.

[130] Wikipédia, "Optimisation par essaim particulaire", http ://fr.wikipedia.org/wiki/Optimisation_par_essaims_particulaires.

[131] R. Attia, M. Adjadji, "Différentes techniques d'optimisation de la commande flou. Application sur la colonne d'absorption", Projet de Fin d'Etude, Ecole Nationale Polytechnique, 2009, Alger.

[132] A. Rioland, A. Audes, "Optimisation essaim particulaire pour un problème d'ordonnancement et affectation de ressource", Rapport de projet 3ème Année, Institut Supérieur d'Informatique de Modélisation et de leurs Applications, France, 2007.

[133] H. Dubreil, "Méthodes d'optimisation de contrôleurs de logique floue pour le paramétrage automatique des réseaux mobile UMTS", Thèse de Doctorat, Ecole Nationale Supérieure des Télécommunications, Paris, France 2005.

[134] M. Clerc, "Optimisation par essaim particulaire : versions paramétriques et adaptatives", Hermes science - Lavoisier, Paris, 2005.

[135] M. Clerc, P.Siarry, "Une nouvelle métaheuristique pour l'optimisation difficile : la méthode des essaims particulaires", rapport de recherche, J3eA, Vol.3-7, France, 2004 ;

[136] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla : An open urban driving simulator.

[137] M. J. Tarr and W. H. Warren. Virtual reality in behavioral neuroscience and beyond. Nature Neuroscience, 5(11) :1089–1092, Nov. 2002.

[138] F. Grzeskowiak. Simulation et expérimentations pour l'évaluation de la navigation de robots dans la foule. Robotique [cs.RO]. Thèse de Doctorat de l'Université de Rennes 1 (UR1), 2021.

[139] M. Saiful Azimi, ZA. Shukri and M. Zaharuddin : Virtual Reality based Mobile Robot Navigation in Greenhouse Environment. Advances in Animal Biosciences : Precision Agriculture (ECPA) 2017, (2017), 8 :2, pp 854–859.

[140] C.Pinciroli, V.Trianni, R.O'Grady. "ARGoS : A modular, ultiengine simulator for heterogeneous swarm robotics," in International Conference on Intelligent Robots and Systems (IROS), Sept 2011, pp. 5027–5034.

[141] L.Guyot, N.Heiniger, O.Michel, F.Rohrer. "Teaching robotics with an open curriculum based on the e-puck robot, simulations and competitions," in Proceedings of the 2nd International Conference on Robotics in Education. Vienna, Austria, 2011.

[142] Coppelia Robotics GmbH, "V-REP simulator," 2016. [Online]. Available : http ://www.coppeliarobotics.com.

[143] M. L. Heilig : Sensorama Stimulator, 1962. United States Patent and Trade Office, Virginia, USA, US-3,050,870.

[144] http ://www.telepresence.org

[145] I.E. Sutherland : The ultimate display. In Proceedings of the IFIP Congress, volume 2, pages 506–508, 1965.

[146] H. Rheingold : La réalité virtuelle : quand l'illusion a toutes les apparences de la réalité. Dunod, 1993.

[147] G. Burdea et P. Coiffet : La réalité virtuelle. Hermès, 1993.

[148] P. Fuchs : Introduction à la réalité virtuelle. In Le traité de la réalité virtuelle, volume 1, chapitre 1. Les Presses de l'Ecole des Mines de Paris, 2003.

[149] S. R. Ellis : Nature and origins of virtual environments : a bibliographical essay. Computing Systems in Engineering, 2(4) :321–347, 1991.

[150] D. Zeltzer : Autonomy, interaction, and presence. Presence : Teleoperators and Virtual Environments, 1(1) :127–132, 1992.

[151] B. McKenna : Life, love and the pursuit of virtual reality : An interview with inventor jaron lanier. Arts & Ideas magazine, 1(3), 2000.

[152] B. Arnaldi, P. Fuchs et J. Tisseau : Le traité de la réalité virtuelle, chapitre 1, page 8. Les Presses de l'Ecole des Mines de Paris, 2003.

[153] L. Devigne, M. Babel, F. Nouviale, V. K. Narayanan, F. Pasteau, and P. Gallien. Design of an immersive simulator for assisted power wheelchair driving. In 2017 International Conference on Rehabilitation Robotics (ICORR), pages 995–1000, July 2017.

[154] F. Hernoux, E. Nyiri, and O. Gibaru. Virtual reality for improving safety and collaborative control of industrial robots. In Proceedings of the 2015 Virtual Reality International Conference, VRIC '15, pages 1–6, New York, NY, USA, Apr. 2015.

[155] R. Li, M. v. Almkerk, S. v. Waveren, E. Carter, and I. Leite. Comparing Human-Robot Proxemics Between Virtual Reality and the Real World. In 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 431–439, Mar. 2019.

[156] J. Wainer, D. J. Feil-Seifer, D. A. Shell, and M. J. Mataric. Embodiment and Human-Robot Interaction : A Task-Based Perspective. In RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication, pages 872–877, Aug. 2007.

[157] A. Staranowicz and G. L. Mariottini. A survey and comparison of commercial and open-source robotic simulator software. In Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments, PETRA '11, pages 1–8, New York, NY, USA, May 2011.

[158] P. Castillo-Pizarro, T. V. Arredondo, and M. Torres-Torriti. Introductory Survey to Open-Source Mobile Robot Simulation Software. In 2010 Latin American Robotics Symposium and Intelligent Robotics Meeting, pages 150–155, Oct. 2010.

[159] J. Craighead, R. Murphy, J. Burke, and B. Goldiez. A Survey of Commercial Open Source Unmanned Vehicle Simulators. In Proceedings 2007 IEEE International Conference on Robotics and Automation, pages 852–857, Apr. 2007.

[160] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), volume 3, pages 2149–2154. IEEE.

[161] B. Gerkey, R. T. Vaughan, and A. Howard. The player/stage project : Tools for multi-robot and distributed sensor systems. In Proceedings of the 11th international conference on advanced robotics, volume 1, pages 317–323. Citeseer, 2003.

[162] O. Michel. Cyberbotics Ltd. WebotsTM : Professional Mobile Robot Simulation. International Journal of Advanced Robotic Systems, 1(1) :5, Mar. 2004.

[163] E. Rohmer, S. P. N. Singh, and M. Freese. V-REP : A versatile and scalable robot simulation framework. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1321–1326, Nov. 2013.

[164] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan. Modular open robots simulation engine : MORSE. In 2011 IEEE International Conference on Robotics and Automation, pages 46–51, May 2011.

[165] A. Konrad. Simulation of mobile robots with unity and ros : A case-study and a comparison with gazebo, 2019.

[166] https ://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatlab.htm#simRemoteA

[167] adept, «Pioneer 3-DX,» adept mobilerobots.

[168] P. Bouvier. La présence en réalité virtuelle une approche centrée utilisateur. Thèse de Doctorat de l'Université Paris-Est, 2009.