وزارة التعليم العالي و البحث

BADJI MOKHTAR UNIVERSITY -ANNABA
UNIVERSITE BADJI MOKHTAR -ANNABA

جامعة باجي مختار

- عنابة -

Année 2016

Faculté des Sciences de l'Ingénieur

Département d'Informatique

MÉMOIRE

Présenté

en vue de l'obtention du Diplôme de Doctorat 3ème Cycle LMD

Approche de Navigation Coopérative et Autonome des Robots Mobiles
(Application sur un chantier de construction)

Option

Reconnaissance des Formes et Intelligence Artificielle (RFIA)

Par

Mr Abdelmadjid BENMACHICHE

DEVANT LE JURY

Président :	Dr M. S. Bousbia	Professeur	U.ANNABA
Examinateur:	Dr H. Seridi	Professeur	U.GUELMA
Examinateur :	Dr Z. Bouras	MCA	EPST-ANNABA
Examinateur :	Dr H. F. Merouani	Professeur	U.ANNABA
Rapporteur :	Dr T. Bouhadada	Professeur	U.ANNABA
Invité :	Dr M. T. Laskri	Professeur	U.ANNABA

Remerciements

Tout d'abord, je remercie le Bon Dieu le tout puissant pour la bonne santé, la volonté et la patience qu'il m'a donné pour accomplir ce travail.

Mes premiers remerciements vont à mes encadreurs : Pr Tahar Bouhadada et Pr Mohamed Tayeb Laskri pour leurs orientations, leur disponibilité et pour tout ce qu'ils m'ont appris.

Mr Tahar Bouhadada pour m'avoir fait confiance et pour m'avoir permis de faire cette thèse.

Mr Mohamed Tayeb Laskri pour sa présence et ses conseils qui m'ont fait avancer et progresser.

Je remercie également Pr M. S. Bousbia, d'avoir accepté de présider ce jury.

Mes remerciements vont également aux Pr H. F. Merouani, Pr H. Seridi, et Dr Z. Bouras d'avoir pris la peine de juger ce travail et d'avoir accepté de faire parti de ce jury.

J'adresse aussi mes remerciements à ma famille et à mes amis qui m'ont aidé et encouragé aux moments opportuns.

Dédicaces

Je dédie ce travail :

A La mémoire de Mon père,

A Ma mère,

A Mon frère et Mes sœurs

À Toute la famille

Et à Tous mes amis

Table de Matières

Liste des figures

Liste des tableaux

Résumé

Cette thèse entre dans le cadre de la problématique de la navigation coopérative de robots mobiles autonomes, et se focalise principalement sur un type de système et d'application spécifiques, en l'occurrence, le chantier de construction. Il s'agit ainsi de produire un environnement de robotique spécifique où les robots mobiles autonomes doivent coopérer pour prendre en charge les taches relevant de ce chantier.

Notre travail offre une vision non exhaustive des thématiques de recherche associées au domaine de la robotique mobile, et présente les verrous scientifiques qui restent à lever pour aboutir au développement d'un robot autonome. L'autonomie de ce dernier requiert la réalisation coordonnée de taches de commande et de perception de l'environnement. Parmi celles-ci, la navigation joue un rôle fondamental dans l'interaction du robot avec son environnement d'évolution. Elle consiste en la détermination de trajectoires réalisables par le robot pour suivre un chemin préétabli, tout en contournant les obstacles mobiles ou fixes. Pour effectuer cette tache, notre approche s'appuie sur les algorithmes génétiques de plus court chemin. Le problème de navigation est alors modélisé sous la forme d'un problème d'optimisation sous contraintes dont la fonction fitness quantifie l'écart entre le meilleur chemin du robot et les autres chemins aléatoires. Les obstacles sont intégrés sous forme de contraintes en pénalisant le déplacement des robots ; l'objectif étant de permettre à ces robots de changer de position en évitant les obstacles. Notre approche a été implémentée et plusieurs cas de figures ont été testés. Les résultats obtenus démontrent la robustesse de la méthode déployée ainsi que ses performances.

Mots-clés: robotique évolutionnaire ; chantier de construction; l'algorithme génétique - robots mobiles - évitement d'obstacles - système multi-robots.

Abstract

This thesis is part of the problem of cooperative autonomous mobile robot navigation, and focuses mainly on a specific type of system and application, in this case, the construction site. He thus comes to producing a specific robotic environment where autonomous mobile robots must cooperate to support the tasks under this site.

Our work offers a partial view of research topics related to the field of mobile robotics, and presents the scientific barriers that remain to be overcome lead to the development of an autonomous robot. The autonomy of the latter requires the coordinated implementation of control spots and perception of the environment. Among these, the navigation plays a fundamental role in the interaction of the robot with its changing environment. It consists of determining feasible paths for the robot to follow a predetermined path, skirting mobile or fixed obstacles. To perform this task, our approach is based on genetic algorithms shortest path. The navigation problem is then modeled as a constrained optimization problem whose fitness function quantifies the gap between the best path of the robot and other random paths. Obstacles are integrated as constraints penalizing the movement of robots; the aim being to allow these robots to change positions while avoiding obstacles. This method is implemented and several scenarios were tested. The results obtained demonstrate the robustness of the method as well as extended performance.

Keywords: evolutionary robotics; construction site; the genetic algorithm - mobile robots - obstacle avoidance - multi-robot system.

ملخص

تتناول هذه الأطروحة مشاكل الملاحة التعاونية من الروبوتات المتحركة المستقلة و تركز على نوع من النظام وتطبيق معين: موقع بناء، أي إنتاج بيئة من الروبوتات محددة، حيث تحتاج إلى تعاون الروبوتات المتحركة المستقلة لتولي المهام في إطار هذا موقع البناء.

يقدم عملنا نظرة جزئية من الموضوعات البحثية ذات الصلة في مجال الروبوتات المتنقلة، ويعرض جميع العقبات العلمية المتبقية للارتقاء و الارتفاع بهذا المجال ، ويؤدي إلى تطوير روبوت مستقل. استقلالية هذا الأخير يتطلب التنفيذ المنسق للمهام و السيطرة والنظرة إلى البيئة. ومن بين هؤلاء المهام تلعب الملاحة دورا محوريا في التفاعل الروبوت و بيئتها. وهو يتألف من تحديد مسارات ممكنة بواسطة الروبوت لمتابعة مسار محدد سلفا مع الحفاظ على عدم الاصطدام مع العقبات الثابتة أو المتحركة. لتنفيذ هذه المهمة، ويستند نهجنا على الخوارزميات الجينية لإيجاد اقصر أقصر الطرق. ثم يتم غرار مشكلة الملاحة وإيجاد الحل الأمثل لأفضل مسار للروبوت من بين مجموعة من المسارات العشوائيةوسط مجموعة من العقبات التي تقلص مجال حركة الروبوتات.

وذلك بهدف السماح لهذه الروبوتات لتغيير المواقف مع تجنب العقبات. يتم تطبيق هذا الأسلوب واختبارها عدة سيناريوهات. أظهرت النتائج متانة وأداء النهج المتبع .

كلمات البحث: *موقع البناء - والروبوتات التطوري -الخوارزمية الجينية - الروبوتات المتحركة - تجنب العقبات - نظام متعدد الوكيل.*

Introduction Générale

Introduction Générale

Le concept de robot mobile autonome est apparu vers la fin des années soixante, de deux sources totalement différentes : tout d'abord des recherches menées à l'institut de Recherche de Stanford sur les possibilités d'équiper des machines de capacités de déduction et de réaction logique à des événements extérieurs. On a ainsi construit le robot Shakey [WWW1], machine à roues reliée à un ordinateur et équipée d'une caméra lui permettant d'acquérir des images de son environnement. Shakey évolue dans un univers de cubes et de pyramides de tailles et de couleurs différentes. Il a pour mission de prendre un objet et de le porter ailleurs, quelque soit sa position ; chaque mission dure près de cinquante minutes. D'autre part, l'industrie nucléaire avait besoin de machines permettant d'agir à distance dans des environnements encombrés et inaccessibles à l'homme. L'entreprise américaine « Général Electric » développe alors un quadrupède pour essayer de résoudre ce problème, tandis que les projets Luna [HAR07] et Mars Rover [WWW2] s'échafaudent dans le but d'explorer des planètes sans que l'homme ne prenne part au voyage.

Pendant plusieurs années, laboratoires, industriels, informaticiens et mécaniciens vont continuer leurs travaux en parallèle. On accède ainsi côté industriel à la télé-opération et à une partie de la robotique classique, tandis que du côté informatique, on assiste à de grands progrès dans le domaine de l'intelligence artificielle. Ainsi, vers la fin des années soixante-dix, trois pôles géographiques principaux se distinguent (France, Japon, Etats-Unis). La synthèse de tous les travaux réalisés jusqu'alors donne enfin naissance aux robots mobiles autonomes (du robot domestique au robot militaire).

L'industrie de production, les sociétés d'exploitation minière, les expéditions de recherche sous-marine ...etc, les domaines d'utilisation de robots autonomes sont très variés, allant de la production en chaîne dans une usine de voitures à l'exploration d'autres planètes, comme c'est le cas avec Mars Pathfinder, robot mobile autonome d'exploration de la planète Mars.

C'est dans cet environnement de plus en plus automatisé que se fait sentir le besoin d'outils capables, non seulement d'effectuer des tâches répétitives ou encore impossibles à l'homme (porter des charges lourdes, découpage ultra précis, …), mais aussi de manifester une certaine autonomie de déplacement dans des milieux hostiles à l'homme. On en voit désormais les applications sur des chantiers tels que le désamiantage d'immeubles, la décontamination radioactive, les expériences en milieu dangereux… Aussi a-t-on besoin de robots mobiles autonomes capables de se déplacer d'un point à un autre sur une simple demande de l'utilisateur, qui n'a ainsi plus besoin d'être un expert en pilotage. Le meilleur exemple de planification complexe de la trajectoire est fourni par les drones, ces robots volants destinés aussi bien à l'espionnage militaire qu'aux études botaniques nécessitant des prises de vues aériennes.

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

L'enjeu principal de la robotique mobile actuelle consiste à développer des systèmes de navigation intelligents, où la navigation autonome est un axe de recherche qui vise à donner à une machine la capacité de se mouvoir dans un environnement sans assistance, ni intervention humaine pour accomplir un but désiré [LAT91], [LAU01]. La tâche de la navigation consiste à donner au robot la possibilité d'obtenir les informations dont il a besoin pour raisonner et le doter de capacité de locomotion adaptée à son environnement [BOR96]. Cependant, elle implique des systèmes complexes dans la réalisation, où leur maîtrise pose d'importants problèmes non seulement technologiques mais aussi scientifiques. Un robot mobile autonome est un système mécanique qui doit être en mesure de prendre des décisions pour effectuer des mouvements en fonction des informations sur sa position et sur l'environnement dans lequel il évolue [CHE14].

L'intérêt est d'atteindre un objectif visé, tout en s'adaptant à certaines variations des conditions de fonctionnement sans intervention humaine [SHU06].

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

La recherche en robotique mobile s'intéresse à la conception des systèmes intelligents dotés par des techniques de commande efficaces pour le déplacement d'un robot mobile, où la sécurité soit prioritaire par rapport à l'optimalité [CUE05], [LAU01]. Initialement, le système de navigation dispose d'un modèle de l'environnement dans lequel sont représentés les principaux éléments fixes: obstacles, murs, portes, meubles…etc. Ces systèmes de navigation traditionnels présentent certaines difficultés dans les applications temps réel puisqu'ils nécessitent une grande capacité de calcul et de mémoire [KHA86], [BOR91], exigent une planification complète de l'environnement, et restent incompatibles avec les exigences des robots en temps réel en termes de rapidité et de réactivité. Cependant, lorsque l'environnement devient plus complexe (*i.e.* partiellement connu ou dynamique), il apparait indispensable que le robot mobile soit doté de capacités décisionnelles lui permettant de : réagir automatiquement, sans collision, avec les objets imprévus, percevoir, analyser et modéliser son environnement. Ensuite, à partir de l'information disponible, le robot doit pouvoir planifier sa trajectoire de mouvement. En fin, le système de contrôle doit exécuter la séquence des actions élaborées.

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

Plusieurs techniques ont été proposées pour résoudre le problème de navigation autonome d'un robot mobile, les plus utilisées sont celles de l'intelligence artificielle, telles que les algorithmes génétiques, la logique floue, les réseaux de neurones artificiels ou l'apprentissage automatique. L'approche génétique est basée sur l'utilisation des techniques d'évolution artificielle pour développer des contrôleurs pour des robots. L'avantage de cette approche est que la représentation utilisée pour les contrôleurs est très libre. Ainsi, il est possible de faire évoluer des programmes avec des techniques de type programmation génétique. Les techniques les plus utilisées aussi sont celles qui font appel à des techniques d'auto-apprentissage [SUT98] et les systèmes neuronaux [AND95]. Les capacités d'apprentissage des réseaux de neurones peuvent être utilisées pour résoudre ce problème. L'idée de base est d'entraîner le robot mobile pour un ensemble de situations et la capacité de généralisation des réseaux de neurones lui permet de prendre les décisions adéquates dans le cas de nouvelles situations [THR95], [FIE98], [GAU99], [JAN04]. L'apprentissage supervisé nécessite la disponibilité d'un nombre convenable de paires situation-action. Malgré sa convergence rapide, il est difficile d'obtenir un nombre suffisant de données pour l'apprentissage. D'autre part, l'apprentissage non supervisé semble être une solution prometteuse pour résoudre ce problème puisqu'il ne requiert pas la disponibilité de paires situation-action pour l'apprentissage [SUT98]. Au lieu de programmer un robot pour qu'il effectue une mission, on peut le laisser seul apprendre sa propre stratégie. La technique consiste à déterminer quelles sont les meilleures actions à réaliser. Si ces actions permettent au robot d'atteindre son objectif, leurs liens d'activation sont renforcés. L'idée fondamentale de l'apprentissage non supervisé est d'améliorer un comportement après chaque interaction avec l'environnement. Au début il faut que le robot explore son environnement pour associer les situations et les actions à des récompenses positives ou négatives [WAT92], [GLO99].

Les algorithmes génétiques ont montré leur efficacité pour la navigation des robots mobiles [HER95], [ABD04], ils ont déjà une histoire relativement ancienne, puisque les premiers travaux de John Holland sur les systèmes adaptatifs remontent à 1962 [HOL62]. L'ouvrage de David Goldberg [GOL89] a largement contribué à sa vulgarisation. De très nombreuses variantes des algorithmes génétiques existent aujourd'hui. On trouve dans la littérature plusieures méthodes de réglage des algorithmes génétiques avec d'autres approches de l'intelligence artificielle, telles que: les réseaux de neurones [JAN93], [GOD99], l'apprentissage par renforcement [BER92], [JOU98], les colonies de fourmis [BOU09], ou bien combinées [TOU97], [CAN03], [ZHO07]…etc. Ces méthodes dites hybrides combinent les propriétés de chaque approche afin d'optimiser les paramètres des systèmes d'inférence flous. Elles sont capables de générer une solution optimale ou quasi-optimale.

Contribution et objectifs

L'objectif de notre étude porte sur les potentiels donnés par les technologies d'automatisation de développer des procédés de construction durables pour les bâtiments. Les aspects de tels processus, économiques, environnementaux et sociaux, sont de ce fait appréciés.

Les principaux facteurs qui poussent à un changement dans ce processus de construction sont la diminution du nombre des travailleurs, l'augmentation du coût et de la durée limitée des projets, la pression sur l'industrie de construction et la consommation couteuse de matériaux, dans le secteur de la construction. Dans ce contexte, notre objectif vise à développer une approche qui permet à un groupe d'agents (les robots) de coopérer pour construire un mur dans un chantier de construction.

Notre travail consiste à implémenter un système de planification pour robots mobiles. Nous y abordons le problème d'un robot évoluant dans un environnement d'intérieur, a priori, méconnu. Pour réaliser la navigation de ce robot mobile dans de telles conditions, il sera question, dans la première étape, de le doter de moyens de perception de l'environnement et d'analyse d'informations, et dans la deuxième étape (étape de la classification), d'attribuer des activités (des tâches) pour les robots (acteurs) employeurs de ce chantier. Ainsi il est possible de séparer les activités et diviser le travail entre les deux catégories des robots employeurs, maçons et ouvriers. La troisième étape regroupera les thématiques liées à la planification de chemins, la génération de trajectoires et la commande des robots de manière générale. Pour optimiser la navigation des robots ouvriers, nous utilisons les algorithmes génétiques, et nous proposons un modèle de planification pour la navigation autonome des robots mobiles. En d'autres termes, nous utilisons plusieurs robots dans un environnement dynamique de façon optimale où ils devront déterminer le plus court chemin avec un générateur aléatoire, de trajectoires (chemins). Les obstacles générés de façon aléatoire sont insérés dans des endroits inconnus, et les robots devront les éviter en utilisant des sémaphores.

La performance du système de navigation final en termes de temps, de calcul et de précision dépend donc de celle de chacune des étapes de calcul précédentes, mais aussi du choix effectué quant au modèle des capteurs et à celui de l'environnement.

Le présent travail a pour but la mise-en-œuvre d'un système de navigation basé sur des algorithmes génétiques. Le modèle de l'environnement étant une matrice 2D, l'ensemble des robots, qui ne connaissent pas l'environnement, doivent atteindre des points d'arrivée à partir de points de départ. Ils doivent déterminer la trajectoire la plus optimale (le plus court chemin), éviter les obstacles, ainsi que la collision inter-robots.

Organisation du mémoire

Le document est composé de cinq (05) chapitres:

Le premier chapitre est consacré à la présentation des robots mobiles. Un aperçu général sur le domaine de la robotique mobile est abordé pour examiner la typologie des robots mobiles, les différentes parties constitutives et les architectures de contrôle existantes pour un robot mobile.

Dans le deuxième le deuxième chapitre, nous présentons un état de l'art de la navigation en général et la navigation coopérative et autonome dans des environnements dynamiques avec contournement d'obstacles en particulier.

Dans le troisième chapitre, nous décrivons la coopération dans les systèmes multi-robots est présenté. Nous montrons les limites de la décomposition par tâche des missions robotiques et le problème de déploiement d'une mission sur une flotte de robots en particulier.

Dans le quatrième chapitre, consacrée à notre contribution à la problématique de recherche, nous présentons une étude conceptuelle sur les techniques de communication entre les entités autonomes intelligentes (agents ou robots) distribués dans notre système de navigation où les robots doivent coopérer pour prendre en charge les tâches relevant du chantier de construction.

Dans le dernier chapitre, nous aborderons l'application d'un planificateur génétique pour la navigation autonome de plusieurs robots mobiles pour accomplir une tâche spécifique par le robot qui est la poursuite d'une trajectoire de référence, en d'autre terme, la convergence vers un but et l'évitement d'obstacles. Des exemples de simulations sont fournis afin de mettre en évidence les résultats des méthodes proposées pour la navigation coopérative et autonome de plusieurs robots mobiles dans un chantier de construction.

Nous terminons ce mémoire avec une conclusion générale qui résume notre contribution dans le cadre de la problématique traitée, et quelques perspectives intéressantes autour du travail que nous avons mené.

Chapitre I :

Les Robots Mobiles

Chapitre I : Les Robots Mobiles

I.1. Introduction

La robotique qui est un domaine de recherche important fait appel aux connaissances croisées de plusieurs disciplines ; son objectif étant de permettre au robot d'interagir rationnellement avec son environnement sans intervention humaine [LAU01]. Les robots mobiles ont une place particulière en robotique. Leur intérêt réside dans leur mobilité qui ouvre des applications dans de nombreux domaines. Comme les robots manipulateurs, ils sont destinés à assister l'homme dans les tâches pénibles (transport de charges lourdes), monotones ou en ambiance hostile (nucléaire, marine, spatiale, lutte contre l'incendie, surveillance…).

L'aspect particulier de la mobilité impose une complexité technologique et méthodologique qui s'ajoute en général aux problèmes rencontrés par les robots manipulateurs. La résolution de ces problèmes passe par l'emploi de toutes les ressources disponibles tant au niveau technologique (capteurs, motricité, énergie) qu'à celui du traitement des informations par l'utilisation des techniques de l'intelligence artificielle ou de processeurs particuliers (vectoriel, cellulaires).

L'autonomie du robot mobile est une faculté qui lui permet de s'adapter ou de prendre une décision dans le but de réaliser une tâche malgré un manque d'informations préliminaires ou éventuellement erronées. Dans d'autres cas d'utilisation, comme celui des véhicules d'exploration de planètes, l'autonomie est un point fondamental puisque la télécommande est alors impossible par le fait de la durée du temps de transmission des informations.

Aussi, la robotique mobile est un domaine dans lequel l'expérience pratique est primordiale. Au début, les robots ont fait leur apparition dans l'industrie grâce aux capacités des manipulateurs. Ces derniers, qui sont des bras imitant le bras humain, sont capables d'utiliser différents outils pour accomplir diverses tâches. Au fur et à mesure, ces bras ont gagné de nouveaux degrés de liberté à l'aide de plateformes mobiles ; c'est l'apparition des robots mobiles à roues [FIL04]. Les premiers robots autonomes utilisaient la roue, mais pour certaines applications où la géométrie de l'environnement diffère (terrains accidentés, endroits difficiles à atteindre), les recherches, inspirées du monde des animaux, se sont orientées vers d'autres moyens de locomotion. Ainsi, les robots à pattes sont apparus et on connaît déjà les robots hexapodes et les robots bipèdes. Des problèmes tels que la conception mécanique, la perception de l'environnement, la modélisation de l'espace de travail, la planification de trajectoires sans collision, et la synthèse des lois de contrôle non linéaires sont des exemples de différents sujets de recherche dans la robotique mobile [SHU06]. L'intérêt indéniable de cette 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'environnements non structurés dû principalement à la nécessité d'aider ou de remplacer l'intervention humaine.

L'objectif de ce chapitre est de donner un bref aperçu sur le domaine de la robotique mobile, et en particulier, la navigation autonome. Nous y présenterons l'autonomie du robot mobile, ainsi que les grandes classes et les types de capteurs utilisés. Nous nous intéresserons principalement aux méthodes de contrôle développées pour la commande de ces machines autonomes.

I.2. Contexte

L'objet de la robotique est l'automatisation de systèmes mécaniques lesquels, en les dotant de capacités de perception, d'action et de décision, le but étant de leur permettre d'interagir rationnellement, et de façon autonome, avec leur environnement.

Depuis les premiers automates jusqu'aux systèmes disponibles en ce début du XXI^{ème} siècle, on mesure, tout à la fois, le chemin parcouru et celui restant à parcourir avant de réaliser les rêves qui animaient les pionniers. La robotique est un domaine de recherche qui se situe au carrefour de l'intelligence artificielle, de l'automatique, de l'informatique et de la perception par ordinateur. Cette interdisciplinarité est à l'origine d'une certaine complexité. Des applications dans des domaines aussi variés que l'industrie manufacturière, le spatial, l'automobile ou plus récemment les loisirs et le secteur médical, démontrent aujourd'hui l'intérêt économique et social de ces recherches.

La robotique mobile autonome vise plus spécifiquement à concevoir des systèmes capables de se déplacer de façon autonome. Les applications directes se situent notamment dans les domaines de l'automobile, de l'exploration planétaire ou de la robotique de service par exemple.

De nombreuses applications qui ne découlent pas directement des avancées de la robotique, mais qui utilisent ses méthodes et ses développements, restent à découvrir.

Notre thèse a pour cadre général la navigation autonome des robots mobiles et elle se focalise sur un type de système et d'application spécifiques : "*la navigation autonome des robots mobiles, application dans un chantier de construction*".

I.3. Généralités sur la robotique mobile

Depuis les années 1960, la robotique mobile a connu un essor considérable. Les progrès techniques ont permis en effet la construction de robots de plus en plus perfectionnés : de nombreux périphériques, une puissance de calcul accrue et des moyens de communication performants. Ainsi, la mobilité autonome des robots est devenue un sujet de recherche développé par tous les pays industrialisés ; qu'il s'agisse de robots mobiles à pattes, à roues ou même sous-marins et aériens, les applications sont vastes et multiples : robots de services, surveillance, construction, nettoyage, manipulation de charges, automobile intelligente, robots d'intervention, robots d'exploration planétaire ou de fonds marins, satellites, robots militaires, etc. De ce fait, le marché potentiel de la robotique est

devenu considérable, même s'il faut pour cela résoudre des problèmes plus importants et plus fondamentaux que prévus, initialement, dans la quête vers la machine intelligente [TIG96].

I.3.1. Définition d'un robot mobile

Un robot mobile est un système mécanique, électronique et informatique qui agit physiquement sur son environnement en vue d'atteindre un objectif qui lui a été assigné. Cette machine est polyvalente et capable de s'adapter à certaines variations de ses conditions de fonctionnement. Elle est dotée de fonctions de perception, de décision et d'action. Ainsi, le robot devrait être capable d'effectuer, de différentes manières, des tâches diverses et d'accomplir correctement sa propre tâche, même s'il rencontre de nouvelles situations inopinées [www3].

Figure I.1 : Le robot mobile « Sejourner » utilisé pour la mission pathfinder de la NASA [www3]

L'appellation « *robot mobile* » rassemble tous les types de robots dont la caractéristique commune est la faculté de se mouvoir. Cependant, la manière qui dépend du domaine d'utilisation des robots, fait la différence de ces derniers.

Ainsi, la mobilité par les roues est, par exemple, la structure mécanique la plus communément appliquée. Cette technique assure, selon l'agencement et les dimensions des roues, un déplacement dans toutes les directions avec une accélération et une vitesse importantes.

I.3.2. Composants matériels d'un robot mobile

À la base, un robot mobile est constitué de composants matériels et logiciels. Parmi les premiers (composants matériels), on trouve une plateforme mobile à laquelle sont rattachés tous les autres composants comme les capteurs, les actionneurs et une source d'énergie (batteries).

D'autres organes tels que des bras manipulateurs peuvent lui être ajoutés pour une application particulière [BEA06].

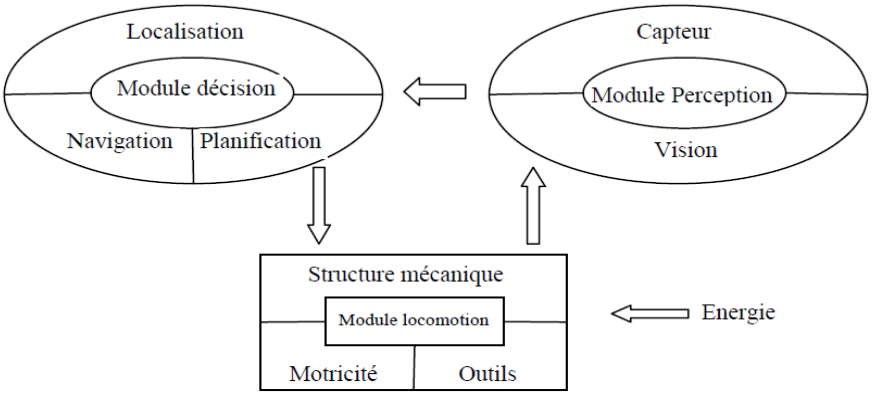


Figure I.2 : Architecture modulaire d'un robot mobile.

I.4. Autonomie d'un robot mobile

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

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

- Il est capable d'accomplir, sans intervention humaine, les objectifs pour lesquels il a été conçu.

- Il est capable de choisir ses actions afin d'accomplir ses missions.

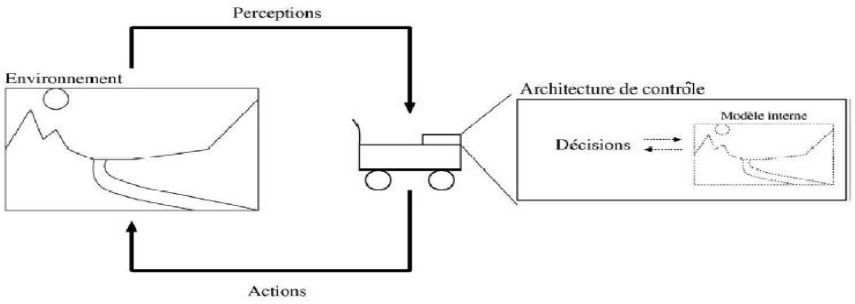
Une machine autonome peut être définie comme étant l'association d'un système d'intelligence artificielle avec des capacités de perception, de modélisation de son environnement et de son propre état. Elle doit être aussi dotée de facultés d'action sur son propre état et sur son environnement. Pour cela, le robot doit suivre le schéma correspondant au paradigme (*Percevoir-Décider-Agir*) [REI94]. La figure I.3 (a) présente l'interaction du robot avec son environnement. La manière dont le robot mobile gère ses différents éléments est définie par son architecture de contrôle qui peut éventuellement faire appel à un modèle interne de l'environnement ou une stratégie intelligente pour lui permettre de planifier ses actions à long terme.

Bien que, comme nous le verrons par la suite, plusieurs architectures existent au niveau de la satisfaction de ce paradigme, l'activité d'un tel robot se ramène aux tâches suivantes comme illustré sur la figure I.3 (b) :

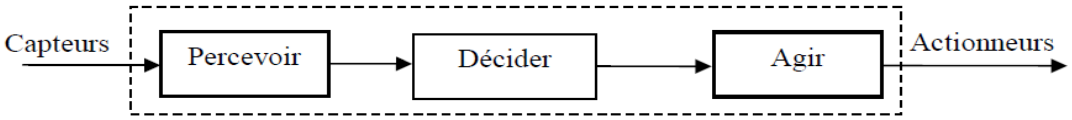
- **Percevoir :** le robot doit acquérir des informations sur l'environnement dans lequel il évolue par l'intermédiaire de ses capteurs. Ces informations permettent de mettre à jour un modèle de l'environnement (architectures hiérarchiques ou délibératives) ou peuvent être directement utilisées comme entrées de comportement de bas niveau (architectures purement réactives).

- **Décider :** le robot doit définir des séquences d'actions résultant d'un raisonnement appliqué sur un modèle de l'environnement ou répondant de manière réflexe à des stimuli étroitement liés aux capteurs.

- **Agir :** il doit exécuter les séquences d'actions élaborées en envoyant des consignes aux actionneurs par l'intermédiaire des boucles d'asservissements.



(a)



(b)

Figure I.3 : (a) Interaction entre le robot et l'environnement et (b) étape de traitement automatique

Selon *Steels*, un système est autonome s'il développe les lois et les stratégies qui lui permettent de contrôler son comportement [STE95]. En fait, l'autonomie est une capacité relative et on peut considérer qu'il existe une progression insensible du niveau le plus bas au plus élevé. Elle représente la capacité à choisir une stratégie, en termes de sous-buts pour arriver à un but fixé. La limite basse de l'autonomie est constituée par un système automatique qui s'autorégule en fonction de lois préétablies, i.e. qui ne génère pas les lois que les activités de régulation cherchent à satisfaire.

L'autonomie désigne littéralement la capacité d'une entité à « se gouverner par ses propres lois ». Cela signifie entre autres le choix de ses buts. Dans la littérature robotique, on peut distinguer trois points de vue :

- L'autonomie au sens fort qui nous ramène à des questions de volonté et de but propre.
- L'autonomie au sens faible qui désigne la capacité de maintenir sa structure au sein d'un milieu complexe à travers des mécanismes tels que l'auto-organisation, l'évolution, l'adaptation et l'apprentissage.
- L'automatisme ou l'absence de contrôle extérieur est le sens implicite que l'on attribue généralement au mot autonomie.

« S'adapter » consiste, pour un robot, à modifier son comportement pour faire face à des changements internes ou externes afin de maintenir certaines propriétés. L'adaptation se rapporte toujours à quelque chose : adaptation d'un groupe à la panne d'un individu, adaptation à la présence de passants dans l'environnement, …etc. Les bases de l'autonomie sont concrètement les propriétés qu'un robot doit exhiber pour être autonome, par exemple:

- Navigation et localisation pour accéder à tous les points de l'environnement.
- Surveillance, c'est-à-dire détection d'évènements anormaux.
- Efficience dans son travail, en adoptant une stratégie de patrouille.

Un robot complètement autonome n'est pas forcément souhaitable. On peut vouloir en prendre le contrôle à certains moments ou intervenir indirectement à travers la modification de certains paramètres. Le terme d'autonomie ajustable désigne justement l'interruption temporaire de l'automatisme, du fait d'un superviseur ou de l'agent lui même, afin d'accroître l'efficacité du système [SEM04]. Le choix du degré d'automatisme dont doit être doté un agent autonome dépend de l'architecture de contrôleur.

1.4.1. Autonomie du mouvement et autonomie décisionnelle

L'autonomie du mouvement passe par la détermination des déplacements, de manière planifiée et réactive, et par la commande des déplacements. Si les méthodes mises en jeu sont fortement dépendantes des modèles cinématiques et dynamique des engins considérés, des méthodes génériques commencent à apparaître, surtout pour la planification de trajectoire et la commande référencée sur des éléments de l'environnement. Dans ce contexte, la taille de l'environnement considéré nous amènera plus particulièrement à nous intéresser à l'intégration de méthodes locales de gestion des mouvements avec des méthodes plus globales, qui considèrent l'ensemble des informations disponibles sur l'environnement et la mission à réaliser. Ces derniers aspects relèvent surtout de l'autonomie décisionnelle : il s'agit de déterminer les modalités de déplacement à adopter et les informations à acquérir.

Les systèmes automatiques sont entièrement prévisibles, du moment que leur état interne est connu, alors que les systèmes autonomes ne le sont pas, puisqu'ils sont capables de prendre eux-mêmes des décisions en fonction de critères qui peuvent échapper à l'observateur. L'autonomie implique la liberté de contrôle. Un système autonome prend ses décisions lui-même, il n'est pas contrôlé par un agent externe. La prise de décision implique une capacité à évaluer des alternatives en fonction d'un état courant et de l'expérience acquise par le passé [ARN00].

Posée dans le contexte d'environnements larges, éventuellement dynamiques, enrichie par la considération de systèmes multi-robots pas nécessairement homogènes, et de contraintes de communication et de gestion des ressources, cette problématique est encore très ouverte. La possibilité d'interactions avec des opérateurs, distants ou non, doit aussi être considérée à différents niveaux au sein des robots. Ces problèmes d'autonomie ajustable passent notamment par le développement de concepts d'organisation des processus décisionnels (architecture de contrôle).

1.4.2. Autonomie envisageable pour les robots actuels

Il ne paraît pas suffisant de viser le développement d'un ensemble de fonctionnalités indépendantes, sous forme de comportements ou de modules, mais un ensemble cohérent où toutes les fonctionnalités puissent être mise en œuvre sur des démonstrateurs dans une même unité de lieu et de temps. La phase d'étude des missions a permis d'extraire des tâches "robotisables" que l'on peut décomposer en comportements sensorimoteurs ou en fonctions de surveillance. Les premiers se distinguent des suivants par la présence d'actions motrices. Ces éléments sont listés ci-après [DAL01]:

- faire un déplacement dans une direction et selon une distance données ;
- se diriger vers un amer visuel (désigné au robot par un opérateur humain) ;
- suivre un guide visuel au sol (bord de mur, de trottoir, d'accotement, couloir) ;
- s'orienter dans une direction donnée ;
- contourner un obstacle ;
- explorer une zone délimitée ;
- porter ou déposer une charge ;
- collecter un objet ;
- se servir d'un outil spécifique ;
- utiliser des armes non létales ;

I.5. Classification des robots mobiles

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

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

- *Les contraintes dynamiques*, liées aux effets du mouvement (accélérations, vitesses bornées, présence de forces d'inertie ou de frottement). Ces facteurs influent sur le mouvement exécuté.

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

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

I.5.1. Robot mobile à roues

Compte tenu de la simplicité du mécanisme de locomotion utilisé, ce type de robot est le plus répondu actuellement. La plupart des robots mobiles à roues opèrent dans des sites aménagés, des sites industriels ou des environnements intérieurs; mais il existe également des applications en

environnements extérieurs, comme l'exploration spatiale [FIL04]. La grande majorité des robots de ce type présente des contraintes de non holonomie qui limitent le mouvement instantané que le robot peut réaliser, car il existe pour toutes ces roues un point unique (*centre instantané de rotation* (CIR)) de vitesse nulle autour duquel le robot tourne de façon instantanée. La commande se fait par la motorisation des roues installées. Ces contraintes augmentent la complexité du problème de planification de trajectoire et son contrôle [LAU01], [SUT98]. Les robots mobiles à roues peuvent être classés en plusieurs types avec des propriétés intéressantes: uni-cycle, différentielle, tricycle, de type voiture, omnidirectionnelle et à traction synchrone [BAY07]. Les modèles des robots mobiles à roues existant sont illustrés sur les figures suivantes (Figure I.4 jusqu'à la Figure I.9).

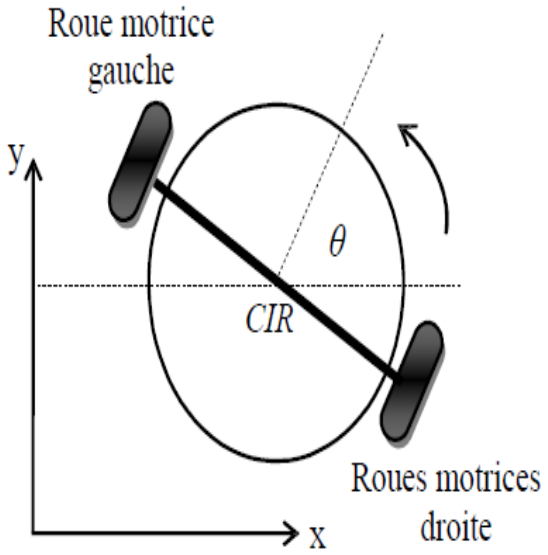


Figure I.4 : Robot mobile de type uni cycle.

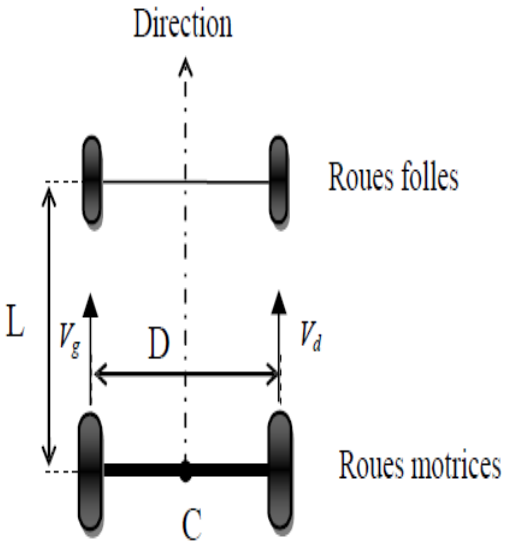


Figure I.5 : Robot mobile de type différentiel.

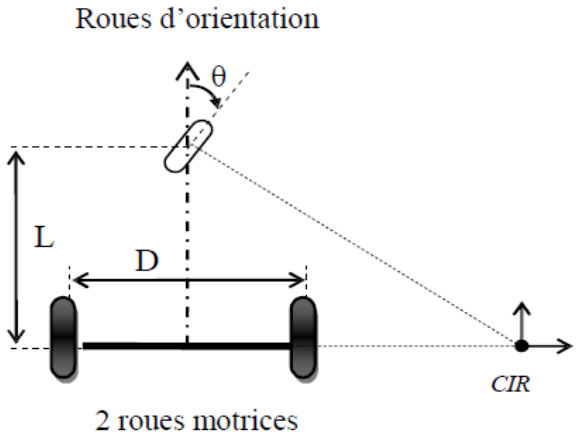


Figure I.6 : Robot mobile de type tricycle.

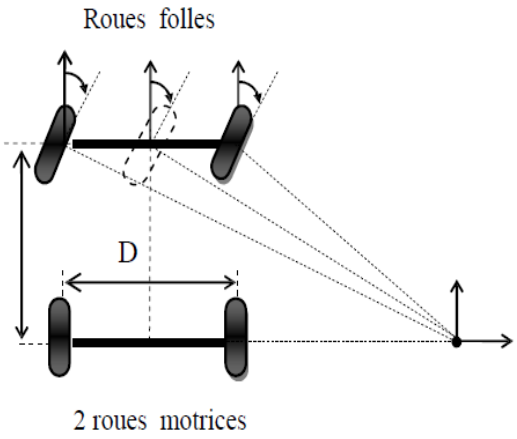


Figure I.7 : Robot mobile de type voiture.

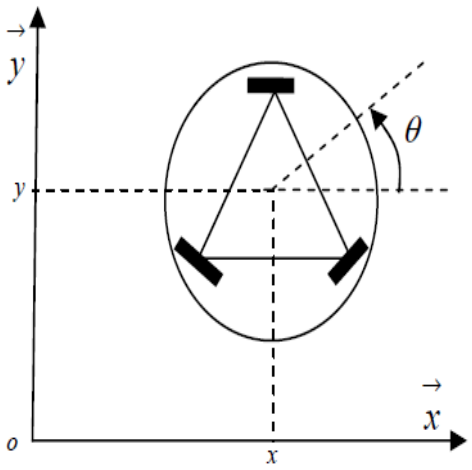


Figure I.8 : Robot omnidirectionnel.

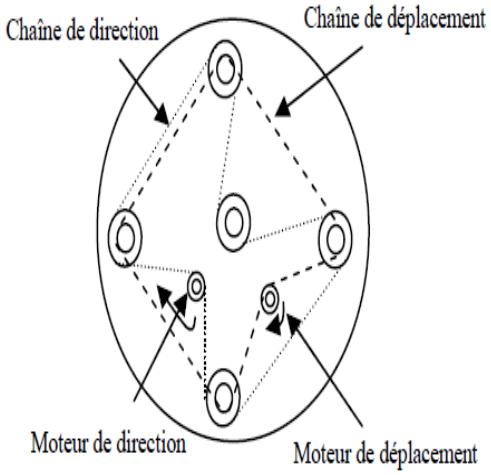


Figure I.9 : Robot mobile à traction synchrone.

I.5.2. Robot mobile utilisant la chenille

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

I.5.3. Robot mobile à pattes

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

I.5.4. Autres moyens de locomotion

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

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

I.6. Les capteurs

La commande des robots mobiles est basée sur deux types d'informations importantes; les informations proprioceptives et les informations extéroceptives [BOR96], [SHU06]. Le système de perception est très important pour la sécurité du robot si l'environnement est encombré d'obstacles fixes

ou bien mobiles (autres robot). Pour se focaliser sur le problème de navigation, nous allons nous restreindre dans ce chapitre aux capteurs utiles pour la tâche de navigation.

I.6.1. Capteurs intéroceptifs

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

I.6.2. Capteurs extéroceptifs

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

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

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

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

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

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

I.7. Les Actionneurs

Pour bouger à l'intérieur de son environnement et interagir avec celui-ci, un robot est équipé d'actionneurs, peut être muni d'un ou de plusieurs moteurs pouvant faire tourner ses roues afin d'effectuer des déplacements.

Généralement, les roues du robot sont contrôlées par deux commandes motrices, soit une vitesse d'avancement et un taux de rotation. Habituellement, ces commandes s'expriment en mètres par seconde (m/s) et en degrés de rotation par seconde (deg/s)[BEA06].

Le type de locomotion définit deux types de contraintes [LAR03]:

-Les contraintes cinématiques, qui portent sur la géométrie des déplacements possibles du robot;

-Les contraintes dynamiques, liées aux effets du mouvement (accélérations bornées, vitesses bornées, présence de forces d'inertie ou de friction).

I.8. Composants décisionnels d'un robot mobile

Afin de faire fonctionner un robot mobile, plusieurs modules logiciels sont mis à contribution. Ces modules peuvent servir à interpréter les données perçues par les capteurs afin d'y extraire des informations, ou à traiter des commandes de haut niveau pour générer d'autres commandes à un niveau inférieur. Les modules les plus fréquemment utilisés sont les modules de vision, localisation, de navigation, de planification et de séquence ment d'activités du robot. Ces aspects de ces modules logiciels seront donnés avec plus de détails dans les sections suivantes.

I.8.1. La vision

La vision fournit une grande quantité de données en provenance de l'environnement et permet d'entreprendre une interaction intelligente avec les environnements dynamiques (Évitement d'obstacles mobiles, rendez-vous avec autres agents mobiles. . .). De ce fait, il n'est pas surprenant de trouver une grande quantité de recherches sur le développement de capteurs qui essaient d'imiter le système visuel humain. De plus, les capteurs visuels utilisés par les robots intelligents doivent avoir les mêmes sensibilités et réponses à la lumière que notre système de vision. En robotique, au cours des deux dernières décennies, les innovations technologiques concernant la fabrication de caméras et l'évolution des ordinateurs ont permis d'intégrer des systèmes complexes de vision dans les systèmes embarqués, que ce soit sur des robots mobiles pour la navigation autonome ou sur des véhicules pour l'aide à la conduite. La vision artificielle revêt une importance toute particulière car elle permet de fournir à la machine les capacités nécessaires pour réagir avec son environnement ; elle fournit les représentations à partir desquelles le robot prend des décisions [SIE04], [AVI05].

I.8.2. Localisation

Un robot mobile doit toujours connaitre sa situation pour se déplacer d'un point à un autre, soit en identifiant des repères, on parle de localisation absolue, soit tout simplement en mesurant les distances parcourues et les directions empruntés depuis sa position initiale. Les méthodes de localisation se regroupes en deux catégories, soit : [BAY09], [DUR89]

- La localisation à l'estime ou relative qui est obtenue par des informations issues de capteurs proprioceptifs ;

- La localisation absolue qui est obtenue par des informations issues de capteurs extéroceptifs.

- Localisation à l'estime ou relative: obtenue par des informations issues des capteurs proprioceptifs, et consiste à déterminer la variation des coordonnées de position lors d'un déplacement en mesurant tout simplement les distances parcourues et les directions, empruntées depuis sa position initiale.

- Localisation absolue: obtenue par des informations issues de capteurs extéroceptifs; le robot doit toujours connaître sa situation pour se déplacer d'un point à un autre en identifiant des repères artificiels, la méthode des balises est la plus employée.

I.8.3. Représentation de l'environnement

La localisation a besoin, sous une forme ou une autre, d'une représentation informatique de l'environnement. Il existe deux types principaux de cartes : les grilles d'occupation et les graphes. Dans le premier cas, l'environnement est quadrillé en cases.

Durant l'exploration, à l'aide des sonars ou d'un télémètre laser, chacune de ces zones se voit attribuer une probabilité d'être occupée par un obstacle. Le résultat est une image probabiliste des murs et des espaces libres. Cette représentation donne lieu à une localisation sous forme de coordonnées qui désignent une case du quadrillage.

L'un des inconvénients des grilles d'occupation vient de la grande quantité de données qu'il faut manipuler. L'exploitation de ces représentations est coûteuse en temps de calcul [NAG98].

Les cartes topologiques ne s'intéressent qu'aux points clefs du milieu. Elles ont la forme d'un graphe où les sommets figurent des endroits spécifiques (intersections de couloirs, changement de classe de perception...) et les arcs l'existence d'un chemin entre deux sommets. C'est une vue plus abstraite de l'environnement que celle proposée par les grilles d'occupation. Elle vise à ne conserver que l'information utile.

Si le quadrillage est précis, il manque de discernement. Un long couloir uniforme ne nécessite pas à priori une accumulation de données sous formes de milliers de probabilités. Notons que ces deux types de cartes ne sont pas exclusifs [THR96].

D'autres formes de représentation existent, dont les représentations implicites. Ici, il n'y a pas de cartes mais apprentissage d'un comportement adapté au milieu.

- **Notion d'environnement dynamique**

Un environnement est dit dynamique s'il comporte des obstacles susceptibles de changer au cours du temps. Soit le cas des obstacles qui se déplacent (un piéton, un véhicule, …), ou ceux qui changent de forme ou de ceux qui apparaître/disparaître (une porte coulissante semble « disparaître » dans le mur quand elle s'ouvre).

- **Notion d'incertitude**

Une information est incertaine, si elle est bruitée (mauvaises conditions de mesures), incomplète (obstruction d'un capteur ou portée limitée, absence d'informations sur l'évolution d'un objet ou d'un phénomène) ou imprécise (les glissements des roues par rapport au sol sont observables mais rarement mesurables avec précision).

I.8.4. Planification

Une fois que le robot dispose d'une représentation de l'environnement et d'une estimée de sa position, il convient de s'en servir pour générer les tâches qui lui permettront d'atteindre son objectif.

Pour cela plusieurs méthodes sont possibles. Le choix d'une méthode de planification est guidé par deux questions :

1. Quel type d'espace utilisé : l'espace de travail ou l'espace des configurations?

2. Quel type de méthodes : des méthodes exactes ou des méthodes approchées?

Les méthodes exactes sont basées sur une exploitation complète de la description de l'environnement. Par opposition, les méthodes approchées réalisent tout d'abord une discrétisation de l'environnement sous forme de grilles régulières ou irrégulières. L'espace libre ainsi représenté est un sous-ensemble de l'espace libre réel.

Alors que les méthodes exactes sont susceptibles d'être complètes, les méthodes approchées ne le sont jamais [REI94].

I.8.5. Navigation

La navigation autonome d'un robot mobile est tout simplement la capacité d'aller d'une position initiale à une position finale de manière autonome.

Réaliser une tâche telle que se déplacer vers le point de coordonnées (x, y), aussi simple puisse-t-elle paraître à un humain, requiert la mise en œuvre de fonctionnalités potentiellement complexes de perception/décision/action.

I.9. Détection d'obstacles et localisation

La perception de son environnement d'évolution est la base de tout système autonome.

Sans une bonne perception et interprétation de ce qui l'entoure, un robot ne peut pas prendre de décision correcte. Cette partie vise à décrire les différents moyens mis à disposition au robot pour localiser les obstacles qui l'entourent. Ensuite les différentes méthodes de localisation du robot lui-même sont abordées. L'idée est de permettre au final de créer un modèle, plus ou moins simplifié, des interactions entre le robot et son environnement. Cette étape est nécessaire et primordiale pour la navigation d'un robot mobile autonome.

Pour cela, un robot est équipé de capteurs proprioceptifs qui fournissent des informations sur le robot lui-même, et extéroceptifs qui fournissent des informations sur ce qu'il y a autour de lui (son environnement).

I.9.1. Détection d'obstacles et cartographie

Les capteurs permettant de fournir des informations sur l'environnement extérieur peuvent être classés en deux catégories, passifs et actifs [PRU96]. Dans le premier cas, on se contente de recueillir et d'analyser une énergie fournie par l'environnement, typiquement la lumière. Dans le second cas, c'est au capteur de générer une énergie, et de récupérer cette énergie après interaction sur le milieu extérieur. C'est le principe de base des télémètres (capteurs de mesures de distances), qui sont largement utilisés pour tracer des cartes en ligne de l'environnement dans lequel évolue le robot. Les télémètres laser à balayage sont fréquemment utilisés pour la navigation de robots avec de très bonnes performances notamment en intérieur.

Le principe de ces télémètres repose sur le calcul du temps aller-retour mis par une impulsion lumineuse pour revenir sur le capteur. Une onde infrarouge de faible puissance est émise par la diode laser, et au même moment un chronomètre informatique est lancé. L'onde se réfléchit sur le premier objet rencontré en chemin, et revient sur le détecteur du capteur. Le temps mis par l'onde pour faire l'aller-retour permet de déterminer la distance de l'objet. Un miroir tournant motorisé permet de balayer toute une gamme d'angles devant le télémètre, dans le plan de balayage qui est parallèle au sol. La précision de ces appareils et leur robustesse aux variations de température en font des outils très intéressants pour les applications en robotique mobile de faible/moyenne vitesse.

Les capteurs ultrasonores utilisent des ondes sonores de fréquence non perceptible par l'oreille humaine, généralement dans la fourchette 20-200 khz. De la même manière que les télémètres laser, ils sont basés sur le principe de la mesure du temps aller-retour lors de la réflexion sur un obstacle. C'est la méthode employée par certains animaux pour percevoir leur environnement, comme les chauves-souris ou les dauphins : l'écholocalisation. Un avantage de ces capteurs est que contrairement aux télémètres, l'onde qu'ils émettent n'étant pas focalisée, ils perçoivent beaucoup plus facilement des éléments filiformes comme des pieds de chaises ou des grillages. Par contre leur portée est faible, et ils sont moins adaptés aux milieux de propagation non isotropes comme l'air.

Un des inconvénients des capteurs ultrasonores par rapport aux télémètres lasers est la divergence importante du faisceau ultrasonore, qui s'apparente plus à un cône qu'à un faisceau.

Généralement l'ouverture de l'angle est de plusieurs dizaines de degrés, ce qui rend la localisation des obstacles imprécise. Ces capteurs sont donc plutôt utilisés pour des mesures à courte distance (de quelques centimètres à quelques mètres). Ils sont relativement sensibles aux variations de température, et la fréquence de mesure dépend de la distance maximale de détection (plus cette distance est grande, moins la fréquence d'acquisition des mesures est élevée).

L'avantage de ces capteurs est qu'ils sont moins onéreux qu'un télémètre laser, et ils sont souvent utilisés dans des applications en intérieur avec des espaces de navigation assez restreints.

Les capteurs passifs se servent directement de l'énergie émise par l'environnement. C'est typiquement le cas des systèmes de vision par caméra en stéréo vision. La reconnaissance de primitives entre deux images permet d'évaluer la position/orientation d'un objet, et ainsi d'évaluer la profondeur. L'utilisation simultanée de deux caméras est cependant nécessaire pour y parvenir. Plus de deux caméras peuvent également être utilisées, de manière à améliorer la robustesse de la méthode.

La vision omnidirectionnelle s'avère également très intéressante dans le cadre d'applications en robotique mobile, dans le sens où elle permet de surveiller en même temps tout ce qui se passe autour du robot. La caméra est placée face à un miroir parabolique ou hyperbolique. L'image est complètement distordue ce qui complique la mesure de distances, mais l'aspect vision panoramique offre des avantages pour l'évitement d'obstacles dynamiques en environnement structuré. Les verticales deviennent des radiales, et les horizontales des arcs de cercle. L'utilisation de systèmes à base de vision est fortement développée, et pas seulement dans le domaine de la robotique. Mais globalement, ce type de système reste fortement tributaire de la qualité de l'énergie recueillie : influence de la luminosité ou encore du contraste.

I.9.2. Fusion de données multi capteurs et cartographie

La localisation d'un robot mobile s'effectue par la mise en correspondance de différentes sources extéroceptives et proprioceptives. Généralement il s'agira de confronter les mesures de déplacements prises par odométrie avec une méthode de localisation absolue : soit reconnaissance et calcul de distance par rapport à des balises de position connue, soit mise en correspondance avec une carte construite en ligne et/ou présente dans une base de données, soit encore localisation externe du robot par des capteurs dans l'environnement (GPS). Le principe le plus simple pour effectuer cette mise en correspondance, consiste à utiliser les mesures de localisation absolue pour recaler périodiquement l'état du robot, obtenu par intégration des déplacements mesurés par l'odomètre. Cette méthode, bien que simple à utiliser, présente le problème de ne pas utiliser conjointement les différents moyens de mesure, mais successivement. Ainsi on ne tient pas compte des incertitudes liées tant à l'odométrie qu'à la méthode de localisation absolue.

Cependant nous savons que quelque soit la technologie utilisée pour la prise d'informations, aucune mesure n'est parfaite, il existe toujours une part d'incertitude sur celle-ci. Ces incertitudes peuvent provenir soit du principe de mesure lui même, soit des imperfections technologiques. Typiquement, pour une mesure de distance avec un télémètre laser, on trouve des erreurs systématiques d'une quinzaine de millimètres en moyenne (erreur constante intrinsèque au télémètre utilisé), et une erreur statistique de 5 mm environ. Lorsque plusieurs méthodes de mesure sont utilisées conjointement, le principe utilisé pour mettre en concordance les informations consiste à effectuer une moyenne pondérée des différentes mesures par la confiance que l'on accorde à chacune (inversement à leur variance donc).

Pour des obstacles statiques, nous obtenons des mesures récurrentes selon une certaine fréquence d'acquisition. Pour des mesures récursives, la précision peut être améliorée en utilisant un filtre de Kalman. Ce type de filtrage, très utilisé notamment en automatique, est un filtre statistique qui permet de réduire les incertitudes au fur et à mesure de l'acquisition de nouvelles mesures. Cette méthode est particulièrement utilisée en robotique pour la localisation du robot relativement aux obstacles [JET99]. L'algorithme utilise les connaissances sur la dynamique du robot et du système de mesure et sur les incertitudes associées à chaque mesure. Le calcul s'effectue en deux phases : une phase de prédiction de la mesure et de sa variance, suivie d'une phase de mise à jour de celle-ci par l'acquisition de nouvelles mesures. L'historique des mesures n'a pas besoin d'être gardé en mémoire, et le processus est récursif.

Pour pouvoir planifier les déplacements du robot, il est nécessaire d'établir une modélisation de l'environnement à partir des mesures des positions relatives des obstacles par rapport au robot. Concrètement il s'agit d'établir une cartographie locale des espaces où le robot pourra circuler, ou non, en considérant la position absolue du robot comme connue. Il existe 2 grands types de représentation pour l'environnement local : les cartes géométriques et les grilles d'occupation. Les premières peuvent être obtenues par traitement des données issues de mesures télémétriques, et en effectuant une reconnaissance des formes simples (murs, coins). A partir d'une connaissance des déplacements du robot et en comparant la carte courante avec des cartes mises en mémoire au fur et à mesure que le robot se déplace, la robustesse de la carte peut être améliorée [CAN01]. Les grilles d'occupation sont des cartes discrétisées, généralement sous forme de grille d'un certain nombre de lignes et de colonnes. A chaque case de la grille est soit associée une valeur booléenne pour dire si la case est

accessible par le robot ou non (occupée par un obstacle ou libre), soit une probabilité d'occupation (loi de Bayes).

I.9.3. Localisation

Les outils permettant la localisation d'un robot dans son environnement peuvent être classés en deux catégories : ceux par localisation à l'estime et ceux par localisation absolue [PRU96].

Le principe de la première catégorie consiste à intégrer des informations sur les vitesses ou les accélérations fournies par des capteurs proprioceptifs (odomètres, centrales inertielles).

L'avantage de ces méthodes est qu'elles sont indépendantes de l'environnement, par contre leur souci est leur manque de précision dû à la dérive temporelle. En effet les erreurs s'intégrant elles aussi au fur et à mesure du temps, il est nécessaire d'apporter régulièrement des recalages (Figure I.10 et Figure I. 11).

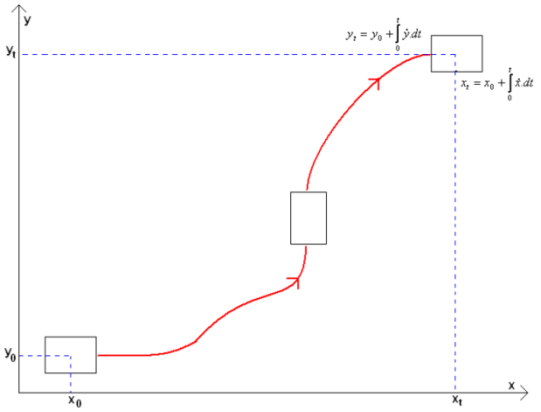


Figure I. 10 : Localisation à l'estime

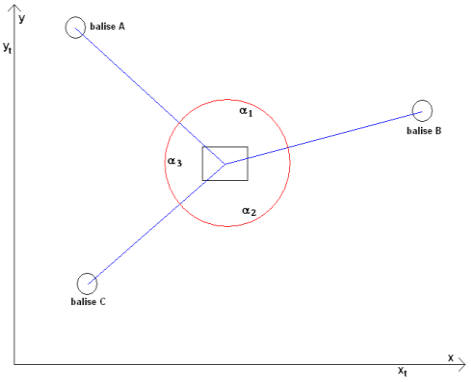


Figure I. 11 : Localisation absolue (méthode par triangulation)

Parmi les méthodes de localisation à l'estime, le système le plus simple et le plus couramment utilisé pour la mesure de déplacement du robot est l'odmétrie. L'hypothèse de roulement sans glissement que nous avons vu précédemment, nous permet de relier directement les déplacements du robot à la vitesse de rotation des roues. Par intégration des déplacements à chaque instant, on en déduit la position relative du robot par rapport à son point de départ.

L'odométrie est une méthode de localisation très courante, simple, mais également très rapidement imprécise. En effet à cause du glissement des roues sur le sol, les erreurs s'accumulent au fur et à mesure que le robot avance, ce qui implique d'importantes erreurs sur les longs parcours s'il n'y a pas de recalage régulier. Cette méthode est de ce fait fortement tributaire de la qualité du sol sur lequel le robot se déplace.

Les incertitudes sur le diamètre exact des roues, sur les paramètres géométriques du robot, sur la résolution des codeurs, génèrent des erreurs de type systématique, qui vont s'accumuler très rapidement en odométrie. Cependant, ces erreurs peuvent être identifiées et évaluées pour faire un re-calibrage du système et ainsi améliorer sa précision. Les erreurs non systématiques comme les glissements ou les irrégularités du sol, génèrent moins rapidement des erreurs, mais ne peuvent par contre pas être re-calibrées puisqu'on ne peut pas les prévoir.

Pour l'exploration martienne, où le terrain est fortement accidenté, l'utilisation de système d'odométrie classique est impossible. Pour cette application, Cheng a proposé une technique d'odométrie alternative, dite odométrie visuelle, basée sur la reconnaissance de points singuliers dans l'image vidéo du sol fournie par une caméra montée sur le robot [CHE06] (les points de Harris).

Connaissant le positionnement de la caméra par rapport au robot, le déplacement de ces points dans l'image permet d'évaluer les vitesses de déplacement du robot, et par intégration, de retrouver sa localisation relative par rapport à sa position initiale.

Le second type de méthode pour la localisation est la localisation absolue. Ces méthodes utilisent des éléments repérables par le robot dans l'environnement de navigation, de position connue, pour permettre au robot de se repérer relativement à ceux-ci. Ces éléments sont appelés des balises ou amers et sont dits soit réels, s'ils ont été placés spécialement pour permettre la localisation, soit virtuels s'il s'agit d'éléments présents naturellement [CHE06].

I.9.4. Localisation et cartographie simultanées

En robotique mobile, le SLAM (*simultaneous localization and mapping*) consiste, pour un robot évoluant en milieu inconnu, à tracer une carte de l'environnement et localiser simultanément le robot dans celle-ci. La carte est construite de manière incrémentale au fur et à mesure que le robot évolue dans le terrain. En croisant les données perçues avec les informations géographiques dont il dispose en mémoire, le véhicule est capable de se localiser par rapport à des cartes préexistantes. L'idée du SLAM est donc de traiter conjointement les problèmes connexes que sont la navigation et la localisation d'un robot autonome.

La méthode alternative, moins gourmande en temps de calcul, consiste à utiliser la « mémoire visuelle » du véhicule, en repérant des éléments caractéristiques, pour se localiser par rapport à ceux-ci. Ces éléments sont extraits sous forme de primitives visuelles, que le robot cherchera à retrouver dans les images qu'il perçoit pour suivre un chemin précis [COU09], [PRA06].

I.10. Problèmes en robotique mobile

On distingue sans trop d'ambiguïté un certain nombre de problèmes en robotique mobile. Bien évidemment, l'aspect matériel, qui consiste à choisir et dimensionner aussi bien la structure mécanique du système que sa motorisation, son alimentation et l'architecture informatique de son système de contrôle-commande apparaît comme le premier point à traiter. Le choix de la structure est souvent effectué parmi un panel de solutions connues et pour lesquelles on a déjà résolu les problèmes de modélisation, planification et commande. Le choix des actionneurs et de leur alimentation est généralement assez traditionnel. De la même façon, les architectures de commande des robots mobiles ne sont pas différentes de celles des systèmes automatiques ou robotiques plus classiques. On y distingue cependant, dans le cas général, deux niveaux de spécialisation, propres aux systèmes autonomes : une couche décisionnelle, qui a en charge de la planification et la gestion (séquentielle, temporelle) des évènements et une couche fonctionnelle, chargée de la génération en temps réel des commandes des actionneurs. Bien évidemment, l'architecture du robot dépend fortement de l'offre et des choix technologiques du moment. Les problèmes spécifiques à la robotique mobile n'apparaissent finalement que lorsque l'on dispose d'une structure mobile dont on sait actionner les roues. Tous les

efforts du roboticien vont alors consister à mettre en place les outils permettant de faire évoluer le robot dans son environnement de manière satisfaisante, qu'il s'agisse de suivre un chemin connu ou au contraire d'aller d'un point à un autre en réagissant à une modification de l'environnement ou à la présence d'un obstacle [BAY09].

I.11. Les grands axes de recherche dans la robotique mobile

Il existe de nombreuses thématiques de recherche dans le milieu de la robotique mobile autonome, ce qui montre qu'aujourd'hui encore le problème est entier. Le Groupement De Recherche en Robotique, qui regroupe la communauté des chercheurs internationale dans le domaine de la robotique, a dégagé 4 grands axes de travail autour desquels s'articulent les colloques du Groupe de Travail:

- Techniques de localisation et cartographie : cet axe regroupe tous les développements autour de la perception et de la localisation du robot. On y retrouve notamment les méthodes SLAM (Localisation et Cartographie Simultanées). Plus récemment l'utilisation de bases de données sous forme de cartes 2D ou 3D, mais également sous forme SIG (Système d'Informations Géographiques) a ouvert de nouvelles perspectives dans ce domaine. De manière générale la fusion de données est également un thème important, tant la nécessité de coupler diverses sources de mesures apparaît nécessaire pour améliorer la précision et garantir l'intégrité des informations,

- Contrôle et commande des véhicules : cet axe regroupe les thématiques liées à la planification de chemin, la génération de trajectoires, et la commande des robots de manière générale. Une prise en compte de plus en plus poussée des contraintes et de la dynamique des robots est nécessaire, pour adapter au mieux les robots à leur environnement. La bonne gestion des obstacles et la prise en compte des incertitudes de mesures sont également des points clés de cette thématique,

- La communication inter-véhicules : on retrouve ici tous les travaux liés à la coopération entre robots, et le contrôle de flottilles de véhicules,

- L'interprétation de scènes : les recherches dans ce domaine visent à pousser plus loin la perception de son environnement par le robot, que la simple reconnaissance des objets. En effet dans certaines applications il est nécessaire que le robot appréhende plus finement son environnement que par une simple détection et localisation des obstacles. Les travaux concernent notamment la perception multi-capteurs et la représentation dynamique des scènes.

La perception d'une part et la commande au sens large d'autre part sont donc les deux thèmes majeurs de recherche pour obtenir un robot mobile parfaitement autonome. Parmi les problématiques liées à la commande, celle de la navigation tient un rôle important : elle consiste à déterminer les trajectoires que le robot sera capable de suivre pour lui permettre d'évoluer correctement au milieu d'obstacles, en considérant qu'il dispose d'une méthode de cartographie.

I.12. Conclusion

Dans ce chapitre nous avons résumé toutes les notions de base nécessaires à la compréhension du domaine de la robotique mobile, ainsi qu'un panorama de techniques appliquées dans la planification et la navigation d'un robot mobile autonome. Après cet aperçu, dans le chapitre suivant nous présenterons une vision sur le domaine de la navigation autonome des robots mobiles. Nous allons aussi présenter les principales architectures utilisées pour le contrôle des robots mobiles en citant les avantages et les inconvénients de chacune.

Chapitre II :

La Navigation

Chapitre II : La Navigation

II.1. Introduction

La navigation autonome est un domaine de recherche très actif, c'est un thème de recherche dans des laboratoires de haut niveau. Le but pour un robot est de trouver le plus court chemin en évitant les obstacles d'un point de départ jusqu'au point d'arrivé. Ces dernières années, la navigation collective attire de plus en plus l'attention des chercheurs que la navigation d'un seul robot à cause de son efficacité, sa rapidité, sa robustesse et sa capacité à résoudre des tâches complexes. Dans ce chapitre nous présenterons un état de l'art, ce dernier offre une vision non exhaustive des thématiques de recherche associées au domaine de la navigation autonome, et présente l'ensemble des verrous scientifiques qu'il reste à lever pour aboutir au développement d'un robot mobile.

II.2. Le problème de navigation

Un navigateur désigne généralement un module chargé de faire se mouvoir le robot mobile dans son milieu d'évolution. Si on fait l'analogie avec le pilotage d'une voiture, se déplacer d'un endroit A à un endroit B nécessite au moins 3 tâches bien distinctes :

- réfléchir à un itinéraire pour se rendre au point B : prendre telle autoroute en direction de telle ville, prendre la sortie numéro tant, tourner à droite au niveau du premier rond point etc…

C'est ce que nous appelons la planification de chemin. Il est préférable de déterminer son itinéraire à l'avance, et pour y parvenir quand on effectue le trajet pour la première fois, il est nécessaire d'utiliser des cartes, ou une représentation préétablie, pour trouver le meilleur itinéraire possible,

- sur la route, il faut suivre du mieux possible l'itinéraire, en adaptant sa vitesse à la route, la circulation, aux limitations de vitesses, aux conditions météorologiques… etc. Il faut prendre correctement les ronds points, les virages, se frayer un passage dans la circulation. Il faut déterminer des trajectoires qui vont permettre de suivre du mieux possible l'itinéraire prévu, tout en s'adaptant aux conditions de circulation. Ces trajectoires seront dépendantes du véhicule que l'on conduit (voiture, moto … etc), chaque véhicule ayant des capacités propres. Ce travail d'adaptation, c'est le travail du navigateur.

- enfin, pour pouvoir suivre une trajectoire donnée, il faut appuyer sur l'accélérateur, tourné le volant, de manière générale il faut agir sur le véhicule. C'est le travail du pilote, qui doit agir sur les actionneurs du système pour appliquer la trajectoire souhaitée.

Dans notre étude, le navigateur a donc pour mission de générer des trajectoires pour le robot. La notion de trajectoire est différente de celle de chemin dans le sens où elle intègre des informations temporelles le long du parcours du robot, c'est-à-dire des vitesses et parfois des accélérations (dérivées premières et secondes). Concrètement, un navigateur se différencie d'un planificateur de chemin par les éléments suivants :

- Il travaille sur une carte de l'environnement plus localisée autour du robot.
- Il travaille à plus court terme que le planificateur, il prévoit moins loin dans le temps.
- Il intègre les contraintes liées au robot, il doit s'assurer que le robot est capable physiquement de réaliser les trajectoires demandées.

Dans l'architecture de contrôle, le navigateur reçoit un chemin de la part du planificateur, qui peut être sous différentes formes plus ou moins complètes : série de points de passage (*x, y*) pouvant contenir des informations supplémentaires comme l'angle d'orientation du robot, des

vitesses indicatives …etc. Concrètement, le rôle du navigateur consiste à générer des trajectoires qui soient réalisables par le robot (c'est-à-dire qui respectent le modèle cinématique du robot, les contraintes de non holonomie ainsi que les différentes contraintes liées aux actionneurs), qui n'intersectés pas avec les obstacles dans l'environnement local de navigation, et qui suivent au mieux le parcours proposé par le planificateur.

En fonction des méthodes utilisées, nous allons nous apercevoir que les délimitations entre Pilotage/navigation/planification sont plus ou moins définies, il peut y avoir un recouvrement des tâches entre ces 3 grandes fonctions. Par exemple le rôle de navigateur peut-être tenu en partie ou complètement par le planificateur de chemin, si celui-ci intègre les possibilités de mouvement du robot dans son calcul de parcours. Mais le navigateur est un point clé pour un robot mobile autonome, puisque c'est lui qui fait le lien entre les capacités de mouvement du robot et l'environnement.

II.3. Planification de mouvement

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

II.3.1. Planification globale de trajectoire

C'est la modélisation de l'espace de l'environnement par un graphe, ou la recherche de la trajectoire est basée sur l'utilisation des algorithmes des graphes, on peut citer: le graphe de visibilité, la décomposition cellulaire…etc [SHU06], [FIL04].

II.3.2. Planification locale de trajectoire

Généralement, l'environnement du robot mobile est inconnu, et le robot ne dispose pas, à priori, aucune information sur l'environnement. Il est nécessaire donc de réaliser une planification locale de type réflexe. Pendant le déplacement, le robot mobile doit analyser son environnement et prendre la décision en fonction de cette analyse [FIL04], [BOR91]. Les méthodes réactives de l'intelligence artificielle sont considérées comme des approches de planification locale.

II.4. Localisation

Afin d'exécuter le mouvement planifié, le robot doit se localiser dans son environnement en estimant la position et l'orientation par rapport à un repère fixe. L'estimation peut s'effectuer soit par une mesure des déplacements du robot, soit par une mesure de sa position absolue dans l'environnement. Plusieurs approches de localisation sont utilisées en robotique mobile. Elles peuvent être classées en trois catégories: localisation basée sur une carte, localisation par rapport à des balises et localisation par rapport à d'autres robots [BOR96] (section I.9.3.).

II.5. Le cheminement

Le cheminement est une mission importante pour un robot mobile. Il consiste à calculer les commandes envoyées aux actionneurs permettant de réaliser le mouvement planifié.

Un robot est un système dynamique commandé par bouclage pour la poursuite de sa trajectoire de référence [FIL04], [CHE11]. On s'intéresse dans ce travail à l'étude de cette fonctionnalité.

Dans cette partie, on peut considérer deux grandes familles de méthodes permettent à un robot d'atteindre une position souhaitée. Les méthodes sans trajectoire explicite (champs de potentiels, réseaux de neurones et logique floue) cherchent plutôt à contrôler le mouvement global du robot de manière à le guider vers son but. L'autre famille, celle des méthodes de suivi de trajectoire, telles que les fonctions transverses et les sorties plates, permet au robot de suivre une trajectoire de référence donnée, connaissant les contraintes cinématiques.

II.5.1. Méthodes sans trajectoire

Le premier type d'approche pour résoudre le problème de navigation d'un robot mobile autonome est de rendre la cible du robot attractive pour celui-ci, de manière à ce qu'il cherche à s'en rapprocher. Et inversement rendre les obstacles répulsifs, pour qu'il s'en écarte et reste à bonne distance. Dans cette approche, La communauté des chercheurs en robotique ne cherche pas à anticiper la trajectoire du robot, on le laisse librement décider à chaque instant dans quelle direction il doit se rendre, à partir des informations recueillies concernant son environnement local. On fait l'hypothèse que contrôler son comportement général plutôt que de chercher à calculer une trajectoire, permettra tout de même au robot d'atteindre son but au final.

Nous allons voir 3 méthodes qui reposent sur cette philosophie : les réseaux de neurones, la logique floue et les champs de potentiels artificiels.

- **Champs de potentiels artificiels**

Les méthodes de type APF (*Artificial Potential Field*) furent les premières employées dans les années 80 (Figure II.1.). Le principe des méthodes APF est de construire un champ de potentiels sur l'environnement de navigation du robot. La valeur de ce champ est minimale sur le point que le robot doit atteindre, et croit continûment au fur et à mesure que l'on s'écarte de ce point. Les obstacles génèrent un champ de potentiel répulsif pour le robot, de valeur supérieure à n'importe quel autre point ne correspondant pas à un obstacle du champ de potentiel. Et le champ de répulsion s'étend autour des obstacles avec une intensité inversement proportionnelle à la distance par rapport à l'obstacle.

L'idée est alors de demander au robot de se déplacer dans la direction du gradient de potentiel négatif le plus fort, sur le champ de potentiel global obtenu.

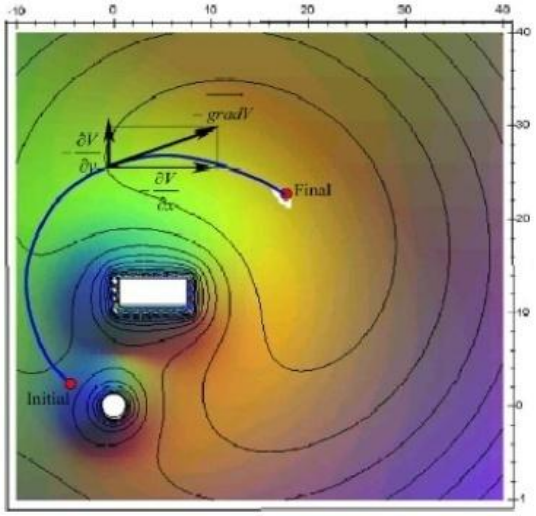


Figure II.1 : Carte des champs de potentiels autour d'un obstacle.

Dans [KHA90], et dans [AND83], les auteurs furent les premiers à imaginer l'idée de forces imaginaires agissant sur le robot. Ces méthodes présentent l'avantage d'être simples à mettre en œuvre, et elles furent les premières à être implantées physiquement sur de vrais robots en 1985 par Brooks [BRO86] et en 1989 par Arkin [ARK89]. Malgré le matériel informatique encore limité à cette époque, les calculs de trajectoires/commande étant très rapides avec ce type d'approche, cela permit de premières expérimentations sur des robots relativement lents. Dans [ADA03], Agirrebeitia propose une extension du principe des méthodes APF pour la navigation de robots dans un espace 3D. L'expérimentation a révélé certains problèmes récurrents liés au principe même de ces méthodes :

- Minima locaux entrainant des situations où le robot est piégé (classiquement le piège en forme de U).
- Pas de passage détecté entre des obstacles assez proches.
- Oscillations en présence d'obstacles et dans les passages étroits.

Dans [KOR91], les auteurs ont démontré mathématiquement les problèmes d'instabilité de ces méthodes (entrainant des oscillations), problèmes qui apparaissent fortement dès lors que ces méthodes sont implémentées sur des systèmes « rapides ».

- **Réseaux de neurones**

Le neurone est l'entité de base du système de réaction animal, et les neurologues McCulloch et Pitts furent les premiers à étudier leur fonctionnement chez la grenouille [LET59]. Les réseaux de neurones désignent à la fois l'étude de ces systèmes biologiques et leur modélisation informatique, plus ou moins simplifiée, à différentes applications, comme la reconnaissance de caractères, son premier terrain d'application historique, ou la navigation des robots mobiles autonomes. Dans ce cadre a été définie la notion de réseaux de neurones artificiels.

Dans un réseau de neurones artificiel (Figure II.2.), chaque neurone possède plusieurs entrées et une sortie. Chacune de ses entrées se voit affecter un poids (dit poids synaptique) différent, et si la somme ainsi pondérée des signaux des différentes entrées dépasse un seuil, la sortie prend une valeur positive (le neurone déclenche).

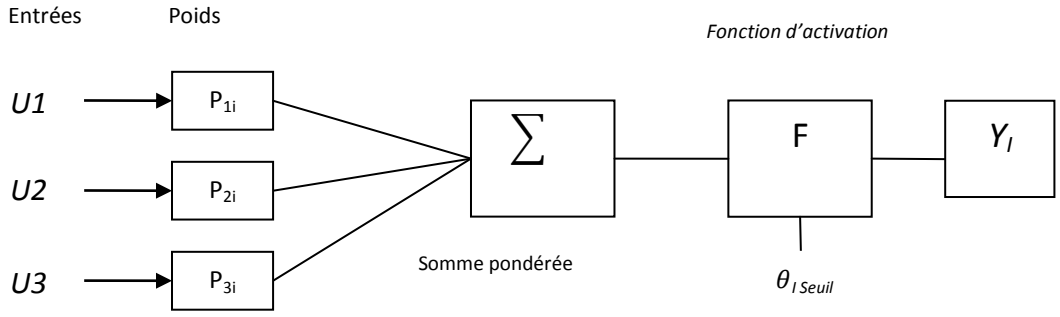


Figure II.2 : Modélisation d'un réseau de neurones artificiel

Un réseau complet est constitué de différentes couches de neurones. Les premières couches correspondent à des couches de détection, en robotique les neurones correspondants sont directement liés aux informations délivrées par les capteurs du robot. Les couches supérieures correspondent plutôt à des couches d'interprétation, et pour un robot elles seront plutôt liées aux effecteurs. De plus il y a généralement un certain nombre de couches intermédiaires entre les couches d'entrée et de sortie.

Un apprentissage du réseau est réalisable en adaptant les poids des entrées synaptiques.

Pour cela, le principe est de donner au réseau un grand nombre d'exemples (set d'entrées donné), et de rétro propager l'erreur obtenue entre la sortie du réseau et la sortie attendue. L'algorithme d'apprentissage aura alors pour tâche de régler les différents poids synaptiques pour minimiser cette erreur.

De nombreux contrôleurs pour robots mobiles ont été construits sur ces principes [LEB02], [BEL02].

Les dernières générations utilisent un codage en fréquence pour la transition de l'information, comme le font les neurones biologiques. Sur une fenêtre temporelle donnée, le nombre de pics émis quantifie par exemple la proximité d'un obstacle. Wang a développé un contrôleur basé sur ce principe, pour un robot uni cycle équipé de capteurs ultrasonores [WAN08].

- **Logique floue**

La logique binaire présente l'avantage de la simplicité, mais est assez éloignée de la façon humaine de raisonner. Si on prend l'exemple de la qualification de la proximité d'un obstacle, la logique

floue permet de faire intervenir des notions telles que « assez près » ou « très loin », au lieu de se cantonner à une définition binaire « obstacle ou pas obstacle ». Celle-ci a été formalisée par Zadeh en 1965 [ZAD65].

Le principe d'un contrôleur basé sur la logique floue se décline en 3 phases :

- Une phase de fuzzification, qui va transformer les variables d'entrée en variables floues ;
- Une phase faisant appel à une table des règles de comportement, règles logiques du type « si (condition 1) et/ou (condition 2) alors (action sur les sorties) » ;
- Une dernière phase dite de defuzzification, permettant de traduire l'action déterminée par les règles de comportement en commande à envoyer aux actionneurs.

- L'étape de fuzzification fait appel à des intervalles flous, qui vont délimiter l'espace des variables d'entrée en un certain nombre de sous ensembles flous (par exemple pour une proximité, on pourra avoir très proche (contact), assez proche, distance moyenne, assez loin et très loin) ; des fonctions d'appartenance sont alors utilisées pour définir le degré de vérité (probabilité d'appartenance) de la variable floue en fonction de la grandeur d'entrée. Ces fonctions peuvent être de type triangle, gaussienne ... etc. Ainsi pour une mesure de distance donnée, la règle d'appartenance nous dira « il y a 95% de chances que l'obstacle soit assez proche, 5% de chance que l'on soit en contact ». Cette notion est basée sur le fait qu'il y a toujours des incertitudes concernant les mesures des capteurs et les informations dont on dispose en général.
- La seconde étape consiste en l'élaboration de règles de comportement pour le robot, suivant la combinaison des variables floues en entrée. La table des règles de comportement est construite manuellement, et est tributaire de l'expérience de la personne qui va régler le contrôleur. Pour un robot mobile, une règle pourra être : «s'il y a un obstacle assez proche sur la droite, il faut tourner à gauche et diminuer la vitesse du robot ».
- La dernière étape, consiste à transformer le comportement obtenu par la table des règles, en commande pour le robot. Une méthode qui peut être utilisée pour faire cela est celle des centres de gravité, qui va consister à faire la moyenne pondérée des commandes à appliquer. La pondération étant liée aux probabilités d'appartenance de chaque variable d'entrée.

Un contrôleur flou pour permettre la navigation d'un robot mobile de type voiture «*Robucar*» à double braquage, a été étudié dans [OUA05]. Ce contrôleur permet au robot d'atteindre sa position finale tout en respectant les contraintes cinématiques du robot, mais ne tient pour le moment pas compte des obstacles. Des travaux sont en cours pour intégrer la prise en compte de l'orientation finale du véhicule.

Dans [CHA01], les auteurs utilisent la symétrie droite/gauche des règles de logique de comportement du robot pour simplifier celles-ci et ainsi diminuer le temps de calcul. Le problème de ces méthodes par logique floue est généralement le même que celui des méthodes APF, le problème des minima locaux qui se traduit par le fait que le robot peut rester piéger dans des culs de sac par exemple. Une technique pour se sortir de ces pièges a été proposée par Xu dans [XU 99]. Cette technique utilise des cibles virtuelles pour sortir le robot du piège dans lequel il est tombé, à partir du moment où il reconnaît être tombé dans un piège (Figure II.3).

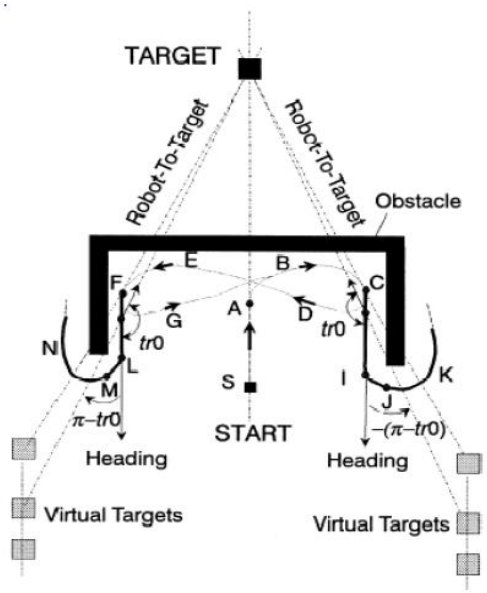


Figure II.3 : Le principe des Virtual Target pour sortir d'un minimum local [XU 99].

Un autre problème inhérent aux méthodes par logique floue est qu'elles sont trop spécialisées pour un type d'environnement donné, et donc elles souffrent de problèmes d'adaptation à des environnements différents. Cependant il existe des méthodes dites d'apprentissage qui permettent au robot de modifier lui même ses règles de comportement au fur et à mesure qu'il explore un nouvel environnement. Le problème de ces méthodes est qu'elles demandent un temps d'apprentissage plus ou moins long avant que le robot ne puisse naviguer efficacement [EMG04].

II.5.2. Méthodes de suivi de trajectoire

Le second type d'approche consiste à contrôler le robot pour lui faire suivre une trajectoire précise. Ces méthodes (Figure II.4.) nécessitent généralement de déterminer un modèle inverse du robot, c'est-à-dire un modèle qui permette de calculer les commandes à envoyer au robot (espace articulaire) connaissant la trajectoire que l'on souhaite lui faire suivre (dans l'espace cartésien).

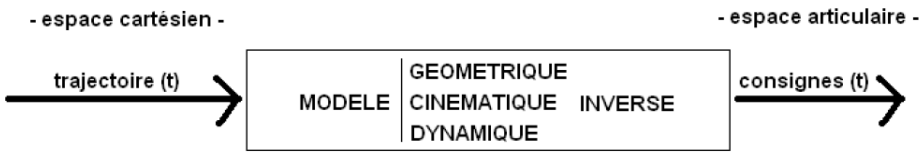


Figure II.4 : Approche par un modèle inverse pour effectuer un suivi de trajectoire

La détermination d'un tel modèle est souvent le point central de ce type de méthode, et le but est d'obtenir la meilleure convergence possible du robot vers la trajectoire de référence.

Les deux méthodes suivantes donnent des résultats intéressants dans ce domaine: les sorties plates et les fonctions transverses.

- **Sorties plates**

La théorie des systèmes plats fut introduite en 1995 par Fliess [FLI95]. Un système plat est un système

$x' = ax + bu$ tel que:

$$\begin{cases} x = f(y, y', y'' \dots y^k) \\ u = g(y, y', y'' \dots y^k) \\ y = h(x, u, u' \dots u^m) \end{cases} \quad (1)$$

Les différentes composantes du vecteur y sont les sorties plates du système. Concrètement cela revient à dire que l'on peut exprimer l'ensemble du système en fonction des sorties plates, et d'un nombre fini de ses dérivées. L'avantage d'écrire un système sous cette forme, dans le cas du suivi de trajectoire, est que cela permet de démontrer la convergence du système vers la trajectoire souhaitée. L'inconvénient est qu'il n'existe pas de méthode systématique pour déterminer la forme plate d'un système.

Une application pour le suivi de trajectoire des robots mobiles est donnée par Fraisse et elle est utilisée pour la navigation d'une flottille de robots [FRA04]. Chaque robot suiveur a pour but de suivre un robot leader, lui-même contrôlé à distance par télé opération (Figure II.5.).

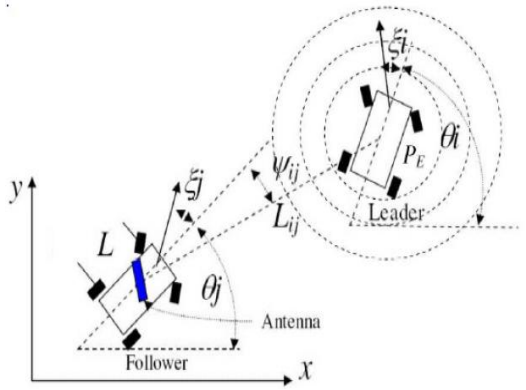


Figure II.5 : Modélisation d'une flottille de robot avec un robot leader et un/plusieurs robots suiveurs [FRA04].

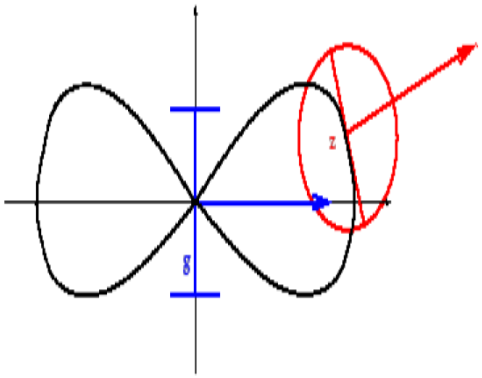


Figure II.6 : Base mobile S et sa transformée omnidirectionnelle Z, avec le repère compagnon en rouge [AND95].

Dans un premier temps les variables d'état du robot (vitesses linéaire, angulaire…) sont exprimées en fonction des sorties plates (positions x et y et leur dérivées). Ces sorties plates sont ensuite paramétrées en fonction du temps. Le problème devient alors un problème d'optimisation par rapport au temps des sorties plates, pour que le robot atteigne l'état désiré en respectant un ensemble de contraintes. Le fait que le système soit plat permet de démontrer la convergence de trajectoires vers les trajectoires désirées. Les sorties plates sont donc un outil intéressant dans le cadre du suivi de trajectoire en télé opération par exemple (trajectoire fournie par l'utilisateur) ou dans le cas d'une flottille de robots (trajectoire déterminée en fonction du leader). Elles garantissent de très bons résultats dès lors que l'on arrive à déterminer un modèle plat du robot à commander.

- **Fonctions transverses**

La synthèse de lois de commande par retour d'état stabilisant est la problématique traitée par l'approche par fonctions transverses. Cette problématique, pour les systèmes non holonomes, est particulièrement ardue dès lors que la trajectoire de référence présente des points fixes. En effet, le

système linéaire n'est plus contrôlable au voisinage des points fixes. L'objectif de stabilisation asymptotique en un point, qui est impossible à atteindre, est alors délaissé pour une stabilisation pratique, qui consiste à stabiliser le système en un voisinage arbitrairement petit de ce point.

L'approche par fonctions transverses, Morin et Samson offrent une solution à ce problème en permettant une stabilisation pratique autour de n'importe quelle trajectoire de référence, réalisable ou non, pour un système contrôlable, et s'étend aux systèmes présentant certaines propriétés de symétrie comme la voiture [MOR06]. Cette approche est basée sur l'utilisation de fréquences variables comme entrées de commande supplémentaire, et repose sur l'existence d'une fonction f vérifiant les conditions de transversalité [FER95]. Cette fonction permet de définir un repère compagnon, dans le voisinage proche du véhicule, et c'est ce repère qui est asservi sur la trajectoire de référence. Cette méthode permet ainsi à un véhicule non holonome tel qu'une voiture ou un uni cycle, de suivre une trajectoire non admissible, comportant par exemple des déplacements transversaux, sous une certaine erreur. Le robot effectuera alors des manœuvres autour de la trajectoire de référence. Cette erreur maximale est garantie à une valeur qui est fixée librement. Le compromis sur cette erreur étant que plus elle est fixée petite, plus le robot fera un grand nombre de manœuvres pour suivre sa trajectoire (Figure II.6.).

La restriction de cette méthode est liée à la difficulté de trouver les fonctions transverses pour certains systèmes, puisqu'il n'existe aucune méthode systématique pour les déterminer. De plus, la question de la robustesse de cette méthode, par exemple aux erreurs de mesures ou aux problèmes de glissement, reste un sujet ouvert.

- **Autres méthodes**

D'autres méthodes pour faire du suivi de trajectoires sont basées sur la reconnaissance d'images le long du parcours. Ces images peuvent être prises avec une caméra, ou avec des télémètres laser. Le principe est de créer une base de données d'images à reconnaître, et pour cela le plus simple est de piloter le robot en manuel dans une phase dite d'enregistrement, en lui faisant suivre le parcours de référence. Une fois la base de données créée, on peut procéder à la phase de suivi de trajectoire en autonome. A chaque instant d'échantillonnage, l'image vue par le robot est comparée à l'image qui correspond dans la base de données, et la corrélation entre les deux images permet de calculer les commandes à envoyer au robot, pour le recaler sur la trajectoire de référence.

Ce type de méthode a été appliqué à la navigation de véhicule en milieu urbain en utilisant des camera vidéo [COU08] [COU09], et en intérieur de bâtiments en utilisant des télémètres lasers [SHU06].

La reconnaissance d'amers est un principe proche de la reconnaissance d'images, sauf que cette fois ci on repère des éléments remarquables fixes dans l'environnement de navigation, pour guider le robot le long du parcours. Cette méthode a été appliquée sur un robot de type véhicule « CyCab», évoluant en milieu extérieur grâce à des balises installées dans l'environnement de navigation [PRA05].

II.5.3. Les algorithmes génétiques

Une des approches évolutionnistes consiste à concevoir et réaliser un algorithme génétique permettant à un robot de trouver un chemin optimal. Les algorithmes génétiques sont des algorithmes

d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle[1] : croisements, mutations, sélection, etc.

Un algorithme génétique recherche le ou les extrêmes a d'une fonction définie sur un espace de données. Pour l'utiliser, on doit disposer des cinq éléments suivants :

II.5.3.1. Description détaillée

Le principe général du fonctionnement d'un algorithme génétique est représenté sur la figure II.7.

- Un principe de codage de l'élément de population

Cette étape associe à chacun des points de l'espace d'état une structure de données. Elle se place, généralement après une phase de modélisation mathématique du problème traité. Le choix du codage des données conditionne le succès des algorithmes génétiques. Les codages binaires ont été très employés à l'origine. Les codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs, pour l'optimisation de problèmes à variables continues. Les algorithmes génétiques utilisant des vecteurs réels [GOL91], [WRI91] évitent ce problème en conservant les variables du problème dans le codage de l'élément de population, sans passer par le codage binaire intermédiaire.

- *Un mécanisme de génération de la population initiale*.

Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.

- *Une fonction à optimiser.*

Celle-ci prend ses valeurs dans R+ et est appelée fitness ou fonction d'évaluation de l'individu. Celle-ci est utilisée pour sélectionner et reproduire les meilleurs individus de la population.

- *Des opérateurs*

Permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'état.

- Opérateur de croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants. Initialement, le croisement associé au codage par chaînes de bits est le croisement à découpage de chromosomes (*slicing crossover*). Pour effectuer ce type de croisement sur des chromosomes constitués de M gènes, on tire aléatoirement une position dans chacun des parents. On

[1] Il est intéressant de trouver dans l'œuvre d'un spécialiste de zoologie, Richard Dawkins, un exemple informatique tendant à prouver la validité de l'hypothèse darwinienne de la sélection naturelle. La méthode utilisée est presque semblable aux techniques génétiques.

échange ensuite les deux sous-chaînes terminales de chacun des deux chromosomes, ce qui produit deux enfants C1 et C2 (Figure II.8.).

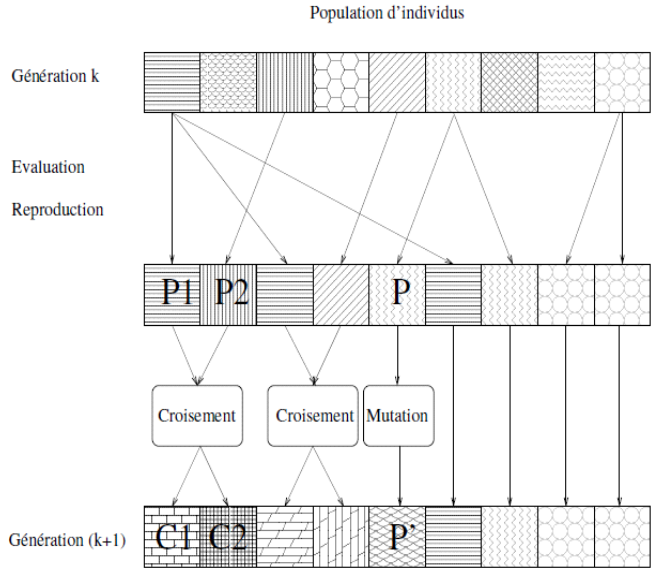


Figure II.7 : Principe général des algorithmes génétiques.

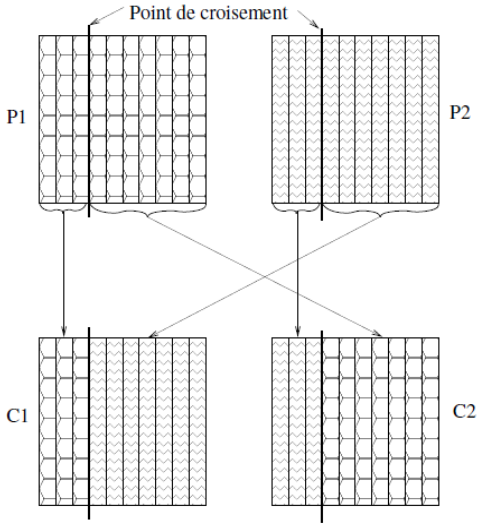


Figure II.8 : Croisement à 1 point

Ce principe peut être étendu en découpant le chromosome non pas en 2 sous chaînes mais en 3, 4, etc [BRI91], (Figure II.8).

Ce type de croisement à découpage de chromosomes est très efficace pour les problèmes discrets. Pour les problèmes continus, un croisement "barycentrique" est souvent utilisé : deux gènes P1(i) et P2(i) sont sélectionnés dans chacun des parents à la même position i. Ils définissent deux nouveaux gènes C1(i) et C2(i) par combinaison linéaire (équation 2) :

$$\begin{cases} C1(i) = \alpha P1(i) + (1-\alpha)P2(i) \\ C2(i) = (1-\alpha)P1(i) + \alpha P2(i) \end{cases} \qquad (2)$$

Où α un coefficient de pondération aléatoire adapté au domaine d'extension des gènes (il n'est pas nécessairement compris entre 0 et 1, il peut par exemple prendre des valeurs dans l'intervalle [−0.5, 1.5], ce qui permet d'engendrer des points entre, ou à l'extérieur des deux gènes considérés).

- Opérateur de mutation

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. Ainsi en toute rigueur, l'algorithme génétique peut converger sans croisement, et certaines implémentations fonctionnent de cette manière [FOG66].

Pour les problèmes discrets, l'opérateur de mutation consiste généralement à tirer aléatoirement un gène dans le chromosome et à le remplacer par une valeur aléatoire (Figure II.11.). Si la notion de distance existe, cette valeur peut être choisie dans le voisinage de la valeur initiale.

Dans les problèmes continus, on procède un peu de la même manière en tirant aléatoirement un gène dans le chromosome, auquel on ajoute un bruit généralement gaussien. L'écart-type de ce bruit est difficile à choisir a priori.

- Principes de sélection

A l'inverse d'autres techniques d'optimisation, les algorithmes génétiques ne requièrent pas d'hypothèse particulière sur la régularité de la fonction objective.

L'algorithme génétique n'utilise notamment pas ses dérivées successives, ce qui rend très vaste son domaine d'application. Aucune hypothèse sur la continuité n'est non plus requise. Néanmoins, dans la pratique, les algorithmes génétiques sont sensibles à la régularité des fonctions qu'ils optimisent. La fonction à optimiser peut i être le résultat d'une simulation. La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. On trouve dans la littérature un nombre important de principes de sélection plus ou moins adaptés aux problèmes qu'ils traitent. Les deux principes de sélection suivants sont les plus couramment utilisés [GOL89]:

- *Roulette wheel selection*;
- *Stochastic remainder without replacement selection*;

Le principe de *Roulette wheel selection*[2] consiste à associer à chaque individu un segment dont la longueur est proportionnelle à sa fitness. On reproduit ici le principe de tirage aléatoire utilisé dans les roulettes de casinos avec une structure linéaire. Ces segments sont ensuite concaténés sur un axe que l'on normalise entre 0 et 1. On tire alors un nombre aléatoire de distribution uniforme entre 0 et 1, puis on "regarde" quel est le segment sélectionné. Avec ce système, les grands segments, c'est-à-dire les bons individus, seront plus souvent choisis que les petits. Lorsque la dimension de la population est réduite, il est difficile d'obtenir en pratique l'espérance mathématique de sélection en raison du peu de tirages effectués. Un biais de sélection plus ou moins fort existe suivant la dimension de la population. *La Stochastic remainder without replacement selection* évite ce genre de problème et donne de bons résultats pour nos applications.

II.5.3.2. Parallélisme

L'intérêt du parallélisme des algorithmes génétiques est de gagner en temps de calcul. Il existe pour cela au moins deux méthodes utilisées classiquement (on pourra se reporter à [BER93], [SMI93], [COL91], [MUH89] et [PET87] pour plus de précisions sur les modèles de parallélisme utilisables dans les AG) :

– la première consiste à diviser la population de taille n en N sous-populations et à les répartir sur l'ensemble des machines dont on dispose ;

– la seconde maintient la population totale sur une seule machine mais se sert des autres pour y confier les évaluations, afin qu'elles se fassent en même temps.

Dans les deux cas, il est nécessaire d'avoir un mécanisme de communication interprocessus.

[2] Dans la littérature, cette méthode porte parfois le nom de méthode de Monte-Carlo.

Les résultats présentés ici ont été obtenus en utilisant PVM. PVM a été réalisé par le Oak Ridge National Laboratory. Il permet à un réseau hétérogène de machines d'apparaître comme une seule entité ayant plusieurs processeurs capables de communiquer entre eux (comparable à un ordinateur à mémoire distribuée). La communication entre les divers composants de cette machine virtuelle se fait par l'envoi de paquets contenant des données ou encore des messages de contrôle. Pour plus de détails, on peut se reporter à [GEI84].

- **Parallélisme par îlots**

L'idée ici est de faire fonctionner plusieurs algorithmes génétiques en parallèle avec une population réduite pour chacun. Le programme maître lance N occurrences d'algorithmes génétiques appelés esclaves sur des machines différentes en transférant à chacune les paramètres nécessaires à leur bon fonctionnement comme le montre la Figure II.9.

Ensuite chaque processus fait évoluer sa population indépendamment jusqu'à ce qu'il décide (selon une probabilité fixée à l'avance) de rassembler ses meilleurs individus pour en transmettre une certaine quantité (proportionnelle à la taille de la population) à une autre machine de son choix. La machine réceptrice intègre la méthode introduit un *clustering* forcé car chaque îlot peut être considéré comme un cluster subdivisé en petits groupes. Chaque machine a la possibilité de converger vers des optima qui seront différents de ceux calculés sur les autres, ce qui correspond au comportement introduit avec le clustering. D'autre part, le surcoût de temps passé pour les communications n'est a priori pas excessif, puisqu'il n'y a de gros paquets de données à transmettre que de temps en temps. Il faut tout de même garder à l'esprit qu'une subdivision en sous-populations de taille trop réduite risque de conduire à faire tourner des algorithmes génétiques non fiables statistiquement. En effet, il faut quand même qu'une population contienne suffisamment d'individus pour que l'espace d'état puisse être exploré de façon correcte, afin que les résultats aient une certaine valeur.

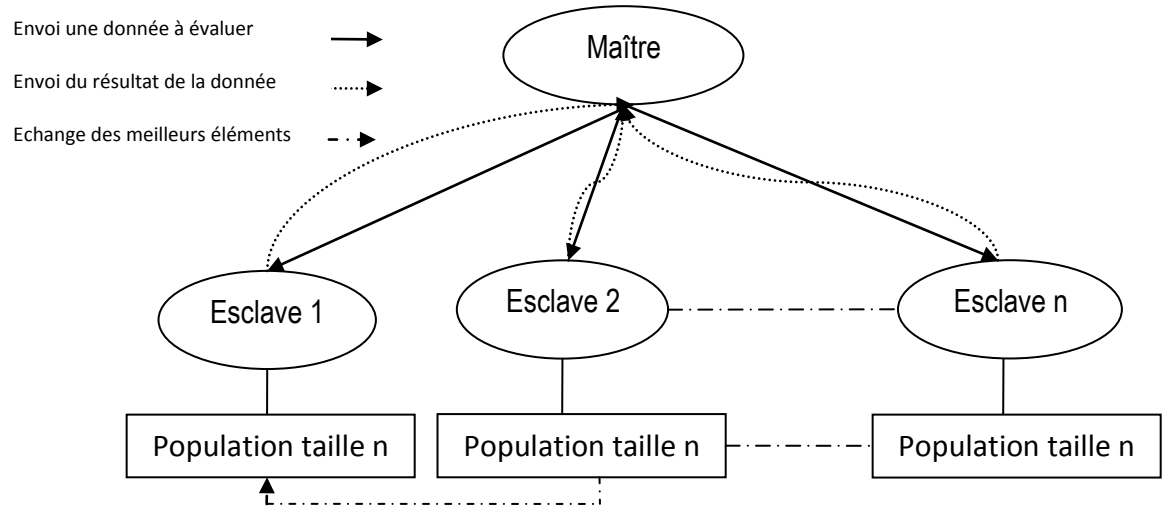


Figure II.9 : Principe de fonctionnement du parallélisme par îlots

- **Parallélisation des calculs**

Contrairement à la méthode qui vient d'être décrite, qui divise la population totale, on utilise ici des "démons" de calcul de fitness dont la seule fonction est de recevoir un individu et de retourner son adaptation. Pour utiliser la puissance de calcul parallèle offerte de manière optimale, il faut retarder au maximum les calculs pour les envoyer en blocs aux démons et faire en sorte qu'aucun ne reste inactif.

Le principe se trouve résumé sur la figure II.10 : le programme maître se charge de faire la sélection, les croisements, en d'autres termes, il fait évoluer la population, puis répartit les calculs dont il a besoin sur l'ensemble des démons. Enfin, dès qu'il a reçu tous les résultats, l'algorithme commence une nouvelle génération. Il faut noter que ce mécanisme demande un grand nombre de communications pour transmettre les données et les évaluations. La méthode n'est donc intéressante que si le temps passé pour un calcul d'adaptation est grand devant le temps de communication, elle sera par conséquent utilisée pour des problèmes dont les évaluations de fitness prennent du temps, on pense essentiellement à des cas faisant appel à des réseaux de neurones, ou à de gros calculs matriciels.

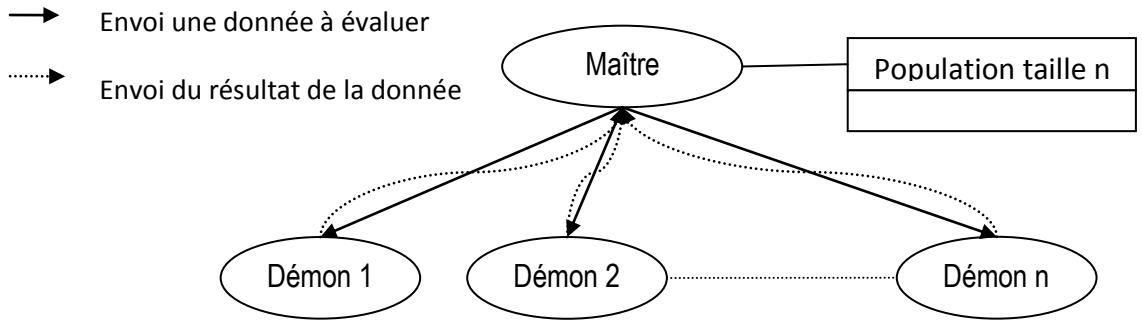


Figure II.10 : Principe de fonctionnement de la parallélisation des calculs

II.5.4. Synthèse des méthodes de navigation

Le problème de navigation en robotique mobile n'est pas un problème parfaitement arrêté. La frontière entre planification de chemin, navigation, et pilotage est plus ou moins définie, il existe souvent des recouvrements entre ces différents domaines connexes. Ces différences d'appréhension du problème se ressentent notamment au niveau des structures de contrôle des robots, qui sont en quelque sorte le reflet de la manière dont chaque équipe perçoit et découpe les différentes tâches nécessaires à la réalisation d'un système robotique autonome. Il existe ainsi beaucoup de définitions différentes du navigateur, d'où certaines différences dans la manière de répondre à la problématique. Par exemple, beaucoup de travaux ne font pas la distinction entre chemin, trajectoire et même pilotage, et tous ces problèmes sont ainsi résolus par un unique module. C'est le cas d'un grand nombre d'approches à base de réseaux de neurones et de logique floue, qui cherchent à résoudre tous ces problèmes en même temps en contrôlant un comportement global du robot. Ce genre d'approche peut donner de bons résultats dans une situation donnée. Mais il n'y pas de planification à long terme de ce que le robot doit faire, le robot n'est donc pas capable de s'adapter à une situation inconnue. Il s'agit généralement d'approcher de la meilleure manière possible un comportement souhaité dans un type de situation donné, ce qui ne correspond pas à la définition d'un système autonome qui serait capable de s'adapter à différentes situations non prévues.

Une autre approche pour la navigation porte sur la détermination de méthodes pour effectuer un suivi de trajectoire ou de chemin. La trajectoire servant de référence peut être fournie par un module de planification, ou encore provenir d'un robot leader qu'il faut suivre à une certaine distance. En considérant un robot classique de type voiture, le problème à traiter provient notamment des contraintes de non holonomie des robots, qui réduisent sa mobilité et l'empêchent donc effectuer certains mouvements pour rattraper la trajectoire de référence. Dans ce domaine, les approches par fonctions transverses ou celles par les sorties plates ont montré d'excellents résultats. La difficulté de ces méthodes est qu'elles nécessitent de déterminer un modèle bien spécifique du robot. Les méthodes pour déterminer ces modèles ne sont pas systématiques, et de plus ces modèles n'existent pas pour tous les robots.

Pour les algorithmes génétiques, les résultats théoriques sont nombreux, cependant la question est comment adapter les operateurs de croisement, de mutation et de sélection pour accélérer considérablement la convergence de l'algorithme génétique ?

Le parallélisme est extrêmement efficace pour accélérer les temps de résolution des algorithmes génétiques. Il faut certes bien étudier le problème afin d'utiliser le bon mécanisme de parallélisme, mais les gains en temps sont alors importants.

II.6. Évitement d'obstacles

Cette partie traite de la fonctionnalité d'évitement d'obstacles. Tout d'abord nous présentons un état de l'art des méthodes d'évitement d'obstacles, et nous montrons leur inadéquation aux spécificités de notre problématique. Ensuite nous proposons une méthode originale d'évitement d'obstacles pour des robots mobiles. Cette méthode utilise une trajectoire de référence qui est déformée de façon à s'éloigner des obstacles et à satisfaire les contraintes cinématiques du système. Enfin nous présentons quelques optimisations de l'algorithme de cette méthode et des extensions de ses applications.

II.6.1. Présentation

La plupart des méthodes d'évitement d'obstacles sont des méthodes locales. En effet les méthodes globales, qui considèrent un modèle complet de l'environnement, se ramènent au problème de la planification de mouvement. Pour des systèmes possédant de nombreux degrés de liberté (plus de 3), la complexité de la planification d'une trajectoire interdit son utilisation en cours d'exécution. Par exemple l'approche de re-planification rapide de Stentz, il partage l'espace de travail en cellules étiquetées «libre» ou «occupée», pour diminuer la complexité [STE94].

Pour les systèmes considérés dans notre travail, la planification d'une trajectoire sans collision dans un environnement fortement contraint est très coûteuse en temps de calcul. Cette étape est donc réalisée «hors-ligne» et nous avons utilisé les techniques d'exploration aléatoire de l'espace des configurations. Le modèle de l'environnement utilisé est construit auparavant par un autre robot ou à partir d'un plan métrique. En plus, nous avons utilisé une méthode de guidage (calcul d'une trajectoire admissible entre deux configurations sans prise en compte des obstacles) pour le système considéré, si elle existe, pour connecter les configurations tirées aléatoirement.

Parmi les méthodes locales d'évitement d'obstacles, on peut distinguer deux catégories : les méthodes utilisant une trajectoire de référence et celles n'en utilisant pas. Le type de système et d'application orientent le choix vers une catégorie ou l'autre.

Enfin un autre type de méthodes doit être mentionné, qui tient compte explicitement de la vitesse estimée des obstacles pour réaliser un mouvement sans collision, ce qui définit une nouvelle problématique. Les travaux de fraisse est constitué un cadre formel et un état de l'art de cette problématique [FRA04].

II.6.2. Méthodes d'évitement d'obstacles sans trajectoire de référence

Pour certains systèmes et dans certains contextes applicatifs, la définition d'une trajectoire comme une série de points de passage est possible. Des méthodes d'évitement réactif d'obstacles ont été développées qui calculent des commandes permettant de rejoindre un point de passage en évitant les collisions avec les obstacles détectés.

- **Champs de potentiel**

Les méthodes de champs de potentiel pour la navigation en robotique, initialement proposées par Khatib pour un bras manipulateur, consistent à construire une fonction de potentiel qui résume les objectifs de la navigation : éviter les obstacles (potentiel répulsif) et atteindre une configuration but (potentiel attractif). À chaque position du robot, une «force» résultant de l'action conjuguée des obstacles et du but est calculée, qui correspond à une direction à suivre par le robot [KHA86].

De nombreuses adaptations de cette technique ont été proposées. Nous pouvons citer Borenstein, qui calcule une direction de mouvement à partir d'informations proximétriques [BOR91].

Ces méthodes purement réactives sont sujettes à des minima locaux, et peuvent nécessiter une re-planification globale. Par ailleurs, elles peuvent entraîner un mouvement oscillatoire du robot dans certaines situations (des passages étroits par exemple).

- **Steering Angle Field (SAF)**

Cette approche a été proposée par Feiten pour un robot de type uni cycle. Il s'agit de calculer pour un ensemble de vitesses linéaires des domaines de l'angle de braquage qui n'entraînent pas de collision avec les obstacles perçus. Cette méthode utilise une discrétisation en grille du plan de travail autour du robot. Elle exploite la possibilité de pré-calculer et de stocker dans des tables de recherche les SAF pour chaque cellule obstacle de la grille [FEI94].

- **Dynamic Window**

Cette technique proposée dans [MIN00] travaille dans l'espace des commandes du robot. La taille du domaine de recherche des vitesses accessibles (c'est à dire n'entraînant pas de collisions) est réduite par la prise en compte explicite de la dynamique (les capacités d'accélération) du système. Les commandes envoyées au robot sont le résultat de la maximisation sur ce domaine de recherche d'une fonction de coût liée à la position but. Cette méthode a été développée pour des robots se déplaçant à des vitesses élevées dans des environnements intérieurs encombrés.

- **Nearness Diagram**

Cette approche proposée par Minguez repose sur un diagramme de proximité des obstacles, mis à jour au fur et à mesure du déplacement du robot [MIN00]. En fonction de l'allure de ce diagramme (nombre de passages sans obstacles, taille des passages, etc.), un comportement adapté (exploration, avancement vers le but, retour en arrière, etc.) est sélectionné. La direction de mouvement la plus prometteuse par rapport au but est alors choisie. Cette méthode a été appliquée à un véhicule de type unicycle grâce à une expression des obstacles dans l'espace des points accessibles en un mouvement élémentaire (un arc de cercle en l'occurrence), obtenue par une transformation dans l'espace dit «Ego-cinématique» [MIN02].

L'exposé de ces différentes méthodes et de leurs hypothèses fait immédiatement apparaître leur inadéquation à des systèmes à cinématique plus complexe évoluant dans des environnements fortement contraints. On ne peut en effet généralement pas présupposer de la forme des trajectoires. De même les discrétisations en grille de l'espace de travail, proposées par certaines méthodes, ne permettent pas les mouvements arbitrairement proches des obstacles.

II.6.3. Méthodes d'évitement d'obstacles avec une trajectoire de référence

Dès lors que la cinématique du système est plus complexe, le calcul préalable d'une trajectoire de référence qui sera adaptée lors de l'exécution s'avère nécessaire. Nous présentons deux méthodes utilisant une trajectoire de référence pour l'évitement réactif d'obstacles.

- **Trajectoires d'évitement**

Une méthode proposée par Laugier et ses collègues dans [LAU99] repose sur l'enchaînement de trajectoires élémentaires (Sensor Based Maneuvers SMB), dans le contexte d'une voiture automatisée roulant sur une route. Les SMB sont par exemple le suivi d'un couloir de route, le changement de file ou le parking parallèle. Les paramètres de chacune de ces modalités sont définis par les informations sensorielles. Un contrôleur, amélioré dans [LAR00] par un réseau de neurones, permet d'enchaîner ces différentes modalités.

- **Bande élastique**

La méthode proposée par Quinlan dans [QUI93] utilise une trajectoire initialement planifiée qui est représentée par une série de boules adjacentes (appelées «bulles») dans l'espace des configurations.

Le rayon d'une boule centrée en une configuration est la distance de cette configuration à l'obstacle le plus proche. Ainsi une trajectoire est sans collision dès lors que les «bulles» qui la composent se recouvrent. Développée pour des systèmes sans contraintes cinématiques, cette technique considère la trajectoire comme une bande élastique, se modifiant sous l'action de forces répulsives générées par les obstacles, et de forces internes de contraction ou d'élasticité.

Du fait de la représentation en bulles, la mise à jour de la trajectoire sous l'action des forces est très rapide.

Cette technique a été étendue à un robot de type voiture par Khatib dans [KHA97]. La «forme des bulles» est ici donnée par la métrique des trajectoires de Reeds et Shepp (combinaisons d'arcs de cercle et de lignes droites) [SOU96]. Le lissage de la courbe joignant les centres des «bulles» est réalisé en utilisant une courbe de Bézier. Mais pour certains systèmes on ne connaît pas la plus courte distance entre une configuration et un obstacle. La méthode de la bande élastique ne peut donc s'appliquer qu'au prix d'approximations sur la forme des trajectoires de tels systèmes, ce qui rend impossible la navigation en environnement très contraint.

L'ensemble de ces considérations, concernant les limitations des méthodes utilisant une trajectoire de référence ou des techniques proposées pour les systèmes à cinématique plus simple, motive le développement d'une méthode générique d'évitement d'obstacles pour les robots mobiles.

II.7. Parking

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

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

II.8. Architectures de contrôle des robots mobiles

Un robot est un système complexe qui doit satisfaire des exigences variées en utilisant un ensemble logiciel appelé architecture de contrôle du robot. Cette architecture permet donc d'organiser les relations entre les trois grandes fonctions: la perception, la décision (planification) et l'action. Ces architectures peuvent néanmoins être classées en trois groupes (approches) [FIL04] comme montré sur la figure II.11.

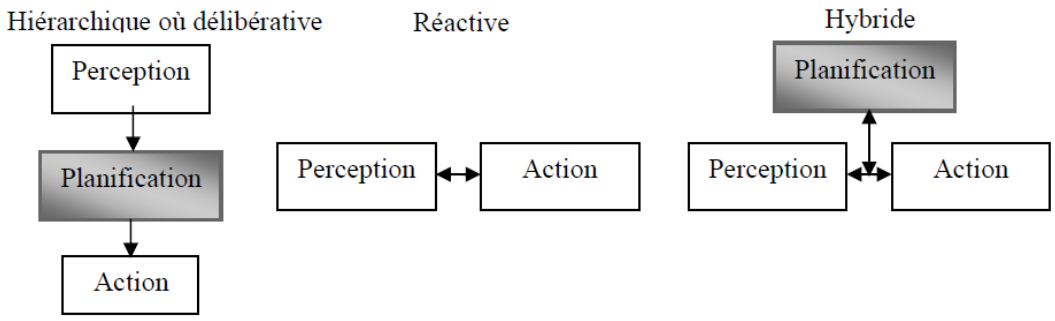


Figure II.11 : Architecture de contrôle pour les robots mobiles.

II.8.1. Approche délibérative (penser puis agir)

Les premiers robots mobiles dérivés des recherches en intelligence artificielle utilisaient des contrôleurs hiérarchiques, dont le fonctionnement repose essentiellement sur la disponibilité d'un modèle de l'environnement permettant de représenter toutes les informations pertinentes pour le déplacement du robot [AZO06]. Elles ont l'avantage de prouver l'existence d'une solution optimale permettant au robot d'atteindre le but assigné. Mais un tel modèle peut être insuffisant dans un environnement inconnu ou dynamique car au moment de la réalisation de l'action, l'environnement peut avoir changé et la décision n'est plus valide. Le traitement est décomposé en une série d'opérations successives comme décrit sur la Figure II.12.

1. Traiter les données sensorielles qui fournissent au robot des informations sur son environnement,

2. Construire ou à mettre à jour, à partir des données acquises, un modèle du monde dans lequel le robot évolue. Ce modèle peut être par exemple une identification du type et de la position des obstacles que le robot doit éviter,

3. Ce modèle est utilisé par la suite pour planifier une suite d'états permettant au robot mobile d'effectuer la tâche visée,

4. Calculer puis d'exécuter les actions afin de suivre le plan généré par l'étape précédente.

Dans ce type d'architecture, le robot utilise toutes les informations sensorielles disponibles et toutes les connaissances internes sauvegardées et raisonne sur les prochaines actions à réaliser.

Le robot doit construire et ensuite évaluer tous les plans possibles pour trouver celui qui atteint le but. La planification requiert l'existence d'une représentation interne du monde qui permet au robot d'avoir une idée de futur et de prédire les résultats d'actions possibles dans les différents états perçus. Le modèle interne doit donc être précis et récent. Quand le temps est suffisant afin de générer un modèle du monde précis, cette approche permet au robot d'agir stratégiquement en sélectionnant la meilleure action pour une situation donnée.

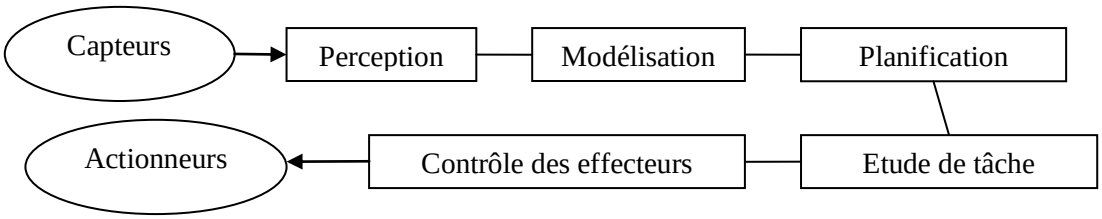


Figure II.12 : Architecture hiérarchique.

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

puissantes. Ces méthodes doivent établir un couplage entre la perception et l'action et le traitement local des différentes tâches [REI94].

II.8.2. Approche réactive

En 1986, Brooks a proposé une approche réactive qui se distingue par l'abandon des phases de *modélisation* et de *planification*. Le contrôle du robot se fait alors sans utiliser le modèle de l'environnement (Figure II.13) [BRO86]. Les stratégies de navigation réactives n'utilisent que les valeurs courantes des capteurs et non des données provenant d'un modèle interne, pour décider de l'action à effectuer. Le principe de l'approche réactive est basé sur la décomposition de la tâche de navigation en un ensemble de comportements actifs de base (explorer, aller au but, éviter les obstacles, suivi des murs,…). Pour guider le robot, il faut donc choisir à chaque instant lequel de ces comportements est à activer? Ce problème est connu dans la littérature scientifique sous le nom de *sélection de l'action*. La solution proposée par Brooks est de diviser la tâche globale en un ensemble des sous tâches secondaires. Cette architecture est appelée "*subsumption*", utilise une hiérarchie des comportements qui se déclenchent donc selon un ordre de priorité en fonction des données de perception (fusion des comportements).

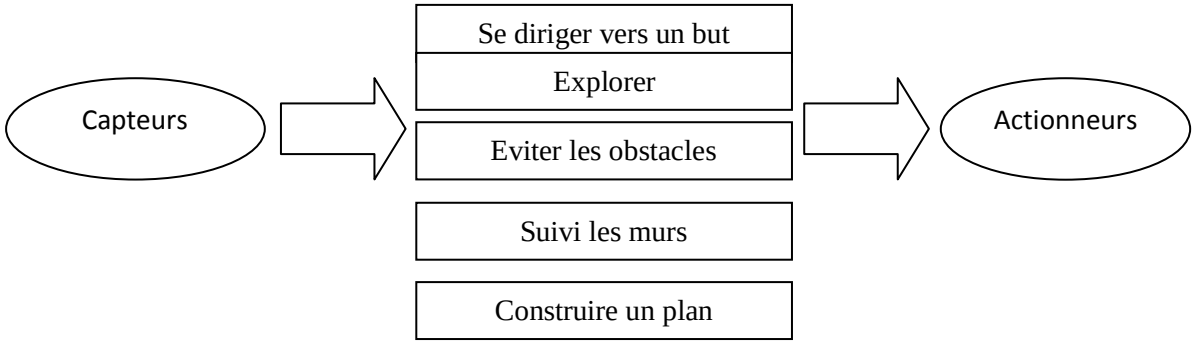


Figure II.13 : Architecture réactive.

La Figure II.14 décrit les différentes couches comportementales d'un robot en utilisant la structure subsumption. Cette architecture peut être définie comme une hiérarchie de comportements plutôt que comme une hiérarchie fondée sur une abstraction des données. Le concept de base est le suivant: si complexe soit-il, peut être décomposé en comportements élémentaires représentant chacun un niveau de compétence. Ces différents niveaux sont placés en couches parallèles et chaque compétence couple directement la perception avec l'action. Une telle architecture a, au moins, trois avantages:

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

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

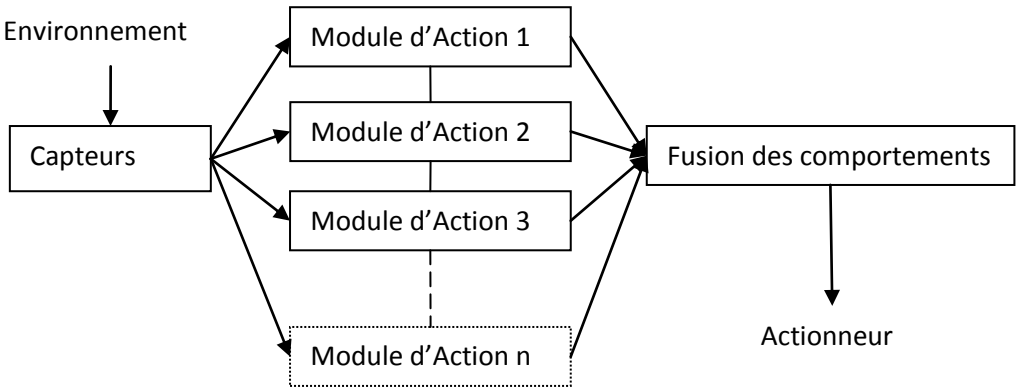


Figure II.14 : Architecture modulaire de subsumption

En effet, sans représentation interne de l'état de l'environnement, il est très difficile de planifier une suite d'actions en fonction d'un but à atteindre. Les robots utilisant cette architecture sont donc en général efficaces pour la tâche précise pour laquelle ils ont été conçus et dans l'environnement pour lequel ils ont été prévus, mais sont souvent difficiles à s'adapter à une tâche différente. Puisque, le robot n'a qu'une vue très réduite de son environnement, ces architectures ne peuvent garantir le succès de la mission en toute circonstance. Les réussites de ces architectures sont liées au couplage direct entre la perception et l'action qui permet une prise en compte très rapide des phénomènes dynamiques de l'environnement et donc une robustesse et fiabilité dans les situations complexes.

II.8.3. Approche hybride

Cette architecture combine les meilleurs aspects des architectures réactives et délibératives: elle combine la réponse temps-réel de la réactivité avec la rationalité et l'efficacité de la délibération [FIL04]. En effet, la composante réactive s'occupe des besoins immédiats du robot comme l'évitement d'obstacles et fonctionne ainsi en un temps court et utilise des données et signaux externes directs, alors que la composante délibérative utilise des représentations internes, abstraites et symboliques du monde et fonctionne en un temps très long. Tant que les sorties des deux composantes ne sont pas en conflit, le système n'a pas besoin de plus de coordination.

Cependant, ces deux parties interagissent pour tirer les avantages de chacune. La composante réactive doit outrepasser la composante délibérative si le monde présente un certains défit immédiat et imprévisible et la partie délibérative doit informer la partie réactive afin de guider le robot vers des trajectoires et des buts plus efficaces. L'interaction des deux parties de l'architecture requiert une composante intermédiaire dont sa construction est particulièrement le plus grand défi dans la conception hybride. Ainsi, les architectures hybrides sont souvent appelées "architectures à trois couches", composées des couches réactive, intermédiaire et délibérative. Un grand effort de recherche a été conduit dans la manière de concevoir ces composantes et leurs interactions.

II.8.4. Approche collective

En cas d'application qui nécessite l'utilisation de plusieurs robots mobiles (la robotique collective), l'architecture de contrôle comportementale collective tire son inspiration de la biologie et essaie de modéliser le comportement des animaux dans leurs environnements complexes d'évolution [AZO06], [FER95]. Les composantes de base de ces architectures sont des comportements observés d'activité

émergeant des interactions entre le robot et son environnement. De tels systèmes sont construits d'une façon ascendante commençant par un ensemble de comportements de survie tel que l'évitement de collisions qui couple les entrées perçues aux actions du robot. Des comportements sont ajoutés pour fournir au robot plus de capacités lui permettant de réaliser des tâches de plus en plus complexes telle que le suivi de mur, la poursuite d'une cible, l'exploration ou le "homing" (rentrez chez-soi). Ces nouveaux comportements sont introduits dans le système incrémentale des plus simples aux plus compliqués jusqu'à ce que leurs interactions aboutissent aux capacités désirées du robot. Comme les architectures hybrides, les architectures comportementales peuvent être organisées en couches mais à l'inverse de ces architectures, les couches ne diffèrent pas beaucoup les unes des autres en termes de temps et représentation utilisées. Toutes les couches sont représentées par des comportements, des processus qui récoltent des entrées et s'envoient des sorties.

II.9. Conclusion

La robotique est la branche de l'intelligence artificielle concernée par l'étude des systèmes automatiques capables d'interagir directement avec le monde physique. C'est une automatisation de ses machines, où l'objectif est d'augmenter les capacités de localisation et de navigation dans son espace de travail. Dans ce chapitre nous avons décrit le domaine de la navigation autonome des robots mobiles, et l'évitement des obstacles en se basant sur les algorithmes génétiques. Nous avons présenté aussi les principales architectures utilisées pour le contrôle des robots mobiles en citant les avantages et les inconvénients de chacune.

La commande d'un robot mobile se divise généralement en trois étapes principales: perception, décision et action. La dernière étape concerne l'exécution des mouvements planifiés, c'est une étape qu'il doit maîtriser efficacement pour accomplir ses missions avec succès.

Notre thèse traite la problématique de la navigation optimale des robots mobiles, en d'autre terme à déterminer le chemin optimal de plusieurs robots mobiles à l'aide des algorithmes génétiques dans un environnement dynamique. Après cet aperçu sur les algorithmes génétiques et l'intérêt du parallélisme, dans le chapitre suivant nous présenterons une vision générale de la coopération dans les systèmes multi-robots (parallélisme). Dans ce cadre, nous montrons les limites de la décomposition par tâche des missions robotiques pour l'utiliser par la suite dans le problème traité.

Chapitre III :

Les Systèmes Multi -Robots

Chapitre III : Les Systèmes Multi -Robots

III.1. Introduction

Malgré le fait que les premières applications de robots parallèles soit apparues bien avant l'avalanche de travaux de recherche en robotique parallèle, ce n'est que depuis quelques années que les résultats de ces travaux ont commencé à conduire à de nombreuses nouvelles applications.

L'un des objectifs finaux de l'utilisation des robots est de remplacer (ou au moins d'aider) les êtres humains dans la réalisation de certaines tâches. C'est particulièrement souhaitable pour les familles de tâches qualifiées en anglais de 3D pour Dirty, Dangerous, or Dull [3][TAK08]. Il s'agit de tâches « sales » telles que le tri d'ordures, dangereuses comme l'exploration de l'intérieur d'une centrale nucléaire ou encore ennuyeuses tel que la surveillance d'un site. Cependant, force est de constater qu'il reste encore un certain nombre de défis techniques à surmonter avant que les robots puissent réaliser de telles tâches sans l'intervention d'opérateurs humains.

Le domaine de la robotique parallèle est fascinant car il semble y avoir une infinité de sujets de recherche, d'une complexité grandissante mais encore surmontable. Voilà pourquoi, il n'est pas surprenant de constater que des milliers de chercheurs s'intéressent à ce domaine aujourd'hui. De plus en plus d'industriels adoptent des architectures parallèles pour leurs produits : simulateurs de mouvement, robots industriels, positionneurs, joysticks, etc.

Nous présentons dans ce chapitre, un état de l'art de la coopération dans les systèmes multi-robots. Dans ce cadre, nous discutons les limites de la décomposition par tâche des missions robotiques et leur intérêt. Le problème traité dans cette thèse vise à proposer un système où les robots mobiles doivent coopérer pour l'organisation d'opérations dans un chantier de construction. Dans ce cadre, nous nous sommes intéressés à la problématique de la coopération automatique dans un système multi-robots. Plus concrètement, nous proposons des solutions à la question suivante:

Comment partager efficacement les tâches aux robots ?

III.2. Problématique

La robotique est une science complexe du fait qu'elle nécessite le concours de plusieurs disciplines comme l'électronique, la mécanique et le génie logiciel. Cette complexité est accentuée avec

[3] Sales, Dangereuses, ou Ennuyeuses.

le passage aux systèmes multi-robots (SMRs) [SIC08], [ARR06]. Cette section est consacrée à ces systèmes et à la problématique de coopération qui découle de leur utilisation.

Elaborer un système multi-robots conduit nécessairement à s'interroger sur les capacités individuelles des robots : dans quelle mesure des approches de décision et de contrôle données peuvent-elles répondre aux exigences et contraintes posées par un tel système ? Suffit-il d'étendre une architecture mono-robot dans un cadre multi-robot, ou alors faut-il considérer la nature des problématiques multi-robots?

Dans un premier temps, nous présentons un bref tour d'horizon des idées qui s'articulent autour des notions de décision et de contrôle, dans un robot. Ensuite, afin de mieux en saisir les spécificités et enjeux, nous proposons trois approches pour appréhender les problématiques rencontrées dans les systèmes multi-robots.

III.3. Architecture et décision mono-robot

La communauté robotique reconnaît qu'aucune architecture n'est parfaite pour répondre à toutes les tâches, et que différentes tâches ont différents critères de succès qui conduisent à différentes architectures. Les architectures de programmation robotique peuvent être regroupées en quatre grandes catégories:

- les architectures centralisées classiques
- les architectures hiérarchiques,
- les architectures comportementales
- les architectures hybrides.

Les premiers travaux concernant les architectures de contrôle robotique étaient inspirés de l'intelligence artificielle, c'est-à-dire organisés autour de processus décisionnels et d'un état symbolique du monde et du robot. Les architectures conçues suivant cette philosophie font partie de la catégorie des architectures centralisées classiques. Elles placent la planification au centre du système et partagent l'axiome suivant lequel le problème central en robotique est la cognition, c'est à-dire, la manipulation de symboles pour maintenir et agir sur un modèle du monde, le monde étant l'environnement avec lequel le robot interagit. Parmi les architectures centralisées, nous pouvons citer: le système de planification STRIPS [FIK71] du robot Shakey dans lequel le plan est statique et le monde supposé constant au cours de l'exécution du plan de mouvement, les architectures Blackboard qui accumulent des données sur le monde et prennent des décisions immédiates basées à la fois sur les objectifs à priori variables et un monde changeant. Pour une tâche donnée, si le système peut modéliser le monde suffisamment bien, et si le monde obéit à son (ses) modèle(s), et si le système peut récupérer l'information pour l'intégrer dans le cœur de la planification centrale, alors une architecture centralisée classique constitue un bon choix pour la réalisation de la tâche. Les architectures centralisées conviennent bien aux tâches pour lesquelles la réactivité et le réflexe ne sont pas des critères essentiels.

Les architectures hiérarchiques décomposent la programmation des applications en niveaux de plus en plus abstraits. Chaque niveau a pour rôle de décomposer une tâche que lui a recommandée le niveau supérieur, en tâches plus simples qui seront ordonnées au niveau inférieur. Le niveau le plus

haut gère les objectifs globaux de l'application, alors que le niveau le plus bas commande les actionneurs du robot. L'instance la plus connue de ce type d'architecture est NASREM (Nasa/nbs Standard REference Model), [ALB89].

Les architectures hiérarchiques ont généralement une réactivité assez faible: compte tenu de la décomposition systématique de la programmation, la chaîne allant des capteurs aux actionneurs en passant par les processus décisionnels capables de répondre à des changements de l'environnement est complexe, entraînant des temps de réponse longs.

Les architectures comportementales sont nées au milieu des années 80 avec l'architecture "subsumption" proposée par Brooks. Elles sont issues de l'observation de comportements animaux simples, et sont basées sur l'idée qu'un comportement complexe et évolué d'un robot peut émerger de la composition simultanée de plusieurs comportements simples. Brooks définit un comportement élémentaire comme "un traitement prenant des entrées capteurs et agissant sur les actionneurs".

L'architecture DAMN (Distributed Architecture for Mobile Navigation) proposée par Rosenblat à l'Université de Carnegie Mellon est une autre variante des travaux de Brooks. Les travaux de Brooks ont mis en évidence l'atout de ce type d'architecture: la rapidité de la réaction du système face aux événements extérieurs ou à des situations spécifiques [BRO86].

Les architectures comportementales ont fait leurs preuves dans de nombreuses et parfois spectaculaires expérimentations concernant la robotique mobile, car la réactivité qui les caractérise permet d'aborder la navigation dans un environnement dynamique. Toutefois, la complexité des applications reposant sur cette approche va rarement au delà de la navigation. En effet, plusieurs comportements sont souvent en concurrence pour le contrôle des actionneurs et on ne peut pas, à priori, assurer la stabilité d'exécution de la loi de commandes complexes telles que celles requises pour le contrôle de bras manipulateurs. Ces approches présentent une autre limitation: les comportements étant préétablis, le système s'accommode difficilement d'un changement de mission impromptu.

Face aux lacunes des deux précédentes catégories d'architectures, certains chercheurs ont proposé des architectures hybrides qui allient les capacités réactives des architectures comportementales et les capacités de raisonnement propres aux architectures hiérarchiques. Ces architectures peuvent être suffisamment souples et puissantes pour que leur domaine d'utilisation en termes de variété de robots contrôlés et de type d'application justifie leur commercialisation. Parmi elles, nous pouvons citer: le CONTROLSHELL vendu par la société californienne RTI (Real-Time Innovations), ou encore l'architecture ORCCAD (Open Robot Controller Computer Aided Design system) de l'INRIA [KAL04]. L'architecture ORCCAD a la particularité d'être indépendante du système à piloter. Elle autorise également la spécification et la validation de missions en robotique.

L'architecture LAAS (qui a fait ses preuves dans les domaines de la robotique mobile, que ce soit sur les plates-formes HILARE ou dans l'expérience MARTHA [BOT99]) (Figure III.1) est un autre exemple d'architecture hybride : au plus prés des aspects matériels, une couche fonctionnelle comprend les processus de contrôle des capteurs et des effecteurs, qui fonctionnent en temps réel, de façon très réactive. Au niveau intermédiaire, la couche de contrôle d'exécution est dédiée au contrôle des requêtes transmises à la couche fonctionnelle. Au plus haut niveau, la couche décisionnelle rassemble les composantes permettant au robot de planifier ses actions et de superviser l'exécution

des tâches. Les fonctions décisionnelles ont des contraintes temps réel moins prononcées que la fonction de la couche fonctionnelle.

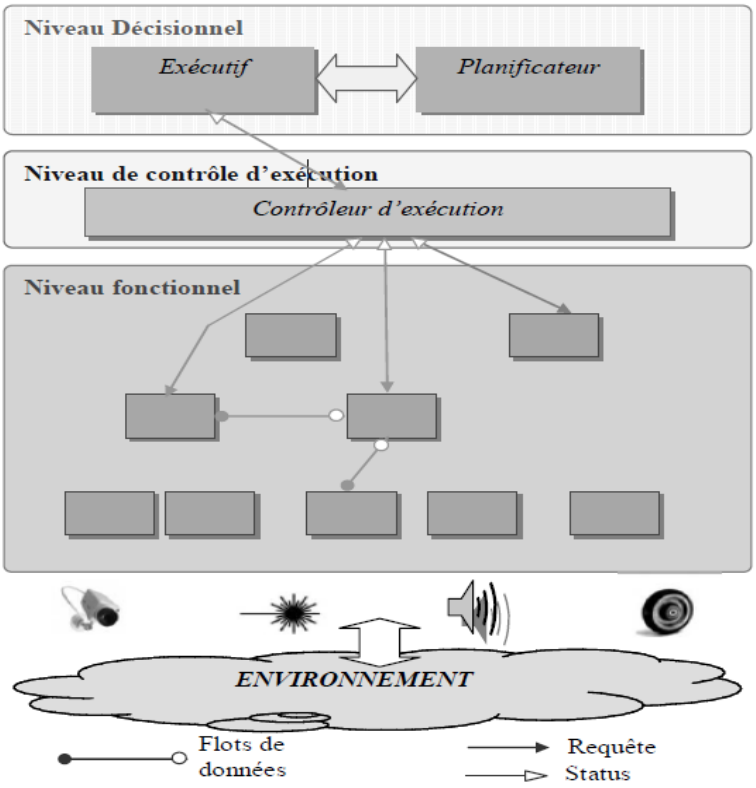


Figure III.1 : L'architecture LAAS

Ces différentes architectures hybrides sont habituellement considérées comme sensiblement équivalentes.

On pourra finalement noter une approche atypique : l'architecture IDEA, dans laquelle chaque composant du système est doté de capacités délibératives et réactives, interagissant en tant qu'agents.

III.3.1. La prise de décision

Pour un robot, la prise de décision est l'activité consistant à anticiper le cours des actions à réaliser dans un futur plus ou moins proche. Dans le schéma habituel, un ensemble de buts doivent être atteints (explicitement demandés par un opérateur, ou automatiquement générés en réponse à un contexte donné).

La mise en œuvre de capacités autonomes de prise de décision en robotique est directement liée aux choix d'architecture : ainsi, dans les architectures de type réactives, la décision est vue en tant que résultante de la combinaison des composantes réactives élémentaires : aucune entité du système n'est dévolue à la prise de décision en tant que telle. C'est à la fois la force (simplicité architecturale, robustesse) et la faiblesse (limitations pour la réalisation de tâches complexes) de ce paradigme.

Dans les architectures délibératives et hybrides, au contraire, un (ou un ensemble de) composant(s) bien défini est en charge de raisonner sur des modèles afin de générer des plans, c'est à dire décomposer des buts en sous-ensembles de tâches élémentaires. Dans un schéma classique de

planification, le planificateur construit un plan de tâches qui, appliquées depuis un état initial (généralement l'état courant du robot), conduisent à satisfaire les buts [GAN05].

III.4. De l'homogénéité à l'hétérogénéité

La majorité des premiers travaux sur les systèmes multi-robots s'est intéressée aux systèmes homogènes constitués de robots identiques. Cependant, ces dernières années ont vu une évolution des travaux de la communauté robotique vers les systèmes multi-robots hétérogènes. Il s'agit de systèmes constitués de robots dotés de capacités différentes notamment au niveau physique : caractéristiques mécaniques, mode de locomotion, capteurs, actionneurs…etc. Ghallab résume bien cette tendance [GHA94].

Les motivations fondamentales de la recherche en systèmes multi-robots sont:

1) la capacité de résoudre des problèmes qui sont intrinsèquement distribués dans l'espace, dans le temps ou dans la fonctionnalité,

2) la capacité de résoudre des problèmes plus rapidement grâce au parallélisme,

3) la capacité d'augmenter la robustesse des solutions par la redondance.

Dans une proportion importante de la recherche sur systèmes multi-robots, les avantages du parallélisme, de la redondance des solutions distribuées dans l'espace et dans le temps sont obtenus par l'utilisation de robots homogènes, qui sont complètement interchangeables. Cependant, un nombre croissant de recherches essaye de répondre aux questions liées à l'utilisation de robots hétérogènes. Ces recherches impliquent généralement un nombre relativement faible de robots – peut-être de l'ordre d'une dizaine de robots ou moins. Même la recherche en robotique homogène est rarement expérimentée avec des équipes de plus de dix à vingt robots.

Les applications complexes futures qui exigent l'utilisation simultanée de grandes équipes avec plusieurs capteurs, qui ne peuvent pas être groupés sur un seul type de robot. Les robots devraient sans doute être conçus aussi avec des tailles plus petites, ce qui limiterait leur charge utile, ou bien rendrait certains capteurs nécessaires trop cher à dupliquer dans toute une équipe de plus de 100 robots. Cela conduit à la nécessité de permettre à de nombreux robots hétérogènes de travailler ensemble en coopération pour résoudre des applications intéressantes».

III.5. Communication et coopération dans les systèmes multi-robots

Du point de vue général, comme conclut Ghallab dans [GHA94], une des questions centrales qu'on doit traiter dans un système multi-robots, quel que soit le domaine d'application, est comment faire coopérer efficacement les robots pendant une mission, d'une manière ou d'une autre, soit automatiquement soit, éventuellement, par l'intervention d'un opérateur externe. La coopération signifie que les robots doivent communiquer pour échanger des informations et coordonner leurs actions dans le but d'accomplir une mission commune globale [ARR06].

La coopération est le point clé pour exploiter le potentiel des systèmes multi-robots [GER02]. La communication est une condition préalable et indispensable pour n'importe quel algorithme de coopération. La présence d'un canal de communication sûr et de débit suffisant permettrait de mettre en place un mécanisme de coopération sophistiqué et efficace.

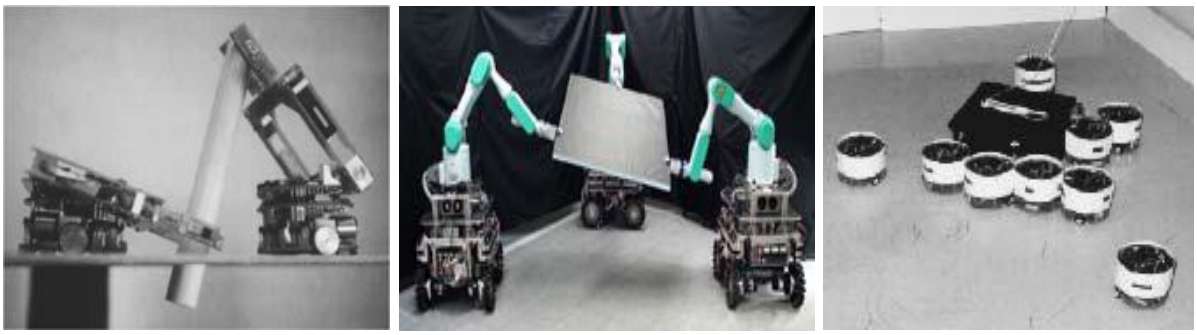
La communication entre les robots dans une équipe peut être réalisée implicitement ou explicitement. La communication implicite, typiquement via l'environnement, est généralement accomplie par les actionneurs et les capteurs des robots. Cela limite à la fois la quantité de données transmises et le degré d'abstraction des informations ainsi échangées. Par conséquent, la communication implicite ne convient pas pour des mécanismes sophistiqués de coopération. Il faut recourir à la communication explicite. D'autre part, avec l'avancement des technologies de communication, les robots d'aujourd'hui sont équipés d'interfaces de communication sans fil haut-débit qui leur fournissent un moyen de communication sûr.

III.5.1. La coopération de robots

La robotique collective est synonyme d'existence d'un groupe de robots. Elle stipule non seulement la mise en place d'un contrôle individuel pour chaque robot, mais impose aussi l'utilisation de stratégies de contrôle appropriées afin que l'assemblage de toutes ces entités robotiques engendre des configurations cohérentes et efficaces pour la réalisation des tâches désirées.

Généralement, faire appel à un groupe de robots au lieu d'en utiliser qu'un seul est motivé par deux facteurs majeurs :

1 - soit la tâche à exécuter nécessite impérativement la coopération d'un nombre minimal de robots, c'est-à-dire, que la tâche en question ne peut s'accomplir sans l'intervention simultanée d'un nombre critique « *Nc* » de robots. Dans [MAR95], les auteurs font coopérer des robots munis de pinces (Figure.III.2(a)) en faisant en sorte qu'ils puissent synchroniser leurs activités dans le but de retirer un long bâton d'un trou. Les travaux menés par Yasuhisa Hirata [HIR02] (Figure.III.2(b)) permettent de contrôler un ensemble de bras manipulateurs montés chacun sur un robot mobile pour réaliser la tâche dite des déménageurs (par exemple, soulever et déplacer un objet trop volumineux et/ou trop lourd par la coordination des activités d'un ensemble de robots). Dans [KUB00], Ronald Kube (Figure. III.2(c)) utilise quant à lui plusieurs robots réactifs afin de pousser un objet trop lourd pour un seul robot. Dans cet exemple, la poussée simultanée d'au moins deux robots est nécessaire pour déplacer l'objet. .



(a) (b) (c)

Figure. III. 2 : Tâches nécessitant la coopération de plusieurs robots

La coopération dans ce premier cas de figure exige donc impérativement, l'intervention d'un groupe de robots pour pallier le manque : d'effecteurs, de puissance, d'informations ou bien plus radicalement le manque d'intelligence embarquée dans chacune des entités robotiques.

2 - soit l'amélioration de certaines performances liées à l'exécution des tâches à réaliser, comme par exemple : la rapidité : on cherche à obtenir un niveau de performance élevé en tirant parti du parallélisme des tâches. e.g., l'exploration parallèle d'un environnement inconnu par ensemble de robots mobiles, dans le but de faire soit une cartographie de l'environnement, soit la réalisation d'une tâche de fourragement [COH96], [STY01]. Ces deux tâches peuvent être réalisées par un seul robot, mais l'ajout d'autres robots va faire en sorte d'accélérer l'exécution des tâches en question :

- fiabilité et la robustesse : les performances du contrôle peuvent être très peu affectées en cas de défaillance d'un agent par exemple [MAT95].

- flexibilité : possibilité d'exécuter les tâches désirées, de diverses manières. Ceci est induit principalement par la redondance des entités robotiques,

- émergence : l'idée ici est de produire une performance collective qualitativement supérieure à celle de l'addition des unités.

Cependant, le fait de contrôler/coordonner plusieurs robots au lieu de n'en contrôler qu'un seul, complexifie considérablement la commande du système, et en raison de l'augmentation :

- De la dynamique des interactions entre entités robotiques dans l'environnement. Ces interactions sont susceptibles, si elles sont mal maîtrisées, d'influer d'une manière néfaste sur l'évolution du système, car les robots peuvent se gêner, se bloquer, se désynchroniser, etc.
- Du nombre de variables régissant l'évolution du système, dû directement à l'augmentation du nombre de systèmes (robots) dans l'environnement, de la complexité du contrôle inhérent déjà à un seul robot, qui doit non seulement agir en fonction des stimuli lui parvenant de l'environnement mais doit aussi adapter son comportement à ceux des autres individus. C'est-à-dire que le robot va essayer de tendre vers un équilibre viable voire optimal pour l'exécution de la tâche coopérative,
- Des incertitudes perceptuelles des robots. Celles-ci sont dues principalement à la nature hautement dynamique qui peut être engendrée par le système multi-robots. Ces incertitudes peuvent complexifier aussi davantage le contrôle du système si elles sont effectives au niveau d'un grand nombre de capteurs.

Les points cités ci-dessus, sont parmi ceux qui font que le contrôle d'un système multi-robots, est plus délicat à mettre en place que le contrôle d'un seul robot élémentaire.

Pour rester dans le même cheminement d'idée quant à l'analogie faite entre un système à asservir et la réalisation d'une tâche par un robot mobile, nous étendons cette analogie pour le cas d'un système multi-robots (Figure. III.3). Nous remarquons d'une part, que les robots partagent le même environnement et d'autre part, que les décisions (commandes) générées par chacun des contrôleurs, sont aussi influencées par les interactions avec les autres robots.

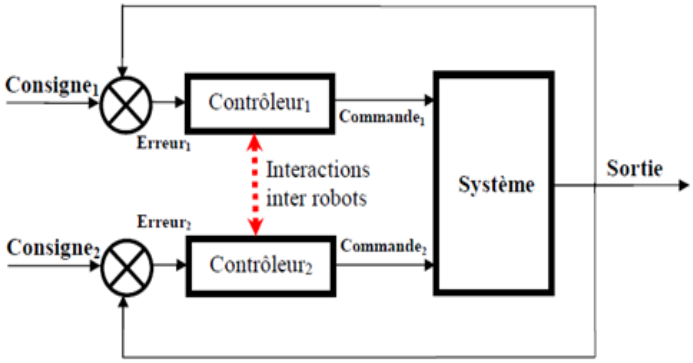


Figure III.3 : Schéma d'asservissement d'un système multi-robots

Pour parler effectivement de robotique coopérative, les robots doivent impérativement partager et/ou intervenir simultanément sur des ressources communes (par exemple, un objet à déplacer collectivement, une voiture à souder ou à peindre dans une chaîne de montage de voiture, etc.), ce qui signifie que, d'une part les robots doivent évoluer dans le même environnement, et d'autre part qu'ils sont appelés à coordonner leurs actions pour réaliser la tâche coopérative. Cette coordination peut être très élaborée, et donc obtenue via des mécanismes de très haut niveau, que l'on peut qualifier de cognitifs. Cette coordination peut aussi être complètement émergente des interactions induites pas les robots, et dans ce cas de figure l'architecture de contrôle est qualifiée de réactive [BRO87].

III.5.2. Projets effectifs et tâches génériques pour la robotique collective

Les projets applicatifs potentiels et effectifs impliquant un ensemble de robots mobiles, sont nombreux et très prometteurs. Ceci est dû principalement aux nombreuses plus-values qualitatives et quantitatives qui peuvent être apportées par rapport aux approches traditionnelles existantes. Parmi les applications majeures qui tirent et qui pourraient tirer profit des recherches menées sur la coopération d'un groupe de robots autonomes, nous pouvons citer : les applications militaires telles que le maintien de patrouille en formation [BAL95b], [BON99]; l'exploration spatiale [HUN03]; le transport coopératif [ALA98a], [AHM01], etc.

Les tâches génériques en robotique collective ont comme principal objectif de fournir des tâches de références "benchmark" pour évaluer les architectures de contrôle proposées dans la littérature. Ces évaluations se font sur plusieurs aspects :

- quantitatif et qualitatif : ceci concerne par exemple les tests d'amélioration de certaines métriques caractérisant les tâches à exécuter.

- méthodologique : au sens que nous trouvons des approches de contrôle émanant de différentes disciplines comme l'automatique, l'informatique, ou les sciences du vivant,

- conceptuel : nous pouvons alors confronter les avantages et les inconvénients d'un contrôle centralisé par rapport à un contrôle distribué, ou d'un contrôle cognitif par rapport à un contrôle réactif.

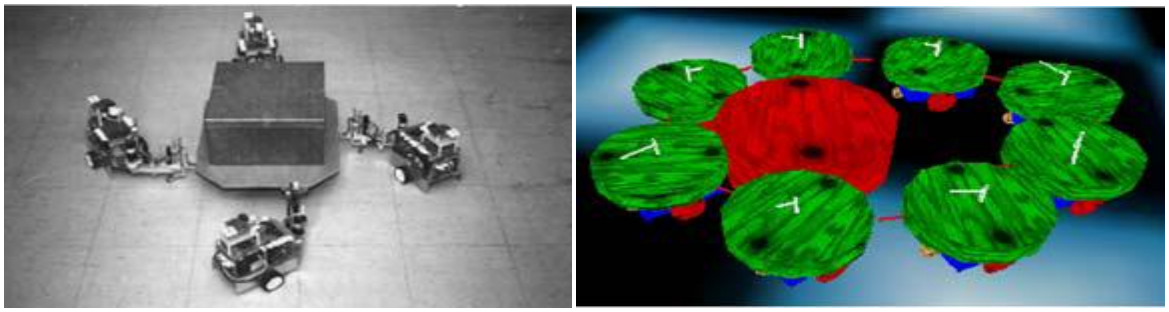
Ces tâches génériques sont censées exciter l'ensemble des modes de fonctionnement du système multi-robots. Par conséquent, la viabilité, la pertinence et les différentes caractéristiques qu'offre le contrôle utilisé sont ainsi systématiquement évaluées. Ces évaluations permettront

ultérieurement d'appliquer l'architecture de contrôle proposée sur des applications effectives, totalement différentes des tâches génériques mais qui partageraient certains points caractéristiques avec l'application désirée.

Nous présentons dans ce qui suit, un aperçu de la multitude de tâches génériques couramment traitées en robotique coopérative.

III.5.2.1. Transport et manipulation coopérative d'objets

L'objectif dans ce type de tâches est de faire coopérer un groupe de robots mobiles pour soulever et/ou porter et/ou réorienter et/ou pousser des objets qu'un seul robot ne pourrait réaliser tout seul. De très nombreux travaux traitent de ce type de tâches et cela vu son aspect pratique et potentiellement transposable à des applications effectives. Dans [STI93], Stiwell utilise plusieurs robots porteurs qui se positionnent en dessous d'une palette chargée pour la déplacer. Les robots mobiles disposent d'un capteur de force qui leur sert d'information pour contrôler le déplacement de la palette d'une manière stable et distribuée. Dans le même contexte Ahmadabadi dans [AHM01] propose une architecture de contrôle distribuée nommée constrain-move strategy, et ce pour soulever, orienter, déplacer, déposer une charge (Figure. III.4 (a)) le long d'une trajectoire contrainte par la disposition des robots. Rus dans [LAB93] utilise des robots pousseurs qui sont guidés par des scénarios appropriés pour orienter collectivement des objets volumineux dans l'environnement. L'approche proposée par Rus dépend d'une multitude d'informations capteurs et d'une communication de haut niveau entre les robots. Dans le projet européen Swarm-Botsun groupe de robots (nommés s-bots) à la possibilité de s'auto-assembler pour former différents types de structures physiques. Parmi elles, on trouve une structure qui permet de pousser-tirer d'une manière coopérative des objets en les entourant complètement (Figure. III.4 (b)) afin de donner ainsi la possibilité à certains robots de pousser et à d'autres de tirer [BAL03]. Dans [KUB00], Kube s'inspire des comportements des fourmis (par exemple, déplacement des proies), pour contrôler un groupe de robots complètement réactifs, afin de réaliser une tâche de poussée d'objets. Il est à noter que Kube, dans son architecture, n'a prévu aucun mécanisme de communication entre robots ce qui rend le contrôle proposé d'autant plus réactif.



(b) (b)

Figure III.4 : Exemple de manipulation coopérative d'objet

Il existe dans la littérature de nombreux autres travaux traitant de tâches en relation avec le transport et la manipulation coopérative d'objets. Nous pouvons citer ceux avec les mécanismes de coopération [CAL90], [DON94], [MAT95], [YAM01] et [MUL98].

III.5.2.2. Mouvement en formation

Naviguer en formation et en présence d'obstacles est la tâche générique qui consiste à simultanément:

- se déplacer vers l'objectif,

- éviter les obstacles statiques,

- éviter les autres robots,

- et maintenir la formation.

Dans [BAL95b], Balch propose de contrôler la navigation de quatre robots mobiles en formation dans un milieu contraint (présence d'obstacles) en utilisant des schémas moteurs spécifiques. Les configurations spatiales testées sont représentées en Figure III.5. Dans [YAM01], Yamaguchi traite cette tâche générique en distribuant complètement le contrôle sur les entités autonomes qui n'exploitent que des informations localisées pour établir leurs actions. Yamaguchi établit aussi un modèle mathématique pour démontrer la faisabilité et la stabilité de l'approche proposée. Parmi les autres travaux qui traitent de la tâche de mouvement en formation, nous pouvons citer : Chen dans[CHE94] et Wang dans [WAN91].

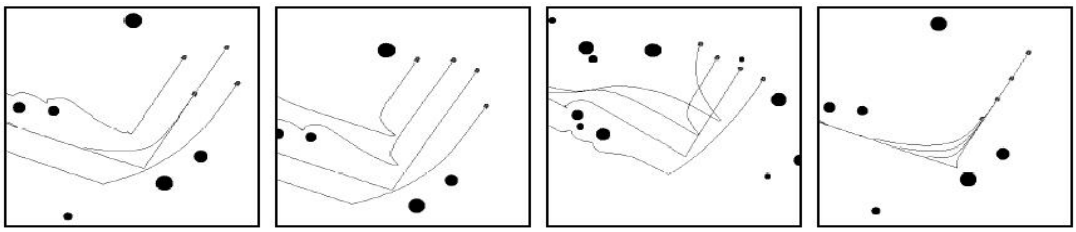


Figure III.5 : Formations pour quatre robots mobiles (de gauche à droite : diamant, cale, ligne, colonne). Les points noirs représentant les obstacles [BAL95b].

III.5.2.3. Fourragement

La tâche de fourragement caractérise à l'origine les sociétés d'insectes, comme celles des fourmis. Cette tâche consiste à rechercher la nourriture dans l'environnement soit d'une manière aléatoire, soit d'une manière dirigée (e.g.., l'utilisation des phéromones pour guider les autres congénères vers la source de nourriture) et à la transporter par la suite au nid. Cette tâche a inspiré bon nombre de travaux en robotique, et cela bien évidemment en remplaçant : les fourmis par des robots, la nourriture par des objets caractéristiques dans l'environnement, les phéromones par des communications directes entre robots, et le nid par une position identifiée dans l'environnement. Dans [MAT04], Mataric' utilise un groupe homogène de sept robots mobiles communiquant entre eux, pour trouver et collecter des palets distribués aléatoirement dans l'environnement. Dans [BAL97], Balch à investigué l'importance de la communication pour réaliser différentes variantes du fourragement en l'occurrence : la mine[4], la consommation[5] et l'exploration[6]. Dans [VAU00a], [VAU00b], Vaughan a implémenté la tâche de fourragement avec quatre robots autonomes. Ces robots tiennent compte entre

[4] Consiste à trouver une ou plusieurs mine(s) et à rapporter le minerai à la base.
[5] Cette tâche a pour effet de consommer le minerai trouvé sur place.
[6] Consiste à parcourir la plus grande surface possible de l'environnement.

autres de leur odométrie pour diffuser les positions des sites de récolte trouvés (en utilisant une communication hertzienne). L'auteur constate une certaine robustesse du système multi-robots, malgré les perturbations évidentes, dues aux erreurs d'odométrie. Il est à noter que le fourragement est l'une des tâches génériques la plus étudiée dans la littérature. C'est ce qui justifie le nombre important de travaux traitant de cette tâche [DRO93], [STE90], [ARK92], etc.

Il y a d'autres tâches génériques en robotique collective. Parmi elles nous pouvons citer : la tâche de tri d'objets " sorting objects" qui consiste à mettre les objets de même nature dans les mêmes tas (par exemple, tri des disques en fonction des couleurs) [MEL01] (Figure. III.6), l'exploration d'environnements inconnus dans le but de faire une cartographie de l'environnement [COH86].

Figure III.6 : Tâche du tri d'objets

III.5.3. La communication

La communication est la base de la résolution coopérative de problèmes. Dans [BAL95a], Balch et Arkin ont testé et confirmé l'importance de différents types de communication entre robots pour l'exécution de tâches coopératives telles que le fourragement. Communiquer d'une manière générale peut être synonyme de partage d'informations, d'expression des états internes, de négociation pour trouver le meilleur compromis aux situations, de requête pour avoir des informations, etc. En fait, ce sont tous les mécanismes qui conduisent, s'ils sont bien traités, analysés et interprétés, à ce que le groupe soit plus efficace. L'importance évidente de la communication nous amène donc à étudier plus en détails ces mécanismes dans le cadre plus précis d'un système multi-robots. On note que cette communication sert essentiellement à orienter les interactions entre robots vers des situations non conflictuelles (par exemple, éviter les conflits spatiaux temporels et/ou coopératives).

La communication entre agents est catégorisée dans ce qui suit, selon plusieurs critères tels que le degré de sophistication, le type d'informations échangées, les mécanismes de fonctionnement.

III.5.3.1. Communication de haut niveau

On entend par communication de haut niveau, toute communication s'exerçant entre individus via des mécanismes de protocoles évolués, tels que :

- l'établissement d'une connexion orientée entre deux entités avec des identifiants (émetteur $_{idx}$ / récepteur $_{idy}$),
- l'échange d'informations de haut niveau, comme par exemple :

• l'état d'achèvement des sous-tâches entreprises par chaque robot, ceci est utilisé par exemple dans l'architecture de contrôle ALLIANCE proposée par Parker [PAR03],

• les plans d'actions propres à chaque robot, pour trouver le meilleur compromis des actions à entreprendre pour chaque robot [AGU97],

• la mise aux enchères des tâches qui doivent être exécutées. Dans [GER02], Gerkey définit le protocole nommé MURDOCH, qui réalise une allocation dynamique des tâches à effectuer par un groupe de robots mobiles. Le protocole qu'il propose est la mise aux enchères (ou avec une forme d'appel d'offre) des tâches qui doivent être exécutées "action-based task allocation" et laisse par la suite le robot le plus apte (le plus offrant) exécuter la tâche en question.

Il est à noter que les types de communication cités ci-dessus exigent l'échange de grands flux d'informations.

- L'existence de protocole de validation de la réception des messages,

- l'existence d'un tableau noir "Blackboard[7]" qui a comme effet de centraliser les communications entre agents. Le projet MARTHA[8] [ALA98a] par exemple utilise une forme de tableau noir pour pouvoir coordonner l'activité de plusieurs dizaines de robots mobiles transportant des conteneurs dans des gares, des ports ou des aéroports. Dans ce système, un superviseur centralise les décisions et les communications globales du système multi-robots.

La Figure III.7 montre un scénario possible d'une communication de haut niveau. Il est à noter que la littérature regorge de travaux et de réalisations en matière de protocoles de communication que nous avons nommé ici haut niveau[9]. Ils ne sont pas tous appliqués en robotique mobile coopérative, mais demeurent très facilement transposables.

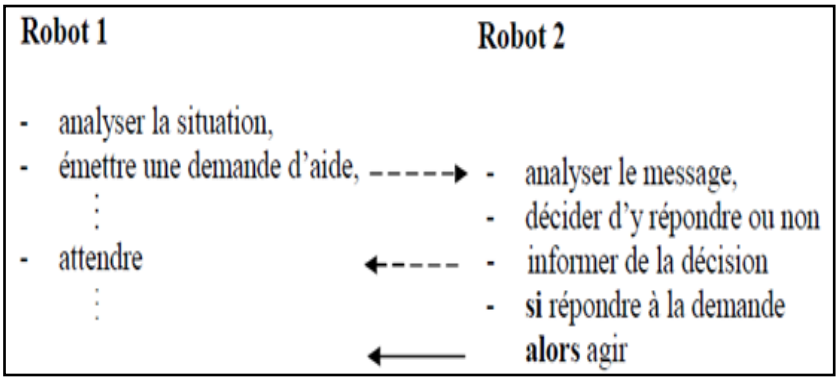


Figure III.7 : Scénario possible d'une communication de haut niveau

III.5.3.2. Communication de bas niveau

La première interrogation qui peut nous interpeller, est de savoir à partir de quel moment juge t'on qu'une communication est de bas niveau ?

[7] 0n définit : toute structure qui centralise les messages émis par les différents agents présents dans l'environnement et ce à la manière d'une base de données partagée par Labidi et Lejouad [LAB93]. L'objectif étant par la suite de pouvoir redistribuer les informations sur les agents qui en formulent la requête. L'idée de cette structure est de mieux maîtriser les flux de communication, et de pallier les problèmes de collisions éventuelles entre messages. Le tableau noir constitue donc une mémoire collective robuste et efficace pour les systèmes multi-agents
[8] Multiple Autonomous Robots for Transport and Handling Applications.
[9] Le plus utilisé d'entre eux est sans doute le protocole TCP/IP (Transmission Control Protocol I Internet Protocol) utilisé par Internet.

Effectivement, la limite n'est pas clairement établie entre une communication considérée comme de haut niveau et celle plutôt de bas niveau. Nous pouvons néanmoins affirmer que les communications de type bas niveau :

- S'établissent d'une manière très localisée autour de l'entité émettrice, et véhiculent des informations très simples,
- N'utilisent pas d'identifiant pour caractériser les émetteurs et les récepteurs de message. C'est-à-dire que les messages sont diffusés spontanément,
- N'utilisent pas de protocole particulier de négociation entre agents.

La Figure III.8 montre un mécanisme simple de communication bas niveau. Il est à noter que ce genre d'interaction entre entités est généralement attribuée à des entités plutôt réactives, fonctionnant par stimulus-réponse, c'est-à-dire que le signal reçu par le robot 2 est considéré comme un stimulus directement exploitable sans qu'il y ait besoin d'un prétraitement particulier pour y répondre.

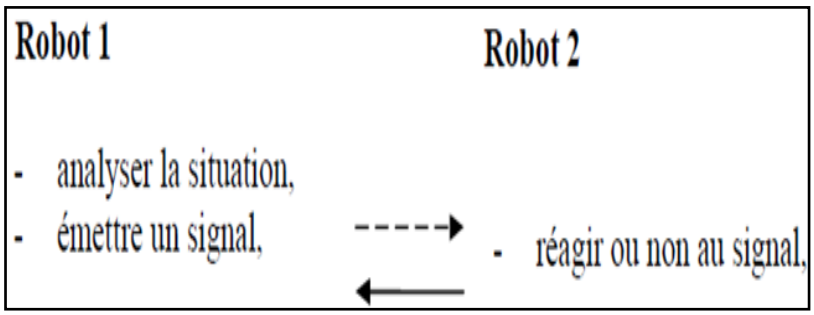


Figure III. 8 : Exemple possible d'une communication de bas niveau

III.5.3.3. Communication indirecte

Dans ce genre de communication c'est l'environnement des agents qui sert de médium de communication [JAN00], [HUG00], et cela à l'exemple des fourmis qui déposent des phéromones dans l'environnement pour guider leurs congénères vers les sources de nourriture. Cette communication indirecte peut être aussi illustrée par ce qui est appelé par Paul Grassé la stigmergie[10] [GRA59].

III.6. Différentes approches méthodologiques pour faire coopérer un ensemble de robots

Les investigations scientifiques traitant de la coordination, de l'organisation, de l'analyse de fonctionnement, du contrôle, de l'auto-organisation [BON99], d'un ensemble d'entités autonomes; l'entité autonome signifiant ici autant, un agent logiciel, un être vivant; qu'un robot mobile. Nous proposons une décomposition en deux ensembles distincts d'approches traitant de l'étude d'un ensemble d'entités autonomes. Le premier ensemble les approches émanant de l'informatique (Système Mufti-Agents), et le deuxième correspond aux approches issues de l'automatique (Robotique

[10] Du grec "stigma", piqûre, point, et "erglia" ou "ergia", action, efficacité, de "ergon", œuvre. Désigne le fait, observé chez les Termites, que le résultat d'un travail exécuté par une collectivité d'individus, dirige à mesure ce travail, ce qui donne l'apparence d'une coordination planifiée. Il s'agit d'une œuvre stimulante. Cette collaboration entre les fourmis a été décrite pour la première fois dans les années 50 par le biologiste français Pierre Paul Grassé (Grassé 1959). Il disait des insectes sociaux : "Ils ne dirigent pas leur travail, ils sont plutôt guidés par lui". À cette forme de stimulation, Pierre Paul Grassé a donné le nom de "stigmergie".

Collective). La richesse et la diversité de ces approchas seront abordées dans ce qui suit. Il est à noter que les travaux des uns et des autres ne sont pas complètement indépendants. Au contraire, les différentes disciplines interagissent, se chevauchent, et se complètent sur bien des points, créant ainsi des ponts entre toutes ces disciplines, et des champs interdisciplinaires pour aborder la coopération entre entités autonomes (Figure. III.9).

III.6.1. Approche Informatique «Système Multi-Agents»

L'intérêt de l'informatique aux notions d'agents intelligents autonomes, a commencé avec les premiers travaux en Intelligence Artificielle "IA" (fin des années 50). Vers les années 70, ces mêmes travaux ont donné naissance à l'Intelligence Artificielle Distribuée "IAD" qui traitent pour leur part de l'intelligence produite par un ensemble d'agents coopératifs [LEN75], [STE90], [AVO92]. De l'IAD a émergé par la suite la notion de Système Multi-Agents "SMA" qui résout des problèmes complexes en les décomposant en un ensemble d'agents autonomes. La spécificité des approches SMA par rapport à celles attribuée à l'IAD est qu'elle impose beaucoup plus de contraintes (structurelles et décisionnelles) par rapport aux entités constituant le système [GRA59].

Les travaux liés au contrôle de systèmes complexes en les décomposant en plusieurs sous-systèmes (agents), ont été abordés dès la fin des années 60 par Michie et Chambers [MIC68]. Les auteurs proposent de commander un pendule inversé par un apprentissage approprié par essai-erreur via la mise en coopération d'un ensemble d'agents (BOX) qui représentent chacun une partition de l'espace d'états atteignable par le système.

Dans [DRE92], Dreyfus critique d'une manière générale le manque d'incarnation (embodiment [BRO90]) des programmes informatiques représentant entre autres les SMA. Ceci engendre selon eux, une incapacité de ces programmes (agents) à interagir de manière significative avec leur environnement et surtout avec leurs semblables (les autres agents). Dans le cadre du contrôle d'entités autonomes situées, l'approche développée par les informaticiens favorise généralement plus l'aspect conceptuel du problème à celui de la prise en compte effective des contraintes physiques caractérisant les agents dans leur environnement. En effet, parmi les tâches coopératives génériques traitées en simulation par cette communauté telle le fourragement, le tri d'objets ou la tâche de poussée d'objets, nous notons le fait que leur exécution se fait généralement sans contraintes physiques explicites. De ce fait, le passage de l'aspect conceptuel vers l'implantation effective des solutions proposées, ne se fait pas toujours hélas sans encombre, car la physique de la manipulation (e.g., soulever, déplacer ou déposer des objets) ou la dynamique de mouvements des entités robotiques ne sont généralement pas modélisées d'une manière précise.

L'autre exemple, caractérisant à la fois, le monde de l'informatique et celui de la coopération d'agents, est celui du génie logiciel. Ce domaine produit des logiciels complexes en faisant coopérer un ensemble d'agents logiciels : classe, objets, ou toute autre forme de technologie basée sur une conception orientée objets[11]. Cette démarche objet qui tend donc à décomposer le système complexe en plusieurs modules (/ou objets) faiblement couplés, a permis d'envisager autrement la conception des logiciels, passant de la notion de programme à celle d'organisation [ERC91].

[11] Tels que : l'ActiveX ou l'OLE "Object Linking and Embedding" qui sont des technologies Microsoft qui permettent de créer et d'éditer des objets partageables par de plusieurs applications

Les secteurs liés au génie logiciel ou aux réseaux de communication [BON99] montrent des succès certains dans leurs applications les plus diverses, et cela malgré la complexité des systèmes traités. Il est à noter cependant que la nature de ces systèmes, tendent à spécifier exactement : les agents qu'ils manipulent (attributs, méthodes, protocoles...), les ressources partagées, les communications et la hiérarchie existantes entre agents. De plus, la nature des interactions entre agents est quasi déterministe. Tout ceci permet donc de réduire considérablement voire éliminer complètement tout type d'aléa quant aux scénarios prévus pour le fonctionnement de ces systèmes complexes.

Les travaux attribués aux informaticiens, comme nous pouvons le constater ci-dessus, traitent des agents sous plusieurs formes, qu'ils soient agents virtuels (plutôt orienté logiciel) ou bien situés dans un environnement (tel qu'un routeur de communication ou un robot mobile). Les apports méthodologiques et conceptuels potentiels qui peuvent être apportés par les informaticiens (ou plus spécifiquement par la communauté SMA) pour faire coopérer un groupe de robots mobiles, sont donc certains.

III.6.2. Approche Automatique «Robotique Collective»

Parler de contrôle en automatique est avant tout synonyme d'existence d'un modèle décrivant le système à contrôler[12]. Même si le système n'est pas trivial à modéliser, alors une estimation de ces paramètres est effectuée pour avoir une représentation la plus proche possible du système à contrôler [KHA99]. Le contrôle d'un robot mobile se fait par exemple en utilisant l'un des modèles qui le représente soit géométriquement, soit cinématiquement ou bien dynamiquement. Le choix d'un modèle se fait en fonction du type de contrôle et de la tâche qu'on veut lui faire exécuter [LAU01], [TOU92]. Ces modélisations ont pour avantage majeur de pouvoir synthétiser mathématiquement un contrôleur approprié pour le système automatique et ainsi adapter, améliorer le contrôle en fonction des performances désirées de l'asservissement [13] (précision, rapidité, robustesse, etc.). Ainsi, si nous restons dans le même ordre d'idée, faire coopérer un ensemble de systèmes robotiques, est par conséquent aussi synonyme d'existence de modèles régissant le système multi-robots[14]. Effectivement, c'est ce qui est appliqué généralement dans le cadre de la robotique coopérative, où chaque robot est modélisé de telle sorte à ce qu'il intervienne dans le modèle global du système multi-robots, et que les interactions entre les systèmes (robots, objet à manipuler, etc.) se fassent par une modélisation appropriée des couplages.

Dans [KHA95], Khatib utilise une architecture décentralisée pour contrôler deux robots mobiles équipés de bras manipulateurs "vehicle/arm" (Figure.III.9 (a)), et ceci pour manipuler d'une manière coopérative des objets. Nous dénombrons deux types de coopération dans ce système robotique :

La première consiste en la coordination des mouvements entre la plate-forme robotique mobile (pour les mouvements lents et peu précis) et le mouvement du bras manipulateur (pour les mouvements rapides et précis), pour avoir une coopération efficace entre ces deux structures, la

[12] Ceci bien évidemment en excluant les nouveaux types de contrôle émergents, basés par exemple sur de la logique floue, des réseaux de neurones ou de l'apprentissage par renforcement, qui n'ont pas besoin de modèle explicite du système à contrôler

[13] est à noter aussi que contrôler un robot mobile au sens automatique-robotique du terme passe aussi par la prise en compte de modèles tels que ceux liés : aux frottements, aux bruits, à l'holonomie ou non des robots, aux couples maximums engendrés par les moteurs des roues, etc. C'est-à-dire tenir compte effectivement, des contraintes physiques et/ou structurelles des robots, ainsi que de leurs interactions avec leur environnement

[14] "Dans ce qui suit nous nous focaliserons sur la coopération de robots mobiles qui manipulent (i.e., déplacent, soulèvent, déposent) une ressource commune

seconde correspond à la coopération des deux "vehicle/arm" pour la manipulation commune d'objet. Khatib a considéré l'ensemble des structures robotiques en contact comme une seule structure redondante. Les forces internes du système (i.e., les forces connectant les effecteurs des bras manipulateurs avec l'objet à manipuler) sont modélisées comme des articulations virtuelles. Pour avoir un modèle cohérent des articulations virtuelles, il faut néanmoins considérer que le contact est rigide et s'effectue sans glissement, ce qui est très difficile à respecter. Afin de rectifier les erreurs inhérentes à la modélisation, Khatib a prévu une communication entre les robots.

(a) (b)

(a) Coopération de deux PUMA 560 placés sur des plateformes mobiles (b) Coopération mixte entre des robots et un humain

Figure III.9 : Coopération de robots au sens automatique-robotique

Dans [HIR00], Hirata propose un algorithme décentralisé de contrôle de deux robots, qui doivent déplacer un objet sans connaissance préalable ni de la position des autres robots, ni de la géométrie de l'objet à déplacer. Le déplacement de l'objet est effectué en utilisant un robot leader et un autre suiveur. Le leader connaissant à tout moment l'objectif à atteindre, il impulse alors une force à travers l'objet à déplacer pour désigner au robot suiveur la direction désirée. Il est à noter que les robots disposent de capteurs d'effort pour détecter le module et l'orientation de la force appliquée via l'objet. Ce principe a été développé par la suite dans [HIR02] pour réaliser une coopération mixte entre des robots et un humain qui impulse les forces et les moments nécessaires pour guider les déplacements de l'objet (Figure. III.9(b)).

Dans [SAS95], Sasaki traite aussi le cas du transport (soulever-acheminer) d'un objet par plusieurs robots. L'objectif de ces travaux étaient de trouver la géométrie optimale des robots en dessous de l'objet à transporter, et ce afin de garantir à la fois, la maximisation de la stabilité de l'objet posé sur les robots et la minimisation de l'énergie fournie par les robots. La Figure III.10 donne les différentes étapes nécessaires pour réaliser la tâche de soulever-acheminer selon Sasaki.

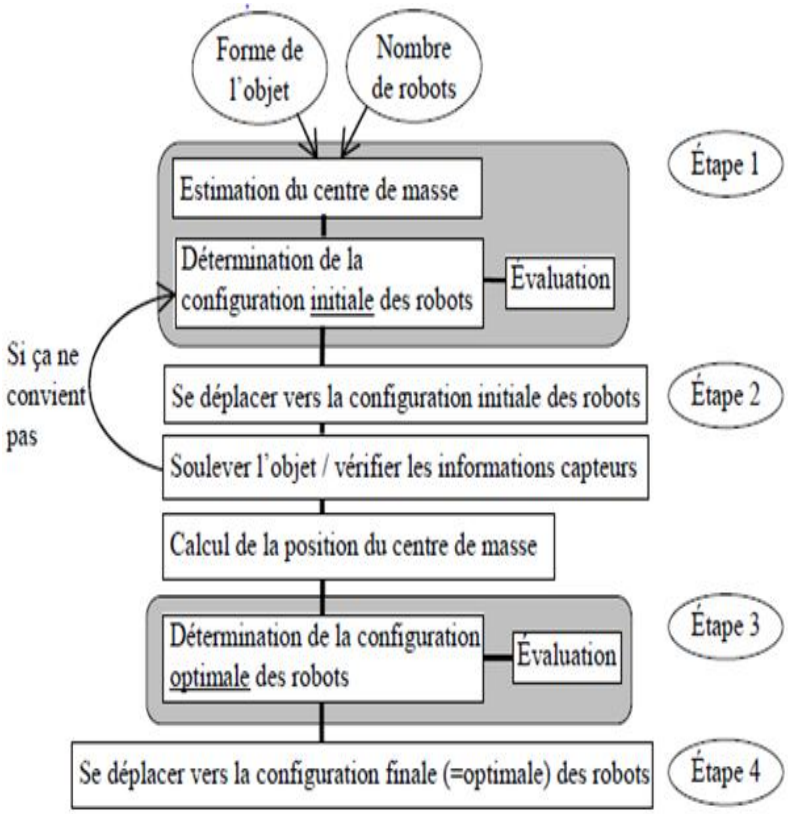


Figure III.10 : Algorithme pour trouver la configuration optimale pour saisir un objet [SAS95]

Dans [STI93], Stilwell utilise une architecture décentralisée avec un leader pour contrôler un ensemble de robots mobiles non holonomes. Les robots se positionnent en dessous d'une palette chargée pour la déplacer d'une manière stable. Chaque robot est modélisé comme une masse et le contact existant entre les robots et la palette est modélisé par un ressort ayant un amortissement omnidirectionnel, ce qui donne une approximation du second ordre à la dynamique des forces transmises entre les robots et la palette à déplacer.

Dans [AHM01], Ahmadabadi propose une architecture de contrôle distribuée nommée constrain-move-strategy pour soulever, orienter, déplacer, déposer, une charge selon une trajectoire contrainte. Le scénario utilisé par Ahmadabadi est inspiré directement des lois de la mécanique, et il consiste à contraindre l'objet à déplacer par un ensemble de robots mobiles (disposant de bras manipulateurs planaires), afin que l'objet ne puisse être déplacé que selon une direction privilégiée. Une fois que ceci est vérifié, il suffit qu'un ou plusieurs robots applique une force dans le sens où l'objet n'est pas contraint afin de le déplacer. L'ensemble des étapes nécessaires pour réaliser cette tâche est implémenté en utilisant une architecture à base de subsomption [BRO87], où les robots communiquent entre eux et connaissent à tout moment leur position et orientation par rapport à l'objet à déplacer.

Il est évident que l'approche prise par les roboticiens pour faire coopérer un ensemble d'entités robotiques, est celle qui permet le mieux de faire une analyse rigoureuse et constructive des solutions proposées. En effet, cette approche permet de quantifier les performances atteintes par le système, au sens automatique du terme : précision, rapidité, robustesse, stabilité, etc. Ceci est effectif vue la disponibilité des modèles régissant le système multi-robots. Cependant, ce qui est flagrant dans ce

genre d'approches, c'est que le nombre de robots en coopération ne dépasse pas quelques individus (deux à quatre robots au maximum), car les modèles et les scénarios proposés deviendraient nettement plus complexes à gérer. Ce qui est notable aussi dans ce genre d'approche, c'est l'utilisation de modèles pour représenter les interactions entre systèmes (robot-robot, robot-objet). Par conséquent, la pertinence du contrôle est étroitement liée à la précision de ces modèles d'interaction. Cependant, ces modèles peuvent s'avérer très peu fiables, surtout si l'environnement est hautement dynamique, et que les robots doivent par exemple se reconfigurer plusieurs fois pour réaliser efficacement leurs manipulations.

Le principal verrou faisant donc que l'approche automatique-robotique ne soit pas en mesure dans l'immédiat de donner des solutions adaptées, viables et surtout exactes au contrôle coopératif d'un grand nombre de robots mobiles (plusieurs dizaines), est lié à la relative pauvreté des modèles existants définissants les interactions entre les robots et leur environnement.

III.7. Conclusion

Ce chapitre nous a permis de présenter les principales spécificités des systèmes multi-robots, et l'élaboration des stratégies qui permettent aux équipes de robots de se configurer de manière autonome en équipes en fonction des objectifs courants de la mission des robots disponibles et de leurs ressources.

L'utilisation de robots hétérogènes capables de jouer plusieurs rôles en même temps, rend le problème de formation difficile à implémenter. Nous proposons dans le chapitre qui suit une réponse qui, sans être idéale, permet de garantir un certain niveau d'optimalité. L'idée est de pouvoir trouver l'équilibre entre les apports (approches, méthodes, outils, etc.) de ces différentes disciplines, pour pouvoir proposer le contrôle le plus fiable et le plus approprié pour faire coopérer un groupe de robots mobiles.

Dans le chapitre suivant, consacrée à notre contribution à la problématique de recherche, nous présentons une étude conceptuelle sur les techniques de communication entre les entités autonomes intelligentes (agents ou robots) distribués dans notre système de navigation où les robots doivent coopérer pour prendre en charge les tâches relevant d'un chantier de construction.

Chapitre IV :

La Conception

Chapitre IV : La Conception

IV.1. Introduction

Le contexte de la navigation en environnement inconnu est particulier. En effet, à part les positions des points de départ et d'arrivée, et parfois les premières observations environnantes, le robot est initialisé sans aucune autre information.

L'optimalité des chemins planifiés dépend de la disponibilité des informations sur le terrain, notamment leur qualité et leur intérêt pour l'atteinte du but, ce qui dépend notamment des endroits où elles sont disponibles.

La conception de nouveaux modèles de planification est l'une des activités critiques des constructeurs des robots mobiles autonomes. En effet, il leur faut pour cela, proposer en permanence des appareils toujours plus performants qui répondent à de nouvelles contraintes réglementaires et qui restent en phase avec leurs clients.

Ce chapitre est une description logique de l'architecture logicielle et les différents modules du notre système. Il constitue une entrée pour les deux dernières phases l'implémentation et le test.

IV.2. Définition du problème

Le travail traite de la problématique de la navigation coopérative de robots mobiles autonomes, on en dégage deux grands axes:

- La communication inter-robots :
  - Contrôle de flottilles de robots.
  - Proposition de l'architecture générale du système.
- Le contrôle et la commande des robots :
  - cet axe regroupe les thématiques liées à la planification de chemins, la génération de trajectoires, et la commande des robots de manière générale.

IV.3. Architecture générale du système

La coopération en robotique consiste à décomposer une mission robotique en un ensemble de tâches qui seront décomposées en sous-tâches jusqu'à trouver la tâche élémentaire (directement réalisable par un robot) selon un mécanisme de coopération. Ce dernier correspond à l'ordre d'exécution et au niveau de synchronisation entre les tâches élémentaires des robots. Une équipe de robots doit collaborer pour atteindre un but commun, ou pour chaque comportement, un ensemble de

conditions est requis. Le rôle du robot reste fixe et ne change pas pendant l'exécution de la stratégie de coopération.

La répartition des tâches dans les systèmes multi-robots est plus performante selon les caractéristiques suivantes :

- Le temps d'exécution.
- La complexité de l'algorithme.
- La robustesse et la tolérance des pannes.

La Figure IV.1 représente l'architecture de notre approche avec les différents agents:

- ***Agent S*** **:** Agent Superviseur, robot chef chantier, il joue le rôle du leader et de coordinateur entre les agents.
- ***Agent C*** **:** Agent Collecteur, il collecte les informations nécessaires pour la construction, les plans, les instructions, l'ordre de l'exécution et prépare les données pour le processus de classification (codage des données).
- ***Agent E*** **:** Agent d'échantillonnage, responsable de la classification des nouvelles données et associe chaque instruction à son propre type (acteur).
- ***Agent O*** & ***Agent M*** **:** les agents de construction (les robots ouvriers et les robots maçons) calculent et exécutent les tâches de construction.
- ***Agent DO*** **:** Agents de décision, valorisent les résultats et choisissent la meilleure proposition à partir d'un ensemble de solutions proposées par les robots ouvriers.
- ***Agent DM*** **:** Agents de décision, valorisent les résultats et choisissent la meilleure proposition à partir d'un ensemble de solutions proposées par les robots maçons.

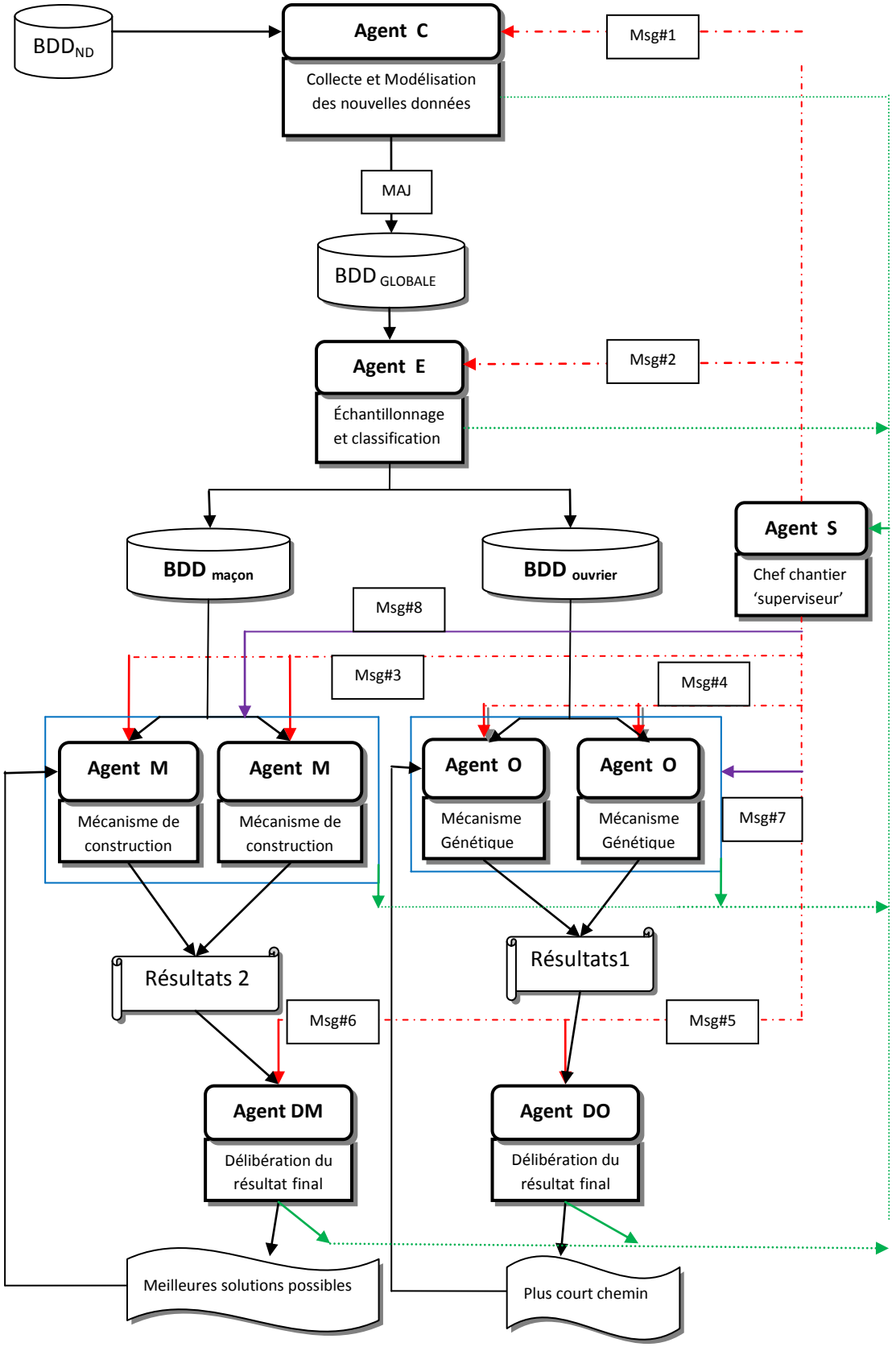
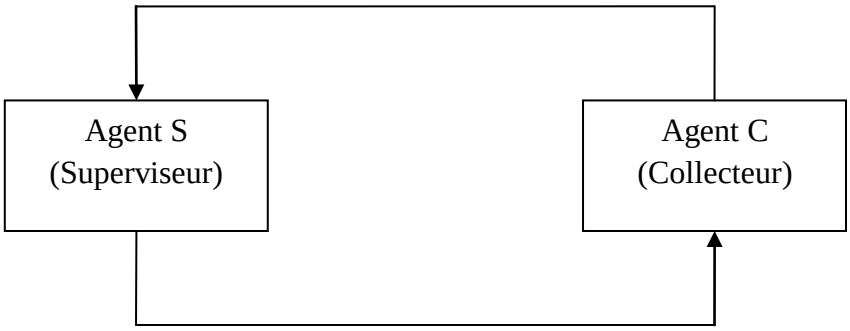


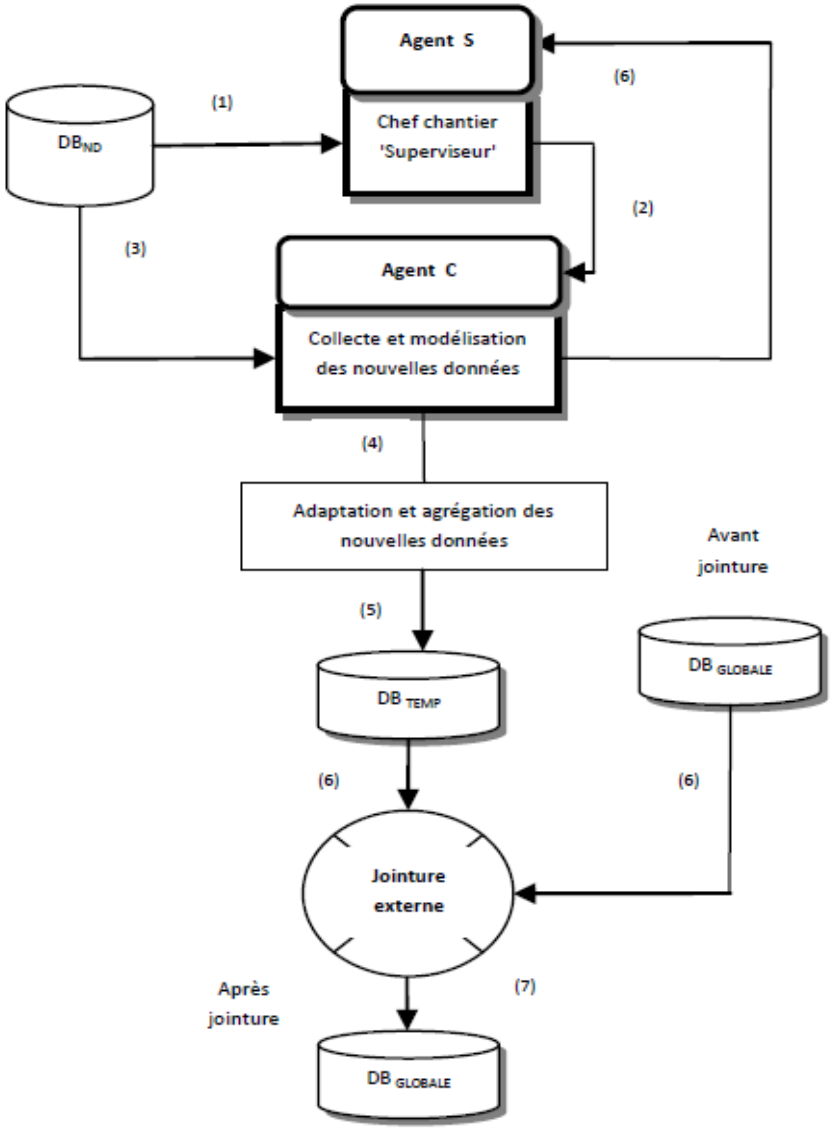
Figure IV. 1 : Architecture générale du système [BEN12b].

IV.3.1. Etape 1 : collecte des nouvelles données

La Figure IV. 2 représente l'interaction entre les deux agents, dans l'étape de collecte des données; c'est un scénario de communication entre l'Agent Superviseur (Agent S) et l'Agent Collecteur (Agent C).



(a) Les agents responsables



(b)

Figure IV. 2 : Schéma de fonctionnement de l'étape de Collecte des nouvelles données.

1- Détection de nouvelles données par l'Agent S.
2- Envoi msg#1 réveil l'Agent C.
3- Accès à BDD_{ND} par l'Agent C.
4- Agrégation et adaptation (codage) des nouvelles données.
5- Jointure externe entre la nouvelle BDD_{ND} et la $BDD_{globale}$.
6- Insertion des nouvelles données et Mise-à-Jour des données existantes.
7- Envoi de message d'achèvement de l'Agent C à Agent S.
8- Suspension de l'Agent C.

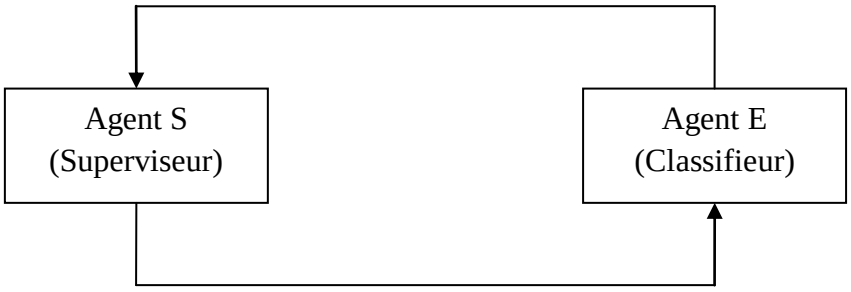
Chaque nouvelle information codée est représentée par : (Tableau IV. 1):

Numéro	Action	Acteur
1	Prendre une brique.	Robot Maçon
2	Casser une brique.	Robot Maçon
3	Mettre une brique.	Robot Maçon
4	Trouver un outil.	Robot Ouvrier

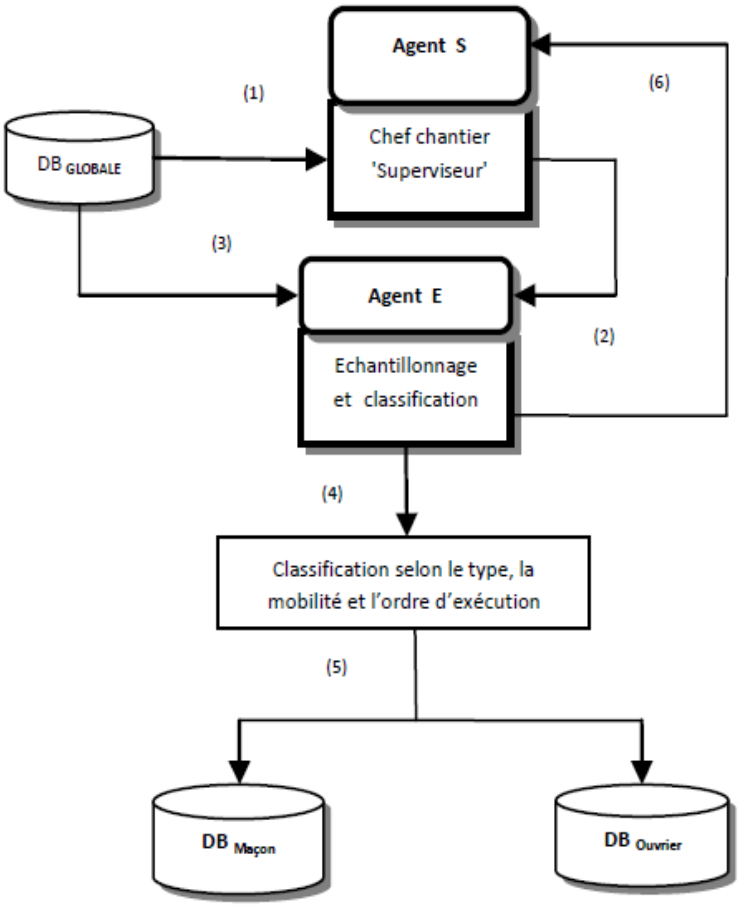
Tableau IV. 1 : Le résultat de la phase de collecte des nouvelles données

IV.3.2. Étape 2 : Echantillonnage et Classification

La deuxième étape, l'étape de classification est un scénario de communication entre l'Agent Superviseur et de l'Agent Classifieur. L'objectif est d'attribuer des activités (des tâches) aux robots employeurs de ce chantier, qui sont l'Agent « M » Maçon et Agent «O » Ouvrier (Figure IV. 3.) :



(a) Les agents responsables



(b)

Figure IV. 3 : Schéma de fonctionnement de l'étape d'Echantillonnage Classification des données.

1- Détection de nouvelles mises à jour dans la $BDD_{globale}$ par l'Agent S.
2- Envoi un msg#2 réveil de l'Agent E par l'Agent S.
3- Etiquetage les données selon leurs types d'activités.
4- Classification et partitionnement de la $BDD_{globale}$ en 2 sous BDD (activités mobiles '$BDD_{ouvrier}$'/activités de construction '$BDD_{maçon}$').
5- Envoi un message d'achèvement de l'Agent E à l'Agent S.
6- Suspension de l'Agent E.

Le but de séparer les activités et de diviser le travail entre les deux catégories des robots employeurs dans le chantier. L'Agent E crée deux bases de données $BDD_{maçon}$ et $BDD_{ouvrier}$.

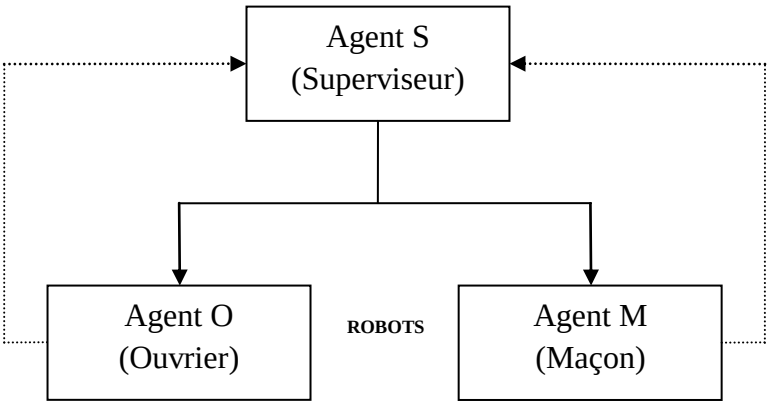
Enfin, insertion dans chaque base de données de l'échantillon correspondant (Tableau IV. 2).

Numéro	Action	Acteur	échantillon	Classe
1	Prendre une brique	Robot Maçon	M	$BDD_{maçon}$
2	Couper une brique	Robot Maçon	M	$BDD_{maçon}$
3	Mettre une brique	Robot Maçon	M	$BDD_{maçon}$
4	Trouver un outil	Robot Ouvrier	O	$BDD_{ouvrier}$

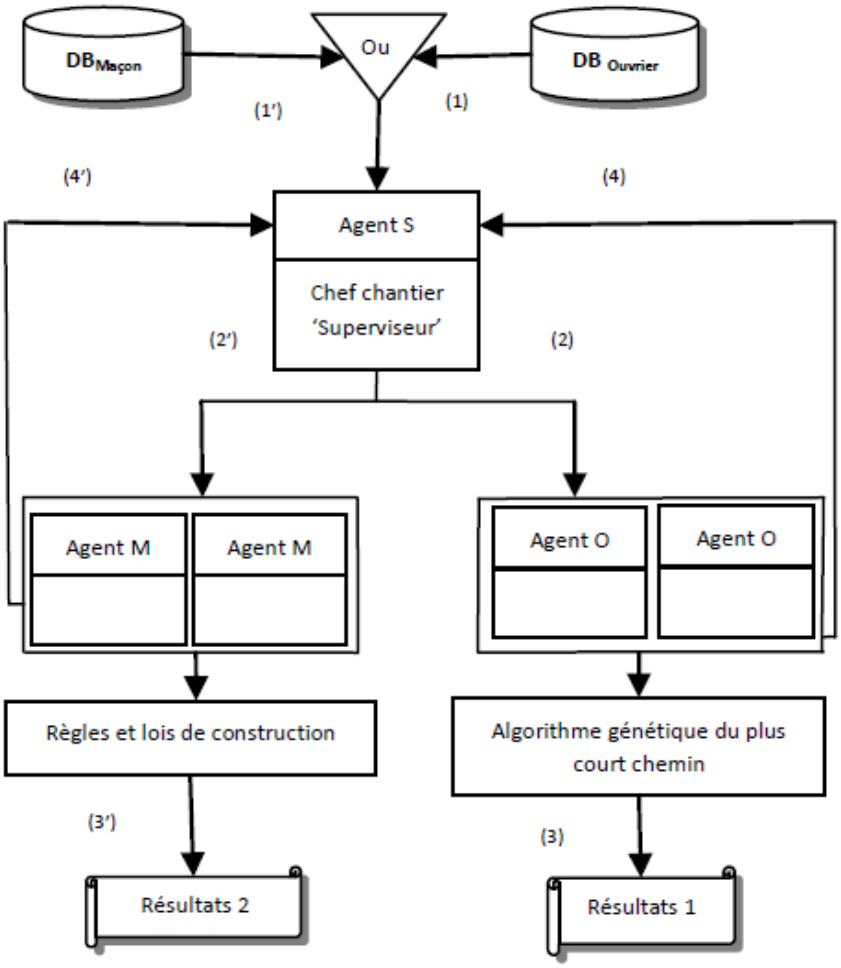
Tableau IV. 2 : Résultat de la phase d'échantillonnage et de la classification

IV.3.3. Etape 3 : Calcul

Cette étape combine deux systèmes; le premier consiste à faire construire un mur par le robot maçon, cette technique est inspirée par la méthode de construction des cellules hexagonales des abeilles. Nous avons remarqué que les abeilles mettent les morceaux de cire une à côté de l'autre, et termine finalement avec de petits morceaux. Dans notre cas, les robots maçon (Agent M) mettent les briques une à côté de l'autre, ils exécutent une fonction d'évaluation de la ligne tracée (briques) et stockent le résultat dans Résultats1. Le deuxième sous-système est lié à la navigation optimale des robots ouvriers en tenant compte de l'évitement des obstacles et la collision entre eux. Pour optimiser la navigation des robots ouvriers, nous utilisons la puissance des algorithmes génétiques pour définir le chemin le plus court. Chaque robot calcule le trajet le plus court entre trois points p1, p2 et p3 (d1 = {p1, p2}, d2 = {p2-p3}}, la distance d1 est la distance entre le robot Maçon et le robot Ouvrier (Agent O); la distance d2 est la distance entre le robot ouvrier et les outils nécessaires. Le résultat est stocké dans Resultats2. (Figure. IV.4).



(a) Les agents responsables

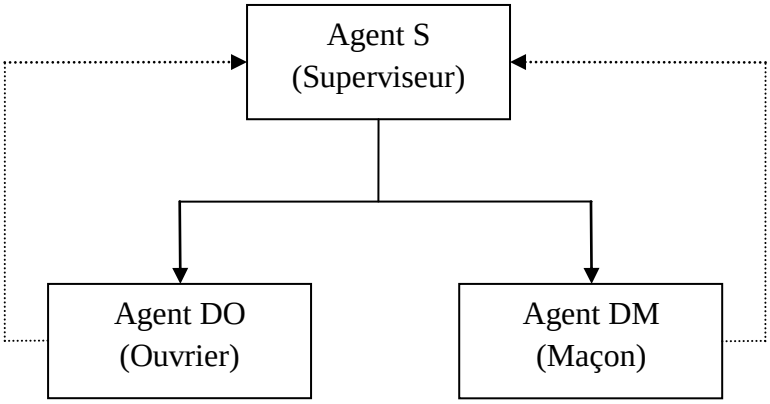


(b)

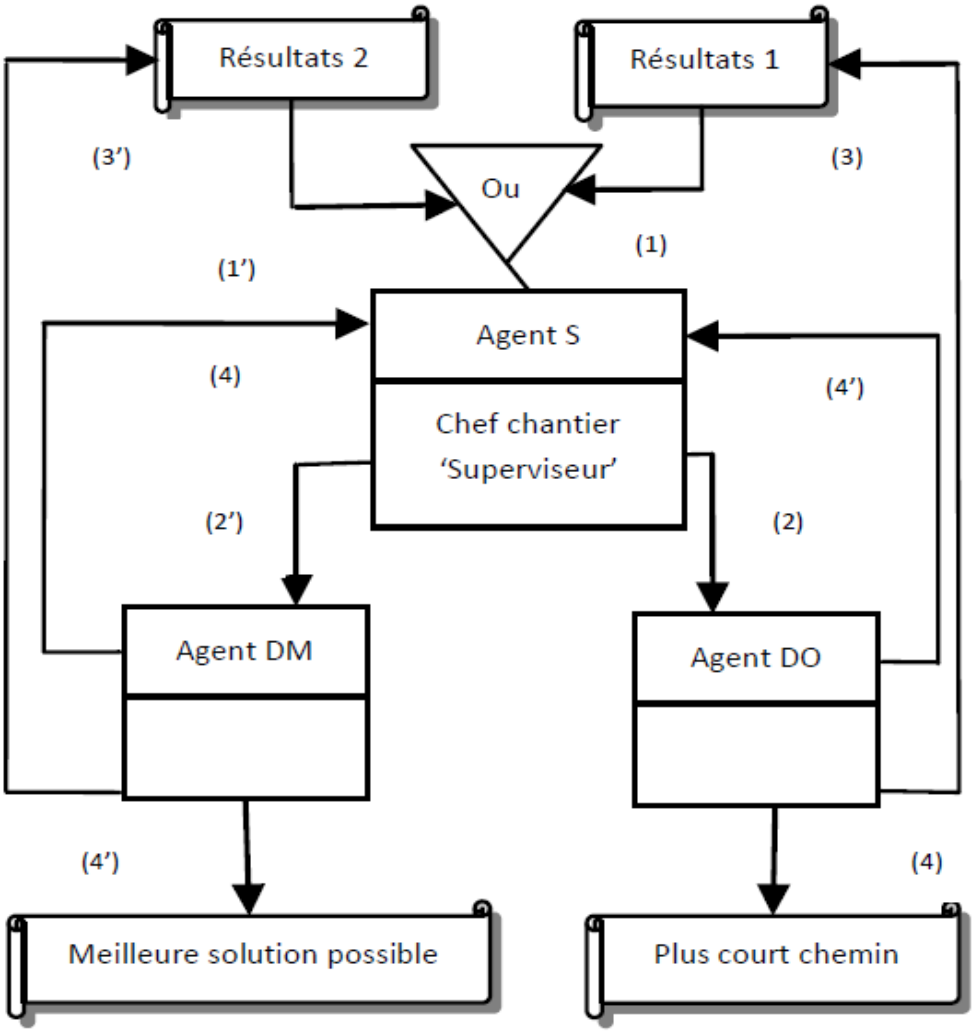
Figure IV. 4 : Schéma de fonctionnement de l'étape de Calcul.

1- Détection des nouvelles mises-à-jour à la 'BDD $_{ouvrier}$ ' par l'Agent S.
2- Envoi un msg#3 réveil Agent O par l'Agent S.
3- Détection des nouvelles mises à jour a la 'BDD $_{maçon}$' par l'Agent S.
4- Envoi un msg#4 réveil l'Agent M par l'Agent S.
5- Si (1) est vrai alors chaque Agent O calcule sa proposition selon des mécanismes de plus court chemin et la stratégie de navigation.
6- Groupement des résultats de calcul des agents Agent O dans le tableau 'Résultats 1'.
7- Si (3) est vrai alors chaque Agent M calcule sa proposition selon des règles de constructions.
8- Groupement des résultats de calcul des agents Agent M dans le tableau 'Résultats 2'.
9- Envoi un message d'achèvement de l'Agent O vers l'Agent S.
10- Envoi un message d'achèvement de l'Agent M vers l'Agent S.
11- Suspension de l'Agent O.
12- Suspension de l'Agent M.

IV.3.4. Etape 4 : de décision



(a)



(b)

Figure IV. 5 : (a) Les agents responsables, (b) Schéma de fonctionnement de l'étape de Décision.

1- Détection des nouvelles mises-à-jour dans le tableau 'Résultats 1' par l'Agent S.
2- Si (1) est vrai alors envoi un msg#5 réveil Agent DO par l'Agent S.

3- Accès au tableau 'Résultats 1' par l'Agent DO.
4- Agent DO délibère la meilleure proposition à partir des résultats stockés dans 'Résultats 1'.
5- Stocker la meilleure solution (proposition) dans 'le plus court chemin' par l'Agent DO.
6- Détection des nouvelles mises-à-jour dans le tableau 'Résultats 2' par l'Agent S.
7- Si (6) est vrai alors envoi un msg#6 réveil Agent DM par l'Agent S.
8- Accès a 'Résultats 2' par l'Agent DM.
9- Agent DM délibère la meilleure proposition à partir des résultats stockés dans 'Résultats 2'.
10- Stocker la meilleure solution (proposition) dans 'meilleure solution possible' par l'Agent DM.
11- Envoi msg d'achèvement par les agents (Agent DO/Agent DM) à l'Agent S.
12- Suspension de l'Agent DO.
13- Suspension de l'Agent DM.

IV.3.4.1. Mécanisme de délibération de meilleures propositions des robots maçons

Construction des murs:

Pour construire un mur sans casser les briques seulement en extrémité, il faut :

- Calculer la longueur du mur,
- Calculer la longueur de la brique,
- Calculer la longueur du mur /longueur de la brique,
  - Le robot «Agent DM» parcourt le mur, brique par brique, et à chaque brique mal positionnée, le compteur du robot-maçon, responsable de cette construction, s'incrémente de 1,
  - Le robot «Agent DM» compte les briques cassées et stocke l'identificateur du robot-maçon qui propose la meilleure stratégie,
  - Le robot «Agent DM» stocke les meilleures stratégies dans «meilleures solutions possibles».

IV.3.4.2. Mécanisme de délibération de meilleures propositions des robots ouvriers

- Les robots-ouvriers déclenchent le mécanisme génétique de calcul du plus court chemin [BEN12a].
- Chaque robot-ouvrier calcule la distance (d_1) entre leur position actuelle et la position de l'outil demandé.
- Chaque robot-ouvrier calcule la distance (d_2) entre la position de l'outil et la position du robot-maçon.
- Le robot « Agent DO » trie les résultats (d_1) obtenus à partir des robots-ouvriers selon un ordre croissant et stocke l'ID du robot qui calcule le chemin le plus court.
- Le robot « Agent DO » trie les résultats (d_2) obtenus à partir des robots-ouvriers selon un ordre croissant et stocke le chemin le plus court.
- Le robot « Agent DO » calcule la distance (d_t) totale par une opération d'addition (dans notre simulation, le chemin est une liste de positions, alors on concatène les deux listes (d_1 et d_2).

- L'agent « Agent DO » met à jour le tableau « plus court chemin » et insère la liste (d_t)

IV.3.5. Etape 5 : de construction

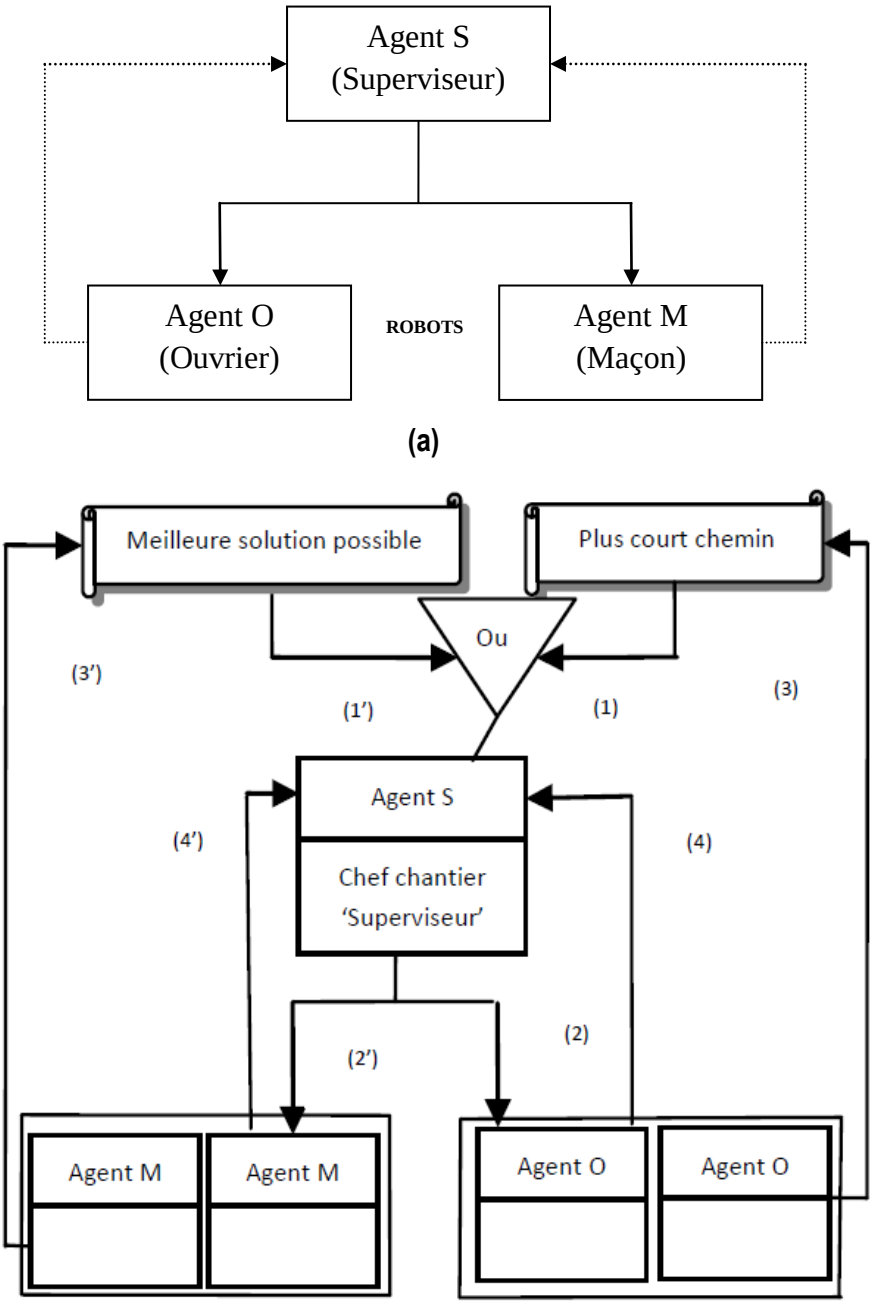


Figure IV. 6 : Schéma de fonctionnement de l'étape Construction.

1- Détection des nouvelles mises-à-jour dans le tableau 'plus court chemin' par l'Agent S.
2- Si(1) est vrai alors envoi un msg#7 réveil l'Agent O associe à la meilleure proposition par l'Agent S.
3- Exécution séquentielle de la meilleure proposition par l'Agent O.
4- Envoi un message d'achèvement à l'Agent S par l'Agent O.
5- Suspension de l'Agent O.
6- Détection de nouvelles mises à jour dans le tableau 'meilleur solution possible' par l'Agent S.

7- Si (6) est vrais alors envoi un msg#8 réveil l'Agent M associe à la meilleure proposition par l'Agent S.
8- Exécution séquentielle de la meilleure proposition par l'Agent M.
9- Envoi un message d'achèvement à l'Agent S par l'Agent M.
10- Suspension de l'Agent M.

IV.4. Le module de raisonnement génétique

Le module de raisonnement génétique définit les mouvements physiques de robot dans son environnement (Figure IV. 7.).

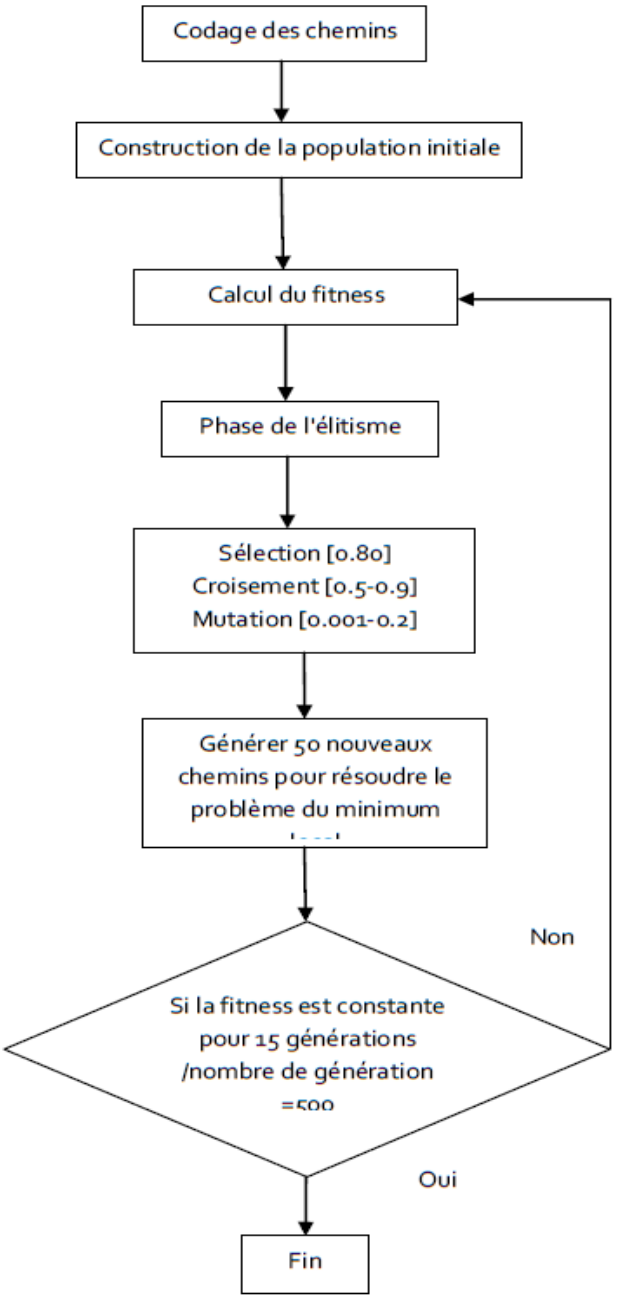


Figure IV. 7: L'algorithme génétique proposé [BEN14].

IV.4.1. Codage

Historiquement, le codage utilisé par les algorithmes génétiques était représenté sous forme de chaînes de bits contenant toute l'information nécessaire à la description d'un point dans l'espace d'état. Ce type de codage a pour intérêt de permettre de créer des opérateurs de croisement et de mutation simples. C'est également en utilisant ce type de codage que les premiers résultats de convergence théorique ont été obtenus.

Un chemin est modélisé sous la forme suivante :

chemin={Xi^1,Xi^2,Xi^3…….,Xi^n}

où :i : est le numéro du robot, l'indice 1..n,

(n) est la longueur du chemin et X un entier ϵ {0, 1, 2, 3}.

Exemple : chemin= {00001333332222211} Où la longueur du chemin n=15.

IV.4.2. Population initiale

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Si la position de l'optimum dans l'espace d'état est totalement inconnue, il est naturel d'engendrer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace d'état, en veillant à ce que les individus produits respectent les contraintes

Elle est générée aléatoirement et correspond aux chemins possibles pour atteindre le point d'arrivée. Avoir une solution acceptable évoque d'envisager 100 chemins possibles afin d'aboutir à la destination.

La population initiale est présentée par une liste de listes pour donner plus de convivialité car la longueur du chemin trouvé n'est pas toujours constante. Où n est le nombre des chemins de chaque génération et x est une variable qui désigne la longueur du chemin.

Pour résoudre le problème de convergence prématurée (minimum local) l'algorithme génère un ensemble de solutions (50 nouveaux chemins), la population initiale est présentée par une liste de listes de donner plus de facilité d'utilisation, car la longueur du chemin trouvé est pas toujours constante, où N est le nombre de trajets de chaque génération et X est une variable qui indique la longueur du chemin (la matrice suivante).

$$\mathbf{Population-initiale} = \begin{bmatrix} \mathbf{w}[1][1], & \mathbf{w}[1][2], & & \mathbf{w}[1][n] \\ \mathbf{w}[2][1], & \mathbf{w}[2][2], & & \mathbf{w}[2][n] \\ \ldots. & \ldots. & \ldots. & \ldots & \ldots. \\ \mathbf{w}[x][1], & \mathbf{w}[x][2], & & \mathbf{w}[x][n] \end{bmatrix}$$

IV.4.3. Fitness

Fitness f est la somme des distances entre deux points consécutifs appartenant au chemin (l'équation (3)), tel que: [BEN15]

$$f = \sum_{i=1}^{S} \text{dis}\ (\text{pos}\ (\text{X}i, \text{Y}i), \text{pos}\ (\text{X}i', \text{Y}i')) \qquad (3)$$

Où pos (Xi, Yi) est la position actuelle du robot et pos (Xi ', Yi') est la position suivante. Le résultat de cette fonction est le nombre de déplacements pour rejoindre le point de départ et le point d'arrivée (l'équation (4)), la longueur du chemin doit être minimum (Figure.IV. 7, section IV.4).

La somme des distances des deux robots doivent être:

$$f = \sum_{j=1}^{M} \sum_{\substack{i=1 \\ i \neq j}}^{M} \frac{\text{dis}\ (\text{pos}\ (\text{X}i, \text{Y}i), \text{pos}\ (\text{X}i', \text{Y}i'))}{2} \qquad (4)$$

Où Dis (pos(Xi,Yi), pos(Xi ',Yi')) désigne les distances de sous-cibles des deux robots. La somme des distances de tous les robots R devraient être : (l'équation (5)):

$$\boldsymbol{f = \sum_{i=1}^{N} \frac{\mathbf{P}(\mathbf{X}i, \mathbf{Y}i)}{\mathbf{P}(\mathbf{X}i, \mathbf{Y}i) + \mathbf{P}(\mathbf{X}i', \mathbf{Y}i')} . R.} \qquad (\mathbf{5})$$

Où P (Xi, Yi) est le chemin du robot i, P (Xi ', Yi') est le chemin inconnu et R une constante positive.

IV.4.4. Sélection

A l'inverse d'autres techniques d'optimisation, les algorithmes génétiques ne requièrent pas d'hypothèses particulières sur la régularité de la fonction objective.

L'algorithme génétique n'utilise notamment pas ses dérivées successives, ce qui rend très vaste son domaine d'application. Aucune hypothèse sur la continuité n'est non plus requise. Néanmoins, dans la pratique, les algorithmes génétiques sont sensibles à la régularité des fonctions qu'ils optimisent. Le peu d'hypothèses requises permet de traiter des problèmes très complexes.

Le cycle de vie des individus apparaît à chaque génération, sur une population de T individus, de sélection est la scène d'un algorithme génétique dans lequel les solutions individuelles sont choisis parmi une population à la reproduction plus tard (de recombinaison ou croisé). Il existe plusieurs algorithmes de sélection de base, 80% y sera conservée, ils peuvent être reproduits (le pourcentage idéal de sélection est entre 70-95). Pour garder les meilleurs individus de chaque génération, notre algorithme utilise une phase d'élitisme: tirant des manières en fonction de leur longueur dans un ordre croissant, puis nous gardons 10% de la population actuelle de les copier dans la nouvelle génération (Figure.IV. 7, section IV.4). Cette opération est utilisée pour définir la probabilité d'un individu i de sélection dans une population, si le fi de remise en forme d'un individu i appartiennent à la population et T est le nombre d'individus dans la population (l'équation (6)), sa probabilité d'être sélectionné est:

$$\frac{fi}{\sum_{i=1}^{T} fi} \qquad (6)$$

IV.4.5. Croisement (Cross-over)

Initialement, le croisement associé au codage par chaînes de bits est le croisement à découpage de chromosomes (*slicing crossover*). Pour effectuer ce type de croisement sur des chromosomes constitués de M gènes, on tire aléatoirement une position dans chacun des parents. Nous avons échangé ensuite les deux sous-chaînes terminales de chacun des deux chromosomes, ce qui produit deux enfants C1 et C2 (Figure II.8, section II.5.3.1.).

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Tout d'abord, les points de rencontre des deux chemins sont déterminés; si les deux chemins ne se rencontrent pas, le fils généré sera identique au père ; sinon nous recherchons le point de rencontre qui optimise le trajet : chemin identique à celui du père de départ jusqu'au point de rencontre, puis chemin identique à celui de la mère du point de rencontre à la fin du chemin maternel. Cela permet de conserver de façon sûre les meilleurs parents dans la génération suivante.

Un algorithme génétique utilise habituellement (opérateur de croisement) pc fixe (0,5-0,9), (l'équation (7)). Le but de la traversée est d'enrichir la diversité de la population par la manipulation de la structure des chromosomes (chemins). Tout d'abord, les points des deux chemins de réunion sont déterminés, si les deux chemins ne répondent pas, le fils générée sera identique au père; sinon, nous cherchons le point de rencontre qui optimise le chemin: chemin identique à celui de son père jusqu'au point de rencontre, chemin alors identique à celle de la mère du point de réunion à la fin du chemin maternel. En outre, nous obtenons la reproduction d'un individu par lui-même, le fils étant alors identique au père. Cela permet de préserver certainement les meilleurs parents dans la génération suivante (Figure.IV.7, section IV.4).

IV.4.6. Mutation

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. Ainsi en toute rigueur, l'algorithme génétique peut converger sans croisement

Un algorithme génétique utilise habituellement un opérateur de mutation fixe (0,001 à 0,2) (l'équation (8)). Pour simuler la présence de mutations, juste après la reproduction, l'individu généré subit aléatoirement des mutations. La probabilité de mutation est de [0.1%, 2%] (ce taux faible est justifié par la probabilité élevée de tomber sur des faux chemins ou des obstacles en mutant les cases c'est-à-dire la probabilité d'avoir un chemin rompu est de 75%, et 25 % pour un chemin complet), chacune des mutations modifie le chemin concerné en lui additionnant un nombre aléatoire, puis en prenant le résultat de la congruence modulo 4 qui représente les 4 directions (Figure.IV.7, section IV.4). Habituellement, l'augmentation des valeurs de la probabilité de croisement (pc) et la probabilité de mutation (pm), améliore les performances de l'algorithme génétique. Nous pouvons contrôler la probabilité de cross-over et de mutation opérateurs dynamiquement en fonction de la situation actuelle. Les opérateurs auto-adaptatifs sont:

$$pc = \frac{(fm - f')}{(fm - f")}, \quad f' \geq f" \qquad (7)$$

$$pm = \frac{(fm - f)}{(fm - f")}, \quad f \geq f" \qquad (8)$$

Où *f* "est l'aptitude moyenne de la population actuelle, *fm* est la condition physique maximale de la population actuelle. *f'* indique la valeur de remise en forme entre deux individus à traverser. Le procédé ci-dessus est répété jusqu'à ce qu'une condition d'arrêt soit atteinte.

IV.5. Algorithme d'évitement

Le principe de l'algorithme d'évitement des obstacles que nous mettons en œuvre joue le rôle d'un gestionnaire d'accès aux boîtes, il utilise les sémaphores de type mutex (un seul robot en même temps dans une boîte).

Dans le cas d'inter-blocage entre 2 robots où le robot « 1 » demande la case du robot « 2 » et inversement, nous avons choisi un robot de façon aléatoire et on considère l'autre comme un obstacle puis le robot choisi fait un appel à l'algorithme génétique et recalcule le chemin en prenant en compte les changements de l'environnement, puis le deuxième robot (considéré comme obstacle) reprend son chemin (Figure.IV.5, section IV.3.4.). Un avantage majeur de cet algorithme vient du fait qu'il parvient aux obstacles dynamiques. En effet, si un obstacle, une personne par exemple, est présent pendant un court instant à proximité du robot, il sera ajouté à la grille et il poursuivra son exécution si le chemin est clair. Ainsi, il est nécessaire d'ajouter un mécanisme de mémorisation de l'environnement. Pour cela, à chaque mise-à-jour de l'environnement, nous avons décidé de multiplier l'ensemble des cases parcourues de la matrice par une constante égale à 0.

IV.6. Conclusion

Dans ce chapitre, nous avons décrit notre approche de la navigation coopérative et autonome de robots mobiles dans un chantier de construction, dans le but de construire le plan de la navigation dans un environnement avec des obstacles. Les robots sont capables de passer d'une position initiale à la position finale sans collision. Le chemin choisi est optimal. La performance de l'algorithme génétique proposée est testée sur des environnements avec une complexité croissante. Les résultats obtenus montrent l'intérêt de notre algorithme génétique, ces algorithmes peuvent trouver le chemin optimal dans un temps très court et a la capacité d'enrichir l'espace de configuration en ensemble différent de mouvements admissibles à l'aide de la sélection, les opérateurs de croisement et la mutation. Cette technique peut être utilisée pour permettre le mouvement de plusieurs robots ensembles dans un lieu commun.

En clair, comme nous venons de le voir, ce chapitre était consacré, à la conception du notre système qui nous ont permis de dégager l'architecture générale de notre application qui nous renseigne sur la relation homme machine de notre application qui sera détaillée dans le prochain chapitre.

Chapitre V :

Réalisation et Implémentation

Chapitre V : Réalisation et Implémentation

V.1. Introduction

L'exploration et l'exploitation collectives d'un environnement inconnu par un ensemble de robots font partie des applications les plus populaires en Intelligence Artificielle Distribuée. De nombreux modèles ont été développés pour trouver des règles simples permettant à des robots d'exploiter leur environnement pour des tâches complexes.

Dans ce chapitre, nous présenterons la réalisation et l'implémentation de notre système, en nous basant sur l'architecture fonctionnelle de l'application, ainsi que sur les outils de développement. Les résultats obtenus seront discutés à la fin du chapitre.

V.2. Les outils de développement

Notre outils a été développé avec le langage C#, qui est un langage de programmation créé par Microsoft en 2002. Nous avons utilisé la classe Semaphore pour contrôler l'accès à des ressources (les cases de l'environnement). Les threads entrent dans le sémaphore en appelant la WaitOne(méthode P), qui est héritée de la classe WaitHandle et libère le sémaphore en appelant la Release (méthode V) [WWW4].

Le compteur d'un sémaphore est décrémenté chaque fois qu'un thread entre dans le sémaphore et incrémenté lorsqu'un thread libère le sémaphore. Lorsque le compteur des threads atteint zéro, les demandes suivantes sont bloquées jusqu'à ce que les autres threads libèrent le sémaphore. Lorsque tous les threads ont libéré le sémaphore, le compteur est à la valeur maximale spécifié lors de la création du sémaphore. Il n'existe aucun ordre garanti, dans lequel les threads bloqués entrent dans le sémaphore.

Espace de noms: System.Threading (dans System.dll).

Hiérarchie d'héritage*:* System.Object→ System.MarshalByRefObject→ System.Threading.Wait Handle→System.Threading.Semaphore.

```
[ComVisibleAttribute(false)]
        [HostProtectionAttribute(SecurityAction.LinkDemand, Synchronization = true,
        ExternalThreading = true)]
        public sealed class Semaphore : WaitHandle
```

V.3. L'architecture fonctionnelle de l'application

Nous nous concentrons sur la situation suivante :

A l'état initial, l'environnement est représenté par une matrice *t* de dimension m*n où le contenu d'une case t[i][j] est soit 0 soit 1. L'ensemble des robots doit atteindre un point d'arrivée à partir d'un point de départ, chaque robot a une position de départ (X_i,Y_i) et une position d'arrivée (X_i',Y_i') où (i) est le rang du robot (i). Les robots ne connaissent pas l'environnement ni la position des obstacles, et doivent se synchroniser (éviter les collisions).

Nous nous sommes intéressés à résoudre les problèmes suivants :

Trouver la trajectoire optimale (le plus court chemin), en évitant les obstacles et la collision inter robots (Figure V.1).

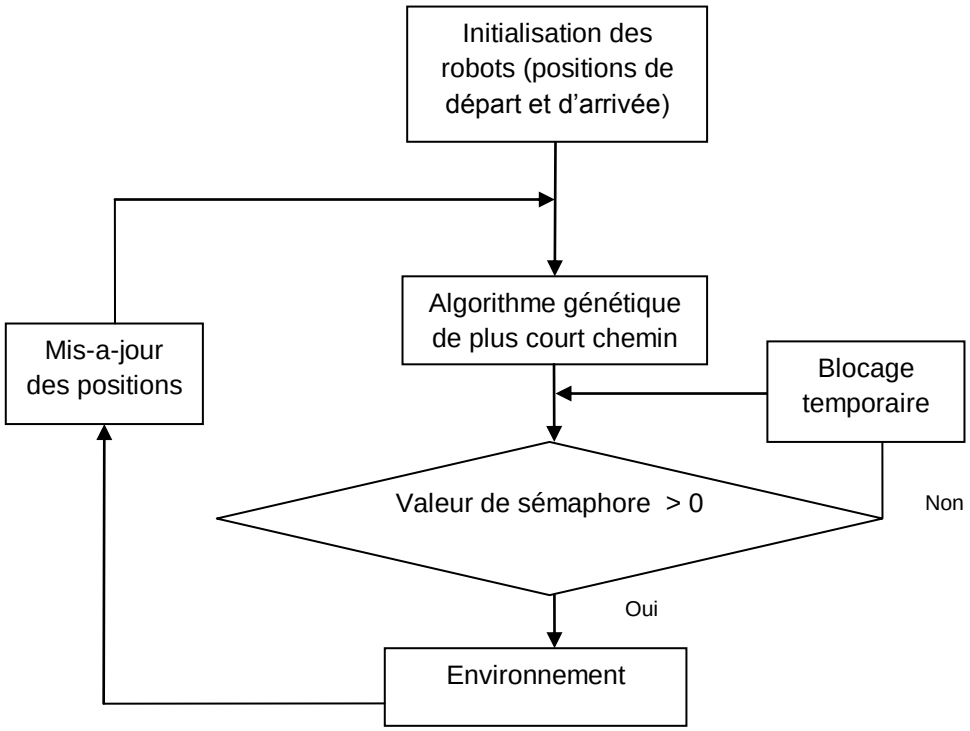


Figure V. 1 : Représentation général du système

1- Le robot se déplace à partir du point de départ dans les 4 directions au hasard.
2- Le robot détecte les obstacles et enregistre les détails de l'environnement.
3- Il trouve un chemin qui relie le point de départ au point d'arrivée.
4- Il rprendre « 1 », « 2 » et « 3 » pour construire la population initiale.

5- Sélectionne les meilleurs chemins.
6- Trouve le chemin le plus court.
7- Attend la fin de calcul pour les autres robots.
8- Exécution parallèle pour les robots.
9- Traitement des cas particuliers (inter-blocage).

V.4. Description des différents modules

V.4.1. Architecture d'un agent

Dans la méthode que nous proposons, un agent doit être capable de raisonner sur la tâche qu'il doit accomplir et doit être capable de synchroniser avec les autres agents.

Un agent est initialisé avec un ensemble de données complémentaires. Ces données permettent d'élaborer sa base, dans laquelle il faut choisir un ensemble de chemins (pour la sélection) (Figure V.2).

Les Agents « C », « S », « E », « DO », « DM », « O », « M » sont des threads. Seuls Les agents «M » et « O » sont des robots.

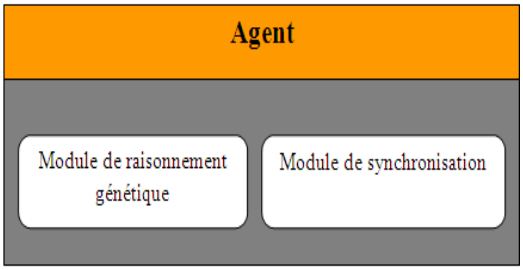


Figure V.2 : Architecture d'un agent

Le module de synchronisation permet de synchroniser entre les agents et joue le rôle d'un gestionnaire d'accès aux cases des robots, il utilise comme outil les sémaphores de type mutex (un seul agent à la fois dans une case).

Code c#:

```
List<List<System.Threading.Semaphore>> semas;
```

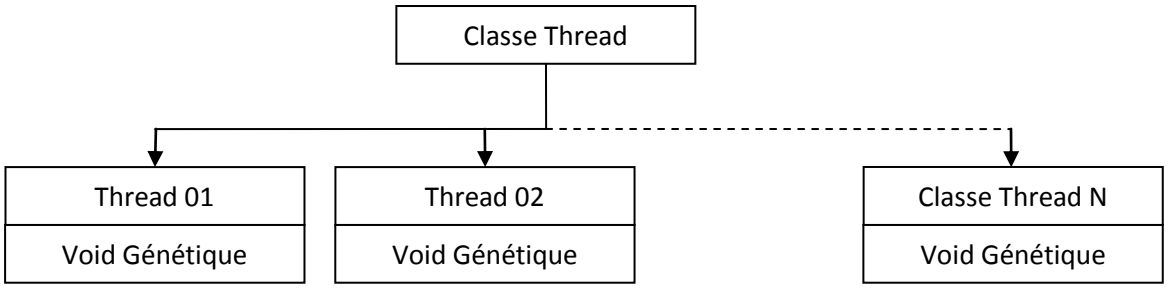


Figure V. 3 : Architecture logiciel des agents O

Le concept de thread a pour objet de permettre à plusieurs threads de partager un jeu de ressources pour travailler ensemble afin d'accomplir une tâche donnée.

Un thread est un processus léger. Les processus servent à regrouper les ressources, les threads sont les entités planifiées pour leur exécution par le processeur.

Le terme multithreading est employé pour décrire la situation dans laquelle plusieurs threads sont présents dans le même processus (Figure V.3).

Pour l'implémentation d'un algorithme génétique du plus court chemin, nous avons été amenés à choisir une modélisation de nos chemins, à laquelle nous pourrions appliquer ces principes généraux. Pour cela, nous avons modélisé un chemin sous la forme suivante :

chemin={Xi^1,Xi^2,Xi^3…….,Xi^n},

où ***i*** c'est le numéro du robot,

n est la longueur du chemin et X un entier ϵ {0,1,2,3}.

Code C#:

```
public  class chromosom{
List<proj_Point> Cases;
int longueur;
        };
```

Avec la modèlisation du case :

```
public  class proj_Point
{
Point point;
int cout;
};
```

V.4.2. Les chemins

Le chemin est une suite de cases où chaque case contient un couple representant les cordonnées de la case (l'emplacement dans l'environnement modélisé par la matrice *t*) de type *Point* et un entier compris entre 0 et 3, qui indique un déplacement du robot. Le tableau constitue une suite de déplacements qu'effectuera le robot mobile en partant du point de départ, en espérant atteindre le point d'arrivé. La longueur du chemin est un indicateur de performance de l'individu (la fitness), c'est-à-dire le nombre de déplacements qu'il doit effectuer pour rejoindre l'arrivée (Figure V.4).

Exemple : chemin= {0013333222211} Où la longueur du chemin n=13.

Le cout de déplacement	0	0	1	3	3	3	3	2	2	2	2	1	1
La direction	→	→	↓	↑	↑	↑	↑	←	←	←	←	↓	↓

Figure V.4 : Schéma représentant un chemin trouvé entre 2 points

V.4.3. L'environnement

L'espace de navigation a besoin, sous une forme ou une autre, d'une représentation informatique. Il existe deux principaux types de cartes : les grilles d'occupation et les graphes. Dans le premier cas, l'environnement est quadrillé en cases. Dans l'état initial la matrice représentant l'environnement avec des obstacles placés aléatoirement, les cases noires qui sont des obstacles portent la valeur 1 et les cases blanches qui sont des cases libres, portent la valeur 0 (Figure V.5). Chaque case (section critique pour les robots mobiles) contient un sémaphore pour organiser les accès concurrents des robots mobiles.

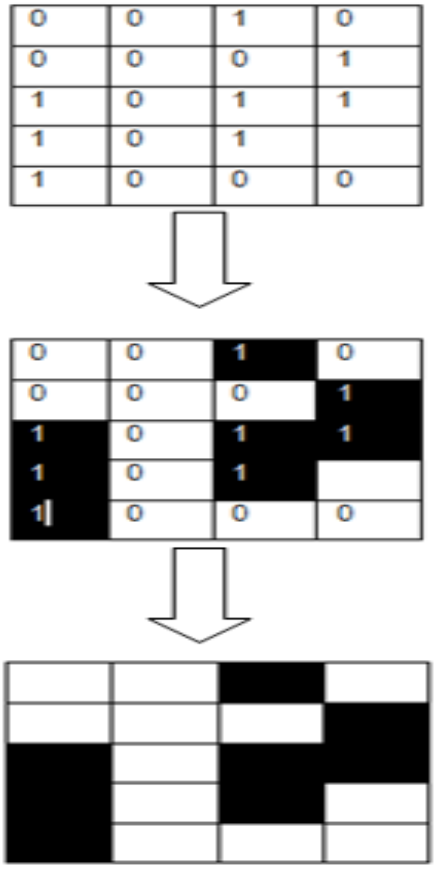


Figure V.5 : Schéma représentant le passage d'une matrice booléenne vers un environnement de navigation.

Code C#:

```
List<List<bool>> matrix;
```

V.4.4. L'algorithme génétique proposé

V.4.4.1. Population initiale

Ce mécanisme doit être capable de produire une population d'individus non homogènes qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre la convergence plus ou moins rapide vers *l'optimum global*. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.

La population est générée aléatoirement et correspond aux chemins possibles pour atteindre le point d'arrivée. Avoir une solution acceptable évoque d'envisager 100 chemins possibles afin d'aboutir à la destination.

Pour résoudre le problème de convergence prématurée (minimum local) l'algorithme génère un ensemble de solutions (chemins), pour chaque génération.

La population initiale est représentée par une *liste de listes* pour plus de flexibilité car la longueur du chemin trouvé n'est pas toujours constante (Figure V.6).

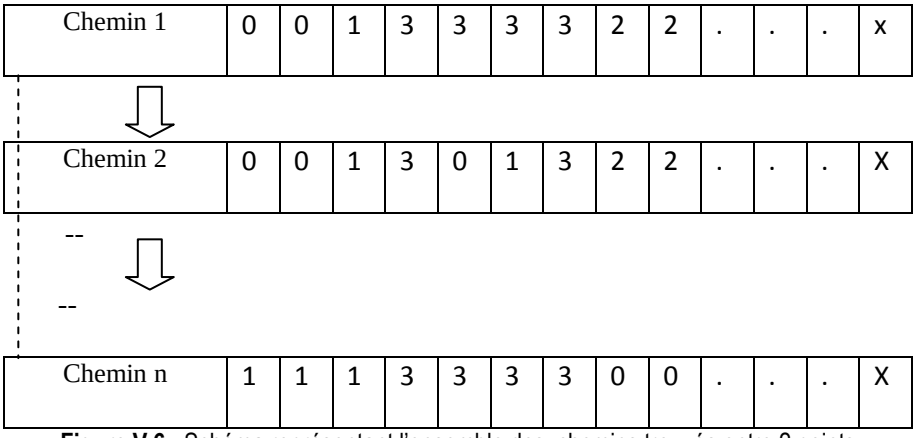


Figure V.6 : Schéma représentant l'ensemble des chemins trouvés entre 2 points

Où **n** est le nombre des chemins de chaque génération et **x** est une variable qui désigne la longueur du chemin.

Code C#:

```
List<List<Point>> chemins;
```

V.4.4.2. Fitness

Celle-ci prend des valeurs entières positives et est appelée fitness ou fonction d'évaluation de l'individu.

V.4.4.3. L'opérateur de sélection

Cet opérateur est chargé de définir quels seront les individus (les chemins) de la population P qui vont être dupliqués dans la nouvelle population P' et qui vont servir de parents (application de l'opérateur de croisement).

Soit n le nombre d'individus de la population P, on doit en sélectionner n/3 (l'opérateur de croisement nous permet de repasser à n individus).

- ***La méthode élitiste***

Cette méthode consiste à sélectionner les n individus dont on a besoin pour la nouvelle génération en prenant les n meilleurs individus de la population P après l'avoir triée de manière croissante selon la fitness de ses individus (longueur des chemins); en effet, la pression de la sélection est trop forte, la variance nulle et la diversité inexistante, du moins le peu de diversité qu'il pourrait y avoir ne résultera pas de la sélection mais plutôt du croisement et des mutations.

V.4.4.4. Les operateurs de croisement et de mutation

Ils permettent de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'état.

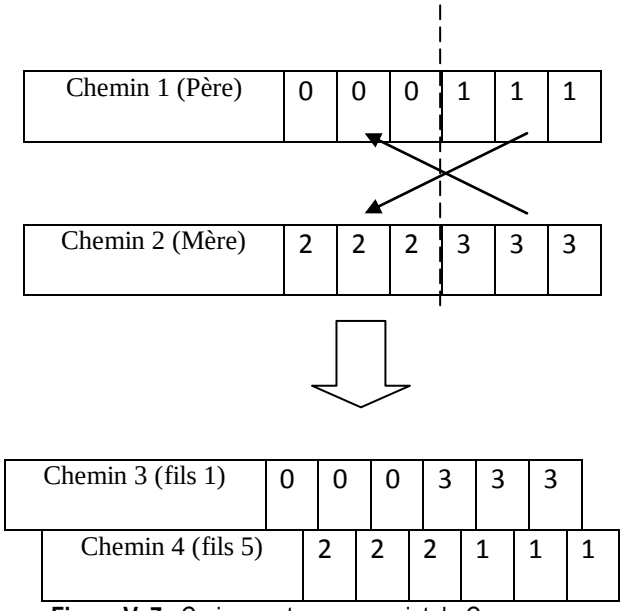


Figure V. 7 : Croisement avec un point de Cross-over

Les croisements sont envisagés avec deux parents et génèrent deux enfants. Cela consiste à échanger les gènes des parents afin de donner des enfants qui portent des propriétés combinées. Bien qu'il soit aléatoire (Figure V.7), cet échange d'informations offre aux algorithmes génétiques une part de leur puissance : quelque fois, de " bons " gènes d'un parent viennent remplacer les " mauvais " gènes d'un autre et créent des fils mieux adaptés aux parents.

Au départ, les points de rencontre des deux chemins sont déterminés.

- Si les deux chemins ne se rencontrent pas, le fils généré sera identique au père.
- Sinon, on recherche le point de rencontre qui optimise le trajet : chemin identique à celui du père du départ jusqu'au point de rencontre, puis chemin identique à celui de la mère du point de rencontre à la fin du chemin maternel.

On définit de plus la reproduction d'un individu avec lui-même, le fils étant alors identique au père. Cela permet de conserver de façon sûre les meilleurs parents dans la génération suivante.

L'aléatoire jouant un grand rôle, on peut constater que si dans la majorité des cas, la convergence vers une solution idéale se fait de façon très rapide (moins de 100 générations), il existe des cas où cette convergence est beaucoup plus lente, même si le parcours à effectuer est en réalité assez court (par exemple du dernier cas du tableau ci-dessus). Il existe des cas où 500 générations ne suffisent pas à obtenir une solution optimale.

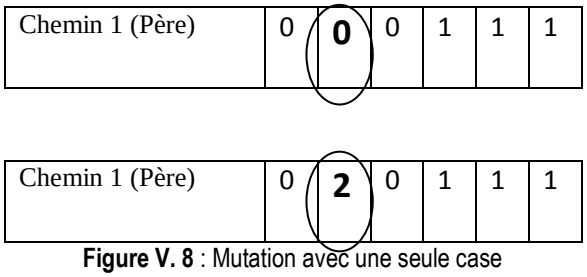


Figure V. 8 : Mutation avec une seule case

La probabilité de mutation est de 1% (ce taux faible est justifié par la probabilité élevée de tomber sur des faux chemins ou des obstacles en mutant les cases, c'est-à-dire, la probabilité d'avoir un chemin rompu est de 75%, et 25 % pour un chemin complet), chacune des mutations modifie le chemin concerné en lui additionnant un nombre aléatoire, puis en prenant le résultat de la congruence modulo 4 qui représente les 4 directions (Figure V.8).

Taux d'élitisme	Te=0.1
Taux de génération des nouveaux chemins	Tg=0.1
Taux de croissement	Tc=[0.5- 0.9]
Taux de mutation	Tm=[0.001-0.2]
Taille de la population	100
Longueur maximale du chromosome	100
Nombre de générations	500

Tableau V. 1 : Les paramètres fixés de notre algorithme génétique

V.4.4.5. Les contraintes de notre algorithme génétique

L'algorithme que nous avons élaboré comporte certaines contraintes :

1. Le nombre de cases que doit parcourir le robot.
2. La taille de la population utilisée.
3. L'opérateur de sélection.
4. L'élitisme (conserve le meilleur individu à chaque génération).
5. Le mode de croisement.
6. Le démarrage des déplacements des robots.
7. Avance pas à pas.
8. La Génération courante.
9. L'Ecart-type courant.
10. La longueur du meilleur parcours.

L'histogramme représenté par la Figure V. 9 montre la variation de la fitness en fonction de l'évolution des générations ; on remarque le tracé descendant qui traduit la minimisation de la fonction de fitness, Les avantages de notre approche sont de deux ordres :

- On peut créer un nombre illimité de robots ce qui rend cette architecture dynamique et extensible
- Le multithreading réduit le temps d'exécution et augmente les performances de notre système.

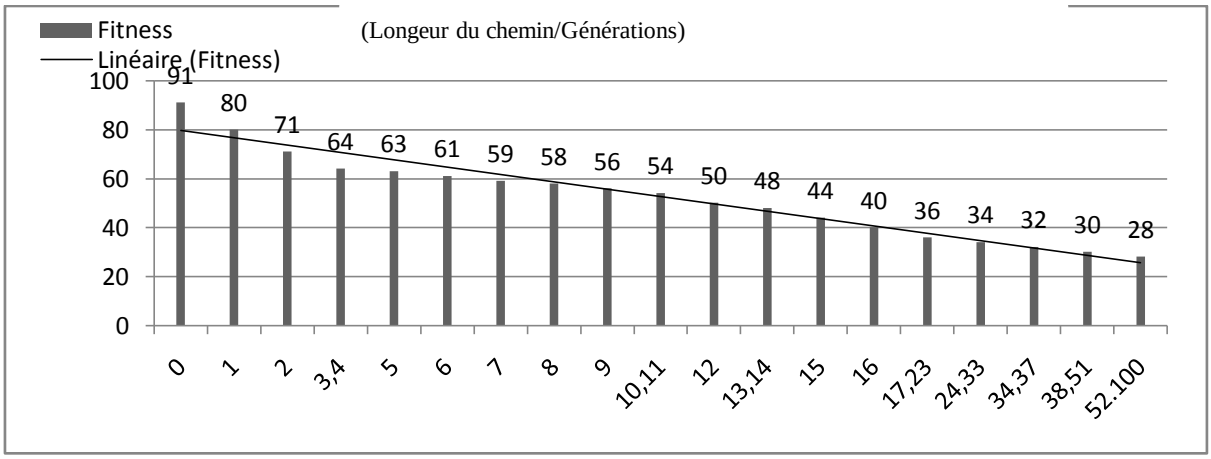


Figure V. 9 : Histogramme représentant la variation du fitness.

V.4.4.6. Cas d'utilisation

Une fois l'environnement établi, les agents prennent leurs positions, en d'autre terme, chaque robot représente un thread et possède un point de départ et un point d'arrivée, ces points peuvent être déterminé de façon aléatoire ou par l'utilisateur.

Au début, le système fournit un ensemble aléatoire d'obstacles ; l'utilisateur peut les ajouter ou les supprimer selon ses besoins, une fois l'environnement configuré, l'utilisateur peut lancer alors l'exécution (Figure V. 10).

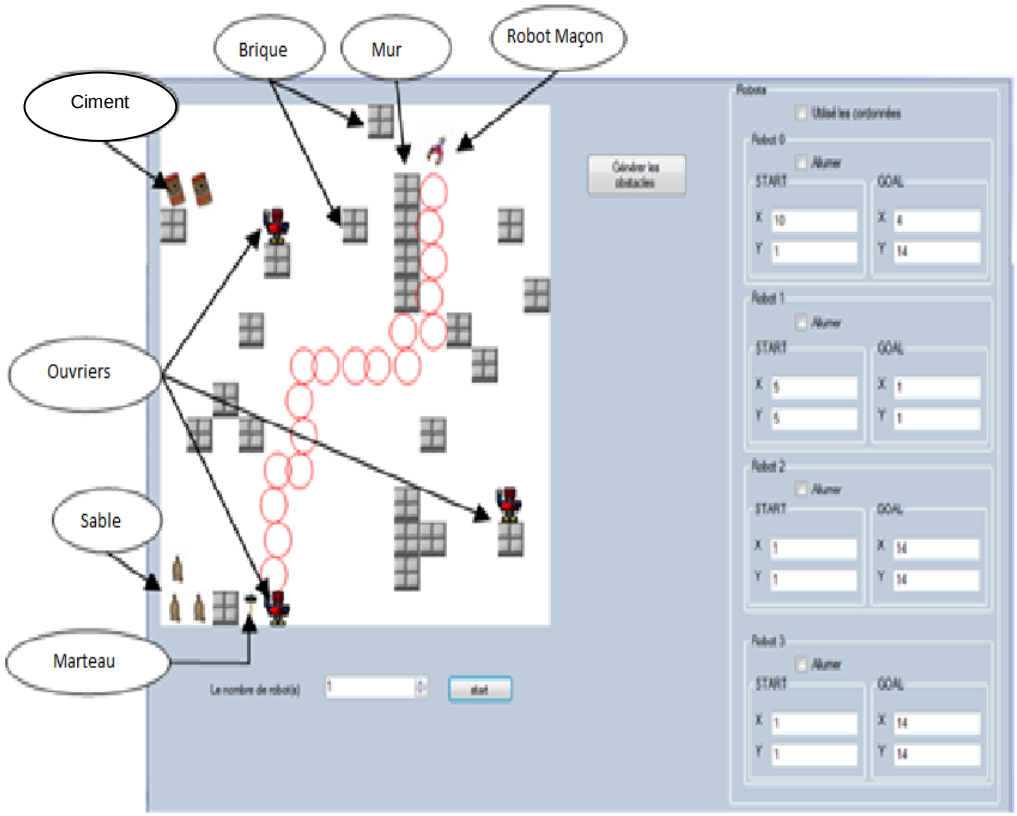


Figure V. 10 : Interface logicielle

Une fois l'exécution lancée par l'utilisateur, les agents doivent se synchroniser pour accomplir une tâche de construction, cette dernière est décomposée en sous tâches, chaque agent essaye de réaliser sa mission et d'atteindre un point d'arrivé en utilisant l'algorithme génétique du plus court chemin, en évitant la collision (Figure V. 10).

Les chemins de la première génération sont créés de façon aléatoire. La fonction du fitness est mesurée par le calcul de distance entre le point de départ et le point d'arrivé (un chemin), les chemins avec les distances les plus courtes sont sélectionnés pour le croisement et la mutation. Le processus est répété jusqu'à l'obtention des résultats optimaux. Les résultats obtenus sont représentés par les tableaux suivants :

Expérience	*Taille de la population*	*Meilleurs fitness*	*Nombre de génération*
1	*10*	*21*	*50*
2	*50*	*19*	*36*
3	*100*	*18*	*6*

Tableau V. 2 : Environnement 10*10 sans présence d'obstacles

Expérience	*Taille de la population*	*Meilleurs fitness*	*Nombre de génération*
1	*10*	*152*	*95*
2	*50*	*144*	*44*
3	*100*	*122*	*12*

Tableau V. 3: Environnement 100*100 sans présence d'obstacles

Expérience	*Taille de la population*	*Meilleurs fitness*	*Nombre de génération*
1	*10*	*28*	*95*
2	*50*	*21*	*72*
3	*100*	*19*	*32*

Tableau V. 4: Environnement avec présence d'obstacles (complexité moyenne)

Expérience	*Taille de la population*	*Meilleurs fitness*	*Nombre de génération*
1	*1000*	*27*	*97*
2	*1500*	*25*	*85*
3	*2000*	*24*	*60*

Tableau V. 5: Environnement avec présence d'obstacles (complexité élevée)

Les deux tableaux (Tableau V. 2 et Tableau V. 3) représentant les résultats sont obtenus après un nombre élevé de générations. Dans les deux premières expériences, en augmentant la taille de la population nous avons obtenus de meilleurs résultats qu'un nombre de générations réduit. Avec la

présence d'obstacles (Tableau V. 4 et Tableau V. 5) le nombre de générations est plus élevé. Donc, pour obtenir de meilleures valeurs de fitness, il faut augmenter le nombre de la population de départ.

Cependant deux problèmes se posent :

- La taille de la population et des générations (itérations) est élevée, ce qui implique une croissance de la complexité.

- Avec l'augmentation de la taille de la population du démarrage en obtient les meilleurs résultats. Cependant, personne ne peut assurer qu'une taille précise de la population ne va pas améliorer les résultats de l'optimisation, c'est-à-dire que les résultats obtenus sont optimaux par rapport à la population de départ, mais on ne sait pas s'ils sont les plus optimaux.

Après les nombreux tests effectués et quelque soit l'état initial des robots de notre environnement, notre approche multi-agents basée sur les algorithmes génétiques a montré des performances satisfaisantes et d'une fiabilité acceptable pour répondre aux requêtes de l'utilisateur.

Généralement les processus de résolution des problèmes du monde réel exigent :

- un minimum temps d'exécution,
- une meilleure qualité de production
- et une haute précision.

L'approche multi-agents basée sur la décomposition de tâches que nous avons utilisé répond à ces besoins.

Les avantages de cette modélisation sont de deux ordres :

- Nous pouvons créer autant d'agents qu'on veut, ce qui rend cette architecture dynamique et extensible.
- Le multithreading réduit énormément le temps d'exécution et augmente les performances de notre système.

En conclusion, nous pouvons créer un monde virtuel (environnement) et implémenter des robots autonomes dans le but de naviguer de façon optimale et synchrone.

V.5. Résultats et Discussion

Pour tester et évaluer les performances de notre approche, nous l'avons comparé avec d'autres algorithmes dans les mêmes conditions: nous avons développé un programme en C # générant n obstacles. Les mouvements des robots ont été contrôlés en utilisant trois (03) algorithmes :

1- L'algorithme génétique est le premier utilisé: un espace sous forme d'une matrice de dimension [m*n] sur laquelle les robots doivent naviguer. Nous avons fait déplacé le robot de la position de départ (0, 0) à la position d'arrivée (m,n). 70 obstacles au total ont été utilisés (pour les 3 algorithmes). Les robots se déplacent d'un pas à chaque mouvement dans une unité de temps.

2- Pour tester l'algorithme réseaux de neurones artificiels, une simulation a été réalisée sous MATLAB. Le temps de seuil a été fixé à 1 seconde. Nous avons fait déplacé le robot de la position de départ (0, 0) à la position d'arrivée (m,n) [PIE02].
3- Le troisième algorithme utilisé pour le problème était A* dans les mêmes conditions: les robots mobiles naviguent dans une carte en 2D avec un degré de liberté de deux degrés, les obstacles placés aléatoirement.

Le tableau suivant présente une étude comparative des algorithmes pour calculer le chemin le plus court (Tableau V. 6).

Nous avons développé un outil logiciel avec le langage C # pour permettre de prendre les résultats de l'algorithme du plus court chemin (*Ri*) (l'équation (9)) :

$$Pi(\%) = \left(\frac{Ri}{Rt}\right)100 \qquad (9)$$

	Environnent	Départ	Arrivé	Obstacles	Distance théorique	**Longeur du chemin**	Temps (ms)	**Taux de convergence**
AG	[15X 15]	0, 0	14, 14	70	28	**28**	141.02	**100%**
RN		0, 0	14, 14	70	28	**31**	298,41	**91%**
A*		0, 0	14, 14	70	28	**28**	102,12	**100%**

Tableau V. 6.Comparaison des résultats de l'algorithme génétique proposé (AG), réseau de neurones (RN) [PIE02] et l'algorithme A* [BEN15].

Nous avons observé le fonctionnement des algorithmes utilisés guider les robots en veillant à ce qu'ils arrivent à la destination finale sans collisions:

Les algorithmes génétiques peuvent être une bonne solution pour résoudre le problème de la navigation autonome de robots mobiles. Néanmoins, leur utilisation doit être conditionnée par certaines caractéristiques:

- Le temps de calcul de la fonction d'évaluation (fitness) doit être raisonnablement court. En effet, celle-ci sera évaluée plusieurs fois (voir section IV.4.3).
- Le nombre de chemins important : les performances des algorithmes génétiques par rapport aux algorithmes classiques sont plus marquées lorsque les espaces de recherches sont importants. En effet, pour un espace dont la taille est faible, il est plus sûr de parcourir cet espace de manière exhaustive afin d'obtenir la solution optimale en un temps qui restera somme toute correct. Par contre, utiliser un algorithme génétique engendrera le risque d'obtenir une solution non optimale en un temps qui restera sensiblement identique.
- Pas d'algorithme déterministe adapté et raisonnable.
- Lorsque l'on préfère avoir une solution relativement bonne rapidement plutôt qu'avoir la solution optimale en une durée indéfinie.

Mais cette approche a aussi ses limites :

- Le temps de calcul : par rapport à d'autres méta-heuristiques, les algorithmes génétiques nécessitent de nombreux calculs, en particulier au niveau de la fonction d'évaluation.

- Ils sont le plus souvent difficiles à mettre en œuvre : des paramètres comme la taille de la population ou le taux de mutation sont parfois difficiles à déterminer. Or le succès de l'évolution en dépend et plusieurs essais sont donc nécessaires, ce qui limite encore l'efficacité de l'algorithme. En outre, choisir une bonne fonction d'évaluation est aussi critique. Celle-ci doit prendre en compte les bons paramètres du problème. Elle doit donc être choisie avec soin.

- Il faut aussi noter l'impossibilité d'être assuré, même après un nombre important de générations, que la solution trouvée soit la meilleure. On peut seulement être sûr que l'on s'est approché de la solution optimale (pour les paramètres et la fonction d'évaluation choisie), sans la certitude de l'avoir atteinte.

Un autre problème important est celui des optima locaux. En effet, lorsqu'une population évolue, il se peut que certains individus qui à un instant occupent une place importante au sein de cette population deviennent majoritaires. À ce moment, il se peut que la population converge vers cet individu et s'écarte ainsi d'individus plus intéressants mais trop éloignés de l'individu vers lequel on converge. Pour vaincre ce problème, il existe différentes méthodes comme l'ajout de quelques individus générés aléatoirement à chaque génération (Voir section V.4.4.3).

Nous avons utilisé l'algorithme génétique du plus court chemin et nous avons observé que l'algorithme prend beaucoup de temps pour la production des solutions initiales, mais donne de bons résultats. Ces résultats évoluent de mieux en mieux à chaque itération. Le résultat final est que l'algorithme est capable de guider le robot du point de départ jusqu'à un point d'arrivé sans collision.

Nous avons également essayé d'utiliser les réseaux de neurones artificiels (RNA). Mais cette approche comporte aussi des limites.

- D'abord sa lenteur : on est vite confronté à ce que l'on appelle l'explosion combinatoire.
- La faiblesse est aussi théorique. La connaissance ne se réduit pas au langage.

En analysant le fonctionnement de l'algorithme A*, nous avons remarqué son efficacité dans l'optimisation de la trajectoire du robot à partir du point de départ a celui d'arrivée, en veillant à l'absence de collisions. Cet algorithme a travaillé de manière très efficace, même dans les moments très chaotiques, pour générer de bonnes solutions dans un environnement limité pour des missions déterminées, mais si les points de départ et d'arrivée sont éloignés (très grande carte), la complexité de l'algorithme augmente.

L'analyse comparative peut être facilement faite en observant les différentes pistes de différents algorithmes. Nous avons remarqué que l'algorithme A* était beaucoup plus efficace et dépasse ses homologues. Il est préférable de trouver les résultats dans un temps minimum (grille limitée). Même si nous savons que la performance des différents algorithmes changera en fonction de la taille de la carte.

L'algorithme réseaux de neurones artificiels échoue dans le cas d'un environnement très complexe tel qu'un labyrinthe. Les besoins de mémoire pour les algorithmes génétiques sont très élevés par rapport aux réseaux de neurones Artificiels et l'algorithme A*. Ce dernier lorsqu'il est appliqué à des entrées de faible taille, fonctionne mieux en terme de temps (vitesse). Le problème survient quand la taille de la carte est trop grande et dépasse les limites.

Le robot est capable de se déplacer de la première position à la position finale dans un environnement dynamique sans collision avec les autres robots ou avec des obstacles ; le chemin choisi étant optimal. Cet algorithme peut être utilisé pour permettre le mouvement de plusieurs robots dans un lieu commun chantier de construction par exemple.

V.6. Conclusion

Dans ce chapitre, nous avons mis au point un processus de décision collective à partir d'un modèle biologique d'optimisation (les algorithmes génétiques). Nous avons montré que ces algorithmes associés à une préférence naturelle (caractéristiques de l'environnement) peuvent conduire un groupe de robots, aux capacités sensorielles et cognitives limitées, à réaliser un choix collectif pour une navigation optimale dans un environnement avec des obstacles. L'aspect le plus intéressant de ces résultats est qu'aucun des robots utilisés n'est capable, individuellement, de telles performances, à cause notamment de leur faible appareillage sensoriel et de leur capacité de traitement limitée.

Conclusion Générale

Conclusion Générale

Notre travail traite la problématique de la navigation autonome des robots mobiles dans des environnements dynamiques et incertains. Ce cadre est défini par les spécificités des applications envisagées, que nous pouvons résumer par la navigation pour robots mobiles à proximité des obstacles.

Notre contribution porte sur le développement de nouvelles méthodes inspirées de l'intelligence artificielle pour assurer une navigation optimisée en évitant les obstacles en toute sécurité quel que soit l'environnement où doit évoluer les robots.

L'objet de notre thèse porte sur les potentiels donnés par les technologies d'automatisation pour développer des procédés de construction durables pour les bâtiments. Les aspects économiques, environnementaux et sociaux de tels processus sont appréciés. Les principaux facteurs qui poussent à un changement dans le processus de construction sont la réduction du nombre de travailleurs de la construction, l'augmentation du coût et de la durée limitée des projets, la pression sur l'industrie de la construction et de la consommation de matériaux de haute dans le secteur de la construction.

Le travail consiste à étudier et appliquer les algorithmes génétiques pour la navigation autonome de plusieurs robots mobiles dans un chantier de construction, afin de permettre aux robots de se mouvoir d'une position initiale à une autre finale en évitant les obstacles et les collisions entre robots.

Nous avons développé une approche de contrôle comportemental, basée sur les algorithmes génétiques, permettant à chaque étape de navigation de sélectionner le comportement adéquat à exécuter par le robot mobile afin d'atteindre la position finale. Cette approche de contrôle a été testée avec succès sur différentes configurations de la cible et des robots ainsi que sur plusieurs environnements distincts. Les robots arrivent toujours à atteindre la cible en évitant les obstacles.

- Nous avons proposé un système ou les robots mobiles doivent coopérer pour l'organisation d'opérations dans un chantier de construction. Dans ce cadre, nous nous sommes intéressés à la problématique de coopération automatique dans un système multi-robots.
- Nous avons développé une méthode d'évitement d'obstacles pour des systèmes robotiques. Cette méthode est inspirée de la gestion des processus dans les systèmes d'exploitation. Nous utilisons comme outil les sémaphores de type mutex (un seul agent à la fois dans une case). Ce dernier prouve la performance, la régulation dans l'évitement d'obstacles.
- Lors de la conception d'un planificateur génétique, le temps de traitement est le facteur le plus important. Pour cela, nous avons opté pour une fonction de fitness simple. Ainsi, les stratégies (de sélection, de croisement et de mutation) choisies ne nécessitent pas beaucoup de calculs.

- Les algorithmes génétiques souffrent de leur incapacité de traiter l'imprécis. Après l'étude et l'observation du fonctionnement de notre algorithme, nous avons constaté qu'il était possible de le rendre souple et ce, en prenant en compte les distances entre les points d'entrée et les points de sortie. Cette approche, appliquée à des cas pratiques, a montré une grande souplesse tout en minimisant le temps de traitement sans pour autant nécessiter de ressources matérielles spéciales. Comme nous l'avons montré dans le cinquième chapitre, la méthode que nous avons proposée simplifie toutes les étapes du traitement génétique, et donne des résultats très intéressants. Il reste à démontrer la validité de cette méthode sur d'autres exemples de processus non linéaires plus compliqués, et dans d'autres applications où les algorithmes génétiques connaissent un grand succès. L'étude et l'analyse de la stabilité d'un tel planificateur seront d'une importance primordiale.

- En robotique mobile, la poursuite de la trajectoire qui est une tâche élémentaire, représente la base de toute autre tâche du robot. Dans la stratégie que nous avons utilisée, la trajectoire à suivre est représentée par un certain nombre de points de passage entre le point de départ et le point d'arrivée. Le robot exécute, point par point, sa trajectoire, et corrige son orientation en fonction de sa position par rapport au point désiré.

- Dans nos travaux futurs, nous essayerons d'améliorer davantage notre algorithme par l'optimisation de la formule utilisée pour le calcul de fitness et ce, afin de rendre notre algorithme plus souple, et s'approcher au maximum de la longueur réelle du chemin.

Enfin, nous pensons que le travail présenté dans cette thèse ouvre de nouvelles perspectives selon les principaux axes suivants:

- La mise en œuvre pratique des méthodes étudiées sur un robot mobile réel. L'objectif étant de développer un planificateur de trajectoires pour un robot mobile de type véhicule, et de tester les capacités de cette approche sur un plan pratique.

- L'application de l'architecture présentée pour des environnements avec des obstacles dynamiques : la version de base de ces systèmes déplace le robot entre positions sans collision. L'opération est régie par un processus perception-action répété à une fréquence élevée. Le résultat est une séquence de mouvements en ligne qui conduit le robot jusqu'au but sans collisions.

- La comparaison des méthodes développées avec d'autres algorithmes stochastiques tels que l'algorithme des colonies de fourmis.

Références Bibliographiques

Références Bibliographiques

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

[ADA03]: Adachi, Y., Saito, H., Matsumoto, Y., Ogasawara, T., "Memory-based navigation using data sequence of laser range finder", Computational Intelligence in Robotics and Automation, Proceedings IEEE International symposium, Vol 1, pp. 479 - 484, 2003.

[AGU97]: Aguilar, L, E., "Commande robuste et coordination de mouvements de robots mobiles", PhD thesis, LAAS W97452, 1997.

[AHM01]: Ahmadabadi, M, N., Nakano, E., "A (constrain and move) approach to distributed object manipulation", IEEE Transactions on Robotics and Automation 17(2), pp.157-172, 2001.

[ALA98a]: Alami, R., Fleury, S., Herrb, M., Ingrand, F., Robert, F., "Multi-robot cooperation in the martha project", IEEE Robotics and Automation Magazine 5(1), pp.36 47, 1998.

[ALA98b]: Alami, R., Ingrand, F., Qutub, S., "A scheme for coordinating multi-robot planning activities and plans execution", in 13th European Conference on Artificial Intelligence, Brighton, UK, 1998.

[ALB89]: Albus, J, S., Mccain, H, G., Lumia, R., "NASA/NBS (National Aeronautics and Space Administration/National Bureau of Standards) standard reference model for tele-robot control system architecture(NASREM)", Technical Report for National Inst. of Standards and Technology- Robot System Div.; Gaithersburg, MD, United States, 1989.

[AND83]: Andrews, J, R., Hogan, N., "Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator", Control of Manufacturing Processes and Robotic Systems, ASME, Boston, pp. 243-251, 1983.

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

[ARK01]: Arkin, R, C., Fujita, M., Takagi, T., Hasegawa, R., "Ethological modeling and architecture for an entertainment robot", in International Conference on Robotics and Automation, pp. 453-458, 2001.

[ARK89]: Arkin, R, C., "Motor Schema-Based Mobile Robot Navigation", the International Journal of Robotics Research, pp. 92-112, 1989.

[ARK92]: Arkin, R, C., "Cooperation without communication: Multi-agent schema based robot navigation", Journal of Robotic Systems 9(3), pp.351-364, 1992.

[ARK98]: Arkin, R, C., "Behavior-Based Robotics ", the MIT Press, 1998.

[ARR06]: Arrichiello, F., "Coordination Control of Multiple Mobile Robots", PhD thesis, Università degli Studi di Cassino, 2006.

[AVI03]: Avila, G, R., Guisset, J., Deneubourg, J., "Synchronization in light-controlled oscillator", Elsevier, in press, 2003.

[AVI05]: Avina J, G., "Navigation visuelle d'un robot mobile dans un environnement d'extérieur semi-structuré". Thèse de doctorat, Institut National Polytechnique de Toulouse, 2005.

[AVO92]: Avouris, N., Gasser, L., "Distributed Artificial Intelligence", Theory and Praxis, Vol. 5, Kluwer Academic Publishers, 1992.

[AWE92]: Awerbuch, B., Peleg, D., "Routing with polynomial communication-space trade-off". SIAM Journal on Discrete Mathematics, vol 5 issue 2, cité page 20, 1992.

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

[BAC01]: Bacchus, F., Ady, M., "Planning with resources and concurrency: A forward chaining approach", in International Joint Conferences on Artificial Intelligence (IJCAI), 2001.

[BAL03]: Baldassarre, G., Nolfi S., Parisi, D., "Evolution of collective behaviour in a team of physically linked robots", in Gunther, R., Guillot, A., Meyer, J,A., editors, Applications of Evolutionary Computing, Springer Verlag, Heidelberg, Germany, pp. 581-592, 2003.

[BAL95a]: Balch, T., Arkin, R, C., "Communication in reactive multiagent robotic systems", Autonomous Robots vol1 (1), pp.27-52, 1995.

[BAL95b]: Balch, T., Arkin, R, C., "Motor schema-based formation control for multiagent robot teams", in Lesser V., Gasser, L., editors, Proceedings of the First International Conference on Multiagent Systems (ICMAS'95), AAAI Press, San Francisco, CA, USA, pp. 10-16, 1995.

[BAL97]: Balch, T., "Integrating motor schemas and reinforcement learning", Technical report, College of Computing GIT-CC-97-11, 1997.

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

[BAY09]: Bayle, B., "Robotique mobile : note de cours", école Nationale Supérieure de Physique de Strasbourg, 2009.

[BEA06]: Beaudry, E., "Planification de tâches pour un robot mobile autonome". Faculté des sciences, université de Sherbrooke, Canada, 2006.

[BEL02]: Belker, T., Schulz, D., "Local Action Planning for Mobile Robot Collision Avoidance", intelligent Robots and System, IEEE/RSJ International Conference on Volume 1, 30, Page(s):601 - 606 vol.1, 2002.

[BEN12a]: Benmachiche, A., Bouhadada, T., Laskri M, T., "Une Nouvelle Approche Evolutionnaire pour la Navigation Optimale de Robots Autonomes", Proceeding of the International Conference on Embedded Systems in Telecommunication, Annaba – Algeria, 2012.

[BEN12b]: Benmachiche, A., Bouhadada, T., Laskri M, T., "Cooperative Approach navigation for autonomous Robots Mobile(Application on a building Site)", Proceeding of the International Conference on Distributes Systems and Decision, Production Systems, page 51, ISSN 2335-1012, Oran – Algeria, 2012.

[BEN14]: Benmachiche, A., Bouhadada, T., Laskri M, T., "New planning Model of mobile robot based on multi-agent theory in a complex environment", Proceeding of the International Conference on Artificial intelligence and Information Technology, Ouargla Algérie, 2014.

[BEN15]: Benmachiche, A., Bouhadada, T., Laskri M, T., "A dynamic navigation for autonomous mobiles robots", iospress: Intelligent Decision Technologies, ISSN 1875-8843 (E), Volume 10, Issue 1, 2016.

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

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

[BER93]: Bertoni, A., Dorigo M., "Implicit parallelism in genetic algorithms", Artificial Intelligence, 61(2) :307–314, 1993.

[BON99]: Bonabeau, E., Dorigo, M., Theraulaz, G., "Swarm intelligence 'from Natural to Artificial systems", Oxford University Press, 1999.

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

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

[BOT99]: Botelho, S., Alami, R., "Multirobot Cooperation in the Martha Project", International Conference on Robotics and Automation (ICRA'99), pp. 1234-1239.1999

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

[BRI91]: Bridges, C, L., Goldberg D, E., "An analysis of multipoint crossover". In Proceedings of the Foundation Of Genetic Algorithms (FOGA), 1991.

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

[BRO87]: Brooks, R, A., "Planning is just a way of avoiding figuring out what to do next", in Working Paper MIT, n 303, 1987.

[BRO90]: Brooks, R, A., "Elephant don't play chess", Robotics and Automation Systems 6, pp.3 15, 1990.

[CAL90]: Caloud, P., Choi, W., Latombe, J, C., Pape C, L., Yim, M., "Indoor automation with many mobile robots", in Intelligent Robots and Systems IROS, Ibaraki-Japan, pp. 67 72, 1990.

[CAN01]: Canou J., Novales C., Poisson G., Marché, P., "Quick primitives extraction using inertia matrix on measures issue from an ultrasonic network", ICRA01-IEEE, International Conference on Robotics and Automation, Séoul, Corée, 2001.

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

[CAP08]: Cappelle, C., Elbadaoui, M., Pomorski, D., Charpillet, F., "Détection, suivi et géolocalisation d'obstacles à l'aide d'un modèle 3D géo-référencé, une caméra et un GPS : validation avec un lidar", Conférence Internationale Francophone d'Automatique, CIFA'08, Bucarest, Roumanie, 2008

[CAS11]: Castaneda G, E., Monroy, D, J., Aponte, J, A., Aviles, O, F., "Design and construction of a mobile type rover robotics platform Robotics", Symposium IEEE IX Latin American and IEEE Colombian Conference on Automatic Control and Industry Applications (LARC), 2011

[CHA01]: Chatterjee, R., Matsuno, F., "Use of single side reflex for autonomous navigation of mobile robots in unknown environments", Robotics and Autonomous Systems, Volume 35, Issue 2, pp 77-96, 2001.

[CHE06]: Yang, C., Maimone, M, W., Matthies, L., "Visual odometry on the Mars exploration rovers - a tool to ensure accurate driving and science imaging", IEEE Robotics & Automation Magazine, Volume 13, Issue 2, pp 54-62, 2006.

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

[CHE14]: Cherroun, L., "Navigation Autonome d'un Robot Mobile par des Techniques Neuro-Floues". Thèse de doctorat, Université Mohamed Khider de Biskra, 2014.

[CHE94]: Chen, Q., Luh, J, Y, S., "Coordination and control of a group of small mobile robots", in International Conference on Robotics and Automation, pp. 2315 2320, 1994.

[CHT01]: Chatterjee, R., Matsuno F., "Use of single side reflex for autonomous navigation of mobile robots in unknown environments", Robotics and Autonomous Systems, Volume 35, Issue 2, pp. 77-96, 2001.

[COH96]: Cohen, W., "Adaptive mapping and navigation by teams of simple robots", Robotics and Autonomous Systems 18, pp.411-434, 1996.

[COL91]: Collins, R, J., Jefferson D, R., "Selection in massively parallel genetic algorithms", in Proceedings of the Fourth International Conference on Genetic Algorithms, 1991.

[COU08]: Courbon, J., Mezouar, Y., Lequievre, L., Eck, L., "Navigation of urban vehicle: An efficient visual memory management for large scale environments", International Conference on Intelligent Robots and Systems (IROS), pp: 1817-1822, 2008.

[COU09]: Courbon, J., Mezouar, Y., Eck, L., Martinet, P., "A generic framework for topological navigation of urban vehicle", International Conference on Robotics and Automation Workshop on Safe navigation in open and dynamic environments Application to autonomous vehicles, ICRA09, Kobe, Japan, 2009.

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

[DAL01]: Dalgalarrondo, A., "Intégration de la fonction perception dans une architecture de contrôle de robot mobile autonome", Thèse doctorat, université de paris sud centre d'Orsay, France, 2001.

[DAW89]: Dawkins, R., "L'horloger aveugle", Robert Laffont, 1989.

[DIA04]: Dias, M, B., "TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments". PhD thesis, Robotics Institute, Carnegie Mellon University. (Cité pages 3, 35, 37 et 47.), 2004.

[DON94]: Donald, B, R., Jennings J., Rus, D., "Analyzing teams of cooperating mobile robots", in International Conference on Robotics and Automation, pp. 1896 1903, 1994.

[DOR91]: Dorigo, M., Schnepf, U., "Organisation of robot behaviour through genetic learning processes", in Tifth IEEE International Conference on Advanced Robotics', Pisa, Italy, pp. 1456 1460, 1991.

[DRE92]: Dreyfus, H, L., "What Computers Still Can't Do: A Critique of Artificial Reason", The MIT Press. Revision of What Computers Can't Do, MIT Press'79, 1992.

[DRO93]: Drogoul, A., "De La Simulation Multi-Agent A La Résolution Collective de Problèmes line Étude De l'Émergence De Structures D'Organisation Dans Les Systèmes Multi-Agents", PhD thesis, Université Paris VI, 1993.

[DUR89]: Durieu C., "Algorithmes de localisation d'un robot mobiles dans un milieu balisé par mesure de distance ou d'angle de gisement", Thèse de Doctorat, Université de Paris sud, 1989.

[ER 04]: Er, M, J., Tan, T, P., Loh, S, Y., "Control of a mobile robot using generalized dynamic fuzzy neural networks", Microprocessors and Microsystems, Volume 28, issue, pp. 491-498 9, 2004.

[ERC91]: Erceau, J., Ferber, J., "Intelligence artificielle distribuée", La Recherche (233), pp.750-758, 1991.

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

[FEI94]: Feiten, W., Bauer, R., Lawitzky, G., "Robust obstacle avoidance in unknown and cramped environments", International Conference on Robotics and Automation, 1994.

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

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

[FIK71]: Fikes R., Nilsson, N., "STRIPS: a new approach to the application of theorem proving". Artificial Intelligence, 2(3-4), pp. 189–208, 1971.

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

[FLI95]: Fliess, M., Lévine, J., Martin, P., Rouchon, P., "Flatness and defect of non-linear systems: introductory theory and examples". International Journal of Control, volume 6(1) :1327–1361, 1995.

[FOG66]: Fogel, L, J., Owens, A, J., Walsh, M, J., "Artificial Intelligence Through Simulated Evolution", Wiley and sons, NewYork, USA, 1966.

[FOX97]: Fox, D., Burgard, W., Thrun S., "The Dynamic Window Approach to Collision Avoidance", IEEE Robotics and Automation Magazine, vol. 4, no 1, pp 23– 33, 1997.

[FRA04]: Fraisse, P., Zapata, R., Zarrad, W., Andreu, D., "Remote secure decentralized control strategy for mobile robots", Journal of Advanced Robotics, in press, 2004.

[FRA11]: Fraisse, P., Gil, A., Zapata, R., "Stratégie de commande décentralisée d'une flottille de robots mobiles terrestres téléopérés", Journées nationales de la recherche en robotique (JNRR 05), Guidel, 2011.

[FRU05]: Fruchard, M., "Méthodologies pour la commande de manipulateurs mobiles non-holonomes", thèse soutenue à l'école nationale supérieure des mines de paris - sophia antipolis , préparée à l'INRIA Antipolis, 2005.

[GAN05]: Gancet, J., "systèmes multi-robots aériens Architecture pour la planification, la supervision et la coopération", thèse de doctorat, LAAS-CNRS, 2005.

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

[GEI94]: Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V., "Pvm 3 user's guide and reference manual", Technical report, Oak Ridge National Laboratory, 1994.

[GER02]: Gerkey, B, P., Mataric M, J., "Sold!: Action methods for multirobot coordiantion", IEEE Transactions on Robotics And Automation, 18(5), pp. 758–768, 2002.

[GHA04]: Ghallab, M., Dana, S., Traverso, P., "Automated Planning ", Theory and Practice, Elsevier , ISBN-13: 978-1-55860-856-6, ISBN-10: 1-55860-856-7, pp 1-635, 2004

[GHA94]: Ghallab, M., Laruelle, H., "Representation and control in IxTeT, a temporal planner", in Proceedings of the second international conference on Artificial Intelligence Planning Systems (AIPS-94), University of Chicago, USA, 1994

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

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

[GOL89]: Goldberg, D, E., "Genetic Algorithms in Search, Optimization and Machine Learning". Reading MA Addison Wesley, ISBN:0-262-11170-5, 1989.

[GOL91]: Goldberg, D, E., "Real-coded genetic algorithms, virtual alphabets and blocking", Complex Systems, vol 5, pp. 139–167, 1991.

[GRA59]: Grassé, P, P., "La construction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. la théorie de la stigmergie : essais d'interprétation du comportement des texmites constructeurs", Insectes Sociaux 6, pp. 41-81, 1959.

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

[HAR07]: Harvey, B., "Soviet and russian lunar exploration", Springer Praxis, ISBN 0-387-21896-3, 2007.

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

[HIR00]: Hirata, Y., Kosuge, K., Asama, H., Kaetsu, H., Kawabata, K., "Coordinated transportation of a single object by multiple mobile robots without position information of each robot", in IEEE/RSJ'00 International conference on Intelligent Robots and Systems, pp. 2024-2029, 2000.

[HIR02]: Hirata, Y., Hatsukari, T., Kosuge, K., Asama, H., Kaetsu, H., Kawabata, K., "Transportation of an object by multiple distributed robot helpers in cooperation with a human", Transactions of the Japan Society of Mechanical Engineers 68(668), pp.181-188, 2002.

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

[HOL62]: Holland, J., "Outline for a logical theory of adaptive systems", Journal of the Association of Computing Machinery, 3-9, 1962.

[HOL75]: Holland, J., "Adaption in Natural and Artificial Systems", University of Michigan Press. (1975).

[HOR93]: Horn, J., Nafpliotis, N., "Multiobjective optimization using the nitched pareto genetic algorithm", illigal Report 93005, University of Illinois at Urbana, 1993.

[HOR99]: Hornby, G., Fujita, M., Takamura, S., Yamamoto, T., Hanagata, O., "Autonomous evolution of gaits with the sony quadruped robot", in Proceedings of Genetic and Evolutionary Computation Conference (GECCO), pp. 1297 1304. Horton I., Visual C f 6, Eyrolles, 1999.

[HUA01]: Huang, H., Liang, C, C., Lin, C, W., "Construction and soccer dynamics analysis for an integrated multi-agent soccer robot system", in proceedings of the national science council, Vol. 25, pp. 84-93, republic of china, 2001.

[HUG00]: Hugues, L., "Collective grounded representations for robots", in fifth Symposium on Distributed Autonomous Robotics Systems, 2000.

[HUN03]: Huntsberger, T., Pirjanian, P., "A control architecture for tightly coupled coordination of multi-robot systems for planetary surface exploration", IEEE Trans. Systems, Man and Cybernetics, Part A: Systems and Humans, Special Issue on Collective Intelligence 33(5), pp.550-559, 2003.

[ING92]: Ingber, L., Rosen, B., "Genetic algorithms and very fast simulated re-annealing", Mathematical Computer Modeling, 16(11), pp. 87-100, 1992.

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

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

[JET99]: Jetto, L., Longhi, S., Vitali, D., "Localization of a wheeled mobile robot by sensor data fusion based on a fuzzy logic adapted Kalman filter", Control Engineering Practice, vol 7, pp. 763-771, 1999.

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

[JUN00]: Jung, D., Zelinsky, A., "Grounded symbolic communication between heterogeneous cooperating robots", Autonomous Robots journal, special issue on Heterogeneons Multi-robot Systems, Kluwer Academic Publishers, Baleh, Tucker and Parker, Lynne E. (eds.) 8(3), pp. 269-292, 2000.

[KAL04]: Kalay, Y, E., "Architecture's new media: principles, theories, and methods of computer-aided design", Massachusetts: MIT press, 2004.

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

[KHA90]: Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", IEEE International Conference on Robotics and Automation, St. Louis, Missouri, pp.500-505, March 25-28, 1990.

[KHA95]: Khatib, O., Yokoi, K., Chang, K., Ruspini, D., Holmberg, R., Casai, A., Baader, A., "Force strategies for cooperative tasks in multiple mobile manipulation systems", in International Symposium of Robotics Research, Munich, 1995.

[KHA97]: Khatib, M., Jaouni, H., Chatila, R., Laumond, J, P., "Dynamic path modification for car-like non holonomic mobile robots", International Conference on Robotics and Automation. Albuquerque, NM, USA, 1997.

[KHA99]: Khalil, W., Dombre, E., "Modélisation, identification et commande des robots", Hermès, Paris, 1999.

[KOR91]: Koren, Y., Borenstein, J., "Potential field methods and their inherent limitations for mobile robot navigation", Robotics and Automation, vol 2, Sacramento, California, pp. 1398-1404, 1991.

[KUB00]: Kube, C., Bonabeau, E., "Cooperative transport by ants and robots", Robotics and Autonomous Systems 30(1/2), pp. 85-101, 2000.

[KVA00]: Kvarnström, J., Doherty, P., Haslum, P., "Extending TALplanner with concurrency and resources", in Proceedings of the 14th European Conference on Artificial Intelligence (ECAI), Berlin, Germany, 2000.

[LAB93]: Labidi, S., Lejouad, W., "De l'intelligence artificielle distribuée aux systèmes multi-agents", Rappot n°2004, programme 2 : Calcul symbolique, programmation et génie logiciel, INRIA Sopia-Antipolis, 1993.

[LAR00]: Large, F., Sekhavat, S., Laugier, C., Gauthier, E., "Towards robust sensor-based maneuvers for a car-like vehicle", International Conference on Robotics and Automation. San Francisco, CA, USA, 2000.

[LAR03]: Large, F., "Navigation autonome d'un robot mobile en environnement dynamique et incertain", Thèse de doctorat, Institut National de recherche en informatique et en automatique, Université de Savoie, 2003.

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

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

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

[LAU99]: Laugier, C., Fraichard, T., Garnier, P., Paromtchik, I, E., Scheuer, A., "Sensor-Based Control Architecture for a Car-Like Vehicle", Autonomous Robots, vol. 6, no 2, 1999.

[LEB05]: Lebedev, D, V., Steil, J, J., Ritter, H, J., "The dynamic wave expansion neural network model for robot motion planning in time-varying environments", Neural Networks, Volume 18, Issue 3, pp. 267-285, 2005.

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

[LEN75]: Lenat, D, B., "Beings: Knowledge as interacting experts", editions `LICAI-4', in Kauffinan Los Angeles, pp. 126-133, 1975.

[LER98]: Leroy S., "Outils géométriques pour la planification de trajectoires de robots mobiles non holonomes", PhD thesis, LAAS W98534, 1998.

[LET59]: Lettvin, J, Y., Maturana, H, R., McCulloch, W, H., "What the Frog's Eye Tells the Frog's Brain", Proceedings of The Institute of Radio Engineers, New York, Vol. 47, No. 11, pp. 1940-51, 1959.

[LIU01]: Liu, J., Wu, J., "Multiagent Robotic Systems", CRC Press LLC, Taylor & Francis Group, pp 3, 2001.

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

[MAH92]: Mahfoud, S, W., Goldberg, D, E., "Parallel recombinative simulated annealing: A genetic algorithm", illigal report 92002, University of Illinois, Urbana, IL 61801-2996, 1992.

[MAR95]: Martinoli, A., Mondada, F., "Collective and cooperative group beahviours: Biologically inspired expermients in robotics", in Proceedings of the Fourth Symposium on Experimental Robotics ISER, Vol 223, pp. 3-10, Stanford, USA, 1995.

[MAS86]: Mason, M, T., "Mechanics and planning of manipulator pushing operations", The International Journal of Robotics Research 5(3), pp.53 71, 1986.

[MAT01]: Mataric, M, J., "Learning in behavior-based multi-robot systems: Policies, models, and other agents", Cognitive Systems Research, special issue on Multi-disciplinary studies of multi-agent learning, 2(1), pp.81-93, 2001.

[MAT04]: Mataric, M, J., "Interaction and Intelligent Behavior", PhD thesis, MIT, 2004.

[MAT92]: Mataric, M, J., "Minimizing complexity in controlling a mobile robot population", in IEEE International Conference on Robotics and Automation, Nice-Rance, pp. 830-835, 1992.

[MAT95]: Mataric, M, J., Nilsson, M., Simsarian, K., "Cooperative multi-robots box-pushing", in 'IEEE International Conference on Intelligent Robots and Systems', Vol. 3, pp. 556 561, Pittsburgh, PA, 1995.

[MEL01]: Melhuish, C., "Strategies for Collective Minimalist Mobile Robots", Vol. ERS 6, Professsional Engineering Publishing, 2001.

[MEZ09]: Mezaache, H., Foudil, A., "Simulation of the Navigation of a Mobile Robot by the QLearning using Artificial Neuron Networks", Proceedings of the 2nd Conférence Internationale sur l'Informatique et ses Applications (CIIA'09), Saida, Algeria, 2009.

[MIC68]: Michie, D., Chambers, R., "Boxes: an experiment in adaptive control", In Machine Intelligence, vol 2, pp. 137-152, 1968.

[MIC91]: Michalewicz, Z., Janikov, C, Z., "Handling constraints in genetic algorithms", In Proceedings of the Fourth International Conference on Genetic Algorithm, ICGA, 1991.

[MIC92]: Michalewicz, Z., "Genetic algorithms + Data Structures = Evolution Programs", Springer-verlag, 1992.

[MIF03]: Mufioz, A., "Coopération située : une approche constructiviste de la conception de colonies de robots", PhD thesis, Université Pierre et Marie Curie, Paris VI, 2003.

[MIH89]: Muhlenbein, H., "Parallel genetic algorithms, population genetics and combinatorial optimization", in Proceedings of the Third International Conference on Genetic Algorithms, 1989.

[MIN00]: Minguez, J., Montano L., "Nearness diagram navigation (ND): a new real time collision avoidance approach", International Conference on Intelligent Robots and Systems. Takamatsu, Japan, IEEE/RSJ, 2000.

[MIN02]: Minguez, J., Montano, L., Victor J, S., "Reactive navigation for nonholonomic robots using the ego-kinematic space", International Conference on Robotics and Automation IEEE. Washingtom, DC, USA, 2002.

[MIN83]: Mintzberg, H., "Structure in Fives: Designing Effective Organizations", Prentice Hall United States ISBN 013855479X, 1983.

[MIN88]: Minsky, M., Papert, S., "Perceptrons", Expanded Edition, MIT Press'69, 1988.

[MOR06]: Morin, P., Samson, C., "Trajectory tracking for non-holonomic vehicles: overview and case study", 4th International Workshop on Robot Motion Control (RoMoCo), pp. 139–153, 2006.

[MOU06]: Mourioux, G., Novales, C., Poisson, G., Vieyres, P., "Omni-directional robot with spherical orthogonal wheels : concepts and analyses", Proceedings of IEEE International Conference on Robotics and Automation, ICRA 06, Orlando, Floride USA, 15-19, 2006.

[MUF03]: Mufioz, A., "Coopération située : une approche constructiviste de la conception de colonies de robots", PhD thesis, Université Pierre et Marie Curie, Paris VI, 2003.

[MUL00]: Muller, P, A., Gaertner, N., "Modélisation objet avec UML", Eyrolles, 2000.

[MUL98]: Muller, J., Petin, J., Morel, G., Vachon, B., Pegard, C., Brassart, E., Hutin, N., Dembelé, S., Janex, A., Morelo, B., Ravassard, J. & Bourjault, A., "Conception de systèmes de transport collectif d'objets", in Premières Journées du Pôle Microrobotique, pp. 61 66, 1998.

[NAG98]: Nagatani, K., Choset, H., Thrun, S., "Towards exact localization without explicit localization with the generalized Voronoi graph", in Proceedings of the IEEE International Conference on Robotics and Automation, pp. 342-348, 1998.

[OUA05]: Ouadah, N., Azouaoui, O., Hamerlain, M., "Implémentation d'un contrôleur flou pour la navigation d'un robot mobile de type voiture", Troisième Congrès francophone, Majecstic, Rennes, France, 2005.

[PAR03]: Parker, L, E., "Multi-Robot Systems", Swarms to Intelligent Automata, Vol 2, chapter The effect of heterogeneity in teams of 100 mobile robots, pp. 205–215, 2003.

[PET87]: Pettey, C., Leuze, M., Grefenstette, J., "A parallel genetic algorithm", in Proceedings of the Second International Conference on Genetic Algorithms, 1987.

[PIE00]: Pierre, A., "Des moutons et des robots : architecture de contrôle réactive et déplacements collectifs de robots", PPUR presses polytechnique, 2000.

[PIE02]: Piepmeier, J., Kenneth, A., Bishop, B., "The Use of MATLAB for Robotic Control in an Undergraduate Robotics", in Proceedings of American Society for Engineering Education Annual Conference and Exposition, 2002.

[PRA05]: Pradalier, C., Hermosillo, J., Koike, C., Braillon, D., Bessière, P., Laugier, C., "The CyCab: a car-like robot navigating autonomously and safely among pedestrians", Robotics and Autonomous Systems (RAS) 50(1), pp. 51-67, 2005.

[PRA06]: Pradel, G., Hoppenot, P., "Symbolic trajectory description in mobile robotics", Journal of Intelligent & Robotics Systems, vol. 45, pp. 157-180, 2006.

[PRU96]: Pruski, A., "Robotique mobile, la planification de trajectoire", ED. Hermes, Paris, 1996.

[QUI93]: Quinlan, S., Khatib, O., "Elastic Bands: Connecting Path Planning and Control", International Conference on Robotics and Automation. Atlanta, USA, 1993.

[RAY86]: Raynal, M., "Algorithms for Mutual Exclusion", MIT Press, ISBN 0-262-181193, 1986.

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

[RUS03]: Russell, S, J., Norvig, P., "Artificial Intelligence: A Modern Approach Upper Saddle River", N.J.: Prentice Hall, pp. 97-104, ISBN 0-13-790395-2, 2003.

[RUS95]: Rus, D., Donald, B., Jennings, J., "Moving furniture with teams of autonomous robots", in Intelligent Robots and Systems (IROS'95'), Vol. 1, pp. 235 242, 1995.

[SAF97]: Saffiotti, A.., "The Uses of Fuzzy Logic for Autonomous Robot Navigation", Soft Computing, vol 1, no 4, pp.180-197, 1997.

[SAS95]: Sasaki, H., Nagura, K., Ishino, M., Tobioka, H., Kotani, K., Sasaki, T., "Cloning and characterization of cell adhesion kinase beta, a novel protein-tyrosine kinase of the focal adhesion kinase subfamily", The Journal of Biological Chemistry, 270(36):21206-19, 1995.

[SEM04]: Sempe, F., "Auto-organisation d'une collectivité de robots : Application à l'activité de patrouille en présence de perturbations". Thèse de Doctorat de l'Université Pierre et Marie Curie, Paris VII, 2004.

[SHU06]: Shuzhi, S, G., Lewis, F, L.., "Autonomous Mobile Robots, Sensing, Control, Decision, Making and Applications", Taylor and Francis Group, 2006.

[SIC08]: Siciliano, B., Khatib, O., "Springer Handbook of Robotics", Springer-Verlag Berlin Heidelberg, pp 3, 2008.

[SIE04]: Siegwart, R., Nourbakhsh, R., "Introduction to autonomous mobile robots", The MIT Press155, 2004.

[SMI93]: Smith, R, E., Perelson, A, S., Forres,t S., "Searching for diverse, cooperative populations with genetic algorithms". Evolutionary Computation, 1(2), pp. 127-149, 1993.

[SOU96]: Soueres, P., Laumond, J, P., "Shortest paths synthesis for a car-like robot", IEEE Transactions on Robotics and Automation, vol. 41, no 5, pp. 672– 688, 1996.

[STE90]: Steels, L., "Cooperation between distributed agents through self-organisation", First European Workshop on Modelling Autonomous Agents in a Multi-Agent \Vorld, Cambridge, England', North-Holland, Amsterdam, pp. 175-196, 1990.

[STE94]: Stentz, A., "Optimal and Efficient Path Planning for Partially-Known Environments", International Conference on Robotics and Automation, 1994.

[STE95]: Steels, L., "When are robots inteligent autonomous agents?", journal of robotics, Volume 15 pp 3-9, 1995.

[STI93]: Stilwell, D., Bay, J., "Toward the development of a material transport system using manas of ant-like robots", in International Conference on Roboties and Automation, pp. 766-771, 1993.

[STY01]: Sty, K., "Using situated communication in distributed autonomous mobile robots", in Seventh Scandinavian Conference on Artificial Intelligence (SCAI01), 2001.

[SUT98]: Sutton, R, S., Barto, A, G., "Reinforcement Learning: An Introduction", MIT Press, Cambridge, MA, 1998.

[TAK08]: Takayama, Z., Ju, W., Nass, C., "Beyond dirty, dangerous and dull: What everyday people think robots should do", In Proceedings of the 3rd ACM / IEEE international conference on Human robot interaction, pp 25-32, 2008.

[THR95]: Thrun, S., "An Approach to Learning Mobile Robot Navigation", Robotics and Autonomous Systems, vol 15, pp. 301-319, 1995.

[THR96]: Thrun, S., Bucken, A., "Integrating Grid-based and topological maps for mobile robot navigation", In Proceedings of the Thirteenth National Conference on Artificial Intelligence, pp. 944-950, 1996.

[TIG96]: Tigli, J, Y., "Vers une Architecture de contrôle pour Robot Mobile orientée Comportement", Thèse de doctorat, Université de Nice-Sophia, Antipolis, 1996.

[TOU92]: Tournassoud, P., "Planification et contrôle en robotique Application aux robots mobiles et manipulateurs", Hermes, 1992.

[TOU97]: Touzet, C., "Neural Reinforcement Learning for Behaviour Synthesis", Robotics and Autonomous Systems, vol 22, no 3, pp. 251-281, 1997.

[VAU00a]: Vaughan, R, T., Stoy, K., Sukhatme, G, S., Mataric, M, J., "Blazing a trail: insect-inspired resource transportation by a robot team", in International symposium Distributed Autonomous Robot Systems, 2000.

[VAU00b]: Vaughan, R, T., Stoy, K., Sukhatme, G, S., Mataric, M, J., "Whistling in the dark: cooperative trail following in uncertain localization space, in Proceedings of the fourth international conference on Autonomous agents, 2000.

[VEL03]: Velghe, J., "Localisation et suivi d'objets par vision", Diplôme d'Études Approfondies – DEA IAP, Laboratoire d'Automatique de Besançon, 2003.

[VLA96]: Vladimir, E., "The role of selection in genetic algorithms", Technical Report FIT-TR-1996-04, 1996.

[VOL97]: Volpe, R., Balaram, J., Ohm, T., Ivlev, R., "Rocky 7: A next generation mars rover prototype", Advenced Roboties 11(4), pp. 341-358, 1997.

[WAN08]: Wang, X., Hou, Z., Zou, A., Tan, M., Cheng, L., "A behavior controller based on spiking neural networks for mobile robots", Neurocomputing Vol 71, Issue 4-6, pp. 655-666, 2008.

[WAN91]: Wang, P., "Navigation strategies for multiple autonomous mobile robots moving in formation", Journal of Roboties Systems 8(2), pp.177 195, 1991.

[WAT92]: Watkins, C., Dayan, P., "Technical Note, Q-learning", Machine Learning, vol. 8, pp. 279-292, 1992.

[WRI91]: Wright, A, H., "Genetic algorithms for real parameter optimization", in Proceeding of the Foundation of Genetic Algorithms, FOGA, 1991.

[XU 99]: Xu, W, L., "A virtual target approach for resolving the limit cycle problem in navigation of a fuzzy behaviour-based mobile robot", Institute of Technology and Engineering, College of Sciences, Massey University, Palmerston North, New Zealand, 1999.

[YAM01]: Yamaguchi, H., Arai, T., Beni, G., "A distributed control scheme for multiple robotic vehicles to make group formations", Robotics and Autonomous Systems, vol 36, pp.125-147, 2001.

[YIN93]: Yin, X., Germay, N., "A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization", in Proceedings of the Artificial Neural Nets and Genetic Algorithms, 1993.

[ZAD65] Zadeh, L., "Fuzzy sets", Information and Control, 8(3):338- 353, 1965.

[ZHO07]: Zhou, Y., Er, M, j., Wen, Y., "A Hybrid Approach for Automatic Generation of Fuzzy Inference Systems without Supervised Learning", Proceedings of American Control Conference, USA, pp. 3371-3376, 2007.

Webographie

[WWW1]: https://sites.google.com/site/3agaelleamelie/1967-shakey-premier-robot-mobile-controle-par-ordinateur-stanford-research-institute, 2016.

[WWW2]: http://mars.nasa.gov/mer/mission/comm_busy.html, 2016.

[WWW3]: Le robot mobile "Sejourner" utilisé pour la mission pathfinder de la NASA, www.robocup2014.org, 2014.

[WWW4]: https://msdn.microsoft.com/fr-fr/library/system.threading.waithandle(v=vs.110).aspx, 2016