République Algérienne Démocratique Et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Abou-Bekr Belkaid - Tlemcen
Faculté de technologie
Département de Génie électrique et électronique
Filière nationale Génie industriel

Projet Fin D'étude
En vue d'obtention du Diplôme de Master en Génie industriel
Spécialité Ingénierie des Systèmes

Intitulé

Conception, réalisation et implémentation d'un robot mobile dans un environnement de travail

Filière: Génie industriel

Spécialité : Ingénierie des systèmes

Réalisé par : BOUSIF Fares

Présenté devant le jury composé de :

M. BENHABIB Mohamed Choukri	Président	MCA-Université de Tlemcen
M^{me}. HOUBAD Yamina	Examinateur	MAA-Université de Tlemcen
M. GUEZZEN Amine Hakim	Encadrant	MCB-Université de Tlemcen

Année universitaire : 2019/2020

Dédicaces

Je dédie ce travail à :

Mes parents, que dieu me les garde,

À mes frères et sœurs,

Et à tous ceux que j'aime.

Remerciement

Après louange à Allah,

Je tiens, tout d'abord, à exprimer mes vifs remerciements à mon encadrant M. GUEZZEN AMINE pour m'avoir inspiré le sujet de ce projet de fin d'étude et pour m'avoir guidé et conseiller tout au long de l'élaboration de ce travail.

Je suis profondément reconnaissant à Monsieur BENHABIB Mohamed Choukri (Maître de conférences à l'Université de Tlemcen) pour m'avoir fait l'honneur de présider le jury du soutenance.

Nous remercions vivement M^{me} HOUBAD Yamina, Maître assistante à l'Université de Tlemcen, pour avoir accepté d'examiner notre travail.

Enfin, je n'oublie pas d'adresser mes vifs remerciements à toute ma famille, qui m'a accompagné tout au long de mes études par son soutien permanent.

BOUSIF Fares

Table des matières

Chapitre II : État de l'art sur la planification de trajectoire

Chapitre III : Algorithme d'odométrie, et résultats de simulation

Chapitre IV : Réalisation et Implémentation

Listes des Figures

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

Introduction Générale

Introduction Générale

L'un des défis de l'homme aujourd'hui est de copier la nature et de reproduire des modes de raisonnement et de comportement qui lui sont propre. Cette envie a fait naître le concept d'intelligence artificielle. Ce concept est étroitement lié à la robotique.

Les robots mobiles feront bientôt une partie intégrante de notre vie quotidienne. Le progrès en robotique mobile s'est avancé en un point où il devient possible de construire des robots qui peuvent agir dans nos bureaux, maisons, écoles, hôpitaux et laboratoires de recherches. Les robots rempliront bientôt plus de rôles que leurs applications courantes en tant qu'assembleurs industriels. Nous avons maintenant atteint un point où nous pouvons concevoir des robots qui peuvent agir en tant qu'associés avec nous (enseignants, assistants de santé, assistants domestiques, chirurgiens, acteurs et collaborateur scientifiques) au lieu d'être simplement un outil ou un jouet ! Cela signifie que nous pouvons nous attendre à ce que le robot mobile agisse et réagisse d'une manière autonome comme un être humain, par exemple, comprendre nos directions pour accomplir une tâche, nous guider à apprendre quelques informations et nous aider quand nous aurions besoin d'un coup de main [1].

Des recherches intensives ont été réalisées au cours de ces dernières années dont le but est d'améliorer l'autonomie d'un robot mobile face à son environnement pour qu'il puisse, sans intervention humaine, accomplir les missions qui lui ont été confiées. Le spectre de ces missions est immense, il couvre des domaines aussi variés que l'industrie manufacturière, le spatial, l'automobile et plus récemment les loisirs et le secteur médical, ce qui démontre l'intérêt croissant porté aujourd'hui au sein de la communauté de la robotique mobile au développement d'un système intelligent autonome capable à la fois de percevoir correctement son environnement et également de savoir comment réagir en conséquence suivant le niveau d'autonomie. C'est à lui de planifier son parcourt et de déterminer avec quels mouvements va-t-il atteindre son objectif ?

Le contexte dans lequel s'inscrit notre travail, consiste à concevoir et réaliser un robot mobile et d'appliquer des techniques pour la navigation autonome de ce robot dans un environnement de travail, afin de permettre au robot de se mouvoir d'une position initiale à une autre finale en évitant les obstacles.

La problématique générale abordée dans ce mémoire est celle de la planification de mouvement dans un espace de travail dynamique, un espace de travail qui comporte des obstacles fixes ainsi que des obstacles mobiles.

Pour cela, des études ne cessent d'améliorer et de développer la perfection et la précision du robot mobile, de façon générale, vers une autonomie totale. Cet objectif essentiel ne peut être atteint que par un développement performant des trois fonctions principales : la locomotion, la perception, et la décision ainsi que la conception d'un comportement proche d'une intelligence artificielle tout en implantant un programme, qui permettra au robot d'être capable de collecter des informations grâce à des capteurs et de réaliser des actions.

Ce mémoire est organisé de la manière suivante:

Dans le chapitre 1, nous allons faire un tour rapide dans le monde de la robotique mobile pour examiner en bref les différentes parties constitutives d'un robot mobile.

Dans le chapitre 2, nous allons parler sur les déférents Approches de Navigation (un état de l'art sur la planification de trajectoire) d'un robot mobile dans un environnement dynamique.

Dans le chapitre 3, nous allons parler sur les principes algorithmes de navigation (algorithme d'odométrie et le filtre de Kalman), et bien aussi on va aborder une solution pour le problème du SLAM (localisation et cartographie simultanées), puis on va présenter les résultats de simulation.

Dans le chapitre 4, nous allons présenter la conception, puis la réalisation pour deux robots déférents dans leurs architectures, ensuite on va présenter quatre modes de réalisation et chaque mode à son utilité, Après on va finir par l'implémentions du robot dans un environnement de travail.

Chapitre I :

Généralité sur la robotique

1. Introduction

L'objet de la robotique est l'automatisation de systèmes mécaniques. En dotant le système de capacités de perception, d'action et de décision, l'objectif est de lui permettre d'interagir rationnellement avec son environnement, et de façon autonome. 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.

Ce chapitre a pour but d'introduire le sujet de ce travail en situant brièvement quelques généralités concernant la robotique mobile en général, ainsi que tous les points devant être abordés pour une meilleure compréhension de la navigation en robotique mobile.

2. Aperçu historique

En 1921, Karel Capek introduisit le terme "robot" dans sa pièce de théâtre "R.U.R." (Rossum's Universal Robot), du tchèque "robota" signifiant travail forcé, corvée. Il raconte l'histoire d'un savant appelé Rossum, ayant réussi à mettre au point des créatures semblables physiquement à des êtres humains, que son fils exploita au sein de son entreprise. Le terme "robotique" fut lui amené par l'écrivain Isaac Asimov, qui proposa les trois lois de la robotique suivantes [2]:

- Loi 1 : Un robot ne peut blesser un être humain ni, par son inaction, permettre qu'un humain soit blessé.
- Loi 2 : Un robot doit obéir aux ordres donnés par les êtres humains, sauf si de tels ordres sont en contradiction avec la Première Loi.
- Loi 3 : Un robot doit protéger sa propre existence aussi longtemps qu'une telle protection n'est pas en contradiction avec la Première et/ou la Deuxième Loi.

La Tortue construite par Grey Walter dans les années 1950 (Figure 1.1), est l'un des premiers robots mobiles autonomes. Grey Walter n'utilise que quelques composants analogiques, dont des tubes à vide, mais son robot est capable de se diriger vers une lumière qui marque un but, de s'arrêter face à des obstacles et de recharger ses batteries lorsqu'il arrive dans sa niche. Toutes ces fonctions sont réalisées dans un environnement entièrement préparé, mais restent des fonctions de base qui sont toujours des sujets de recherche et de développement technologiques pour les rendre de plus en plus génériques et robustes [9].

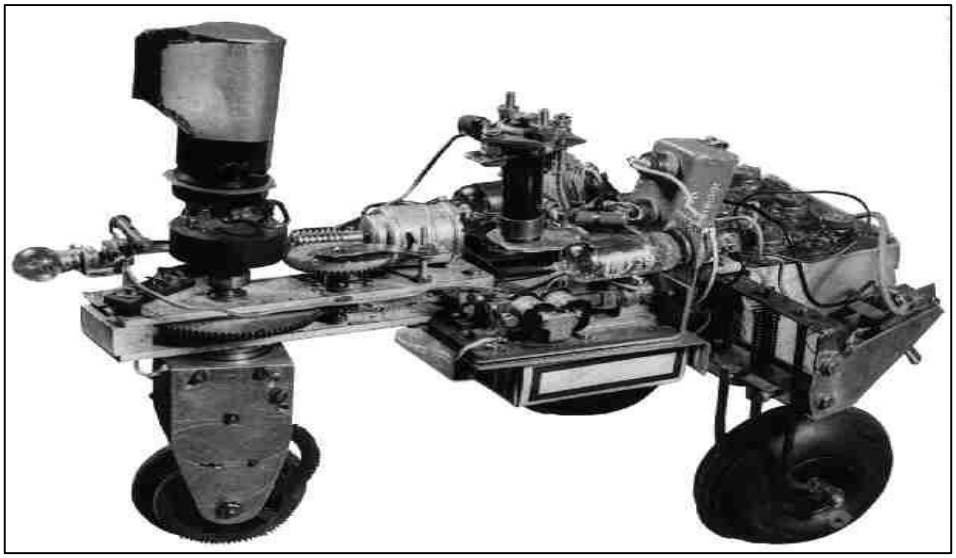


Figure 1.1 : La tortue de Grey Walter (Machina Speculatrix ou Elsie) avec l'illustration [94]

Dans les années 60, les recherches en électronique vont conduire, avec l'apparition du transistor, à des robots plus complexes mais qui vont réaliser des tâches similaires. Ainsi le robot "Beast" (Figure 1.2) de l'université John Hopkins est capable de se déplacer au centre des couloirs en utilisant des capteurs ultrason, de chercher des prises électriques (noires sur des murs blancs) en utilisant des photo-diodes et de s'y recharger.

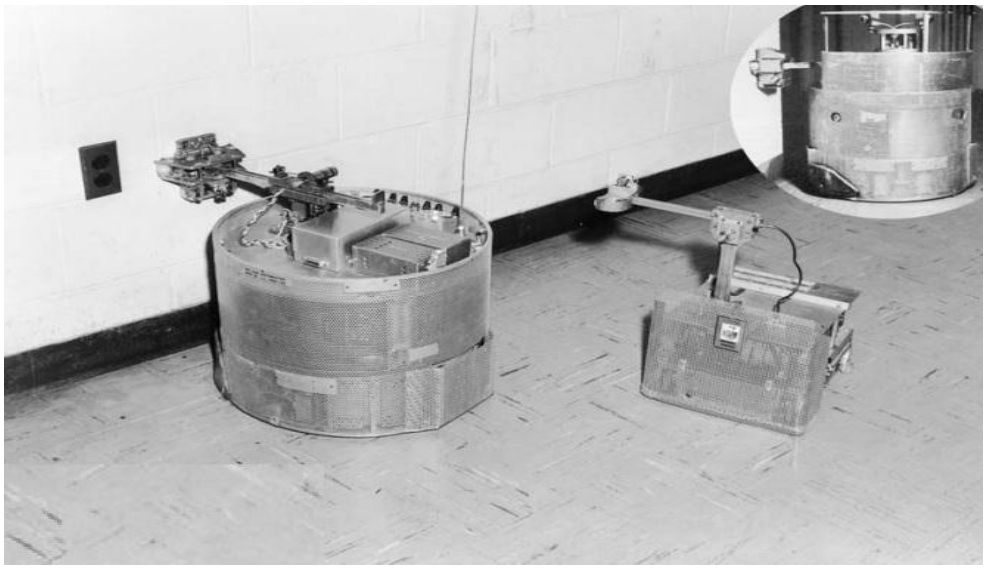


Figure 1.2 : Robot « Beast » de l'université John Hopkins [15]

Les premiers liens entre la recherche en intelligence artificielle et la robotique apparaissent à Stanford en 1969 avec Shakey (Fig.1.3). Ce robot utilise des télémètres à ultrason et une caméra et sert de plate-forme pour la recherche en intelligence artificielle.

Figure 1.3: Shakey de Stanford : Une plate-forme pour les recherches en intelligence artificielle [9]

Le Stanford Cart date de la fin des années 1970 (Fig.1.4), avec notamment les premières utilisations de la stéréo-vision pour la détection d'obstacles et la modélisation de l'environnement.

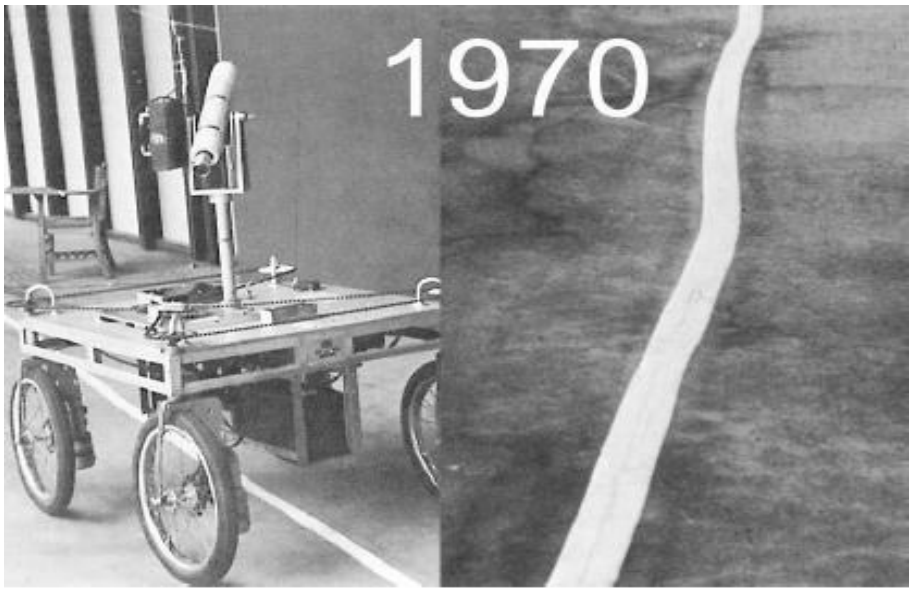


Figure 1.4: Le Stanford Cart de la fin des années 1970 [9]

En France, le robot Hilaire était le premier robot construit au LAAS à Toulouse (Fig.1.5).

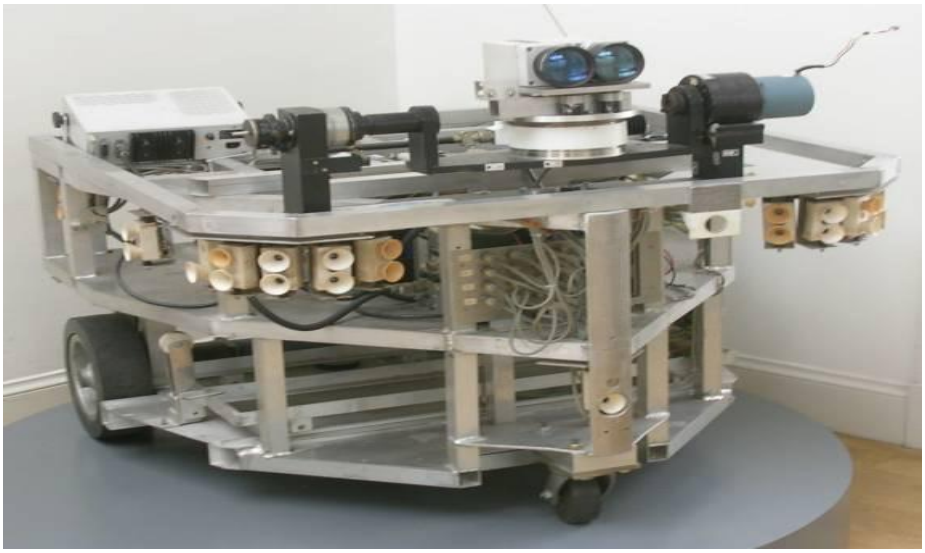


Figure 1.5: Robot Hilare du LAAS 1977 [95]

Une étape importante est à signaler au début des années 1990 avec la mise en avant de la robotique réactive, représentée par Genghis (Fig.1.6), développé par Rodney Brooks au MIT. Cette nouvelle approche de la robotique, qui met la perception au centre de la problématique, a permis de passer de gros robots très lents à de petits robots beaucoup plus réactifs et adaptés à leur environnement.

Figure 1.6: Genghis de Rodney Brooks 1990 [95]

3. Concepts de base

3.1. Définition d'un robot

Un robot est un système mécanique composé de corps mobiles reliés par des actionneurs qui lui donnent des capacités de mouvement dans l'espace physique .les structures mécanique utilisés sont de types très divers, qu'il s'agisse de robots manipulateurs constitués d'un bras terminé par un outil ou un organe de préhension (pince, main multi-doigts, etc.), ou encore de robots mobile avec des principes de locomotion adaptés a divers environnements (roues, chenilles, pattes, etc.) [3].

Enfin il existe également de nombreux autres types de robots mobiles (robots marins, sous-marins, drones volants, micro et nano robots), généralement l'étude de ce type de robot se fait dans des thématiques spécifiques avec des problèmes particuliers à l'application visée [1].

3.2. Exemples d'application

Aujourd'hui, le marché commercial de la robotique mobile est toujours relativement restreint. Mais, il existe de nombreuses perspectives de développement qui en feront probablement un domaine important dans le futur. Les applications des robots peuvent se trouver dans de nombreuses activités "ennuyeuses, salissantes ou dangereuses", mais également pour des applications ludiques ou de service, comme l'assistance aux personnes âgées ou handicapées. Parmi les domaines concernés, citons :

- La robotique de service (hôpital, bureaux)
- La robotique de loisir (Aibo, robot 'compagnon')
- La robotique industrielle ou agricole (entrepôts, récolte de productions agricoles, mines)
- La robotique en environnement dangereux (spatial, industriel, militaire)

Figure 1.7: Exemples de robots utilisés dans différentes applications.

3.3. Non-holonomie et holonomie

3.3.1. Non-holonomie

Les systèmes mobiles dit non-holonomes sont ceux que l'on rencontre le plus dans la vie courante (voiture particulière, bus, camion, …etc.). Ces systèmes ont une structure mécanique relativement simple (des roues motrices, des roues directrices et des roues libres). Une roue peut avoir une, deux ou trois fonctions. Mais tous ces systèmes ont une caractéristique commune : la direction de la vitesse d'entrainement (vitesse linéaire) est imposée par la direction des roues directrices.

3.3.2. Holonomie

Les systèmes holonomes sont beaucoup plus rares dans notre vie quotidienne. Ils ont une structure mécanique complexe qui leur permet de se déplacer dans toutes les directions sans manœuvre.

Dans le domaine des robots mobiles, le terme de robot mobile holonome est appliqué à l'abstraction appelée robot, ou base, sans tenir compte des corps rigides qui constituent le mécanisme proprement dit. Ainsi, tout robot mobile ayant une liberté de mouvement de trois degrés dans le plan est connu comme un robot mobile holonome, exemple le robot Nomadic XR4000 and PUMA 560 (figure 1.8) [4].

Figure 1.8: Robot holonome XR4000 and PUMA 560

3.4. Différence entre un AGV et un AMR [5]

- Un AGV (Figure 1.9) dispose d'une intelligence minimale et ne peut obéir qu'aux instructions programmées.

- Pour naviguer, il a besoin d'être guidé par des rails, des bandes magnétiques ou des capteurs, qui exigent l'installation d'équipements importants (et coûteux), durant laquelle la production peut être perturbée. L'AGV se restreint à suivre une route préétablie, qui, en cas de changements, nécessite une nouvelle intervention, donc des coûts additionnels et une désorganisation.

- L'AGV peut détecter les obstacles devant lui, mais n'est pas capable de les contourner, donc il s'arrête et ne pourra continuer que si l'obstacle est retiré.

- Au contraire, l'AMR navigue selon des cartes qui ont été créées dans son logiciel ou bien selon des parcours préenregistrés.

- Cette capacité peut être comparée à une voiture avec un GPS et une sélection de chemins préalablement répertoriés. Lorsque l'on renseigne le trajet entre son domicile et son travail, il calcule le circuit le plus court par rapport aux positions sur la carte, c'est de cette façon que le robot AMR fonctionne avec les emplacements où il doit enlever et déposer les pièces.

- L'AMR (Figure 1.10) utilise les données transmises par les caméras, les capteurs et les lasers comme un logiciel sophistiqué, ce qui lui permet de surveiller son environnement et de

choisir le parcours le plus efficace pour atteindre sa destination. Il travaille en totale autonomie et si un convoyeur, une palette, une personne ou tout autre obstacle se trouve sur son passage, l'AMR va manœuvrer en toute sécurité pour passer autour de celui-ci et utiliser la meilleure alternative de trajet possible. Cela optimise la productivité, tout en s'assurant que les flux de matériels restent programmés [5].

Figure 1.10: Autonomous Mobile Robot (AMR) [5]

Figure 1.9: Automated guided vehicle (AGV)

4. Architecture du système de commande

L'objectif principal d'un robot mobile consiste à réaliser un mouvement en reliant un point source à un point destination. L'exécution de cette tâche avec un certain degré d'autonomie nécessite l'utilisation d'un ensemble de ressources et d'une structure assurent une coopération efficace entre elles, les performances et capacités du robot dépendent à la fois de la qualité des ressources et à leur gestion [6].

4.1. Généralités sur l'architecture

L'architecture du robot décrit l'organisation des divers modules qui le compose ainsi que leurs interactions. Les modules sont à retirer en permanence ou encore pour une tâche définie à un instant donné. En général, l'informatique à bord est distribuée, permettant ainsi un fonctionnement indépendant et parallèle des modules. Nous pouvons décomposer un robot mobile en quatre niveaux [7] :

- Niveau interface homme-machine.
- Niveau planification de tâches.
- Niveau contrôle.
- Niveau fonctionnel.

4.1.1. Niveau interface homme-machine

Ce niveau d'architecture gère les communications avec le programmeur ou l'ordinateur de tâche. L'opérateur décrit selon le mode de communication (écran/clavier, écran/souris…) la tâche que doit exécuter le robot. Celle-ci sera soumise selon une forme adéquate aux générateurs de plan.

La communication machine-homme transite également par ce niveau. Des accusés de réception des résultats de traitement ou éventuellement des précisions sur la tâche demandée sont transférés vers l'homme selon toute forme de support visuel ou sonore. [7]

4.1.2. Générateur de plan ou planificateurs de tâche

À ce niveau de l'architecture se construit le plan de la tâche défini au niveau supérieur. Le générateur de plan va déterminer une suite séquentielle d'actions en utilisant les ressources disponibles et en définissant les modes d'activation. La génération de plan repose sur plusieurs notions:

- La connaissance de la situation initiale.
- La stabilité des connaissances en cours d'exécution.
- La connaissance parfaite de l'impact des actions.

Si ces trois notions sont valides, alors le plan d'action est généralement simple. La planification est nettement plus délicate lorsque une ou plusieurs des notions ci-dessus n'est plus vérifiée. Dans ce cas, le déroulement du plan ou encore l'élaboration du plan doit subir des modifications par rapport d'informations supplémentaires en cours d'exécution par reconnaissance de situation.

Cette dernière est établie selon deux types d'applications :

- Si le robot en télé-opérer, alors l'opérateur intervient pour interpréter les informations en sa possession et de prendre les décisions concernant la modification du plan.
- Si le robot possède une forte autonomie, alors la reconnaissance de situation consiste à identifier des propriétés courantes de l'environnement utiles à la mission.

Les réactions du robot autonome sont trois types : l'action réflexe, la récupération d'anomalies, la re-planification [8].

L'action réflexe consiste à faire appel à une ressource sans qu'elle soit prévue par le plan original. Cet appel est réalisé au vue d'une situation particulière. L'exécution de cette action à pour objet de se dégageait des situations locale en espérant se rapprocher au plan initial. L'exemple le plus typique et le contournement des obstacles.

La conséquence de l'introduction de cette action est double : soit la réaction permis de se dégager de problème, soit elle conduit vers une situation non récupérable. Le plan d'origine devient alors caduc.

La non-récupération du plan est souvent due à un manque de connaissance de la situation. Donc il est nécessaire d'utiliser toutes les ressources disponibles afin d'analyser la situation courante. Une bonne connaissance de la situation accroît fortement les chances de récupération de l'anomalie. On parle de récupération ou de planification locale.

La troisième méthode de réaction concerne la re-planification. Le robot se trouve dans le cas extrême ou le plan initial a échoué. Alors dans ce cas les actions réflexes et la récupération d'anomalies sont inopérantes. Si la situation courante est connue, il est possible de replanifier la mission [7].

4.1.3. Contrôleur d'exécution

Le contrôle d'exécution orchestre la mise en œuvre du plan établi au niveau du planificateur. Lors du déplacement, il adapte le comportement du robot aux situations instantanées. Ainsi, si les situations successives concordent avec le plan établi, celui-ci se déroule comme prévu, dans le cas contraire, le contrôleur applique la méthode convenue.

Les informations nécessaires à la réactivité, c'est-à-dire l'application des comportements face à une situation, sont issu d'un module de surveillance de l'environnement. Ce dernier se compose de plusieurs sous modules travaillant en parallèle.

Ce module a pour rôle de fournir un ensemble d'informations binaires ou numérique sur l'état de l'environnement, afin de garantir la sûreté des informations et l'exécution du comportement qui en est la conséquence, le contrôleur d'exécution possède un module proposant un diagnostic du fonctionnement global.

Ces informations permettent de réagir face à une situation donnée grâce aux potentialités matérielles et fonctionnelles disponibles à l'instant donné. Une telle opération est possible à condition d'avoir une certaine redondance des fonctionnalités du robot.

4.1.4. Niveau fonctionnel

Le niveau fonctionnel et le plus bas dans la hiérarchie de l'architecture. Il se compose de modules capteurs, effecteurs et actions de base. Les modules capteurs sont organisés en plusieurs niveaux : les éléments de détection, c'est-à-dire les capteurs physiques, l'élément de traitement de signal élémentaire.

L'utilisation d'information s'effectue selon les besoins. Pour l'évitement d'obstacles, l'information télémétrique est suffisante alors que la mobilisation dynamique trouvera plus d'informations dans un ensemble de droites.

Un module effecteur assure la réalisation du mouvement du robot. Comme pour le module précédent, celui-ci se décompose en plusieurs niveaux selon le degré de complexité de la consigne (ex : garder une orientation constante, suivre le chemin...).

Le module des actions de base se compose d'un ensemble de primitives dans l'usage est fréquent. Nous pouvons citer notamment l'évitement d'obstacles ou le suivi de murs (dans notre cas le suivi de trajectoire déjà planifié et l'évitement d'obstacle) [7].

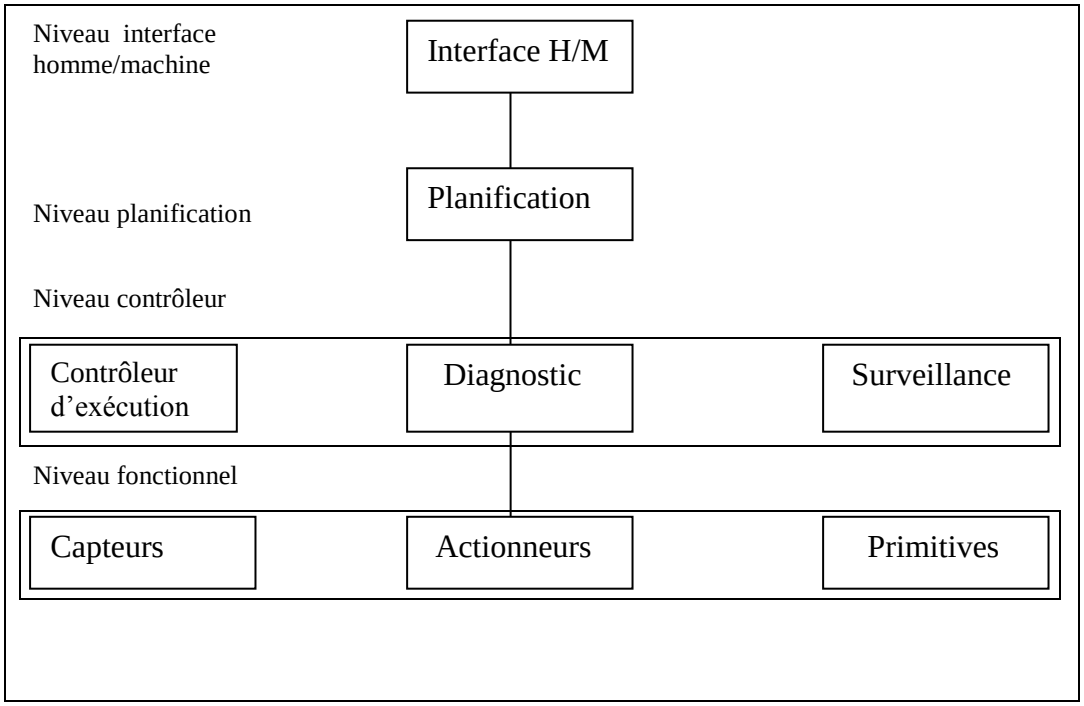


Figure 1.11: Architecture d'un robot mobile. Le block regroupe les différents niveaux d'un robot [7]

5. Systèmes de locomotion

Un système de locomotion assure deux fonctions :

- La propulsion.
- L'appui sur le milieu.

Les modes de propulsions sont variées. Cela s'étend du moteur thermique ou électrique. Le milieu dans lequel évolue le mobile oriente très fortement la mécanique mise en œuvre pour le mouvement.

5.1. Mobiles à roues

Les robots mobiles à roues sont les plus répondus actuellement. La raison est essentiellement la simplicité de conception du mécanisme. La plupart des robots mobiles opérationnels jusqu'à présent, évoluent sur sites aménagés : environnement intérieur ou sites industriels. Ces robots comportent, généralement, trois ou quatre roues. Dans les cas marginaux, ils peuvent prendre six roues voire plus. On peut distinguer plusieurs types de roues [7]:

- Roues motrices.
- Roues motrices directrices.
- Roues libres.
- Roues libres directrices.
- Roues folles.

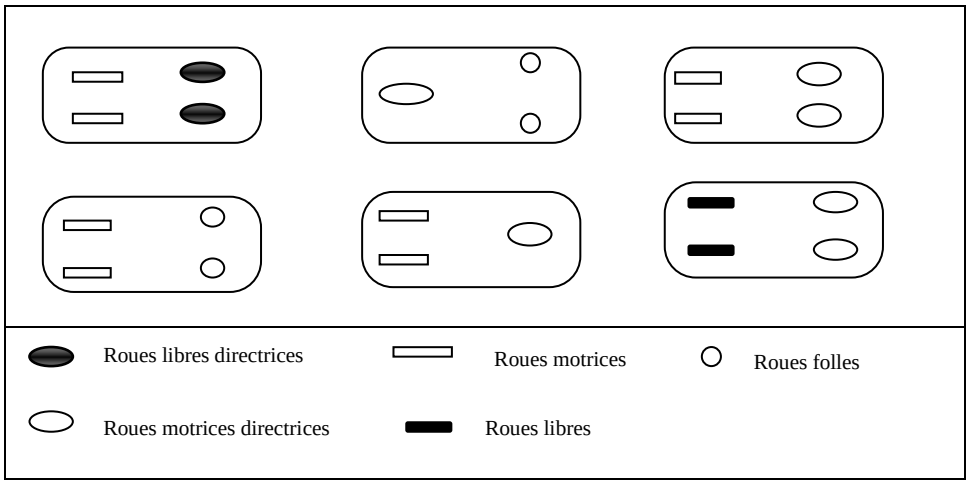


Figure 1.12: Les différentes configurations des roues possibles.

5.2. Mobiles à chenilles

Lorsque la configuration du terrain est plus chaotique, les engins à roues ont du mal à évoluer. Les chenilles sont alors plus performantes (Figure.1.13). Elles permettent d'augmenter l'adhérence au sol. La direction de conduite et définie en fonction de la différence entre les vitesses imposées aux chenilles [7].

Figure 1.13 : Les robots à chenilles.

5.3. Mobiles à pattes

Lorsque le terrain devient encore plus incertain, c'est-à-dire avec de grandes différences d'amplitudes comme un sol jonché de rochers, des engins à roues ou à chenilles ne sont plus efficaces. Dans ce contexte, les mobiles ayant des points d'appuis discrets (Figure.1.14) sont la solution au problème du mouvement. Cette solution est universelle, puisque la très grande majorité des animaux terrestres se meuvent de cette manière. Mais la réalisation d'un tel système et l'établissement de sa commande est complexe [7].

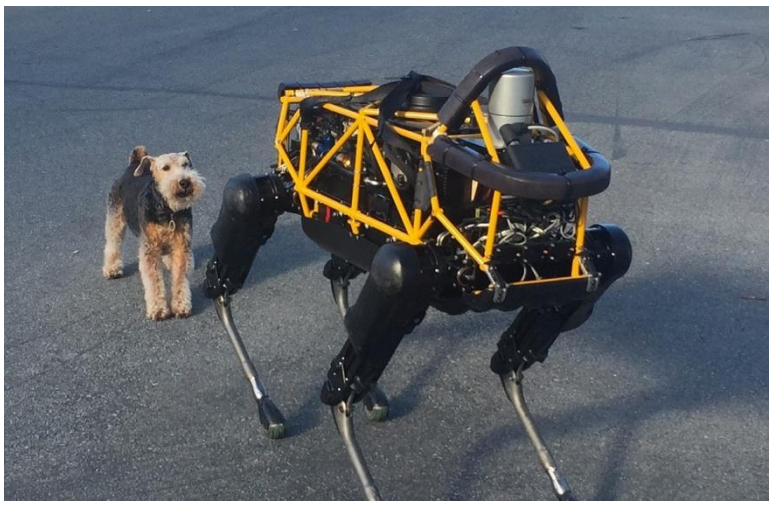


Figure 1.14: Le robot à pattes [102].

Plusieurs autres types de robots mobiles existent, tels que les robots volants (les drones), les robots sous-marins, …etc.

6. Environnement dynamique et incertain

6.1. 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 [7].

6.2. 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) [7].

7. Navigation

La navigation autonome des robots est la capacité à évoluer sans aide dans leur environnement de travail (espace de configuration). La complexité de la méthode de navigation mise en œuvre sur un robot mobile dépend donc de l'environnement dans lequel il doit évolue (milieu intérieur ou environnement naturel, sol plan ou irrégulier, environnement de travail,...). Elle dépend également de la connaissance de cet environnement qui peut être figé ou évolutif et du mode de définition de la trajectoire (apprentissage préalable, planification en ligne, ...).

Le problème de navigation d'un robot mobile consiste de la manière la plus générale à trouver un mouvement dans l'espace des configurations sans collisions, traditionnellement noté C_{free}. Ce mouvement amène le robot d'une configuration initiale $q_0 = q\ (t_0)$ à une configuration finale $q_f = q\ (t_f)$. On peut néanmoins donner des définitions différentes de la tâche de navigation à accomplir, selon le but recherché par exemple on peut souhaiter seulement placer le robot dans une zone donnée et relâcher la contrainte d'orientation, ...etc.

La tâche de navigation ainsi définie est donc limitée à un seul mouvement. Il existe néanmoins une très grande variété de travaux et de méthodes permettant d'aborder ce

problème difficile. Pour différencier les techniques de navigation, on peut de manière générale distinguer deux approches [10] :

- La première consiste à planifier le mouvement dans l'espace des configurations et à l'exécuter par asservissement du robot sur le mouvement de consigne (schéma planification exécution).

- La seconde consiste à offrir un ensemble de primitives plus réactives. Elles correspondent alors à des sous tâches (suivre un mur, éviter un obstacle) dont on estime que l'enchaînement est du ressort d'un planificateur de tâches ayant décomposé la tâche globale.

Les performances du système de navigation sont étroitement liées à la précision, à la fiabilité et au temps de réponse des capteurs et des méthodes mises en œuvre pour localiser le véhicule [11].

La navigation des robots mobiles s'articule autour de trois niveaux principaux [7] :

7.1. Localisation

Le caractère principal d'un robot mobile est la faculté de se mouvoir d'un point vers un autre. Pour ce faire, il est nécessaire d'avoir la connaissance de sa localisation par rapport à un espace de référence dans lequel sont définis les points source du but. De même, la poursuite d'une trajectoire prédéterminée suppose la connaissance instantanée de sa position. Nous pouvons parler de localisation statique lorsque le calcul de la position s'effectue à l'arrêt. La localisation dynamique est évaluée durant le mouvement.

Les outils permettant la localisation d'un robot dans son environnement peuvent être classes en deux catégories : ceux par localisation a l'estime et ceux par localisation absolue [12].

7.1.1. Localisation à l'estime (relative)

La localisation relative permet de déterminer la configuration courante du robot à partir de sa situation antérieure. Elle consiste à déterminer la position par l'intégration de mesure de vitesse ou d'accélération. Les capteurs fournissant ces données sont les odomètres ou gyromètres pour les mesures d'accélération.

Les odomètres sont présents sur l'ensemble des robots mobiles à roues. Cette technique peu coûteuse consiste à fixer sur les roues des codeurs délivrant une impulsion toutes les fractions de tour de roue. L'intégration de ces valeurs permet de déduire la position et l'orientation du mobile par rapport à l'initialisation des compteurs d'impulsion. Cette technique délivre des mesures peu fiables dans le temps.

Les techniques inertielles (accéléromètre, gyromètre) plus coûteuses constituent le haut de la gamme des systèmes de localisation à l'estime. La connaissance de la position référencée à l'origine du déplacement nécessite une double intégration de l'accélération. Ces calculs entraînent une inévitable accumulation d'erreurs qui constitue une dérive d'estimation dans le temps. Un recalage périodique est alors indispensable [13].

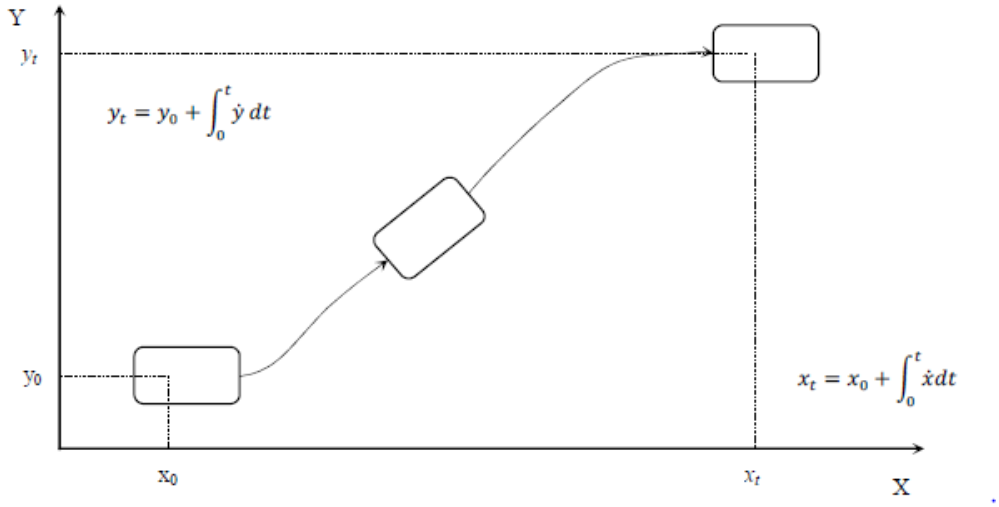


Figure 1.15: Localisation à l'estime

La technique de localisation à l'estime présente l'immense avantage d'être indépendante de l'environnement. Les seules erreurs qu'elle peut générer sont celles dues à son mode de fonctionnement interne. Par contre, l'inconvénient majeur est l'accumulation d'erreurs due aux différentes intégrations.

7.1.2. Localisation absolue

La localisation absolue est une technique qui permet à un robot de se repérer directement dans son milieu d'évolution, que ce soit en environnement extérieur (mer, espace, terre), ou en environnement intérieur (ateliers, immeubles, centrales nucléaires...).

Ces méthodes de localisation sont basées sur l'utilisation de capteurs extéroceptifs Pour répondre à la problématique qui est la localisation d'un robot dans son environnement, deux types de stratégies sont utilisables :

- La première consiste à utiliser des points de repère naturels.
- La deuxième à utiliser des points de repère artificiels.

Quelque soit le cas de figure, la localisation absolue nécessite toujours une représentation de l'environnement. Le robot possède donc «une banque de données» regroupant les caractéristiques des références externes qui est appelée carte de l'environnement.

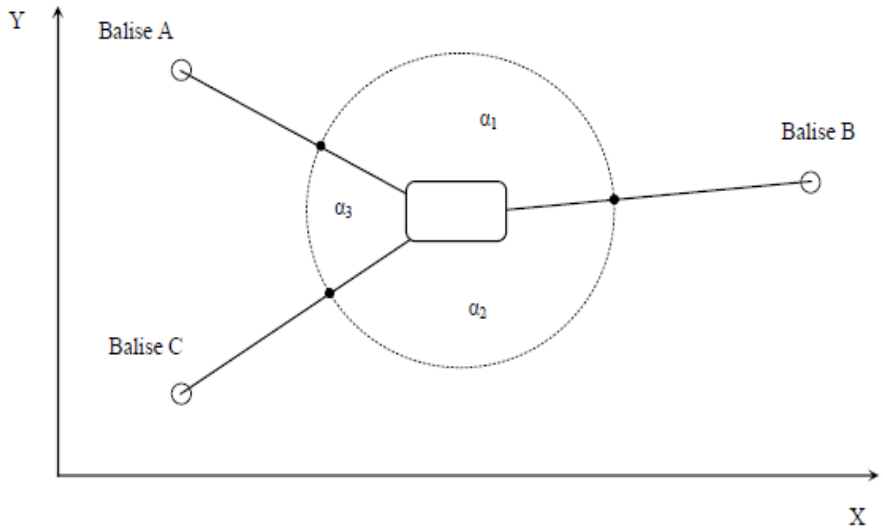


Figure 1.16: Localisation absolue (méthode par triangulation)

7.2. Perception

La notion de perception en robotique mobile est relative à la capacité du robot à recueillir, traiter et mettre en forme des informations qui lui sont utiles pour agir et réagir dans l'environnement qui l'entoure. Elle est donc la faculté de détecter et/ou appréhender l'environnement proche ou éloigné du robot. Alors que pour des tâches de manipulation on peut considérer que l'environnement du robot est relativement structuré, ce n'est plus le cas lorsqu'il s'agit de naviguer de manière autonome dans des lieux très partiellement connus. Aussi, pour extraire les informations utiles à l'accomplissement de sa tâche, il est nécessaire que le robot dispose de nombreux capteurs mesurant aussi bien son état interne que l'état de l'environnement dans lequel il évolue. Le choix des capteurs dépend bien évidemment de l'application envisagée.

La perception est nécessaire pour la sécurité du robot, la modélisation de l'environnement et l'évitement et le contournement d'obstacles.

7.2.1. Capteurs proprioceptifs

Les capteurs proprioceptifs fournissent par intégration des informations élémentaires sur les paramètres cinématiques du système mobile. Les informations sensorielles gérées dans ce cadre sont généralement des vitesses, des accélérations, des angles de giration, des angles d'altitude. Cependant, ils ne peuvent pas procurer de renseignements lors de l'arrêt du système mobile.

On peut regrouper les capteurs proprioceptifs en deux familles :

- Les capteurs de déplacement qui comprennent les odomètres, les accéléromètres et les adars Doppler. Cette catégorie permet de mesurer des déplacements élémentaires, des variations de vitesse ou d'accélération sur des trajectoires rectilignes ou curvilignes.
- Les capteurs d'attitude, qui mesurent deux types de données : les angles de cap et les angles de roulis et de tangage. Ils sont principalement constitués par les gyroscopes, les gyromètres, les gyrocompas, les capteurs inertiels composites, les inclinomètres et les magnétomètres. Ces capteurs sont en majorité de type inertiel [14].

7.2.1.1. Odomètres

Ces capteurs fournissent une estimation en temps réel de la position (x, y) et de l'angle φ d'un véhicule navigant sur un sol plan, par rapport au repère de référence qui était celui du véhicule dans sa configuration précédente.

7.2.1.2. Accéléromètres

Un accéléromètre est un capteur qui, fixé à un mobile ou tout autre objet, permet de mesurer l'accélération linéaire de ce dernier. On parle encore d'accéléromètre même s'il s'agit en fait de trois accéléromètres qui calculent les trois accélérations linéaires selon trois axes orthogonaux.

Par contre, lorsqu'on cherche à détecter une rotation ou vitesse angulaire, on parle de gyromètre. Plus généralement on parle de centrale inertielle lorsqu'on cherche à mesurer l'ensemble des six accélérations.

7.2.1.3. Radars Doppler

Un radar Doppler est un radar qui utilise l'effet Doppler-Fizeau de l'écho réfléchi par une cible pour mesurer sa vitesse radiale. Le signal micro-onde émis par l'antenne directionnelle du radar est réfléchi par la cible et comparé en fréquence avec le signal original allé et retour. Il permet ainsi une mesure directe et extrêmement précise de la composante vitesse de la cible dans l'axe du faisceau.

Les radars Doppler sont utilisés pour la défense aérienne, pour le contrôle du trafic aérien, pour la surveillance des satellites, pour les contrôles de vitesse sur route, en radiologie et dans les réseaux d'assainissement.

7.2.1.4. Gyroscope

Un gyroscope est un capteur de position angulaire et un gyromètre un capteur de vitesse angulaire. Le gyroscope donne la position angulaire (selon un, deux ou les trois axes) de son référentiel par rapport à un référentiel inertiel (ou galiléen).

7.2.2. Capteurs extéroceptifs

Les capteurs extéroceptifs sont employés en robotique mobile pour collecter des informations sur l'environnement d'évolution du système mobile. Ils sont le complément indispensable aux capteurs proprioceptifs présentés précédemment.

Des méthodes de fusion de données sont alors utilisées pour conditionner et traiter les informations sensorielles de natures différentes. Ils sont notamment utilisés dans les domaines d'application tels que l'évitement d'obstacle, la localisation, la navigation et la modélisation d'environnements.

Les principaux capteurs utilisés en robotique mobile sont : les capteurs télémétriques (les ultrasons, les lasers et les infrarouges), le GPS et les caméras.

7.2.2.1. Télémètres

Un télémètre est un appareil ou dispositif permettant par télémétrie de mesurer une distance.

La télémétrie est un procédé (technique) permettant de calculer ou de mesurer la distance d'un objet lointain par utilisation d'éléments optiques, acoustiques ou radioélectriques (un télémétrique laser, par exemple).

7.2.2.2. Lidar

Un LiDAR est un composant électronique qui fait partie de la famille des capteurs. Plus précisément, il fait partie de la catégorie des capteurs de temps de vol (ToF). Un capteur recueille des données sur un paramètre physique tel que la température, l'humidité, la lumière, le poids, la distance, etc.[96]

L'acronyme LiDAR signifie Light Detection And Ranging. Il s'agit d'une méthode de calcul qui permet de déterminer la distance entre le capteur et l'obstacle visé. Un LiDAR utilise un faisceau laser pour la détection, l'analyse et le suivi. La technologie Lidar est une technologie de télédétection qui mesure la distance entre le capteur et une cible. La lumière est émise par le LiDAR et se dirige vers sa cible. Elle est réfléchie sur sa surface et revient à sa source. Comme la vitesse de la lumière est une valeur constante, le LiDAR est capable de calculer la distance le séparant de la cible [96].

Figure 1.17 : Le RPLIDAR A3M1 de Slamtec [96]

7.3. Raisonnement et décision

Ce niveau consiste l'intelligence du robot. À l'instar de l'home, le raisonnement du robot lui permet de décider d'une action appropriée à une situation donnée, compte tenu d'une mission à réaliser.

Plusieurs tâches élémentaires peuvent être exécutées en parallèle pour synthétiser un comportement. La façon dont elles interagissent est définie par l'architecture décisionnelle du robot.

Cette décision est ensuite transmise au niveau fonctionnel pour opérer les différentes parties qui satisfait cette demande.

8. Thématiques scientifiques de 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 spécifique des robots mobiles autonomes est entier. La communauté des chercheurs dans le domaine de la robotique à dégage 4 grands axes de travail autour desquels s'articulent les recherches actuelles en robotique mobile [12]:

- 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 nécessite de coupler diverses sources de mesures apparait 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-véhicule : on retrouve ici tous les travaux lies 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.

9. Conclusion

Ce chapitre a passé en revue les notions de base de la robotique mobile, quelques types d'applications ont été présentés. Ainsi on a concentré sur l'architecture d'un robot mobile et ses différents stratégies de navigation, Certains points sont mentionnés plus en détail que d'autres, car ils sont plus étroitement liés au présent projet.

Donc La perception d'une part et la commande 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 qui tient un rôle important, elle consiste à déterminer les trajectoires à suivre par le robot pour évoluer correctement dans des milieux encombré d'obstacles.

Chapitre II : État de l’art sur la planification de trajectoire

1. Introduction

Notre travail s'intéresse à la navigation d'un robot mobile dans un environnement de travail dynamique. Soit un système robotique équipé de capteurs lui permettant de percevoir son environnement et d'actionneurs lui permettant de se déplacer. Notre but consiste alors à déterminer un mouvement du robot qui:

- Respecte les contraintes sur le mouvement du système robotique.
- Mène à un but prédéterminé.
- Evite toutes collisions avec les obstacles au cours de la navigation.

La détermination du mouvement pour un robot autonome a été largement abordée au cours de ces dernières années [66].

L'objectif de ce deuxième chapitre est de donner un bref état de l'art de la robotique mobile. Les deux modèles de robots mobiles à roues rencontrés le plus fréquemment sont d'abord présentés, la deuxième partie présente une synthèse sur les différentes techniques de planification de trajectoire là ou distingue habituellement deux grandes familles d'approches : les approches délibératives et les approches réactives. Le principe des approches délibératives est de calculer au préalable un chemin ou un ensemble de trajectoires entre une position initiale et une position finale à partir de la connaissance a priori de l'environnement dans lequel évolue le robot. Les approches réactives utilisent l'environnement perçu afin de générer un mouvement à exécuter sur un pas de temps, à la suite duquel un autre mouvement est généré et ainsi de suite. Ces mouvements sont appelés ‘‘mouvements partiels’’. Une représentation de l'environnement est ainsi construite au fur et à mesure du déplacement : la navigation est donc possible en environnement incertain comme en environnement dynamique. Ainsi les principales stratégies d'évitement d'obstacles fixes et/ou mobiles sont abordées.

Il existe différents problèmes de commande pour le robot mobile [16] :

- Le suivi de chemin où l'objectif est qu'un point lié au robot suive une courbe prédéterminée en imposant au robot une vitesse donnée ;
- La stabilisation de trajectoires consistant à prendre en compte la dimension temporelle : la trajectoire de référence dépend du temps et la vitesse du robot n'est plus fixée à l'avance ;
- La stabilisation de configurations fixes : il s'agit de stabiliser asymptotiquement le système dans une position d'équilibre donnée.

Dans ce mémoire, nous nous intéressons uniquement au problème de la poursuite de trajectoire pour un robot mobile non holonome de type unicycle. Un état de l'art sur ce sujet est donné dans la dernière section de ce chapitre.

2. Modélisation et propriétés structurelles

Il existe de nombreux modèles de robots mobiles selon leur structure, le type de roues utilisées, etc. Nous ne présentons ici que les deux classes de modèles les plus fréquemment rencontrés : les robots de type unicycle et ceux de type voiture [16].

2.1. Hypothèses

Généralement pour la commande de robots mobiles, un modèle de commande en vitesse est utilisé plutôt qu'un modèle de commande en couple. Les principales raisons de ce choix sont les suivantes :

- Le calcul de la commande est plus simple pour un modèle cinématique que pour un modèle dynamique.
- Il n y a pas de paramètres géométriques ou inertiels compliqués à identifier pour un modèle cinématique.

Enfin, dans le cas des robots mobiles miniatures utilisés dans notre application (robot unicycle), l'inertie est faible et la dynamique de la motorisation électrique est très rapide.

Pour ces raisons, nous ne considérons dans la suite que des modèles cinématiques en prenant en compte les hypothèses simplificatrices suivantes :

- le robot mobile est considéré comme un véhicule rigide évoluant dans un plan horizontal ;
- Les roues conventionnelles sont supposées indéformable, de rayon notée ci-dessous r ;
- chaque contact roue/sol est réduit à un point ;
- les roues roulent sans glisser sur le sol.

Nous restreignons notre étude aux robots mobiles de type unicycle et de type voiture.

2.2. Modèle cinématique du robot de type unicycle

La cinématique des robots mobiles pose des problèmes nouveaux par rapport à celle des robots manipulateurs [3]. Le robot mobile utilisé dans notre application est un robot mobile de type unicycle. Il est composé de deux roues motrices indépendantes à l'arrière du corps

principal du robot et une roue non commandée à l'avant (roue folle), Cette dernière est destinée uniquement à assurer la stabilité du robot (figure 2.2).Il existe d'autres robots qui peuvent être classés dans la catégorie des robots mobiles de type unicycle, notamment le robot mobile avec deux roues motrices indépendantes au milieu du corps principal du robot (figure 2.1).

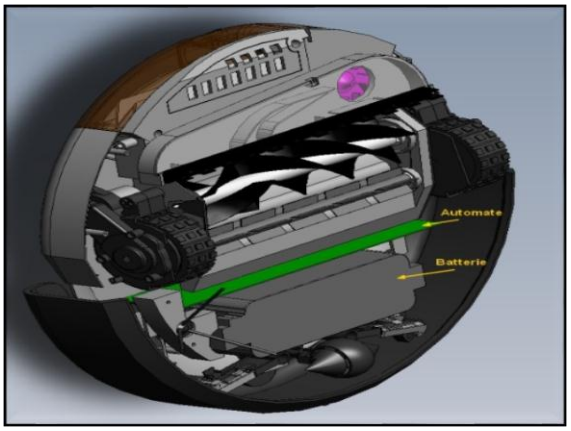


Figure 2.1 : Robot mobile avec deux roues motrices à l'arrière et une roue non commandée à l'avant [98].

Figure 2.2 : Robot mobile avec deux roues motrices indépendantes au milieu du corps principal [97].

Le modèle d'un robot mobile unicycle correspond à celui d'une roue roulant dans un plan [17]. La configuration de l'unicycle est caractérisée par la donnée des nombres *x, y,* θ et ϕ , où *x* et *y* dénotent les coordonnées cartésiennes du point de contact de la roue avec le sol dans un repère cartésien donné, θ l'orientation de la roue et ϕ l'angle de la roue mesurée à partir de la verticale (figure 2.3). Sachant que le roulement est sans glissement le modèle cinématique de la roue s'exprime par les formules suivantes [16]:

$$\begin{cases} \dot{x} - r\dot{\phi}\cos\theta = 0 \\ \dot{y} - r\dot{\phi}\sin\theta = 0 \end{cases} \qquad (1.1)$$

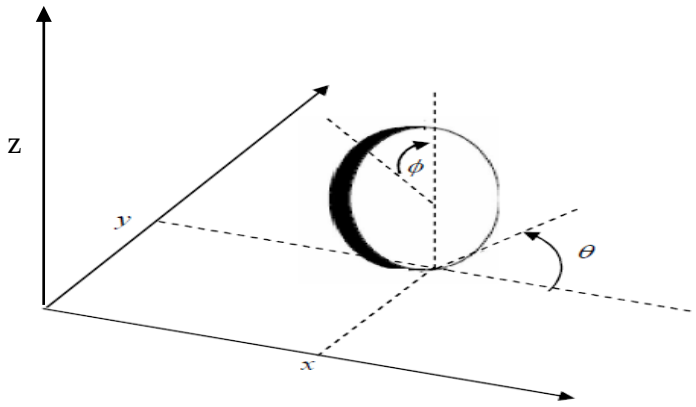


Figure 2.3 : Caractérisation du roulement sans glissement

Où *r* représente le rayon de la roue.

Le modèle (1.1) peut être transformé pour faire apparaître les composantes des vitesses dans les plans de la roue et perpendiculaire à la roue, les contraintes cinématiques suivantes sont alors obtenues :

$$\begin{cases} \dot{x}\sin\theta - \dot{y}\cos\theta = 0 \\ \dot{x}\cos\theta + \dot{y}\sin\theta = r\dot{\phi} \end{cases} \quad (1.2)$$

Ces deux équations sont des contraintes de type non holonome, signifiant que l'on ne peut pas les intégrer de façon à ne faire apparaître que les coordonnées généralisées. Cela peut se démontrer simplement à l'aide du théorème de Frobenius /Bloch 2003/ /Warner 1983/ Nijmeijer 1990/, [18], [19], [20].

En notant la vitesse longitudinale de l'unicycle avec $v = r\dot{\phi}$, et $w = \dot{\theta}$, sa vitesse angulaire, on déduit de (1.1) le modèle cinématique du robot mobile non-holonome de type unicycle :

$$\begin{cases} \dot{x} = v\cos\theta_0 \\ \dot{y} = v\sin\theta_0 \\ \dot{\theta} = w \end{cases} \quad (1.3)$$

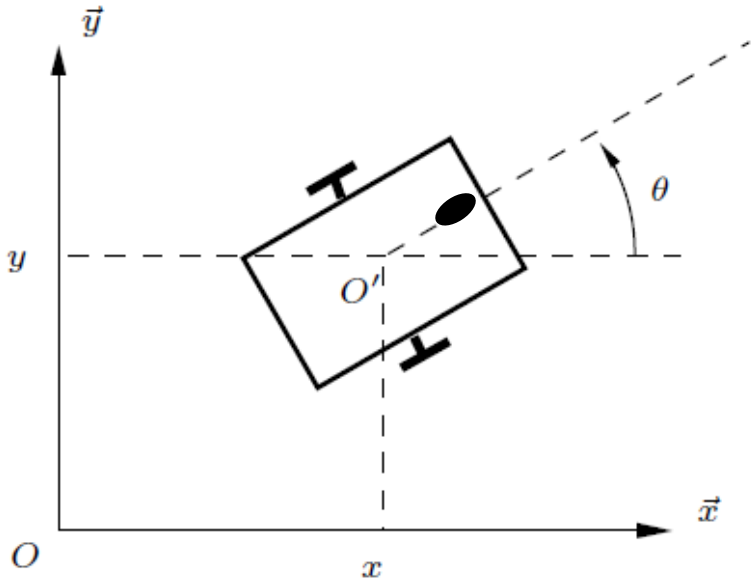


Figure 2.4 : Robot mobile de type unicycle

2.3. Modèle cinématique de type voiture

Un robot mobile de type voiture est composé d'un train moteur à l'arrière du corps principal et de deux roues de direction et à braquage différentiel à l'avant (figure 2.5). La configuration du véhicule de type voiture peut être représentée par le quadruplet $q= [x, y, \theta, \phi]$, où le point de cordonnées (x, y) est le centre de l'essieu arrière de la voiture, θ est l'orientation du véhicule, ϕ l'orientation des roues avant et d la distance entre les essieux avant et arrière.

Les conditions de roulement sans glissement s'obtiennent en écrivant que les vitesses latérales des roues avant et arrière sont nulles [16]:

$$\begin{cases} \dot{x}\sin\theta - \dot{y}\cos\theta = 0 \\ \dot{x}\sin(\theta+\phi) - \dot{y}\cos(\phi+\theta) - d\cos\phi\theta = 0 \end{cases} \quad (1.4)$$

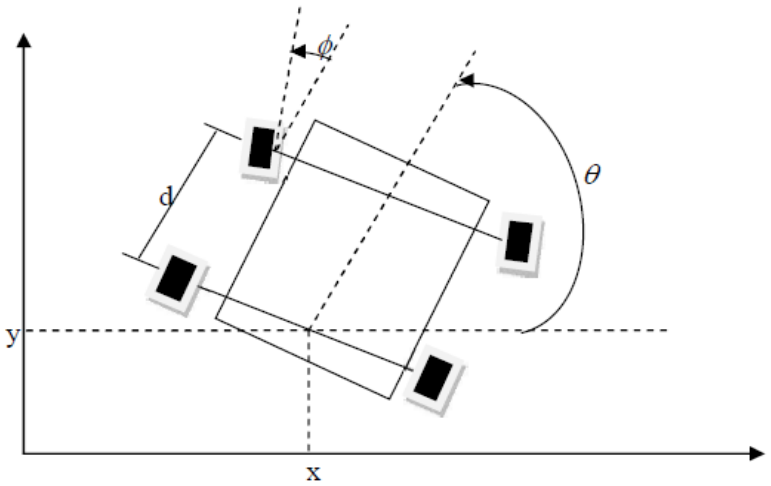


Figure 2.5 : Robot mobile de type voiture

Ces contraintes sont non intégrables, le véhicule de type voiture est donc non holonome. En utilisant des transformations sur l'équation (1.4) [17], le modèle cinématique d'un robot mobile non-holonome de type voiture est obtenu :

$$\begin{cases} \dot{x} = u_1\cos\theta \\ \dot{y} = u_1\sin\theta \\ \dot{\theta} = u_1\dfrac{tan\phi}{d} \\ \dot{\phi} = u_2 \end{cases} \quad (1.5)$$

Où u_1 correspond à la vitesse longitudinale du corps du robot, alors que u_2 correspond à la vitesse angulaire des roues directrices par rapport au corps du véhicule.

2.4. Commandabilité

Le système linéarisé du modèle (1.3) autour d'un point d'équilibre arbitraire (x_0, y_0, θ_0) est donné par les équations suivantes :

$$\begin{cases} \dot{x} = v \cos\theta_0 \\ \dot{y} = v \sin\theta_0 \\ \dot{\theta} = w \end{cases}$$

Le critère de Kalman montre que ce système n'est pas (complètement) commandable. Par contre, le système initial l'est. En effet, le modèle cinématique du robot mobile (1.3) peut être réécrite sous la forme suivante :

$$\dot{q} = X_1(q)\, v + X_2(q)\, w \tag{1.6}$$

$Ou\ q = [x, y, \theta]^T, X_1(q) = [cos(\theta), sin(\theta), 0]^T et\ X_2(q) = [0,0,1]^T$

Par simple calcul du crochet de Lie des champs de vecteurs X_1et X_2, on montre que :

$$[X_1(q),\ X_2(q), [X_1(q),\ X_2(q)]] = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ \sin(\theta) & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{1.7}$$

3. Planification de trajectoire

Le problème de la planification de trajectoire est généralement formulé de la manière suivante [16] : on considère un robot mobile ***A*** se déplaçant dans un espace de travail ***W***, l'objectif est de trouver les chemins qui relient la position du départ du robot $\boldsymbol{P_i}$ à sa position finale $\boldsymbol{P_f}$. Ces chemins doivent être réalisables par le robot, c'est-à-dire, respecter les contraintes cinématiques du robot.

Les chemins doivent aussi éviter des obstacles fixes ou mobiles. Dans cette section, les contraintes cinématiques à prendre en compte dans la planification sont présentées, puis les principales approches pour définir des trajectoires acceptables sont développées.

3.1. Notion de trajectoire

Jusqu'à présent, il n'a été question que de planification de chemin ; un chemin étant une courbe continue de l'espace des configurations du robot. On ne s'est pas préoccupé de l'indexation temporelle de cette courbe, de la façon dont le robot allait se déplacer au cours du temps le long de ce chemin. Ce problème particulier est abordé dans le cadre de la planification de trajectoire. Une trajectoire est une fonction continue du temps qui spécifie à chaque instant la configuration du robot. L'introduction de la dimension temporelle permet de traiter un spectre beaucoup plus large de problèmes de planification de mouvement. En particulier, ceci permet de considérer un espace de travail encombré d'obstacles mobiles et ensuite, cela autorise la prise en compte de contraintes de la nature dynamique auxquelles peut être soumis le robot (force, accélération, vitesse). Dans le cadre de la planification de trajectoire, il est, en général, plus pertinent d'optimiser la durée d'une trajectoire plutôt que sa longueur. Dans la majorité des cas, on recherche donc la solution qui est optimale en temps et non en distance comme la planification de chemin [21].

3.2. Approches délibératives

Les approches dites délibératives consistent à résoudre un problème de planification de mouvement. La planification de mouvement est la d'extermination a priori d'une stratégie de mouvement entre une position initiale et une position finale du robot à partir d'une représentation de l'environnement dans lequel il évolue.

Le problème étant pose, nous présentons ici un bref aperçu des principales approches délibératives ayant retenu notre attention. Elles peuvent principalement être regroupées en deux catégories :

- méthodes d'exploration d'un graphe de recherche.
- méthodes incrémentales de construction d'un arbre de recherche.

Celles-ci sont d´écrites ci-dessous

3.2.1. Méthodes par graphes

Le principe de ces méthodes est de tenter de capturer la topologie de l'espace de recherche (espace de configuration ou espace d'état du robot) dans le but de simplifier le problème à une recherche dans un graphe. Elles sont donc constituées de deux étapes :

- Construction du graphe dans l'espace de recherche approprie.
- Parcours du graphe dans le but de déterminer un chemin ou une trajectoire entre les configurations initiale et finale.

Le parcours du graphe s'effectue la plupart du temps de la même manière : un Algorithme heuristique est utilisé dans le but d'éviter l'exploration complète de l'espace de recherche [22-23]. La construction du graphe peut, quant à elle, fortement varier : alors que les premiers travaux de planification s'intéressaient à trouver un chemin pour des systèmes dépourvus de contraintes sur leur mouvement ou pouvant se déplacer dans toutes les directions (systèmes holonomes), les recherches actuelles prennent en compte les contraintes cinématiques et dynamiques des robots étudiés et planifient des trajectoires dans l'espace-temps afin de considérer le mouvement futur des obstacles mobiles. La représentation de l'espace de recherche, s'est en conséquence adaptée à ces évolutions [24-25].

Nous présentons ci-dessous les principales méthodes de représentation de ces espaces de recherches pour des systèmes et environnement de plus en plus contraints.

3.2.1.1. Graphe de visibilité

C'est la première méthode de planification de chemin connue, elle consiste à relier chaque sommet des enveloppes convexes d'obstacles polygonaux à tout autre sommet visible de cet ensemble (Figure 2.6). On obtient ainsi un graphe dans lequel on peut effectuer une planification après avoir relie les positions initiale et finale aux sommets de cet ensemble les plus proches. Notez que cette technique autorise les configurations de contact entre le système mobile **A** et les obstacles, c'est l'une des raisons pour lesquelles elle est relativement peu utilisée [26].

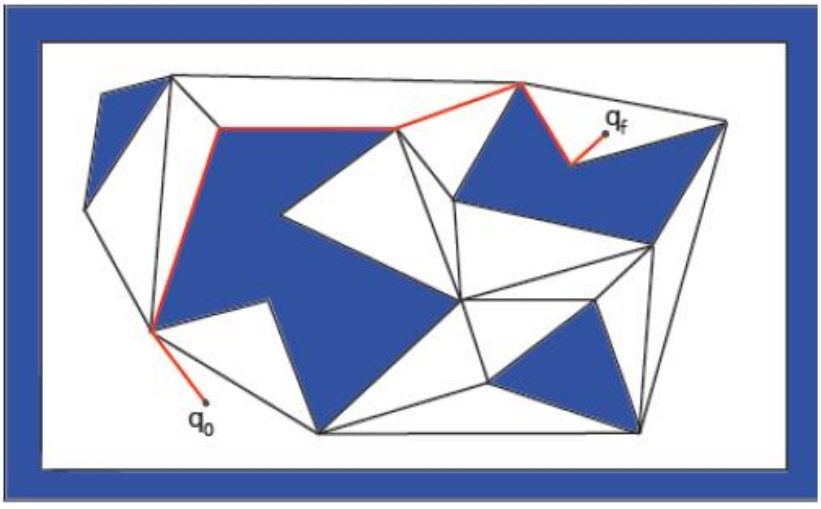


Figure 2.6: Chemin déterminé entre deux configurations à partir d'un graphe de visibilité

3.2.1.2. Décomposition en cellules

Une autre approche consiste en une décomposition cellulaire de l'espace de configuration. A partir d'une représentation simple de l'espace de configuration, cette approche consiste à diviser cet espace en un nombre fini de sous-espace convexes. Par exemple, dans un espace de configuration de dimension 2 dont la représentation des obstacles est polygonale (Figure.2.7), une décomposition cellulaire verticale [27] ou par triangulation [28] peut être facilement calculée. Un graphe est alors construit comme suit : les nœuds sont définis aux barycentres des cellules obtenues et au milieu de leurs côtés. Une arête relie ensuite chaque nœud défini sur un côté au barycentre des deux cellules adjacentes. Comme précédemment, en reliant la configuration initiale q_0 avec la configuration finale q_f aux nœuds du graphe les plus proches, une recherche heuristique dans le graphe résultant permet de trouver un chemin liant ces deux configurations.

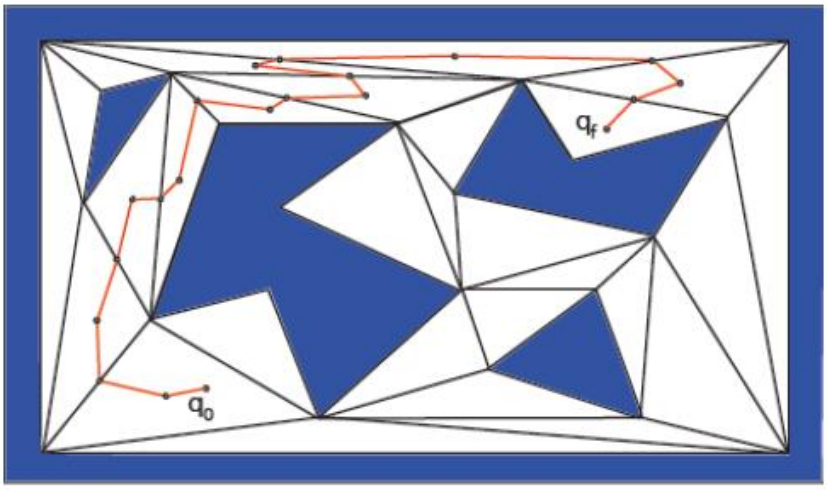


Figure 2.7 : Chemin déterminé entre deux configurations q_0 et q_f à partir d'une décomposition cellulaire.

Des méthodes de décomposition cellulaires plus complexes telles les balles connectées ont été conçues pour des environnements non structures et des espaces de recherche (configurations ou états) de plus fortes dimensions [29-30]. En couvrant L'espace de recherche libre par des sphères se chevauchant de la dimension de cet espace, il est possible comme précédemment de construire un graphe connexe dont une arête est définie entre chaque couple de sphères dont l'intersection est non vide.

Bien qu'attirantes par leur simplicité, les méthodes combinatoires présentées ci-dessus sont en pratique très peu souvent utilisables: pour un système robotique peu complexe (forte dimensionnalité de l'espace de configuration, systèmes redondants, topologie des obstacles

quelconque), une représentation des obstacles C_{obs} dans l'espace de configuration est difficilement accessible, et leur calcul explicite est bien trop complexe. De plus, la géométrie du graphe ne se soucie guère des contraintes sur le mouvement du système. Un robot disposant par exemple de contraintes non holonomes sera absolument incapable de suivre les arêtes du graphe sans une adaptation du plan. Des méthodes alternatives ont alors été apportées afin d'éviter la caractérisation de C_{obs}.

3.2.1.3. Echantillonnage par grilles régulières

Lorsque la complexité du système robotique **A** est telle qu'il est difficile de déterminer la topologie de l'espace de configurations **C** (et l'espace C_{obs} des configurations du robot en collision avec un obstacle de l'environnement), une méthode alternative consiste a discrétiser **C** et a construire une approximation conservative de l'espace libre $C_{libre} = C \setminus C_{obs}$. Pour ce faire, un échantillonnage de l'espace par une grille régulière de dimension **n** peut être effectue. Le graphe **G** en résultant est construit en définissant un nœud pour chaque échantillon et en le liant par une arête à chacun de ses **2n** voisins directs. Une planification peut alors être effectuée entre deux configurations q_0 et q_f en assimilant chacune de ces configurations a un nœud du graphe **G**, puis en explorant itérativement a partir de q_0 ses nœuds voisins jusqu'à atteindre q_f. La détermination de l'obstruction des nœuds et des arêtes du graphe par les obstacles peut ainsi être effectuée uniquement lors de leur exploration par un module de vérification de collision (souvent considéré comme une boite noire pour ce genre d'approches). L'espace libre atteignable par une telle grille est un sous-espace de C_{libre} La garantie de trouver une solution s'il en existe une est donc amoindrie et dépendante de la résolution de la grille.

Afin d'accélérer le processus de recherche sur le graphe, une représentation de l'environnement par grilles multi résolution fut proposé par la suite [31-32]. Une recherche de chemin peut alors être effectuée en premier lieu à basse résolution. Si aucune solution n'est trouvée, les résolutions supérieure sont examinées jusqu'à ce qu'une solution ait été trouvée.

Cet échantillonnage par grilles a également été la source de représentation des espaces libre C_{libre} et obstrue C_{obs} par grilles d'occupation probabilistes. Une grille d'occupation probabiliste est une grille représentant une discrétisation régulière de l'espace dans laquelle chaque cellule est caractérisée par une probabilité d'occupation évaluée lors de la navigation par les capteurs du système robotique. La vérification de collision en chaque nœud du graphe induit est alors effectuée en estimant la probabilité de collision de la cellule de la grille

probabiliste associée. L'avantage de telles grilles est sa possibilité de mise à jour au cours du temps; lorsqu'un obstacle mobile se déplace ou lorsqu'un nouvel obstacle est détecte par les capteurs du système, les probabilités d'occupation des cellules de la grille associées sont réévaluées. Le graphe induit peut en conséquence être modifie, mais s'il désactive un nœud ou une arête choisis lors de la planification, un nouveau mouvement doit être replanifier.

3.2.1.4. Cartes de routes probabilistes

Dans le cas d'espace de recherche de forte dimension, une discrétisation régulière de l'espace peut s'avérer bien trop coûteuse. Afin de diminuer la taille du graphe sur cet espace, une solution consiste à construire une carte de route probabiliste (Probabilistic roadmap [33] sur celui-ci. La construction d'une carte de route probabiliste s'effectue en sélectionnant des configurations aléatoires dans l'espace de recherche (Figure2.8).

Chaque nouvelle configuration est ajoutée au graphe si elle n'est pas en collision avec les obstacles de l'environnement. Dans ce cas un nouveau nœud est créé dans le graphe pour celle-ci. Des arêtes sont alors ajoutées à partir de cette configuration vers d'autres configurations appartenant déjà à la carte de route, dans le cas où ces deux configurations sont suffisamment proches, et qu'il existe un chemin libre de collisions entre celles-ci. La construction de la roadmap se termine généralement lorsqu'un nombre prédéterminé de nœuds constituent celle-ci.

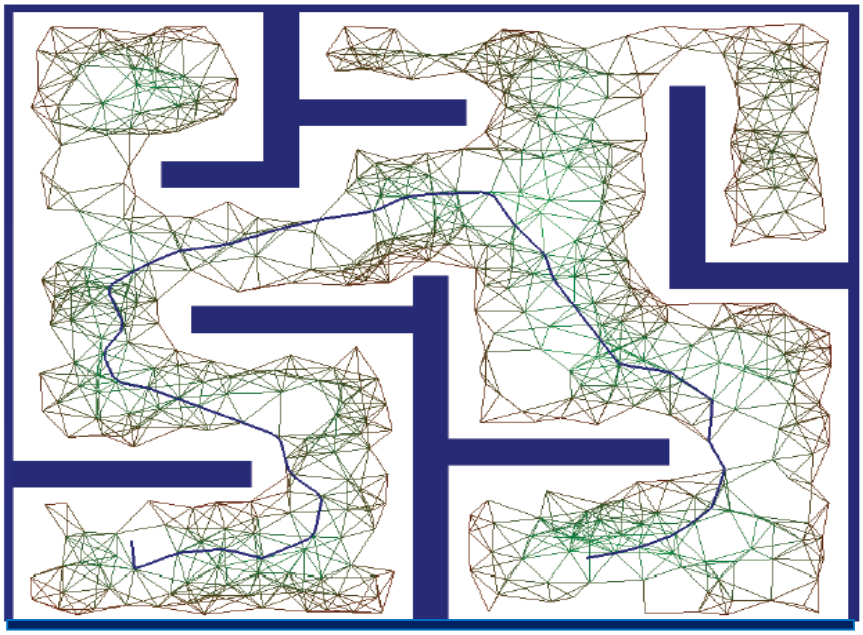


Figure 2.8: Chemin planifié pour un robot de type différentiel à partir d'une roadmap calculée dans l'espace de configuration du robot mobile.

La qualité de telles approches est alors évaluée par rapport à deux critères : sa densité et sa dispersion. La dispersion reflète la taille maximale de l'espace libre $\mathbf{C}_{\mathbf{libre}}$ non couverte par les nœuds composant la carte de route probabiliste. La densité s'illustre par son aptitude à s'approcher aussi près que possible de toutes les configurations de l'espace libre Clibre. De nombreuses extensions de ce type d'approches ont vu le jour afin d'optimiser ces deux paramètres tout en limitant le nombre de nœuds nécessaires.

Tout d'abord, [34] proposent un échantillonnage aux bornes de $\mathbf{C}_{\mathbf{libre}}$ afin de pouvoir déterminer simplement un chemin vers une configuration proche des obstacles de l'environnement. Pour cela, lorsqu'une configuration aléatoirement choisie se trouve en collision, elle est déplacée aléatoirement, puis une fois libre, elle est autant que possible connectée au reste du graphe. A l'opposé, [35] essaient de maximiser l'espace libre autour de chaque nœud du graphe.

Dans [33], une probabilité de distribution est associée à chaque nœud du graphe. Lors de l'insertion d'un nouveau nœud, son positionnement est optimisé afin de maximiser la probabilité de distribution sur l'ensemble de l'espace libre $\mathbf{C}_{\mathbf{libre}}$ et de pouvoir atteindre ainsi de nouvelles configurations non visibles jusqu'alors.

Une carte de route par visibilité (visibility roadmap) est proposée en [37] et [36]. Pour celle-ci, deux types de nœuds sont définis : les gardes et les connecteurs. Un nœud est un garde s'il n'est visible dans $\mathbf{C}_{\mathbf{libre}}$ par aucun autre garde. Un connecteur est un nœud visible par au moins deux gardes. En construisant un graphe connecté tel que toute région de $\mathbf{C}_{\mathbf{libre}}$ soit visible par un garde et que le nombre de nœuds total soit minimal, on obtient une carte de route de faible taille (limitant ainsi le temps de recherche) et pouvant aisément connecter n'importe quel couple de nœuds de $\mathbf{C}_{\mathbf{libre}}$.

Enfin dans le cas de navigation en environnement dynamique, [38] propose une mise à jour dynamique de la carte de route probabiliste. En présence d'obstacles mobiles, certains nœuds et arêtes sont ainsi désactivés ou réactivés afin d'adapter la planification aux mouvements des obstacles sans devoir reconstruire la carte intégralement.

3.2.1.5. Méthodes de re-planification

En présence d'obstacles mobiles, certaines méthodes présentées ci-dessus permettent une adaptation du graphe sur lequel le chemin ou la trajectoire du robot est planifiée, par exemple en déplaçant, activant ou désactivant des nœuds et des arêtes du graphe. Dans le cas ou le plan

initial passait par un nœud ou une arête n'étant plus valide, il est alors nécessaire de replanifier tout ou partie du mouvement suivi. Pour cela, diverses méthodes ont été proposées: Stentz [39] a initialement propose une extension de l'algorithme de recherche heuristique dans le cadre de la navigation de robots mobiles. Lors de l'invalidation d'un plan, celui-ci replanifier localement la partie du plan obstruée par les obstacles. Likhachev et al.[40] ont proposé une extension ''anytime" de l'algorithme : à chaque instant, le robot effectue une recherche le plus loin possible vers le but en un temps de décision constant.

D'autres méthodes de re-planification ont vu le jour modifiant une partie du plan obstrue en cas d'obstacles inattendus et améliorant le chemin suivi lorsque le temps le permet [41].

3.2.2. Méthodes par arbres

En parallèle de la planification classique par exploration d'un graphe de recherche sont apparues les méthodes par arbres. Celles-ci consistent, à partir de la configuration initiale du système, à construire un arbre se développant dans toutes les directions autour du robot dans l'espace de recherche. Elles sont donc bien adaptées dans le cas d'espaces de recherche à forte dimensionnalité.

3.2.2.1. Exploration rapide des arbres aléatoires (RRT)

Elle représente certainement l'une des approches de planification de mouvement les plus répandues à l'heure actuelle [42]. A partir d'une configuration initiale $\mathbf{q_0}$, l'espace de configuration du système est exploré en choisissant aléatoirement à chaque itération une nouvelle configuration $\mathbf{q_{nv}}$ non obstruée par les obstacles vers laquelle se diriger. La branche la plus proche de l'arbre déjà construit est alors déterminée puis étendue en direction de $\mathbf{q_{nv}}$. En répétant le processus, l'espace de recherche est alors rapidement couvert et un chemin vers toutes configurations de cet espace peut alors être facilement déterminé s'il en existe un (Figure 2.9). Le but de la planification étant néanmoins d'atteindre une configuration finale $\mathbf{q_f}$, le processus essaie de déterminer un chemin liant la configuration la plus proche de l'arbre à $\mathbf{q_f}$ après un certain nombre d'itérations de l'expansion de l'arbre.

Dans le cas où le système évolue en environnement dynamique, une extension "anytime" des RRTs a également été proposée [43] : l'arbre de recherche est mis a jour progressivement, et à chaque pas de temps le contrôle à appliquer est déterminé à partir de la racine de l'arbre menant vers la position se rapprochant le plus près du but.

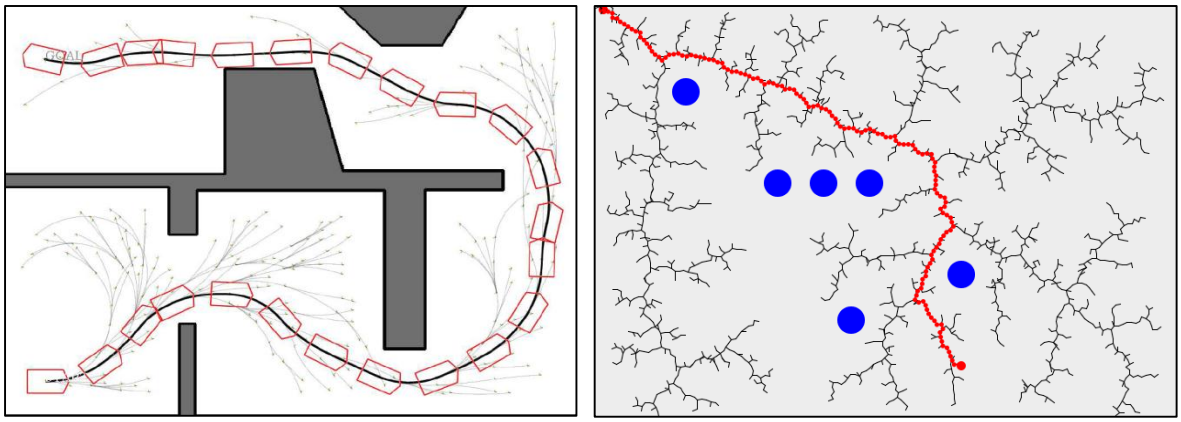


Figure 2.9 : Rapidly Exploring Random Trees : Etapes successives de construction de l'arbre de recherche dans l'espace des configurations du système robotique mobile.

3.2.2.2. Fil d'Ariane

Le fil d'Ariane présenté dans [44] explore l'espace de configuration du système robotique à partir de sa configuration initiale en construisant un arbre de recherche par l'alternance de deux Algorithme (Figure 2.10) :

Explore : Cet algorithme a pour but d'explorer l'espace de configuration libre en y plaçant des balises aussi loin que possible des balises existantes. A l'initialisation, la seule balise disponible est la configuration initiale du système. A chaque nouvel appel de la méthode "Explore", l'arbre est étendu à partir d'une des balises existantes choisie aléatoirement.

Recherche : Cet algorithme recherche autour d'une balise posée s'il est possible d'accéder directement à la configuration finale par un mouvement simple (chemin de Manhattan).

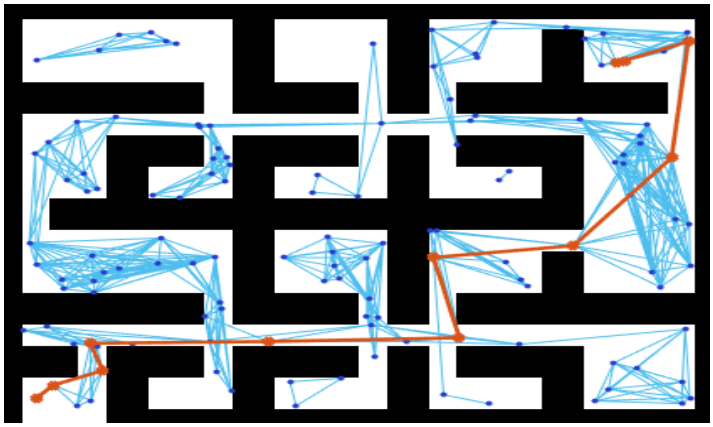


Figure 2.10 : Fil d'Ariane : Méthode de planification alternant deux algorithmes

En répétant successivement ces deux étapes, l'arbre de recherche va s'étendre rapidement sur tout l'espace de configuration accessible à partir de la configuration initiale, jusqu'à converger vers le but, ou s'arrêter s'il n'est plus possible de placer une balise à moins d'une distance minimale de celles déjà posées.

D'autres méthodes moins connues de planification par expansion d'un arbre peuvent être notées : Parmi celles-ci on trouve une planification expansive sur l'espace de configuration [45] consistant étendre itérativement un arbre de recherche en sélectionnant un nœud de l'arbre à étendre à partir d'une probabilité inversement proportionnelle au nombre de nœuds dans la région qui l'entoure. Une nouvelle configuration est alors choisie dans son voisinage proche et un chemin déterminé entre ces deux configurations

Dans la même idée, [46] propose une planification par marche aléatoire. Le principe ici est très simple : un nœud de l'arbre déjà construit est choisi aléatoirement et étendu dans une direction aléatoire. La longueur du chemin à parcourir à chaque étape ainsi que la variation de la direction à prendre par rapport à l'étape précédente est déterminées à partir des observations sur l'environnement, obtenues lors des extensions précédentes de l'arbre.

Ces deux dernières approches bien que simples et fonctionnelles disposent malheureusement de fortes difficultés à traverser de longs espaces libres.

3.3. Approches réactives

Les approches réactives consistent à calculer à chaque pas de temps (après récupération des informations sur l'environnement fournies par les capteurs du système) le contrôle instantané à appliquer sur les actionneurs du système.

3.3.1. Approches par champs de potentiel

Les approches dîtes par champs de potentiel initialement proposées par Khatib [47] consistent à considérer le robot mobile comme une particule soumise à divers champs électromagnétiques régissant son mouvement. Cette méthode s'appuie sur le calcule de deux champs de potentiel (Figure 2.11) :

- Un champ de potentiel attractif provenant de la position finale $\mathbf{q_f}$ du système à atteindre.
- Un champ de potentiel répulsif provenant des obstacles statiques et mobiles de l'environnement.

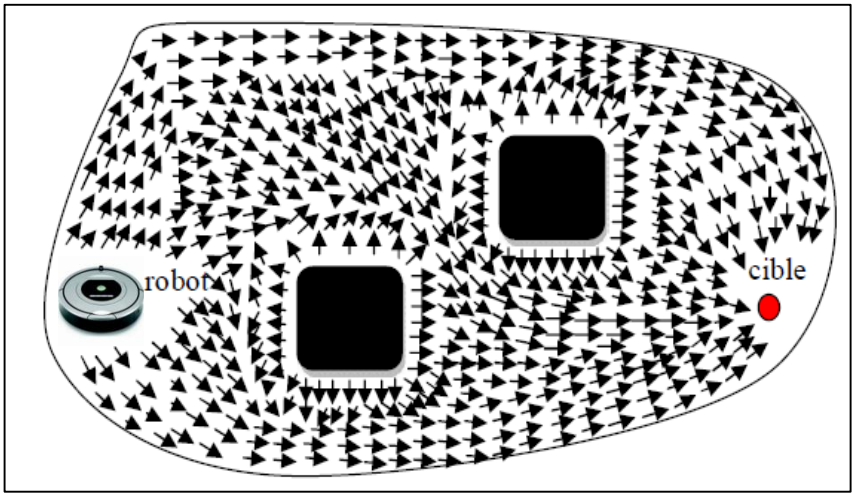


Figure 2.11 : Champs de potentiel répulsif et attractif.

Initialement conçus pour le calcul du mouvement de robot mobile, cette méthode dispose de l'avantage de calculer ces champs de potentiels dans l'espace de travail (espace euclidien IR2 ou IR3 dans lequel une représentation des obstacles est disponible), définissant ainsi une direction privilégiée à suivre par l'élément terminal du robot dans cet espace. Une modification de la configuration du robot dans son espace articulaire est alors déduite de ce champ dans un second temps.

Malgré que cette méthode est simple et élégante mais elle possède de nombreux inconvénients mis en évidence par Koren et Borenstein [48]. D'une part, cette approche est sujette à des minima locaux. Par conséquence, la convergence vers le but n'est pas assurée. D'autre part, ces potentiels peuvent donner lieu à de fortes oscillations du mouvement du robot en présence d'obstacles et principalement lorsque celui-ci navigue dans des passages étroits (couloirs, portes). Enfin, le vecteur de déplacement désigné par le champ de potentiel ne prend en aucun cas en compte la cinématique ou la dynamique du système robotique considéré. Un robot disposant de contraintes non-holonomes aura de sérieuses difficultés à suivre une telle direction.

Malgré ces limitations, de nombreuses techniques de navigation ont découlé de ces champs de potentiels. Ils ont par exemple été adaptés à la navigation au milieu d'obstacles mobiles dans [49] en prenant en compte non seulement la distance aux obstacles mais également la vitesse de ces derniers pour calculer les champs répulsifs.

3.3.2. Histogramme de champs de vecteurs

Dans la lignée des approches par champs de potentiels, sont apparus les histogrammes par champs de vecteurs (Vector Field Histogram VFH). Ceux-ci, introduits par Koren et Borenstein [48] sont nés de la combinaison des champs de potentiels et des grilles d'occupation : un histogramme basé sur une grille cartésienne de l'environnement, cette dernière est construit et mis à jour au fur et à mesure de la navigation pour de reporter la présence d'obstacles à proximité du robot (Figure 2.12). Afin de choisir une direction à suivre, un histogramme polaire est construit à partir de la grille d'occupation : en discrétisant les différentes directions possibles autour du robot, l'histogramme polaire est construit en pondérant pour chaque secteur de la discrétisation polaire les cellules traversées de la grille d'occupation contenant des obstacles. Une fois cet histogramme polaire construit, des ''vallées candidates'' sont déterminées comme les suites de secteurs contigus de l'histogramme polaire libres d'obstacles (Figure 2.13). La direction à prendre par le système est alors déterminée par le milieu de la vallée menant le plus directement au but.

Initialement conçue pour la navigation de robots holonomes (pouvant naviguer dans toutes les directions), cette méthode a été étendue à plusieurs reprises afin de prendre en compte les dimensions du robot (par un espace de configuration implicite) et ses contraintes de vitesse [50]. Plus tard, les VFH ont été combinés à une recherche A* (VFH* [51]) afin de trouver un chemin menant vers le but et d'échapper ainsi aux minimum locaux. Les méthodes VFH disposent néanmoins encore de fortes limitations : Elles ne prennent ni en compte la dynamique du système robotique, ni l'éventuelle présence d'obstacles mobiles ; le mouvement instantané du robot est calculé uniquement à partir des informations sur la position courante des obstacles, leur vitesse n'est en aucun cas considérée.

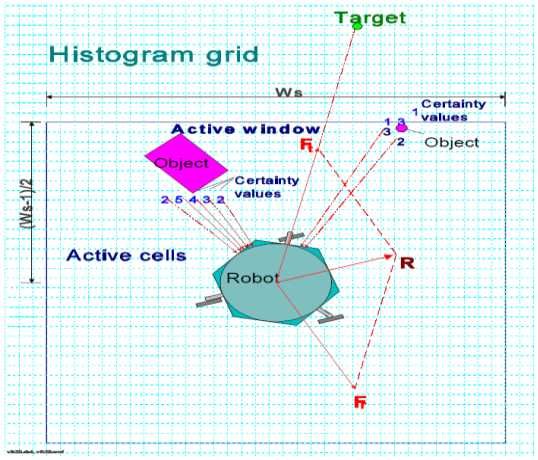


Figure 2.12 : Fenêtre active autour du robot dans laquelle sont mise à jour les probabilités d'occupation par les obstacles.

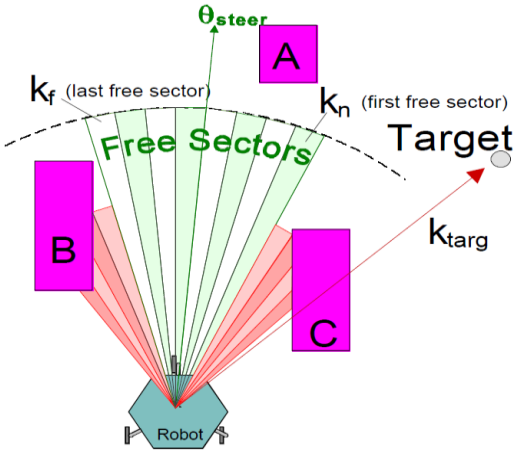


Figure 2.13 : Histogramme polaire calculé à partir de la grille d'occupation. Les vallées libres d'obstacles sont déterminées afin de choisir une direction à suivre

3.3.3. Navigation par diagrammes de proximité

L'approche de navigation par diagrammes de proximité (en anglais : Nearness Diagram Navigation - ND) proposée par Minguez et Montano [52] fortement inspirée des VFH se base sur la topologie de l'environnement pour choisir un contrôle à appliquer à chaque instant. Pour ce faire, deux diagrammes polaires sont construits : l'un représente la distance du centre du robot aux obstacles dans toutes les directions autour de celui-ci (Figure 2.14) ; le second calcule cette même distance à partir des contours du robot afin d'estimer la distance de sécurité conservée.

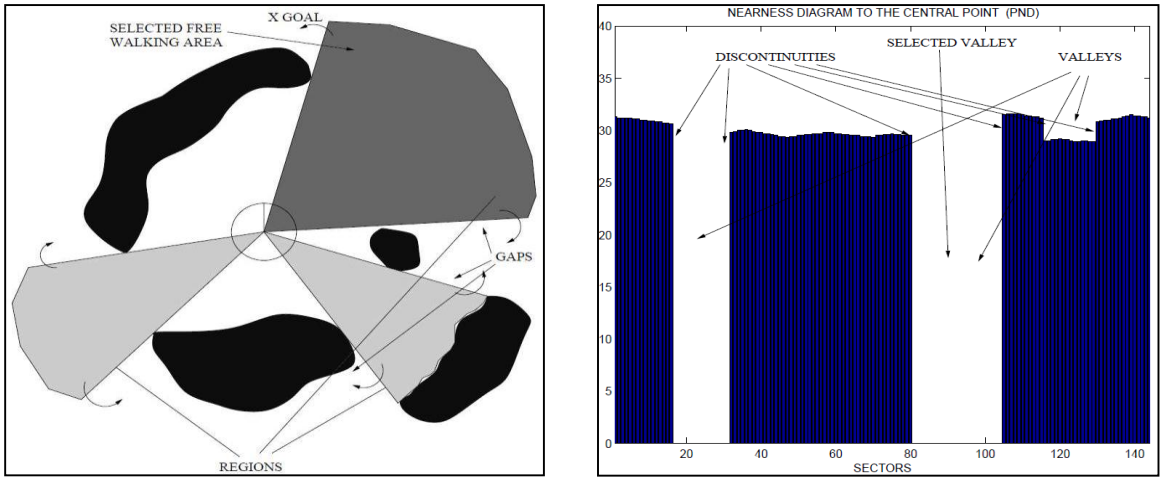


Figure 2.14 : Navigation par Diagrammes de Proximité (ND) : (a) Représentation de l'environnement autour du robot. (b) Caractérisation des vallées disponibles en repérant les discontinuités du graphe de proximité des obstacles

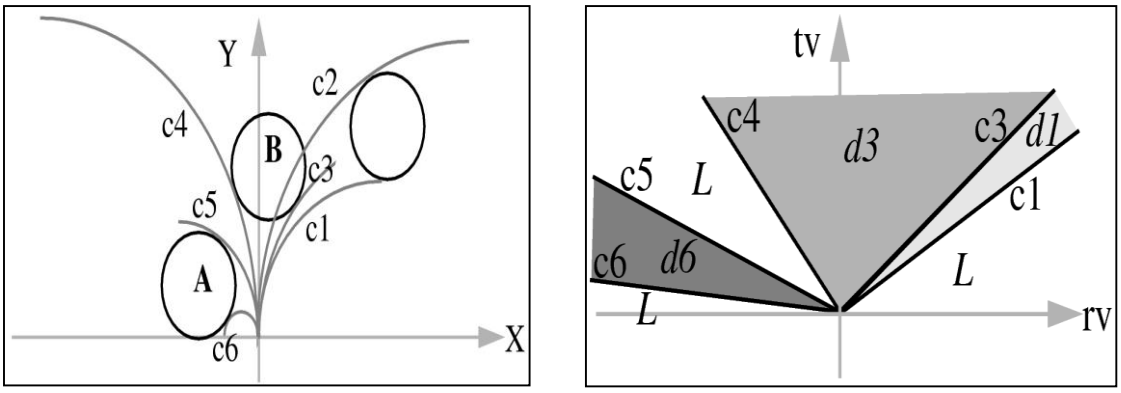
La méthode ND utilise une stratégie déterministe de choix du comportement à adopter en fonction de la distance aux obstacles les plus proches, et en fonction de la topologie des obstacles qui l'entourent. Les options possibles se résument à [1] :

- Contourner un obstacle.
- Passer entre deux obstacles.
- Aller tout droit vers le but ainsi qu'à choisir une vitesse faible ou élevée suivant la proximité des obstacles.

De la même manière que les approches présentées précédemment, cette méthode n'est pas vraiment adaptée pour naviguer au milieu d'obstacles mobiles et ne prends pas en compte la dynamique du système.

3.3.4. Méthode de navigation courbure-vélocité

Différemment aux méthodes précédente, la courbure-vélocité (en anglais Curvature-Velocity- Method - C.V.M) proposée par [53], et au lieu de choisir une direction à suivre en fonction de la position finale et de l'environnement puis de calculer une commande à appliquer dans l'espace des vitesses, la C.V évalue les différentes trajectoires possibles dans l'espace des vitesses (Figure 2.15) puis sélectionne dans cet espace un mouvement à exécuter permettant d'éviter les obstacles et de se rapprocher du but. Afin de choisir quelle commande appliquer, à chaque couple vitesse linéaire / vitesse angulaire (v, w) est associée une fonction de coût basée sur la distance aux obstacles, la modification de l'orientation du système par rapport au but et sur le temps nécessaire pour rejoindre le but. La commande maximisant cette fonction de coût est alors sélectionnée et envoyée au robot lors du prochain pas de temps. Puisque cette méthode ne garantie pas la sécurité du mouvement et peut pousser le robot à passer relativement au près des obstacles. Afin d'éviter de rentrer dans des tel situations cette technique été étendue dans [54] dont le principe est le suivant : Tout d'abord, des "lignes" permettant de passer entre les obstacles en maximisant la distance par rapport à ceux-ci sont déterminées. Ensuite une commande permettant soit de suivre une ligne soit de rejoindre l'une d'entre elles est calculée comme précédemment par C.V.



(a) **(b)**

Figure 2.15 : Méthode de navigation courbure-vélocité, (a) Trajectoires candidates représentées dans l'espace de travail, (b) Contrôles correspondant aux trajectoires candidates dans l'espace des vitesses linéaire et angulaire.

3.3.5. Fenêtres dynamiques (D.W)

Découlant des C.V, l'approche de fenêtre dynamique (Dynamic Window D.W) présentée par [55] conserve le principe de sélection d'un mouvement à suivre dans l'espace des vitesses.

A l'instar de la plupart des approches réactives présentées précédemment, les DWs ont été développées pour des robots de type différentiel, contrôlés en vitesse linéaire v et vitesse angulaire w. A chaque pas de temps, une nouvelle commande constante (v, w) est sélectionnée parmi les vitesses respectant les contraintes suivantes :

- Contraintes cinématiques : $v \in [0; v_{max}]$ et $\omega \in [-\omega_{max}; \omega_{max}]$ ou v_{max} et ω_{max} sont les vitesses maximales admissibles.
- Contraintes dynamiques: Les accélérations linéaire et angulaire γ et η appliquées entre chaque pas de temps doivent être bornées. $\gamma \in [-\gamma_{max}; \gamma_{max}]$ et $\eta \in [-\eta_{max}; \eta_{max}]$ ou γ_{max} et η_{max} sont les accélérations linéaires et angulaires maximales.
- garantie de sécurité passive : Le système doit être certain de pouvoir s'arrêter avant d'entrer en collision avec un obstacle.

Notons $\tilde{u}_s$ l'ensemble des vitesses admissibles respectant les contraintes cinématiques, $\tilde{u}_d$ les vitesses respectant les contraintes dynamiques et $\tilde{u}_a$ les vitesses permettant de s'arrêter avant d'entrer en collision avec les obstacles de l'environnement (Figure 2.16). L'ensemble des commandes candidates $\tilde{u}_r$ est alors défini par :

$$\tilde{u}_r = \tilde{u}_d \cap \tilde{u}_s \cap \tilde{u}_a$$

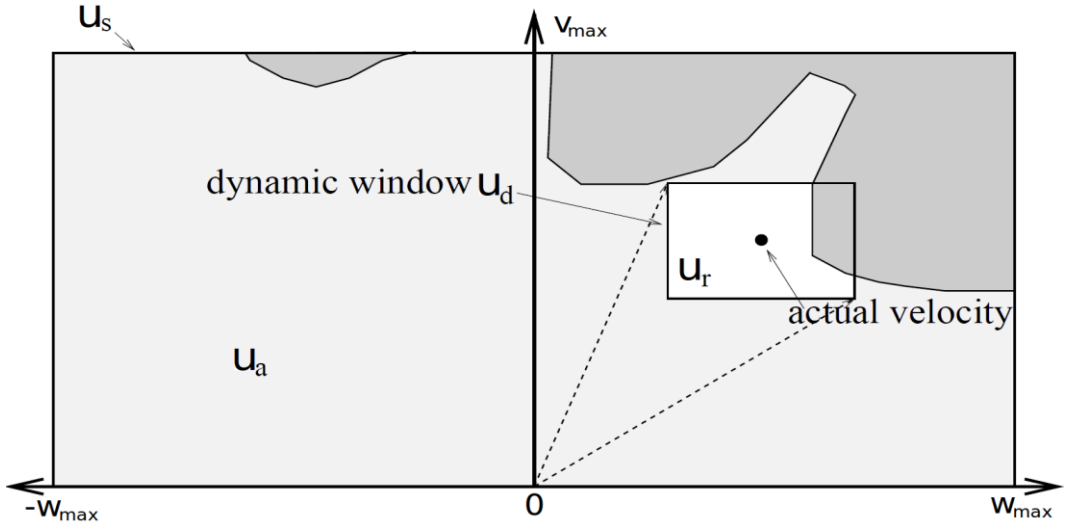


Figure 2.16 : Fenêtre Dynamique (DW) : Calcul du mouvement a appliqué à chaque pas de temps dans l'espace des vitesses

Une fois cet ensemble défini, une fonction de coût similaire au C.V est définie. La commande choisie est de même la commande maximisant cette fonction de coût.

De par sa volonté d'assurer une sécurité passive, l'approche de DW est devenue l'une des approches de navigation réactives les plus populaires de nos jours. Cette approche initiale souffrait d'une limitation importante : seule la position courante des obstacles était prise en compte, mais pas leur mouvement. La navigation en environnement dynamique avec ce type d'approche en était donc fortement compromise. Afin de palier ce problème, les auteurs de [55] ont proposé une extension intitulée ‘‘Time Varying Dynamic Window’’.Celle-ci calcule à chaque instant un ensemble de trajectoires probablement suivies par les obstacles dans le futur. Une vérification de collision à court terme peut donc être opérée.

3.3.6. Représentation des obstacles dans l'espace des vitesses

Dans le but de prendre en compte non seulement la dynamique du système robotique, mais également la dynamique de l'environnement dans lequel évolue ce système, Fiorini et Shiller ont introduit une approche intitulée «Velocity Obstacles (V.O)» [56] Supposant une connaissance à priori du mouvement des obstacles mobiles, l'approche consiste à caractériser parmi les vitesses admissibles (respectant les contraintes cinématiques et dynamiques du système), celles menant à une éventuelle collision avec les obstacles dans le futur (jusqu'à un certain horizon temporel t_h). En supposant qu'un obstacle **B** va conserver une vitesse constante dans un futur proche (par exemple par approximation linéaire de sa vitesse courante), il est possible de déterminer les vitesses relatives du robot à cet obstacle menant à une collision dans le futur. L'ensemble de ces vitesses ‘’interdites‘’ s'illustre graphiquement comme présenté dans la (Figure 2.17(a)) par un cône de vecteurs vitesse interdit. Certaines de ces vitesses ne conduiront bien sûr à une collision qu'après un temps relativement élevé. En limitant les V.Os à un horizon temporel t_h (Figure 2.17(b)), on obtient ainsi une approximation raisonnable des vitesses interdites pour le robot mobile.

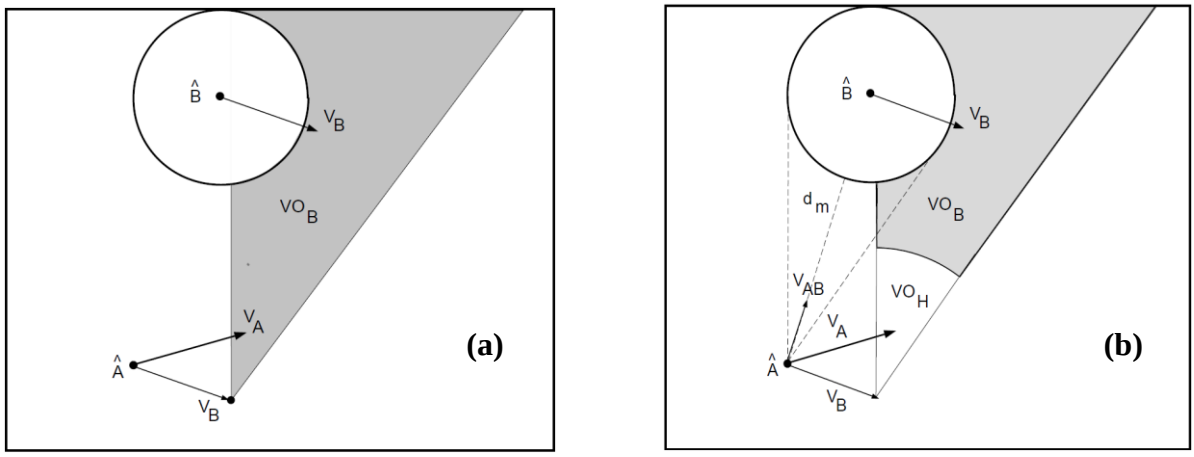


Figure 2.17 : Représentation des obstacles dans l'espace des vitesses (V.O), (a) calcule de cône des vitesses interdites pour un horizon temporel infini, (b) calcule de cône des vitesses pour un horizon temporel limité.

Plusieurs extensions de ces travaux ont vu le jour ces dernières années, telle la prise en compte d'approximation du modèle du futur des obstacles plus complexes [58] ou encore une tentative de définition du ‘‘bon horizon temporel’’ nécessaire à la garantie de la sécurité du mouvement [57]. La détermination de cet horizon temporel est encore un sujet prêtant fortement à débattre à l'heure actuelle.

3.3.7. Navigation basée sur les états de collisions inévitables

Visant désormais comme objectif l'insertion de robot autonome en zones urbaines au milieu de piétons, de véhicules ou d'autres robots, la sécurité du mouvement des systèmes robotiques est devenue un axe prioritaire de recherche dans le domaine. Dans cette optique est née la notion d'état de collision inévitable fortement développée dans [59], [60], [61], [62].

Un état du robot est un état de collision inévitable (E.C.I) si, quelque soit la séquence de contrôle appliquée en entrée du robot, il existe un temps t auquel ce dernier est assuré de rentrer en état de collision.

De par cette définition, il devient clair qu'il est nécessaire pour assurer la sécurité d'un système robotique (et des autres agents évoluant dans son environnement) d'éviter non seulement les états de collisions mais également les E.C.I. La détermination des E.C.I nécessite, comme pour les V.Os de prendre en considération la dynamique du système, mais également celle des obstacles mobiles qui l'entourent (et par conséquent, une prévision de leur mouvement dans le futur). Parthasarathi et Fraichard se sont efforcés de proposer une méthode basée sur une sous-approximation de l'espace des contrôles permettant d'évaluer si l'état d'un système robotique est un E.C.I ou non [60]. Dans la continuité de ces travaux, Martinez et Fraichard [61] ont développé une méthode de navigation réactive permettant à chaque instant de passer d'un état non- E.C.I vers un autre état non- E.C.I.

Cette approche semble être à l'heure actuelle la meilleure option pour se rapprocher d'un risque de collision nul, néanmoins elle reste fortement coûteuse et dépendante de la fiabilité du modèle prévisionnel du mouvement des obstacles considéré.

3.3.8. Planification de mouvement partiel

Une dernière approche réactive mérite d'être notée : il s'agit de la planification de mouvement partiel (Partial Motion Planning (PMP)) utilisée par [53] et [64] Celle-ci consiste à calculer réactivement, en un temps de décision fixe, une trajectoire se rapprochant le plus

possible du but (Figure 2.18). Cette méthode consiste en un algorithme à trois étapes répété à chaque pas de temps :

- Mise à jour du modèle de l'environnement à partir des entrées capteurs du robot.
- Recherche délibérative d'une trajectoire menant à l'état but. Si le but n'a pas été atteint après un temps de décision fixe, la trajectoire calculée s'en rapprochant le plus est choisie comme trajectoire à suivre.
- Enfin, le mouvement planifié au pas de temps précédent est exécuté.

Cette approche permet donc d'être réactive aux diverses évolutions de l'environnement tout en étant capable de sortir d'impasses non détectées à priori. Elle reste sujette à des minima locaux, mais y est néanmoins bien plus robuste que les approches citées précédemment.

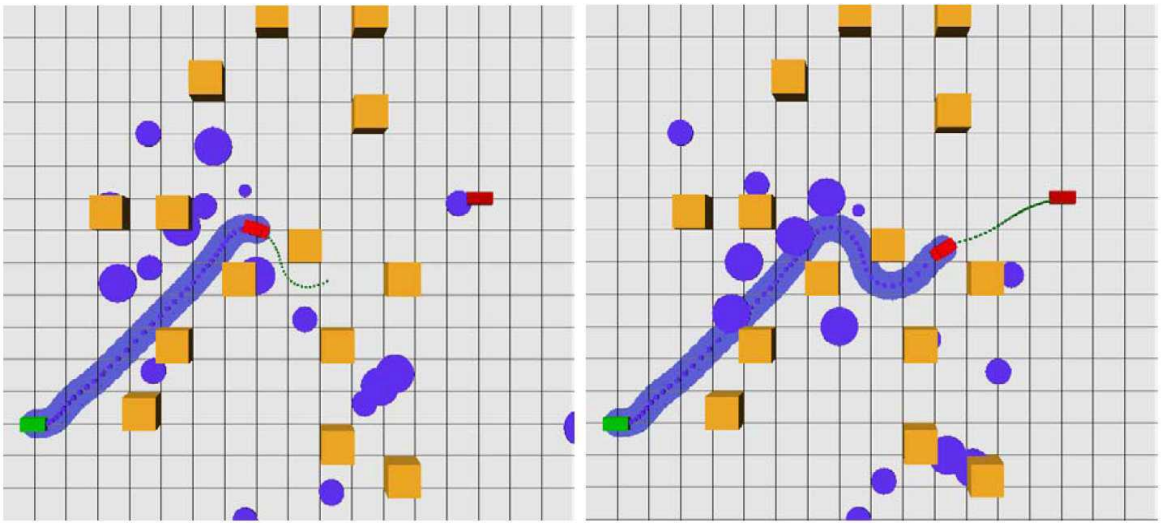


Figure 2.18 : navigation basé sur la planification de mouvement partiel, la méthode consiste à calculer à chaque pas de temps un point à atteindre en rapprochant le plus possible à la cible sans entrer en collusion avec les obstacles.

3.3.9. Défaut de convergence vers le but

Toutes les approches réactives citées ci-dessus disposent d'une complexité suffisamment faible pour être employées en temps réel au cours de la navigation. Cette caractéristique n'assure cependant en aucun cas la sécurité du système robotique (il n'est pour la plupart de ces approches ni garanti qu'il sera capable de s'arrêter s'il rencontre un obstacle et encore moins qu'il sera capable d'éviter un obstacle hostile se dirigeant vers lui).

La capacité à raisonner sur le futur tend à favoriser une navigation sure, c'est donc l'une des caractéristiques que nous souhaiterions intégrer au développement d'une nouvelle technique de navigation.

Malgré ces avantages par rapport aux approches délibératives, aucune de ces méthodes n'est capable d'assurer la convergence vers le but. Nous présentons alors dans la section suivante une méthode de déformation de mouvement essayant à la fois de s'assurer que celui-ci sera bien atteint, tout en évitant les obstacles au cours de l'exécution du mouvement.

3.4. Méthode de déformation de mouvement

Entre approches délibératives calculant un mouvement complet jusqu'au but mais dont la complexité est trop élevée pour être utilisées en environnement dynamique, et approches réactives utilisables en temps réel mais dont la convergence vers le but est compromise, se trouve une large gamme de méthodes.

Afin de palier les inconvénients de ces deux types d'approches, Qinlan et Khatib ont proposé en 1993 une approche basée sur la déformation de mouvement [65]. Son principe est le suivant : Un chemin complet jusqu'au but est calculé a priori et fourni au système robotique. Au cours de l'exécution, la partie du mouvement restant à être exécutée est déformée continuellement en réponse aux informations sur l'environnement récupérées par les capteurs.

Le système peut ainsi modifier son parcours en fonction du déplacement d'obstacles ou de l'imprécision de sa connaissance de l'environnement. La déformation résulte en général de deux types de contraintes : des contraintes externes dues à la proximité des obstacles, et des contraintes internes destinées à maintenir la faisabilité et la connectivité du mouvement. Tant que la connectivité du chemin est maintenue, la convergence vers le but est assurée.

La déformation de chemin souffre néanmoins d'une forte contraint: les déformations sont limitées à des homotopies (déformations continues) du chemin de départ, pouvant ainsi conduire à des comportements inappropriés.

4. Conclusion

Nous avons présenté dans ce chapitre les différents modèles existants dans la littérature pour représenter les robots mobiles non holonomes. Ainsi que les différentes techniques de planification de trajectoire. Un état de l'art résume la navigation autonome d'un robot mobile est présenté aussi dans ce chapitre, à savoir des approches dites délibératives destinées à planifier un mouvement complet entre deux configurations et d'autre part des approches dites réactives. Ces approches consistent à calculer un nouveau mouvement à suivre afin de pouvoir s'adapter au mouvement des obstacles mobiles ou inattendus. Nos travaux se sont donc concentrés sur la résolution du problème de navigation d'un robot mobile dans un environnement de travail. Nous proposons dans le chapitre suivant une approche par d'odométrie afin d'en apporter une solution.

Chapitre III : Algorithme d’odométrie, et résultats de simulation

1. Introduction

Pour améliorer l'autonomie d'un système mobile, il est nécessaire de pouvoir déterminer précisément sa position [67].

Afin de pouvoir se localiser dans un environnement, un robot mobile a besoin de capteurs lui donnant des informations sur sa position absolue dans l'espace. Dans un environnement d'intérieur, les robots mobiles utilisent principalement l'odométrie, basée sur des données proprioceptives ce qui permet par intégration du déplacement, de connaître les coordonnées de la position (x, y) et l'orientation (θ) d'un point du robot dans un repère absolu. Pour cela on utilise un système de mesure relative de position tel que les systèmes inertiels ou les systèmes odométriques [68].

À travers ce chapitre, nous allons présenter les systèmes de masures d'attitude les plus utilisés, à savoir l'algorithme d'odométrie et le filtre de Kalman qu'ils sont utilisés dans ce projet pour nous permettre l'estimation de la position de notre robot. Nous allons aussi présenter le problème du SLAM de manière générale, puis on va présenter les résultats de simulation pour chaque problème.

2. Filtre de Kalman

L'algorithme du filtre de Kalman a été mis au point au début des années 1960, dans le but d'éliminer les bruits de moyenne nulle perturbant un signal et il peut s'appliquer à tout type de système pouvant être décrit par une équation linéaire. Il a forgé sa réputation grâce à son utilisation dans les programmes de navigation inertielle utilisés sur l'ordinateur de guidage d'Apollo développé par la NASA. Il donne de si bons résultats que la plupart des applications pilotées par des capteurs en temps réel fonctionnant dans des environnements bruyants l'utilisent encore aujourd'hui. Qu'est-ce qui rend le filtre de Kalman si intelligent ? Il utilise un processus en deux étapes très simple en apparence pour prévoir le résultat, puis compare cette prévision avec une mesure pour mettre à jour la prochaine prévision. Il conserve un chiffre de "l'incertitude" concernant la mesure du capteur qui est également mis à jour. Si la sortie du capteur commence à devenir plus bruyante, l'algorithme de Kalman réduit l'influence de la mesure sur la sortie et augmente l'importance de sa propre estimation. Les principales choses à savoir sur cet algorithme :

- Le processus à mesurer doit être décrit par un système linéaire. Par exemple ce peut être un système simple de mouvement linéaire basé sur l'accélération, la vitesse et la position. Un filtre de Kalman étendu a été développé par le personnel de la NASA pour traiter les systèmes non linéaires, mais le calcul est encore plus compliqué.
- Une mesure du capteur se compose d'une valeur moyenne et de sa variance. Pour des performances optimales, il faut non seulement que le modèle du système soit exact, mais les variances pour les entrées de signaux doivent l'être aussi. Vous trouverez peut-être des valeurs correctes sur une fiche technique, mais certains tests de laboratoire peuvent être nécessaires.
- Le bruit doit avoir une moyenne nulle .Le filtre va en effet considérer que toute la partie de la mesure qui n'a pas une moyenne nulle.

Le filtre de Kalman est souvent illustré dans la littérature à l'aide d'une seule entrée du capteur, mais il peut gérer deux ou plusieurs entrées fournissant la fusion des capteurs. Bien entendu, l'exigence de traitement en temps réel, même pour une seule entrée est assez conséquente et elle s'intensifie avec plusieurs entrées. Jusqu'à récemment, cela constituait une limitation importante, mais cela est bien moins le cas désormais grâce aux microcontrôleurs 32 bits avec des unités mathématiques à virgule flottante, les DSP et même la manipulation de données parallèle à l'aide de matériels à données multiples et entrée unique (SIMD) comme NEON d'ARM. L'algorithme lui-même peut être simplifié en déclarant certaines de ses nombreuses variables égales à 1 ou 0 "pour des raisons pratiques", mais cela doit être fait avec beaucoup de soin pour ne pas compromettre sa puissance étonnante de réduction adaptative du bruit. La fusion de capteurs avec le filtrage de Kalman offre également autre chose : une fiabilité accrue grâce à la redondance intelligente [69].

2.1. Filtre de Kalman étendu (EKF)

Lorsqu'on souhaite appliquer un filtre de Kalman pour estimer les paramètres d'un système, la première chose à faire est de modéliser le problème. Il se trouve que dans certain cas, les équations qui permettent de modéliser le problème ne sont pas linéaires. De ce fait, le filtre de Kalman n'est plus applicable tel quel. Heureusement, il est tout de même possible d'estimer les paramètres du système à l'aide d'un filtre de Kalman dit étendu. Ce filtre permet en effet de linéariser localement le problème et donc appliquer les équations du filtre de Kalman classique.

Le principe d'un filtre de Kalman étendu est très simple. Tout d'abord, les équations d'état et les équations liant l'état précédent à l'instant suivant qui étaient linéaires dans le cas du filtre de Kalman classique sont maintenant non linéaire. Il est donc impossible de l'écrire sous forme matricielle.

$$\begin{cases} Y=H.X+B \\ \hat{X}_k^+ = A.\hat{X}_k \end{cases} \quad (3.1)$$

On remplace donc les équations par :

$$\begin{cases} Y=h(X,B) \\ \hat{X}_k^+ = f(\hat{X}_k) \end{cases} \quad (3.2)$$

On est donc obligé d'appliquer ces équations non linéaires pour le calcul de la prédiction et la mise à jour du vecteur d'état. On se rend bien compte que l'on linéarisé localement les équations afin d'appliquer le filtre de Kalman. Cette linéarisation est locale, ce qui entraine donc une convergence locale du filtre de Kalman étendu. Ce filtre ne garantis donc pas une convergence globale (à l'inverse du filtre de Kalman classique). La stabilité d'un EKF est donc plus difficile à garantir et dépend souvent de sa bonne initialisation [70].

3. Méthode de Localisation basée sur le modèle odométrique

L'odométrie est certainement la méthode de localisation la plus couramment employée pour les robots disposant d'une structure de locomotion à roues. Son principe consiste à déduire une position, de façon incrémentale, à partir de la vitesse et de la géométrie des roues. La mise en œuvre de cette méthode est des plus simples et ne nécessite qu'une puissance de calcul très limitée.

Cependant, l'odométrie est fragile. Tout d'abord, elle nécessite une connaissance précise de la géométrie du robot: le diamètre des roues doit être déterminé, mais d'autres dimensions, comme l'entraxe ou le point de contact avec le sol, doivent également être prises en compte. Ces paramètres sont généralement difficiles à obtenir de façon précise et, en terrains naturels, dépendent fortement de la nature du sol. De plus, les cas pathologiques sont rarement détectables : les glissements ou patinages sont par exemple dramatiques pour l'estimation de la position et ils ne peuvent en général pas être mesures directement. Selon le type de terrain ils peuvent rendre quasiment inutile l'odométrie en tant que méthode de localisation.

Celle-ci est pourtant incontournable car, outre sa fonction d'estimateur de position, elle permet de réaliser un contrôle très fin des déplacements des plateformes. La position locale est fournie à haute fréquence, autorisant ainsi l'exécution de déplacements élémentaires simples [71].

3.1. Algorithme d'odométrie

L'odométrie est une méthode de mesure du capteur de mouvement ou du capteur de rotation pour estimer le changement de position dans le temps. L'odométrie est utilisée par certains robots, qu'ils soient sur pattes ou sur roues, pour estimer (et non déterminer) leur position par rapport à un emplacement de départ. Cette méthode est sensible aux erreurs dues à l'intégration des mesures de vitesse dans le temps pour donner des estimations de position. La collecte rapide et précise des données, l'étalonnage de l'équipement et le traitement sont nécessaires dans la plupart des cas pour que l'odométrie soit utilisée efficacement. La figure 3.1 explique la géométrie détaillée de l'odométrie pour notre robot mobile. Le véhicule part de (x, y, θ) et se trouve à (x′, y′, θ′). Le centre entre deux roues du robot se déplace le long d'une trajectoire d'arc. En se rappelant que la longueur de l'arc est égale au rayon multiplié par l'angle intérieur [72]:

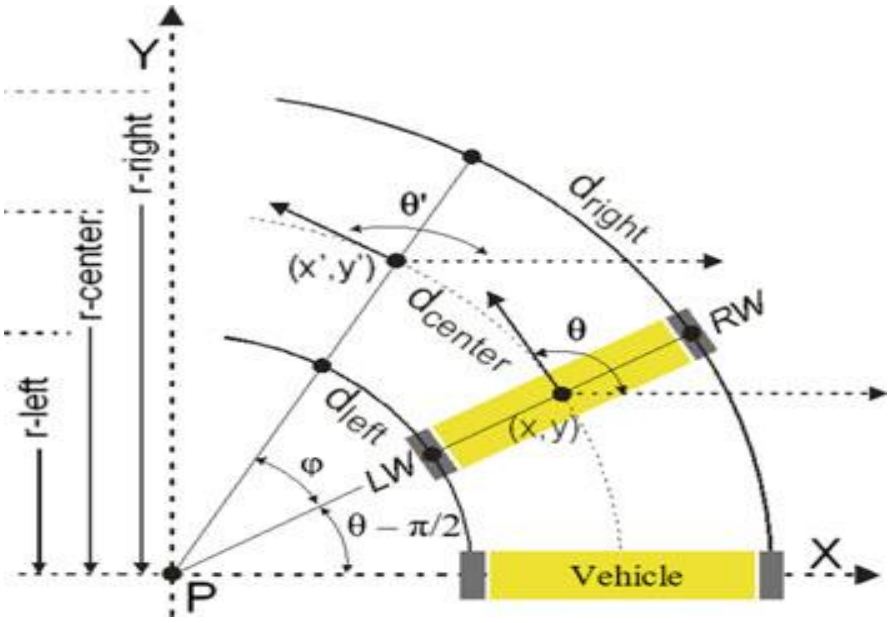


Figure 3.1 : la géométrie détaillée pour l'odométrie

La longueur de cet arc est : $Dcenter = \frac{dleft+dright}{2}$ (3.3)

Sur la géométrie de base, l'équation est:

$$\begin{cases} \varphi r_{left} = d_{left} & (3.4) \\ \varphi r_{right} = d_{right} & (3.5) \end{cases}$$

Si la d_{baseline} est la distance entre les roues gauche et droite:

$$r_{left} + d_{\text{baseline}} = r_{right} \quad (3.6)$$

Soustraire des égalisations (3.4) et (3.5) ci-dessus:

$$\varphi = \frac{d_{right} - d_{left}}{d_{\text{baseline}}} \quad (3.7)$$

Les encodeurs de roue donnent la distance parcourue par chaque roue, roue gauche et droite. Supposons que les roues suivent une trajectoire d'arc pour une courte échelle de temps.

$$\begin{cases} x' = x + d_{\text{center}} \cos\theta & (3.8) \\ y' = y + d_{\text{center}} \sin\theta & (3.9) \\ \theta' = \theta + \frac{d_{right} - d_{left}}{d_{\text{baseline}}} & (3.10) \end{cases}$$

Le calcul de la circonférence de la roue est nécessaire pour savoir jusqu'où le mouvement du robot. Le robot mobile utilise deux roues de 65 mm de diamètre. Chaque roue est équipée d'un encodeur à disque à 20 trous pour une rotation. L'équation est la suivante :

$$\begin{cases} \Delta Tick = tick' \_ tick & (3.11) \\ d_{left} = 2.\pi.r.\frac{\Delta\, tick\_left}{N} & (3.12) \\ d_{right} = 2.\pi.r.\frac{\Delta\, tick\_right}{N} & (3.13) \end{cases}$$

Calcul des vitesses des roues :

Pour calculer les vitesses des roues on se base sur le fait que le robot est positionne en un point de coordonnées (x_0 ety_0) et qu'il se déplace vers un autre point de coordonnées (x et y) avec une orientation $\dot{\theta}$, alors que pour le modèle différentiel (modèle mathématique) d'un robot mobile est définie par :

$$\begin{cases} \dot{x} = \frac{R}{2}\,(v_g + v_d)\cos\theta_0 \\ \dot{y} = \frac{R}{2}\,(v_g + v_d)\sin\theta_0 \\ \dot{\theta} = \frac{R}{L}\,(v_d - v_g) \end{cases} \tag{3.14}$$

Alors les vitesses des roues droite et gauche, respectivement notées v_g et v_d qui sont des entrés dans notre système. Alors qu'on a déjà défini la modélisation cinématique du robot de type unicycle dans le chapitre 2 par :

$$\begin{cases} \dot{x} = \mathrm{v}\cos\theta_0 \\ \dot{y} = \mathrm{v}\sin\theta_0 \\ \dot{\theta} = w \end{cases} \tag{1.3}$$

Par correspondance de ces deux modèles (3.14) et (1.3) (de modèle unicycle et le modèle différentiel) on obtient :

$$\mathrm{v} = \frac{R}{2}\,(v_g + v_d) \longrightarrow \frac{2v}{R} = (v_g + v_d) \tag{3.15}$$

$$w = \frac{R}{L}\,(v_d - v_g) \longrightarrow \frac{wl}{R} = (v_d - v_g) \tag{3.16}$$

De l'équation (3.15) et (3.16), On obtient les équations de la vitesse droite et gauche respectivement:

$$v_d = \frac{2.v + w.l}{2.R} \tag{3.17}$$

$$v_g = \frac{2.v - w.l}{2.R} \tag{3.18}$$

3.2. Principe de fonctionnement du robot

L'organigramme du système est illustré à la Figure 3.2 L'estimation de position est utilisée pour prédire la position du robot en comptant l'impulsion envoyée par l'optocoupleur. La direction du robot peut être connue à partir du résultat du comptage d'impulsions de l'optocoupleur gauche et droit. Le robot lit les coordonnées de destination et vérifie toujours les impulsions d'odométrie (dtick1 et dtick2). Le contrôleur met à jour la dernière position du robot mobile (x_new, y_new, teta_new). Une fois que le robot a obtenu la dernière coordonnée, le contrôleur calcule la différence entre la coordonnée de destination et la coordonnée actuelle.

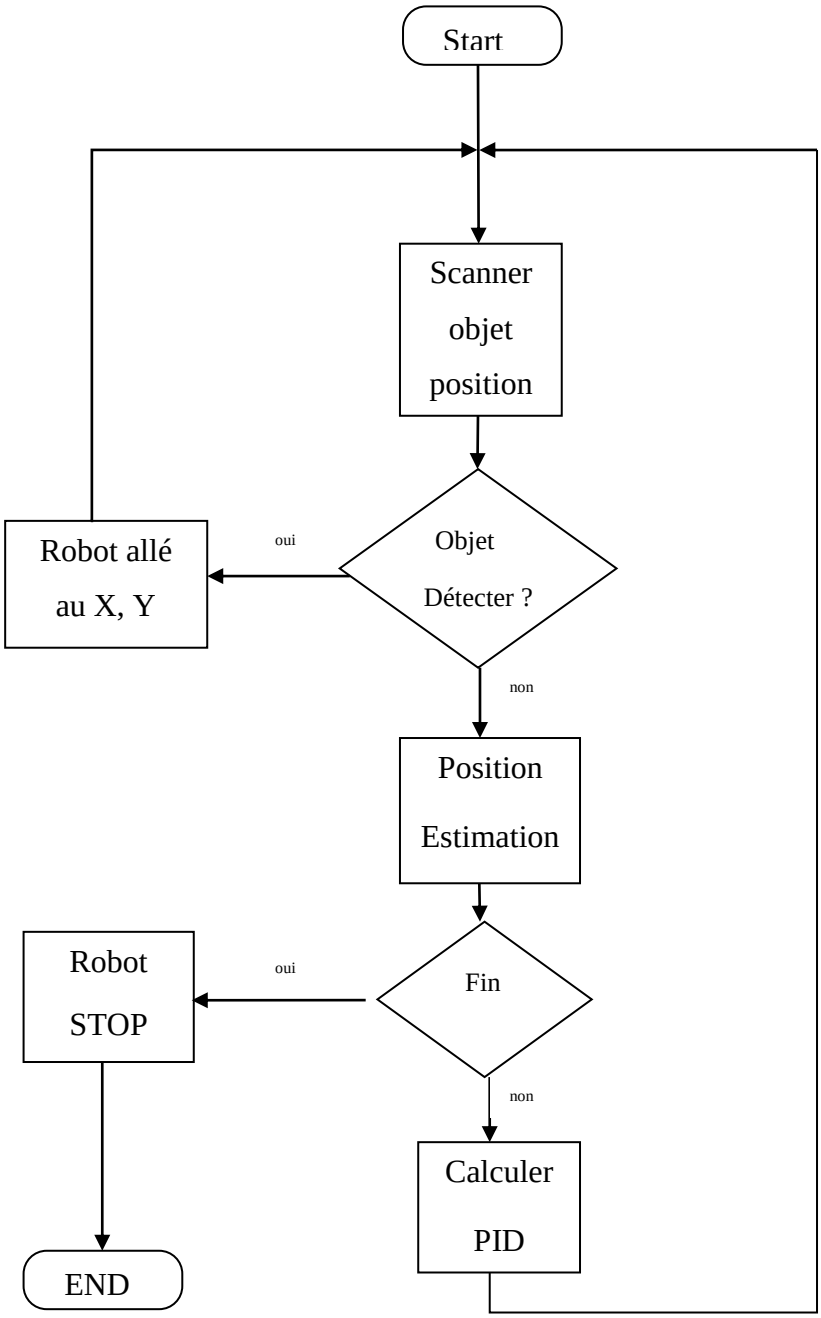


Figure 3.2 : Organigramme sur le principe de fonctionnement du robot

L'erreur est utilisée pour calculer les paramètres PID et déterminer la vitesse du moteur gauche et du moteur droit. Cet algorithme est répété jusqu'à ce que le point de destination soit atteint. Le mécanisme d'évitement des obstacles commence par la lecture de la position de l'obstacle. Le servomoteur fait tourner le capteur ultrasonique et obtient les données de distance entre le robot et les obstacles sous plusieurs angles différents (0, 45°, 90°, 135° et 180°). Chaque distance est sauvegardée et transformée en utilisant la rotation et la translation 2D décrites dans les équations 3.19 et 3.20. L'équation 3.19 est une dérivation de l'équation 3.20 où R(x, y, θ) est une matrice 3 × 3. Le résultat de la transformation donne la coordonnée de la vision du monde représentée à l'équation 3.20.

$$\mathcal{R}\,(x', y', \theta') = \begin{bmatrix} \cos(\theta') & -\sin(\theta') & x' \\ \sin(\theta') & \cos(\theta') & y' \\ 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

$$\begin{bmatrix} x_{di} \\ y_{di} \\ 1 \end{bmatrix} = \mathcal{R}\,(x', y', \theta')\ \mathcal{R}\,(x_{si}, y_{si}, \theta_{si}) \begin{bmatrix} di \\ 0 \\ 1 \end{bmatrix} \tag{3.20}$$

3.3. Erreurs dans l'odométrie

La validité de l'odométrie est basée sur l'hypothèse qui permet de supposer que la rotation propre d'une roue engendre une translation à vitesse connue de son centre. Cette hypothèse, conformément aux différents travaux de Borenstein en la matière, s'avère d'une validité très limitée [73]. Un certain nombre d'erreurs viennent entacher la précision de l'odométrie :

- des erreurs systématiques : erreur sur le diamètre des roues par rapport à la valeur nominale attendue, diamètres différents, erreurs sur la disposition des roues, résolution des codeurs ;
- des erreurs non systématiques : sol non plan ou irrégulier, glissements divers (dus à la nature du sol, à une accélération trop brutale, à un obstacle, un défaut mécanique, etc.), contact au sol non ponctuel.

La propagation des erreurs systématiques au travers de l'odométrie est très gênante car elle est cumulative. En environnement d'intérieur, les erreurs non systématiques auront moins d'importance, notamment parce qu'elles ne s'accumulent pas constamment, comme c'est le cas des erreurs systématiques. Quantitativement, l'estimation de l'incertitude de l'odométrie d'un robot mobile peut être donnée par la matrice de covariance de bruit associée à la posture du robot.

Sa détermination n'est cependant pas une tâche facile. En particulier, à moins de faire des mouvements découplés de translation et de rotation, il est clair que les erreurs correspondantes sont-elles relativement couplées. Par ailleurs, la matrice de covariance de bruit dépend du véhicule, du type de mouvement effectué, des capteurs et de leur modélisation. Pour illustrer l'erreur de manière graphique, on peut également propager une ellipse le long du trajet du robot pour d´écrire l'incertitude en position et un cône pour illustrer l'incertitude en orientation. Les deux sont déterminés à partir de la matrice de covariance de bruit. A mesure que le robot avance et que l'erreur s'accumule ces motifs géométriques grandissent [74].

3.4. Résultats de simulation

Pour montrer l'efficacité de l'approche proposée, une série de simulations a été réalisée en employant des environnements arbitrairement construits contenant des obstacles. La position de tous les obstacles est inconnue ; le robot ne connait que les positions de départ et d'arrivée (cible). Toutes les simulations ont été faites dans l'environnement de MATLAB ou V-REP PRO EDU en utilisant le modèle cinématique du robot.

3.4.1. Simulation sur MATLAB : Comportement de navigation vers une cible :

En absence d'obstacles dans l'espace de travail du robot, la tache de navigation se réduit a une orientation et un déplacement direct vers la cible Les Figures (a) et (b) illustrent les trajectoires de navigation en choisissant différents points de départ et d'arrivés. Ces résultats montrent l'aptitude du contrôleur odométrie à générer les actions de commande les plus appropriées pour accomplir la tâche, ce qui prouve l'efficacité du contrôleur proposé.

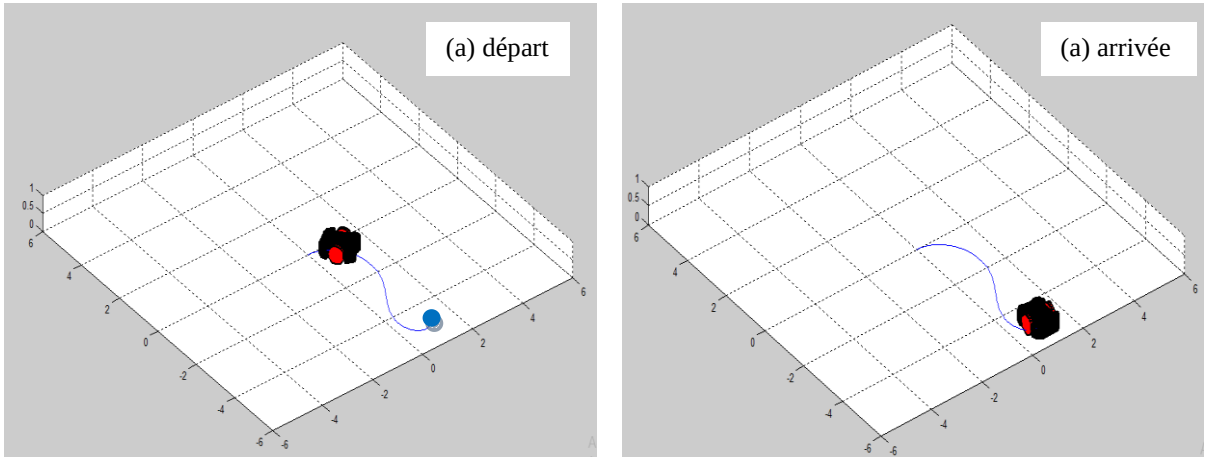


Figure 3.3 : simulation en 3D sur MATLAB pour un déplacement direct d'un robot mobile vers la cible

Sur la même simulation on a déduit la déférente vitesse parcourue par le robot mobile tell que la vitesse linéaire et la vitesse angulaire sur la figure 3.4, et bien aussi on est arrivés à montrer les déférentes erreurs générées par le robot dans la figure 3.5.

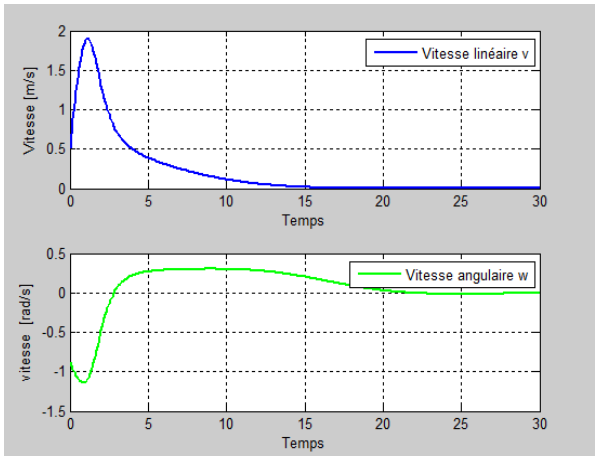


Figure 3.5 : La vitesse linéaire et la vitesse angulaire du robot mobile.

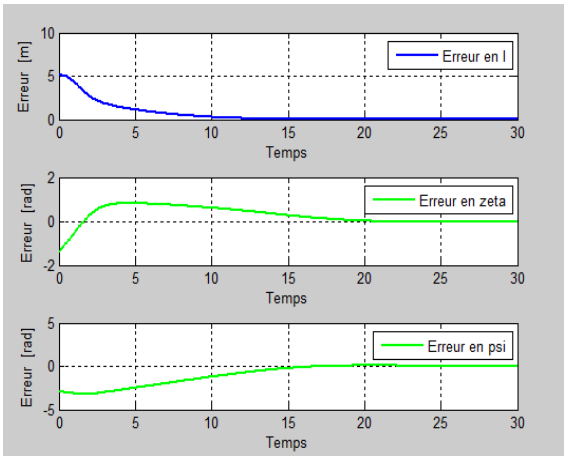


Figure 3.4 : Les erreurs générées par le robot mobile

3.4.2. Simulation sur V-REP PRO EDU : Comportement d'évitement d'obstacles et de navigation vers une cible

Le comportement normal dans la navigation autonome de robot mobile est de lui guider vers son cible. Sur son chemin immédiat, le robot essaye de naviguer vers la cible et quand il s'approche ou rencontre des obstacles, le comportement concerné est exécuté avec un degré convenable. Une fois le robot est en dehors de la zone de collision, il continue sa tâche en se dirigeant vers la cible et en suivant le chemin le plus court.

Les Figures 3.6 et 3.7 montrent des exemples de navigation autonome du robot mobile en présence d'obstacles dans l'environnement. Au départ, le robot essaye de se déplacer en ligne

droite vers la cible, mais lorsqu'il détecte la présence des obstacles dans son chemin, le comportement d'évitement d'obstacles est activé.

Pour cet effet, nous avons réalisé une simulation en 3D sur le logiciel V-REP PRO EDU qui est spécialisé pour la robotique.

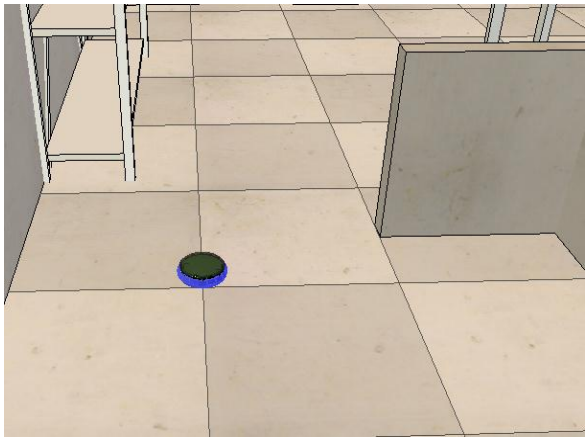


Figure 3.7 : Point de départ d'un robot mobile unicycle dans un environnent de travail (un poste de travail Quel-quand)

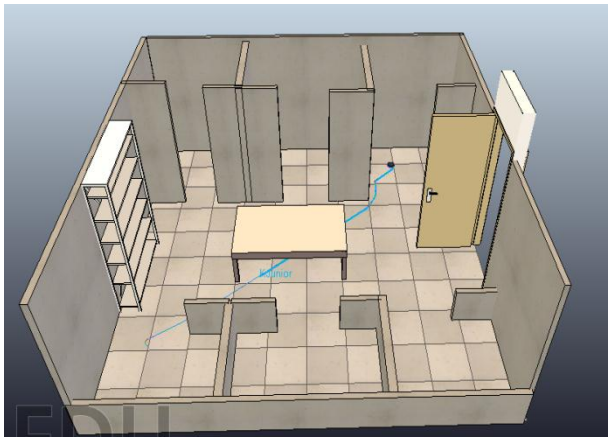


Figure 3.6 : Résultats de simulation sur V-REP PRO EDU après l'arriver du robot au point de cible et d'une situation d'évitement d'obstacles

4. SLAM : présentation générale

4.1. Les origines

Le problème de localisation et cartographie simultanées (SLAM) traite deux questions importantes dans la robotique mobile. La première question est : "Où suis-je ? ". La réponse à cette question définit la localisation du robot. La deuxième question concerne les caractéristiques de l'environnement du robot : "quoi ressemble l'environnement où je me trouve ? " [75].

Dans un système de SLAM, un véhicule robotisé placé dans un environnement inconnu, dans une position inconnue, doit construire la carte de l'environnement tout en essayant de se localiser par rapport à cette carte. Le robot dispose de plusieurs capteurs qui l'aident à récupérer les informations dont il a besoin. La réalisation de cette tâche peut paraître impossible dans la mesure où le robot a besoin d'une carte pour se localiser, mais en même temps il doit connaître sa position (se localiser) pour pouvoir construire la carte. Afin de faciliter le traitement de cette relation " poule-œuf ", les scientifiques ont unifié les deux questions précédentes en une seule question : " Où suis-je susceptible d'être dans la carte la plus probable du monde que j'ai observé jusqu'à maintenant ? ".

La formulation du problème de SLAM ainsi a permis de définir un cadre de résolution probabiliste. L'émergence du SLAM probabiliste a certainement été durant la conférence IEEE Robotics and Automation Conférence en 1986 à San Francisco, California. Plusieurs chercheurs tentaient d'appliquer des méthodes théoriques d'estimation au problème de localisation et cartographie. Les travaux de Smith et Cheeseman [76] et de Durant-Whyte [77] constituent une base des méthodes statistiques de description des relations entre les positions d'amers dans un environnement et l'estimation de l'incertitude géométrique de la carte. L'un des éléments clés de ces travaux traite du degré de corrélation entre les estimations des positions des amers dans une carte.

L'intérêt porté aux problématiques du SLAM a augmenté de manière exponentielle pendant les dix dernières années, notamment grâce aux conférences internationales (ICRA, IROS...) qui attirent de plus en plus la communauté scientifique.

4.2. Formulation du problème de SLAM

Le SLAM est composé d'un ensemble de méthodes permettant à un robot de construire une carte d'un environnement et en même temps de se localiser en utilisant cette carte. La trajectoire du véhicule et la position des amers dans la carte sont estimées au fur et à mesure, sans avoir besoin de connaissances a priori. Considérons un robot se déplaçant dans un environnement inconnu, en observant un certain nombre d'amers grâce à un capteur embarqué sur le robot. La figure 3.8 montre une illustration du problème.

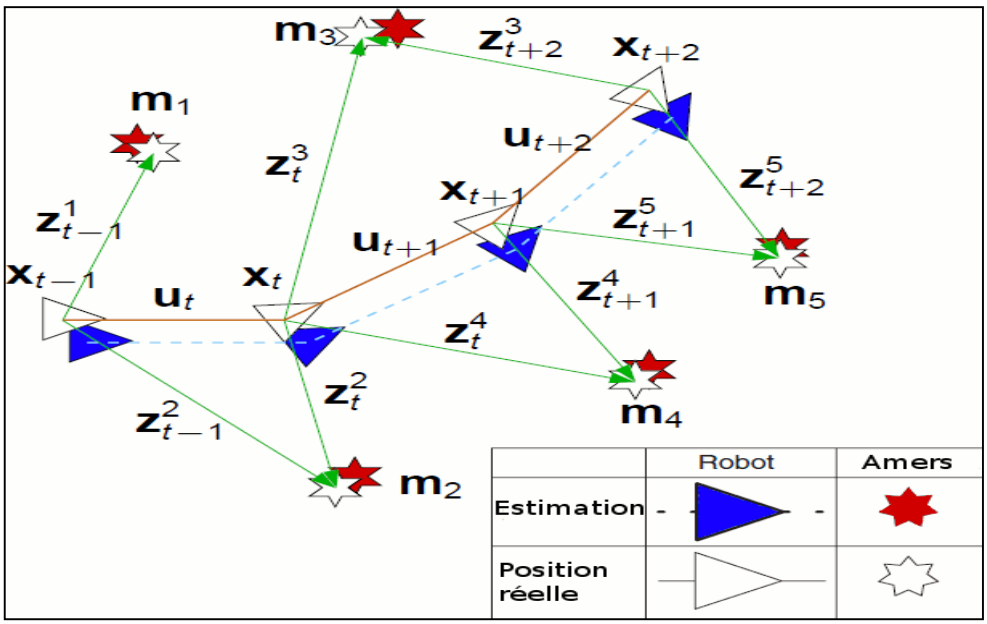


Figure 3.8 : L'idée de base du SLAM

A l'instant k on définit les quantités suivantes :

- x_k: le vecteur d'état. Il contient la position du robot
- u_k: le vecteur de contrôle. L'application de u_k à l'instant k−1 mène le robot de l'état x_{k-1} à l'état x_k
- m_i : vecteur contenant la position de l'amer i.
- z_k : l'observation à l'instant k

On définit aussi les ensembles suivants :

- $x_{0:k}$ = $\{x_0; x1;\ldots;x_k\}$ = $\{x_{0:k-1} ;x_k\}$: l'ensemble des vecteurs d'état jusqu'à l'instant k
- $u_{0:k}$= $\{u_k;u_1;\ldots;u_k\}$ = $\{u_{0:k-1} ;u_k\}$: l'ensemble des vecteurs de commande jusqu'à l'instant k
- $z_{0:k}$= $\{z_0; z_1;\ldots;z_k\}$ = $\{z_{0:k-1} ;z_k\}$: l'ensemble des observations jusqu'à l'instant k
- m = $\{m_1; m_2; \ldots;m_n\}$: la carte de l'environnement contenant une liste d'objets statiques

4.3. Résolution du SLAM

De nombreuses recherches tentent de résoudre le problème de la localisation et la cartographie simultanées, communément appelé SLAM. Les principales méthodes de SLAM sont basées sur des méthodes d'estimation statistiques. Il s'agit de filtrage statistique permettant d'estimer l'état d'un système dynamique à partir des données en provenance d'un ou plusieurs capteurs. On cherche à connaître l'état courant qui correspond le mieux aux données récupérées et, éventuellement, aux informations a priori dont on dispose. Les algorithmes développés peuvent être classés selon plusieurs critères: les types de capteurs utilisés, les méthodes de calcul adoptées, les types de cartes représentant l'environnement ... On trouve ainsi des algorithmes basés sur la vision par ordinateur en utilisant une caméra [78] ou plusieurs caméras [79], et des algorithmes qui utilisent un (ou plusieurs) capteur laser ou sonar [80]. Concernant les méthodes de calcul, deux grandes familles existent. D'une part, les algorithmes basés sur l'utilisation du filtrage de Kalman [81], et d'autre part, on trouve des algorithmes utilisant des filtres à particules [82], ou des filtres particulaires Rao-Blackwellizés un mélange de filtrage particulaire et de filtrage de Kalman - comme FastSLAM [83]. On peut également classifier les systèmes de SLAM suivant l'environnement de travail. Certaines recherches ont porté sur la comparaison des différents algorithmes au niveau des performances et de la vitesse de calcul [84]. Les résultats de ces comparaisons montrent que le problème n'est toujours pas universellement résolu.

4.3.1. Représentation de la carte

Le choix de la représentation de la carte de l'environnement est une étape importante dans le SLAM. Dans [85] l'auteur analyse trois approches fondamentales de représentation de l'environnement :

- L'approche directe [86]
- L'approche basée sur les caractéristiques géométriques (feature-based) [87]
- L'approche basée sur une grille d'occupation (grid-based) [88]

La carte de l'environnement peut aussi être représentée par une approche topologique. Mais cette méthode n'est pas analysée, dans la mesure où elle est basée sur un partitionnement des cartes de types feature-based ou grid-based en régions cohérentes.

4.3.1.1. Approche directe

La méthode de représentation directe de la carte de l'environnement est généralement adaptée à l'utilisation des capteurs Laser. Cette méthode utilise les données brutes des mesures du capteur pour représenter l'environnement sans aucune extraction d'amers ou de caractéristiques prédéfinies (lignes, coins . . .).

Dans le cas d'un capteur laser, chaque mesure est constituée d'un ensemble de points d'impact du faisceau laser sur les objets de l'environnement. On peut ainsi construire une carte simplement en superposant les différents points de mesure. On obtient ainsi une représentation en nuage de points.

4.3.1.2. Approche feature-based

Cette méthode réduit les données des mesures en caractéristiques prédéfinies. On utilise généralement des primitives géométriques, comme des lignes, des cercles ou des coins. La création de la carte consiste ensuite à estimer les paramètres des primitives afin qu'ils correspondent au mieux aux observations. Pour détecter les caractéristiques géométriques, plusieurs méthodes existent. Les plus connues sont :

- La méthode split-and-merge [89] pour la détection des segments de lignes
- La transformation de Hough [90] pour la détection des lignes ou des cercles
- RANSAC [91] pour la détection des lignes ou des cercles aussi

La figure 3.9 montre un exemple d'une carte feature-based. Ce type de cartes est limité aux objets et formes modélisés et prédéfinis. Il est donc incompatible avec les environnements trop complexes et non structurés. Contrairement à la méthode directe de représentation, où on reproduit fidèlement la structure de l'environnement, les approches feature-based sont des approximations seulement.

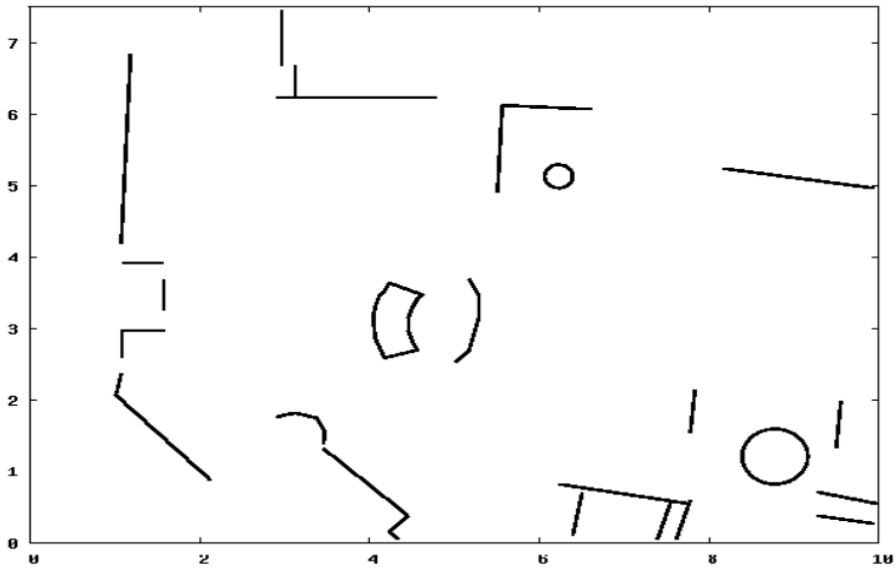


Figure 3.9 : Carte basée sur l'extraction de caractéristiques géométriques de l'environnement

4.3.1.3. Approche grid-based

Les grilles d'occupation ont été introduites par [88]. Dans cette représentation, l'environnement est divisé en cellules rectangulaires. La résolution de la représentation de l'environnement dépend directement de la taille des cellules. En plus de cette discrétisation de l'espace, une mesure de probabilité d'occupation est estimée pour chaque cellule indiquant si celle-ci est occupée ou non.

La mise à jour de l'état de chaque cellule se fait à la réception de nouvelles données.

On trouve dans la littérature plusieurs méthodes pour réaliser cette opération tell que le filtrage bayésien ou La théorie de Dempster-Shafer ou bien La logique floue. Donc La figure 3.10 présente un exemple d'une carte basée sur une grille d'occupation.

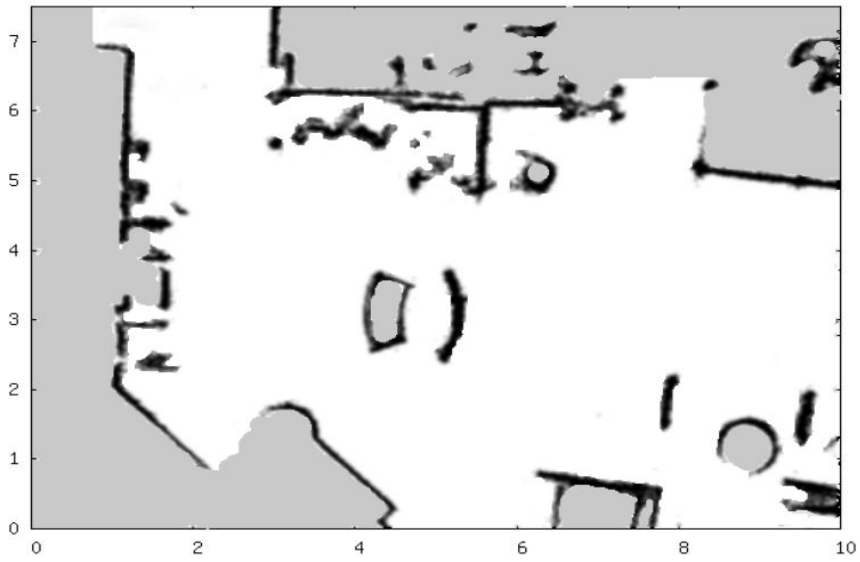


Figure 3.10 : Carte à base d'une grille d'occupation

Les cellules en blanc représentent l'espace libre, tandis que les cellules en noir indiquent la présence d'obstacles. L'approche basée sur une grille d'occupation est efficace lorsqu'il s'agit de modéliser l'incertitude ou fusionner les mesures de différents capteurs.

4.5. Résultats de simulation

Pour montrer l'efficacité de l'approche proposée (SLAM), une simulation a été réalisée en employant des environnements arbitrairement sur ROS (Robot Operating System) et en utilisant une caméra kinect de Xbox 360. Toutes les simulations ont été faites dans l'environnement de Rviz et RTAB – Map.

RTABMAP ou RTAB-Map (cartographie basée sur l'apparence en temps réel) est une approche SLAM basée sur un graphique RGB-D basée sur un détecteur de fermeture de boucle basé sur l'apparence incrémentielle. Pour SLAM, seules les données d'odométrie RTABMAP sont utilisées.

Donc premièrement on a installé ROS sur ubuntu (linux) et puis on a installé RTAB-Map, mais pour montrer les SLAM il faut utiliser des requêtes sur le terminal d'ubuntu qui sont :

1) roslaunch freenect_launch freenect.launch depth_registration:=true
2) roscore
3) roslaunch rtabmap_ros rtabmap.launch
4) Rviz

- La 1ère requête pour installer les drivers de kinect et assurer qu'elle est connectée au pc,

- La $2^{ème}$ Il faut la lancer dans un autre terminal c'est pour lancer ROS mais n'est pas obligatoire tan qu'on a lancé ronslaunch dans la première,
- La $3^{ème}$ c'est pour lancer l'application RTAB-Map,
- Pour La $4^{ème}$ c'est pour lancer Rviz et pour commencer la configuration et puis les simulations.

Comme un premier travail, nous avons essayé de faire la reconstruction de la carte de navigation. La figure 3.11 présente une carte d'une chambre obtenue au cours de nos expériences.

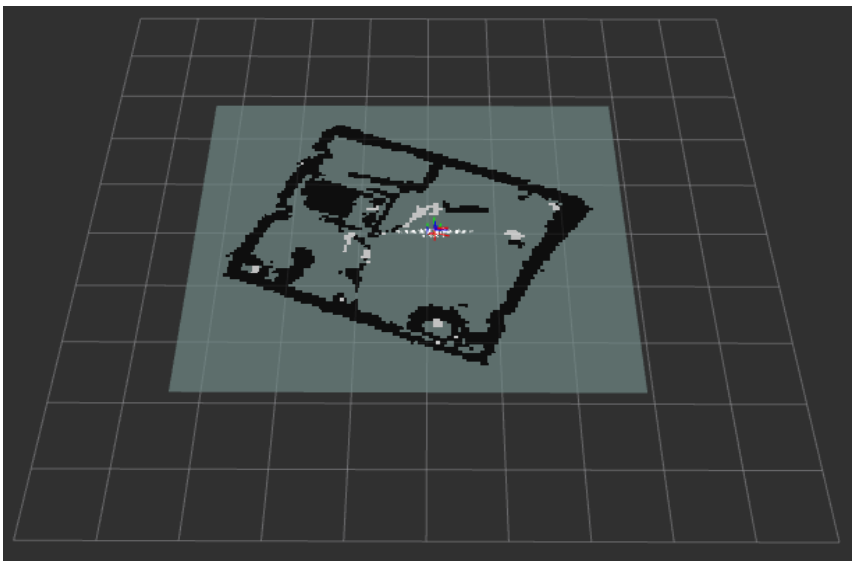


Figure 3.11 : Une carte d'une chambre obtenue au cours de nos expériences.

Après, nous avons pu reconstruire l'environnement (indoor) en 3D par la technique de SLAM comme il est indiqué dans la figure 3.12.

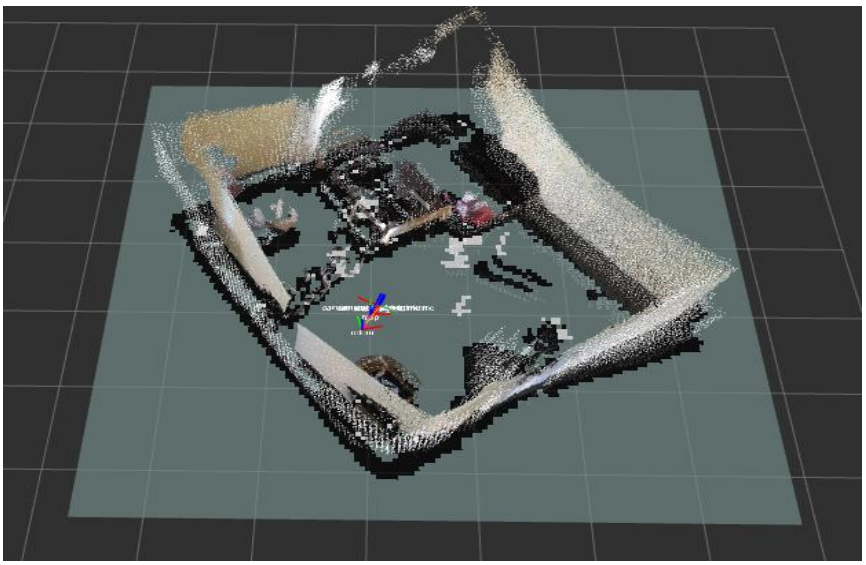


Figure 3.12 : La carte construite en utilisant SLAM

Ou par une simulation directe pour montrer les SLAM et la reconstruction de l'environnement en 3D sur Rviz par l'utilisation d'une caméra kinect. Donc la figure 3.13 représente les premiers résultats de SLAM.

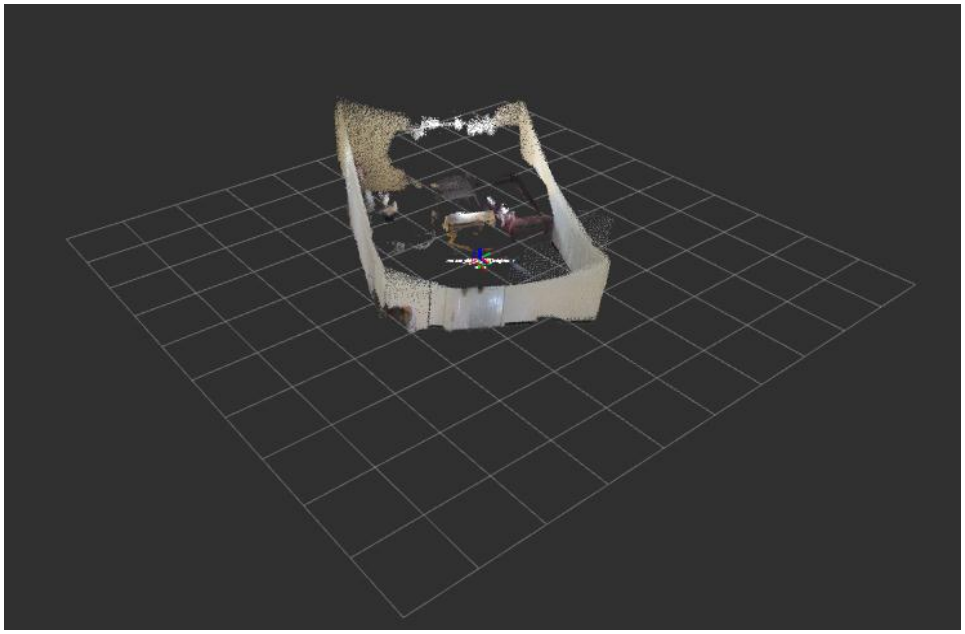


Figure 3.13 : Résultat de simulation du SLAM pour la reconstruction de l'environnement en 3D

5. Conclusion

Dans ce chapitre, un contrôleur comportemental à base d'Odométrie a été conçu pour réaliser la navigation autonome d'un robot mobile dans des environnements inconnus encombrés d'obstacles. L'idée du contrôle comportemental est basée sur la décomposition de la tache de navigation globale en un ensemble de comportements élémentaires tels que la navigation vers la cible et l'évitement d'obstacles,

Les résultats de simulation présentés dans ce chapitre montrent l'efficacité du contrôleur proposé surtout dans les environnements moins compliques, et aussi sur la technique de représentation de la carte de navigation et de la reconstruction de l'environnement (SLAM).

Le chapitre suivant donne une solution prometteuse pour la réalisation de ce robot mobile et puis l'implémentation de ce dernier dans un environnement de travail.

Chapitre IV : Réalisation et Implémentation

1. Introduction

Dans ce chapitre, on s'intéresse à la réalisation du robot et à l'implémentation de l'étude théorique dans notre modèle.

Au cours de ce travail et selon le matériel disponible, nous avons pu réaliser deux robots mobiles, le premier nous aide à montrer la perception et les SLAM (simultaneous localization and mapping), le deuxième a pour but de montrer l'algorithme d'odométrie (ça veut dire la localisation) et le déplacement vers la cible.

D'abord, nous allons présenter les différents pièces que nous aurons besoin et quelles fonctionnalités que nous voulions mettre en œuvre dans notre robot.

Puis on passe à la conception des circuits des différentes parties du robot où on va montré comment raccorder et tester les composantes du robot.

Nous allons introduire l'environnement de développement Arduino. Ensuite, on va élaborer un software complet pour assembler toutes les croquis Arduino dans un seul sketch, l'étape qui va nous facilite la manipulation dans le programme final sur Arduino et Raspberry pi.

2. Conception du robot

Ce schéma montre la structure globale qu'on va suivre pour la réalisation de nos robots différentiels (Fig.4.1).

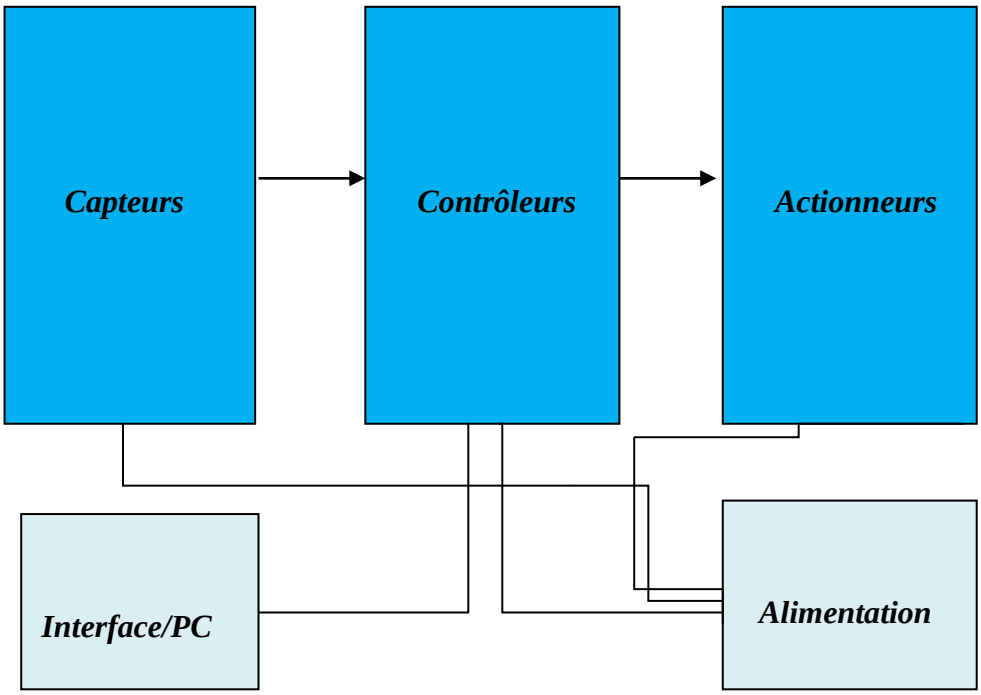


Figure 4.1 : Schéma des différentes unités du robot

3. Cahier de charge

Notre objectif est de réaliser un robot à vitesses différentielles qu'il a les caractéristiques suivantes :

- Il peut effectuer les trois mouvements suivants (marche avant, marche arrière, tourné dans les deux directions)
- Il devrait être facile à monter et à démonter.
- Il devrait y avoir un mode de déplacement avec la capacité de détecter des obstacles avant et les éviter.
- Il devrait avoir un mode de déplacer d'un point de départ à un point de cible à l'aide des capteurs de vitesse disposés dans les roues.
- Il devrait avoir un mode dans lequel il peut être entièrement contrôlé à partir d'un Smartphone.
- Il devrait avoir un mode dans lequel il peut naviguer et localiser d'une manier autonome a l'aide d'une caméra Kinect (Xbox 360).
- Il devrait être facile de manipuler, modifier et améliorer.

On peut déjà remarquer l'émergence de quatre (04) modes :

1) Mode de Perception (robot intelligent)
2) Mode Remote controlled (RC)
3) Mode de navigation autonome (Localisation par la technique de l'odométrie)
4) Mode SLAM (simultaneous localization and mapping)

4. Conception matérielle et mécanique

Pour la plate-forme hardware, on a décidé de choisir le microcontrôleur Arduino , et un mini-ordinateur (Raspberry pi) avant de commencer l'élaboration de la base mécanique de robot, c'était le choix optimal pour notre robot car on peut facilement agencer les autres composantes autour de lui grâce à son plateforme d'entrées/sorties permettant de dialoguer (en temps réel) et exécuter les commandes, et aussi pour son interface qui est largement utilisé partout dans le monde ce qui signifie qu'il y'a beaucoup d'informations et de ressources disponibles.

Avec la liste des fonctionnalités qu'on a établies au-dessus, on peut aller chercher les pièces nécessaires pour la construction de notre robot.

Voici l'ensemble des composants principale qu'on va utiliser pour la conception des deux robots mobiles :

- Carte Arduino Uno ou Mega.
- Motor Driver (L293N).
- Capteur de distance (Capteur ultrason HC-SR04).
- Servomoteur SG90.
- Module Bluetooth (HC-05).
- Plaque d'essai (Breadboard) et câbles.
- Les moteurs à CC.
- Source d'alimentation (batteries).
- Caméra Kinect de Xbox 360.
- Capteurs de Vitesse LM393 (Encodeurs).
- Kit de véhicule (châssis et roues).

4.1. Carte Arduino UNO ou MEGA

La carte Arduino Uno (ou MEGA) sera le cerveau du robot, car il sera en cours d'exécution le composant qui permettra de contrôler toutes les autres parties. (Fig.4.2)

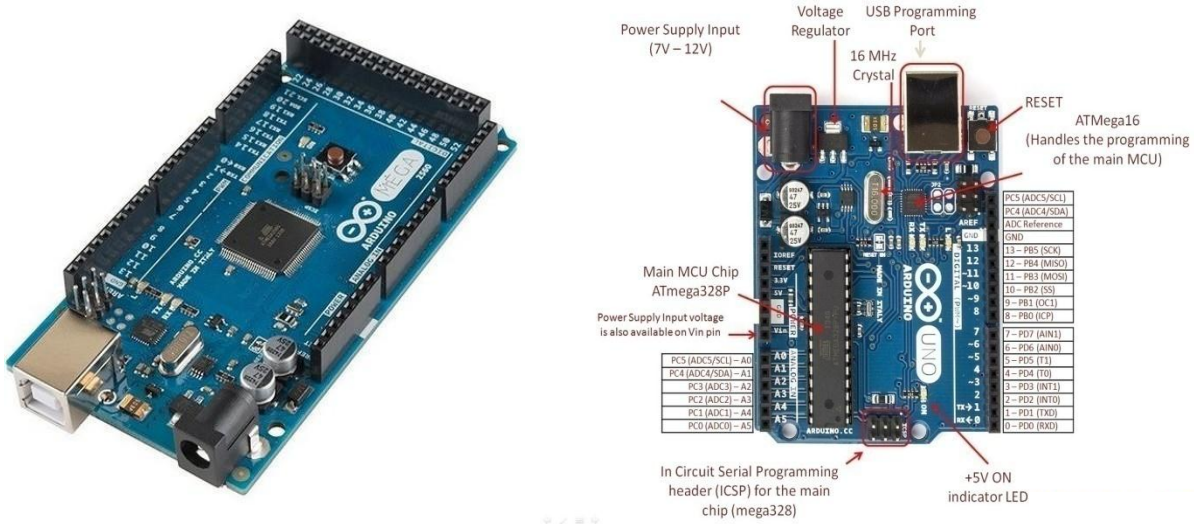


Figure 4.2 : la carte Arduino MEGA et UNO

Le système Arduino est conçu d'une plateforme Open Source installée sur une carte programmée à microcontrôleur AVR permettant l'écriture, la compilation et le test d'un

programme. Les cartes et modules Arduino sont conçus avec des entrées/sorties qui reçoivent des signaux de capteurs ou interrupteurs qui peuvent eux-mêmes commander des moteurs et éclairages par exemple. [92]

4.2. Raspberry pi 3 Model B

Un Raspberry PI est une sorte de mini-ordinateur qui est doté d'un processeur, de mémoire vive, d'alimentation, de ports divers le tout fixé sur une carte mère. Voici le détail de ce qu'on peut retrouver sur un Raspberry PI 3 modèle B par exemple (Fig.4.3), (le matériel varie en fonction de la version que vous prenez) :

- processeur Quad-Core ARM Cortex-A53 1.2 GHz
- 1024 Mo de RAM
- GPU Dual-Core VideoCore IV
- 4 ports USB 2.0
- 1 sortie HDMI
- 1 sortie Ethernet
- 1 prise jack
- Bluetooth/Wifi 802.11 b/g/n

On a utilisé ce Raspberry pi Model B pour but de montrer le Mode SLAM (simultaneous localization and mapping) avec l'utilisation d'une caméra kinect de Xbox 360.

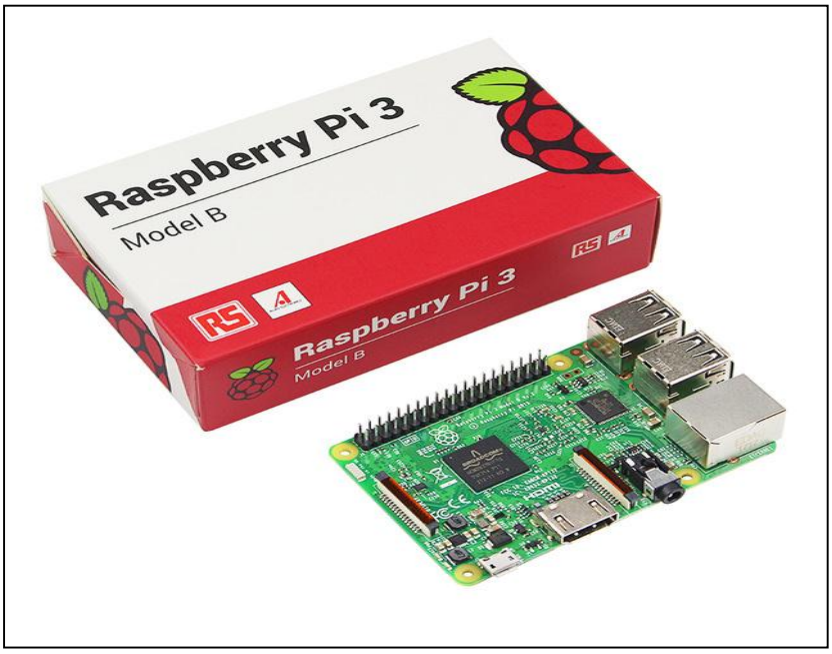


Figure 4.3 : Raspberry pi 3 Model B [99]

4.3. Driver Moteur L298N

La carte Arduino ne peut pas contrôler directement un moteur. La partie délicate est d'être en mesure de faire fonctionner le moteur de manière sélective vers l'avant ou vers l'arrière, ce qui nécessite l'échange des entrées d'alimentation et de masse dans le moteur [15].

On utilise dans ce cas dans notre projet le driver moteur L298N qui est le plus utilisé dans les projets, et très facile à mettre en œuvre [15].

Le module L298N permet de commander séparément 2 moteurs à courant continu DC de 3 à 30 V, ou un moteur pas à pas bipolaire (2 phase step motor). Il fonctionne avec une interface de commande TTL 5V (donc compatible avec Arduino ou avec de nombreux autres microcontrôleurs en basse tension). (Fig.4.4)

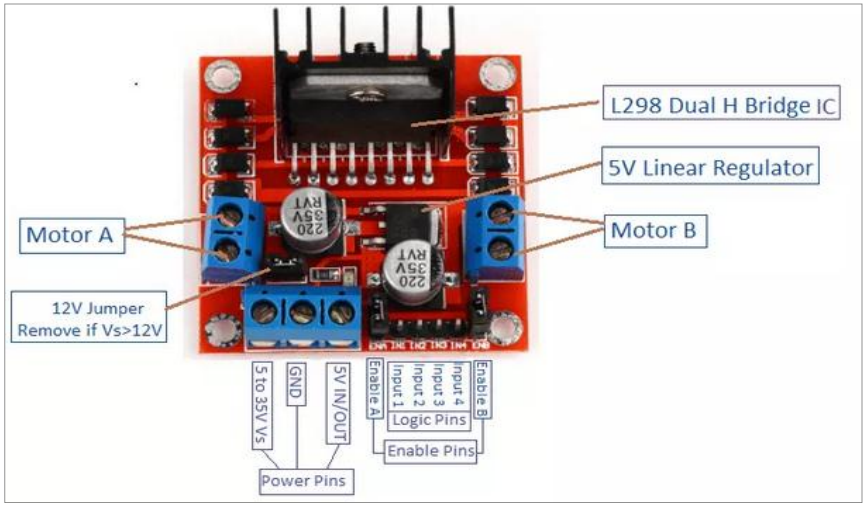


Figure 4.4 : Driver Moteurs L298N

4.4. Capteur de distance (Ultrasonique)

On va utiliser ce dispositif pour empêcher le robot de frapper les murs ou d'autres obstacles sur son chemin.

Nous avons choisi le capteur ultrason HC-SR04, car il dispose simplement de 4 pins: VCC, TRIG, ECHO, GND. Il est donc très facile de l'interfacer à un microcontrôleur (Figure.4.5).

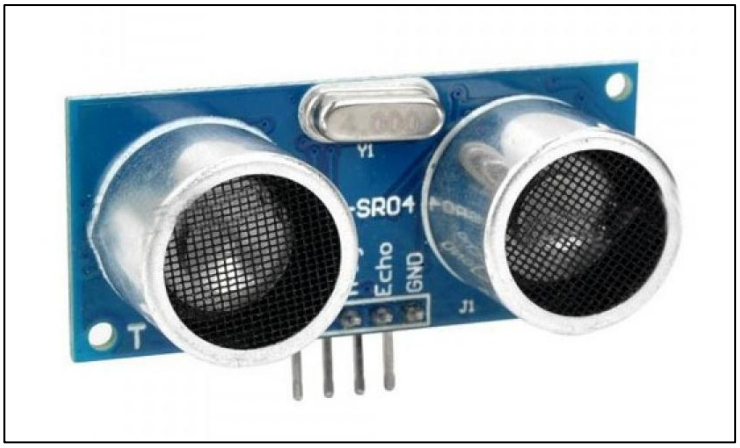


Figure 4.5 : Capteur Ultrason HC-SR04

Le fonctionnement du module est le suivant :

Il faut envoyer une impulsion niveau haut (5v) pendant au moins 10 µs sur la broche 'Trig Input' cela déclenche la mesure. En retour la sortie 'Output' ou 'Echo' va fournir une impulsion + 5v dont la durée est proportionnelle à la distance si le module détecte un objet. Afin de pouvoir calculer la distance en cm, on utilisera la formule suivante [15] :

Distance = (durée de l'impulsion) (en µs) / 58 (basée sur la formule de la vitesse du son)

4.5. Servomoteur

Le servomoteur est un petit moteur avec un réducteur intégré qui permet de faire la rotation d'un axe sur 180°. Il est composé de trois pins, deux pour l'alimentation et un pour le signal.

Dans la plupart des servomoteurs, le signal envoyé correspond à une impulsion comprise entre 1 et 2 millisecondes toutes les 20 millisecondes. La durée du signal correspond à un angle entre 0 et 180° (fig.4.6). [93]

Figure 4.6 : Servomoteur SG90

4.6. Module Bluetooth

La meilleure façon de contrôler le robot à partir d'un Smartphone est via l'interface Bluetooth qu'il est disponible dans tous les Smartphones. Le téléphone va agir comme émetteur de commandes, donc on aura besoin d'un récepteur Bluetooth dans le robot.

Le HC-05 agit comme un port série à travers lequel nous pouvons envoyer et recevoir des données. Donc, en utilisant un terminal série ou une application Bluetooth personnalisée sur l'ordinateur ou par téléphone, nous pouvons contrôler et surveiller notre robot (Fig.4.7) [15].

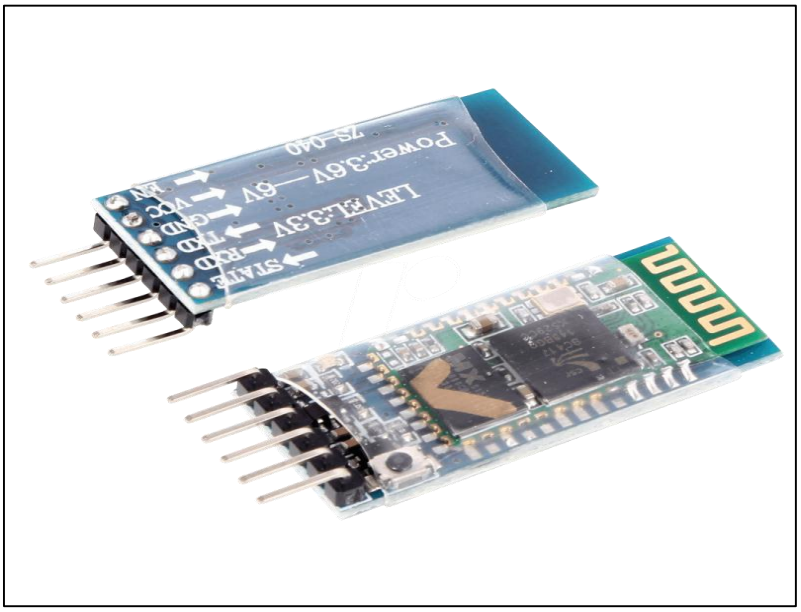


Figure 4.7 : Module Bluetooth HC-05 [100]

4.7. Plaque d'essais et câbles

L'une des restrictions que nous avons décidé d'imposer dans ce projet est que nous ne ferions pas des opérations de soudage, de sorte que nous pouvons assembler et à démonter le robot sans détruire les pièces.

Ensuite, il nous fallait une plate-forme où l'on peut facilement connecter tous les composants ensemble. Pour ce genre de tâche, nous avons utilisé une plaque d'essai (Breadboard).

C'est un dispositif qui permet de réaliser le prototype d'un circuit électronique et de le tester. L'avantage de ce système est d'être totalement réutilisable, car il ne nécessite pas de soudure. (Fig.4.8).

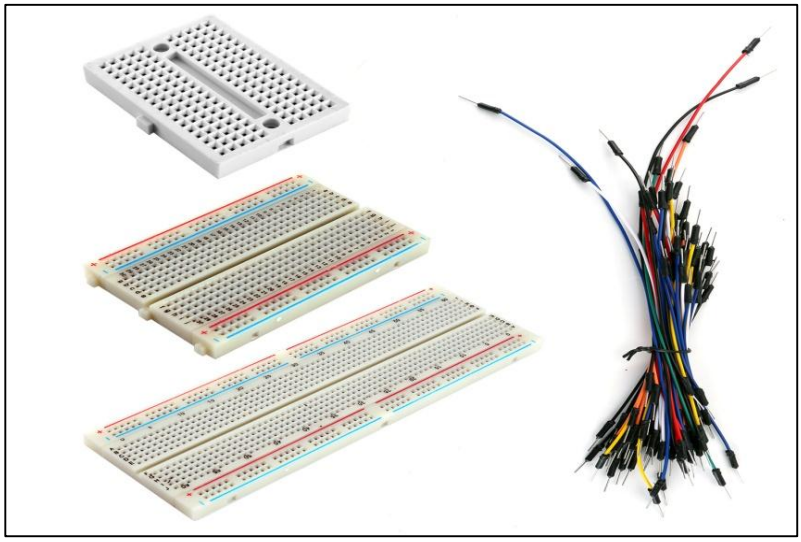


Figure 4.8 : plaque d'assai (Breadboard) et Jumpers

4.8. Moteurs à courant continu (CC)

Parce qu'on a décidé de réaliser un robot différentiel, alors nous avons utilisé deux moteurs à courant continu pour commander les deux roues arrière.

Évidemment, pour pouvoir valider un moteur, il faut connaître les spécifications que nous voulons atteindre.

Pour notre projet, nous avons utilisé des moteurs réducteurs à engrenages à axe unique DC 12V sur le premier robot, et des moteurs à courant continue DC 3V-6V pour le deuxième robot. (Fig.4.9).

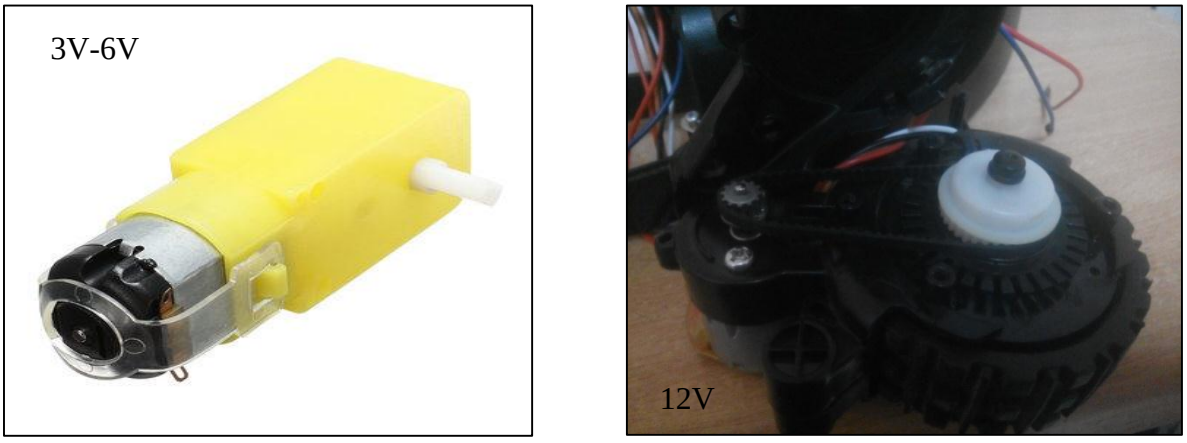


Figure 4.9 : Moteurs DC 3V-6V et des moteurs DC de 12V

4.9. Batteries d'alimentation

Pour alimenter notre robot on va utiliser plus de trois batteries pile Li-ion rechargeable de 3.7V brancher en série entre eux. On a aussi utilisé des batteries des téléphones portables avec un branchement en série entre ces batteries (Fig.4.10).

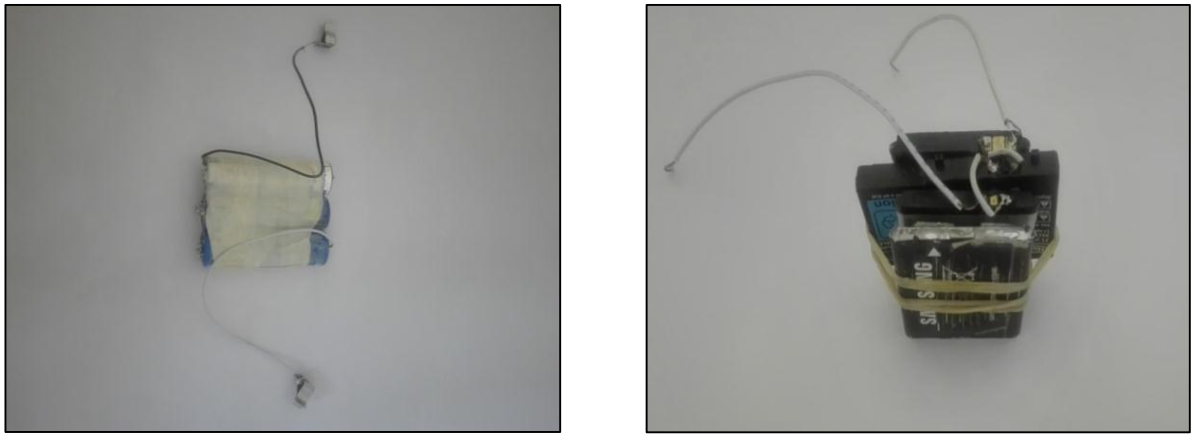


Figure 4.10 : Batteries d'alimentation

4.10. Châssis des deux robots mobiles

Il y a beaucoup de choix pour notre corps de robot. Nos seules exigences étaient qu'il disposait d'une grande plate-forme où toutes les parties peuvent être montées et qu'il est compatibles avec l'emplacement des roues et des moteurs. Sans être très sophistiqué, le châssis définit le type de stratégie choisie face à la structure globale utilisée, lors de la création de notre robot. Il faut organiser tous les éléments pour optimiser le robot : centre de gravité, position des capteurs, moteurs, pneus, batteries, cartes Arduino, plaque d'essaie. Il existe plusieurs solutions possibles ayant chacune ses avantages et inconvénients.

Dans ce cas On a utilisé deux châssis le premier est un châssis pour un robot aspirateur et le deuxième est un kit de Châssis de Robot 2WD Basic (Acrylique), (fig. 4.11).

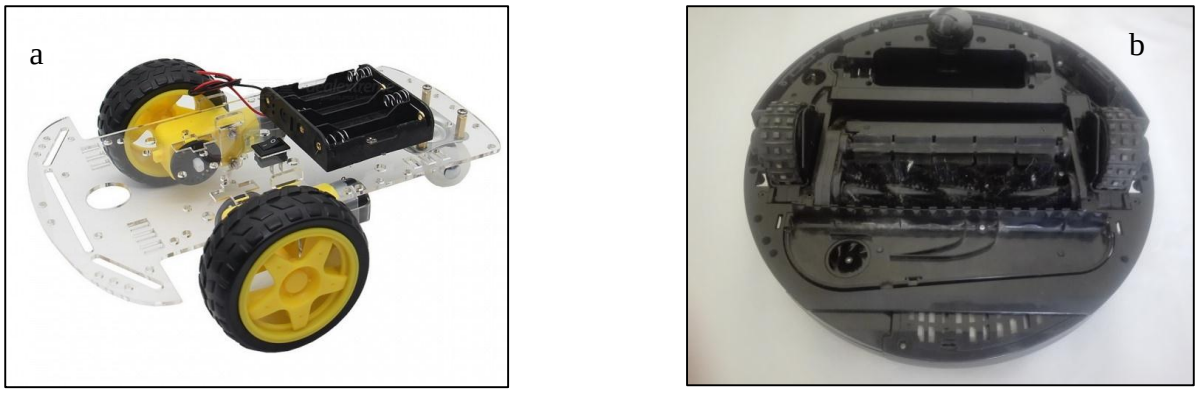


Figure 4.11 : (a) châssis d'un robot mobile unicycle (2WD), et (b) le châssis d'un robot aspirateur unicycle

4.11. Caméra Kinect de Xbox 360

Pour l'instant, nous présentons la Kinect comme une caméra (Figure 4.12), mais on pourrait utiliser le terme anglais "RGB-D camera" puisqu'il y a un objectif couleur (RGB ou rouge vert bleu) et un deuxième objectif permettant de déterminer la profondeur (D pour "depth" qui est le terme anglais pour profondeur).

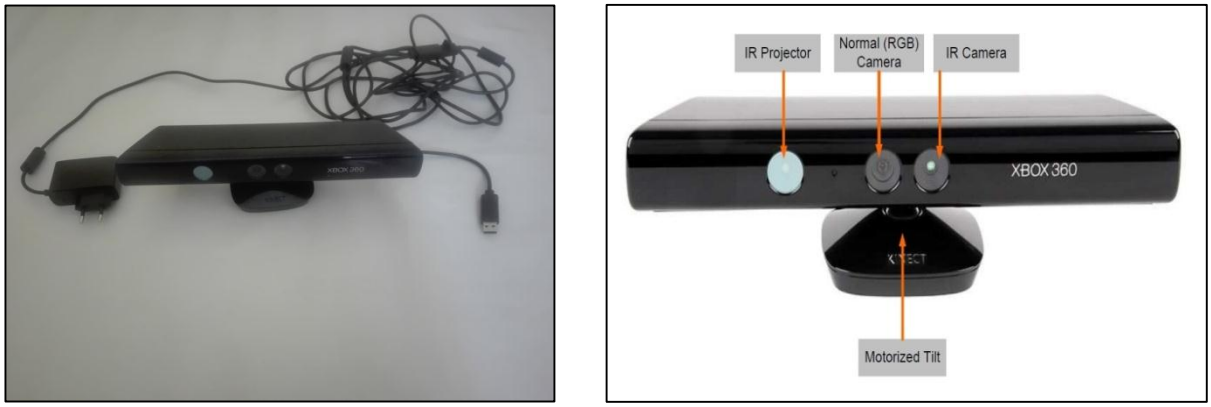


Figure 4.12 : kinect Xbox 360

4.12. Capteur de vitesse par fourche optique

Ce capteur de vitesse prêt à l'emploi utilise une fourche optique. Il est largement utilisé dans la détection de la vitesse de moteurs, le comptage d'impulsions et la détection de positions limites.

Dans ce cas-là, on a utilisé un capteur de vitesse LM393 pour chaque roue disposée sur le châssis du robot 2WD basic (Acrylique), pour but d'estimer sa position (Fig.4.13).

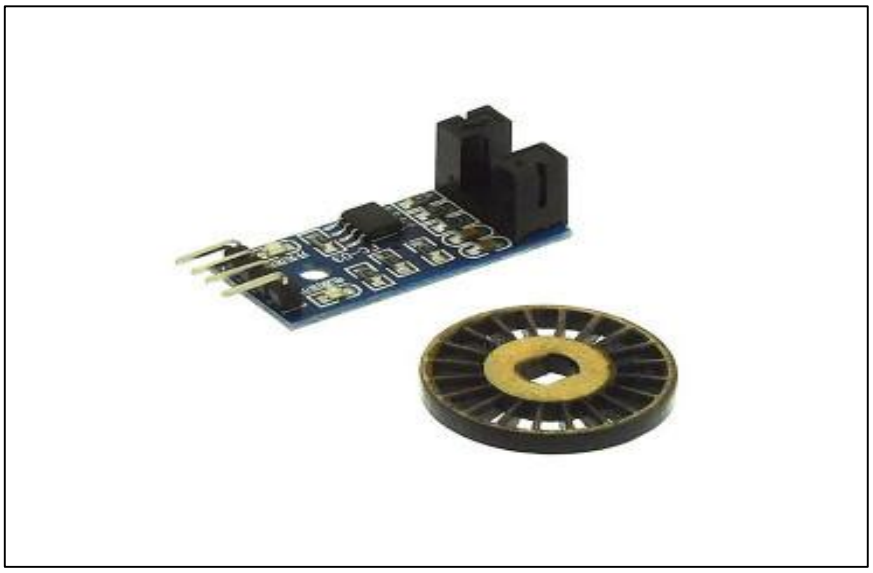


Figure 4.13 : Capteur de vitesse LM393

5. MODE 01 : Perception (Intelligent & Obstacle avoidence)

Dans cette partie on s'intéresse à la réalisation d'un robot qui sera capable de déplacer tout en évitant les obstacles. On a choisi un capteur de distance du type Ulstrasonic HC-SR04. Ce rebot est constitué d'un capteur Ultrason avec un servomoteur, une carte Arduino, deux moteurs à courant continue de 12V, un driver moteur (pont H L298N), et une batterie.

5.1. Circuit du robot intelligent

Dans ce circuit on doit suivre le câblage dans le schéma suivant pour réaliser la tache de mode 01, ce schéma est réaliser sur logiciel Fritzing (Fig.4.14).

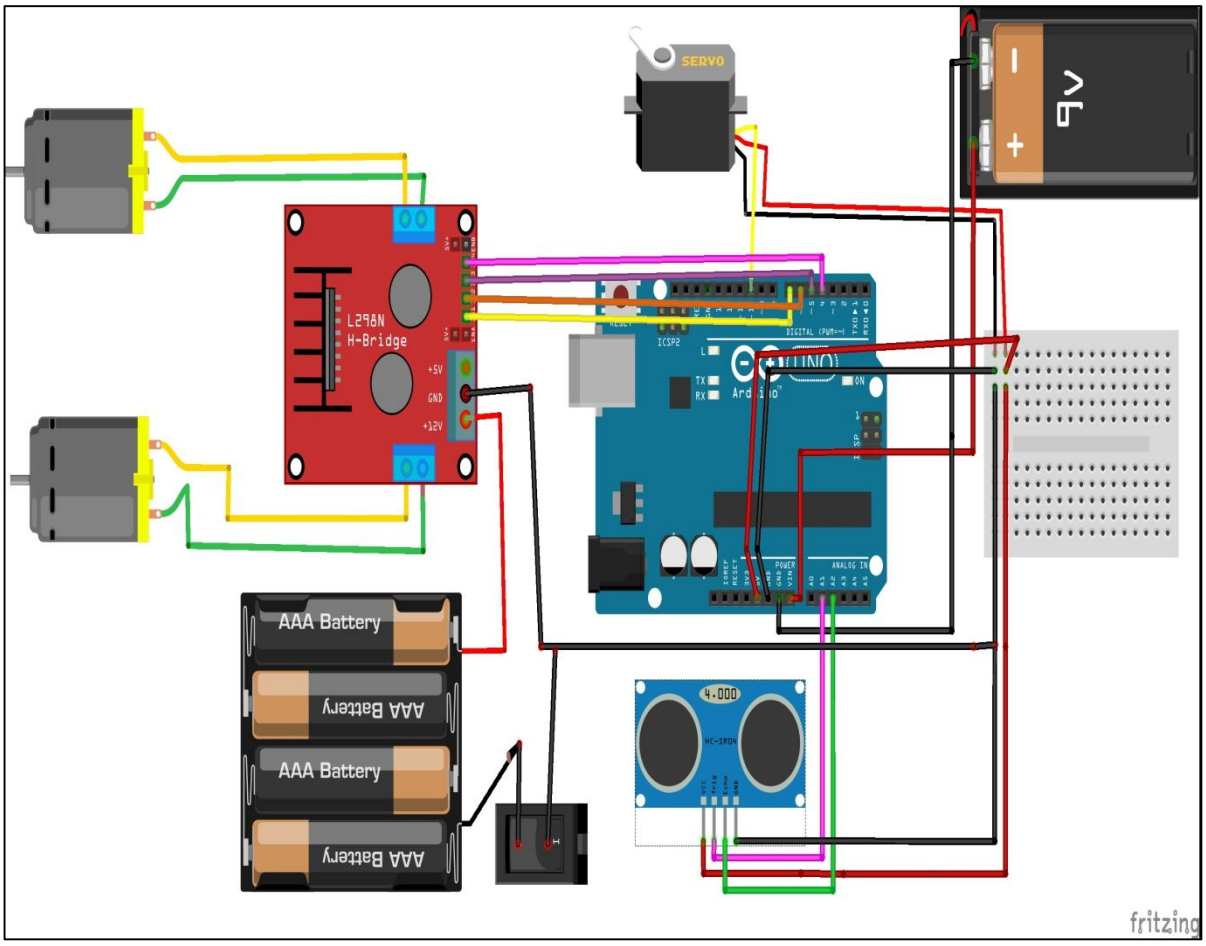


Figure 4.14 : Circuit du robot intelligent par Fritzing

Dans cette réalisation (mode 01) on a décidé de choisir le châssis du robot aspirateur pour intégrer les capteurs.

Le résultat que nous avons obtenu peut être observé sur la figure.4.15, qui représente la plateforme mobile réellement obtenue.

Nous obtenons donc un résultat satisfaisant, soit une plateforme qui est symétrique. Nous obtenons également une structure légère.

Figure 4.15 : La plateforme de robot mobile

6. MODE 02 : Remote controlled via Bluetooth

Pour améliorer la fonctionnalité et l'utilisation du robot nous avons ajouté un module Bluetooth pour lui commander et le contrôler par une application sur un Smartphone à une distance limité.

6.1. Circuit du RC via Bluetooth :

Nous avons ajouté quelques modifications sur le mode 1 pour arriver à contrôler un robot mobile par un Smartphone.

Pour cela, nous avons conservé le montage des moteurs avec le pont H (driver Motors L298N) tout comme le robot qui évite les obstacles (Mode 1). Après, nous avons intégré le Bluetooth dans le circuit comme il est indiqué sur la figure (Fig.4.16).

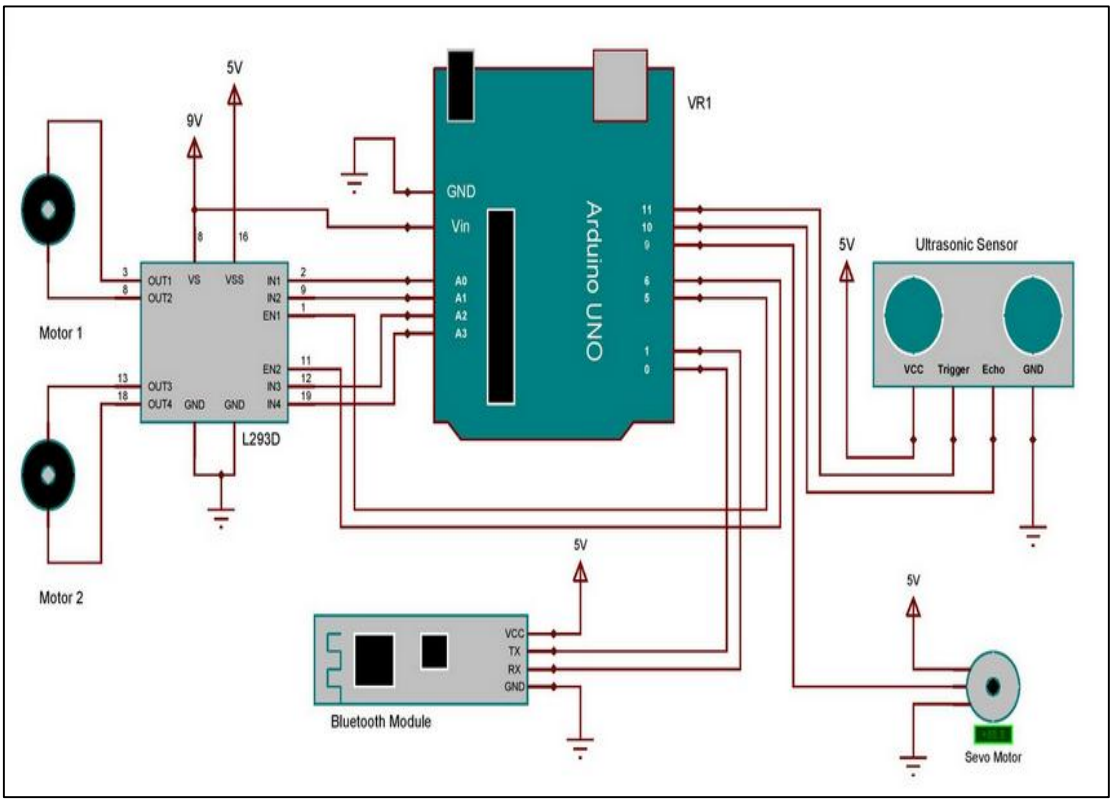


Figure 4.16 : Montage Robot évitant les obstacles et contrôlée par Bluetooth

6.2. Implémentation du robot dans un environnent de travail

Avec l'utilisation d'une application qui est disponible sur Play store, on a arrivé à contrôler le robot mobile par un Smartphone pour lui commander à distance et de contrôler sa direction. Les essais réalisés sur ce robot dans un environnement de travail limité, nous donnent des résultats satisfaisants (Fig.4.17).

Figure 4.17 : Essais du robot mobile contrôlé par une application Android

Pour cela, deux modes sont créés sur l'application Android (Pathfinder), un mode manuel pour contrôler le robot mobile par un Smartphone Android et un mode automatique pour lui laisser naviguer tout seul avec le capteur Ultrason (Fig.4.18).

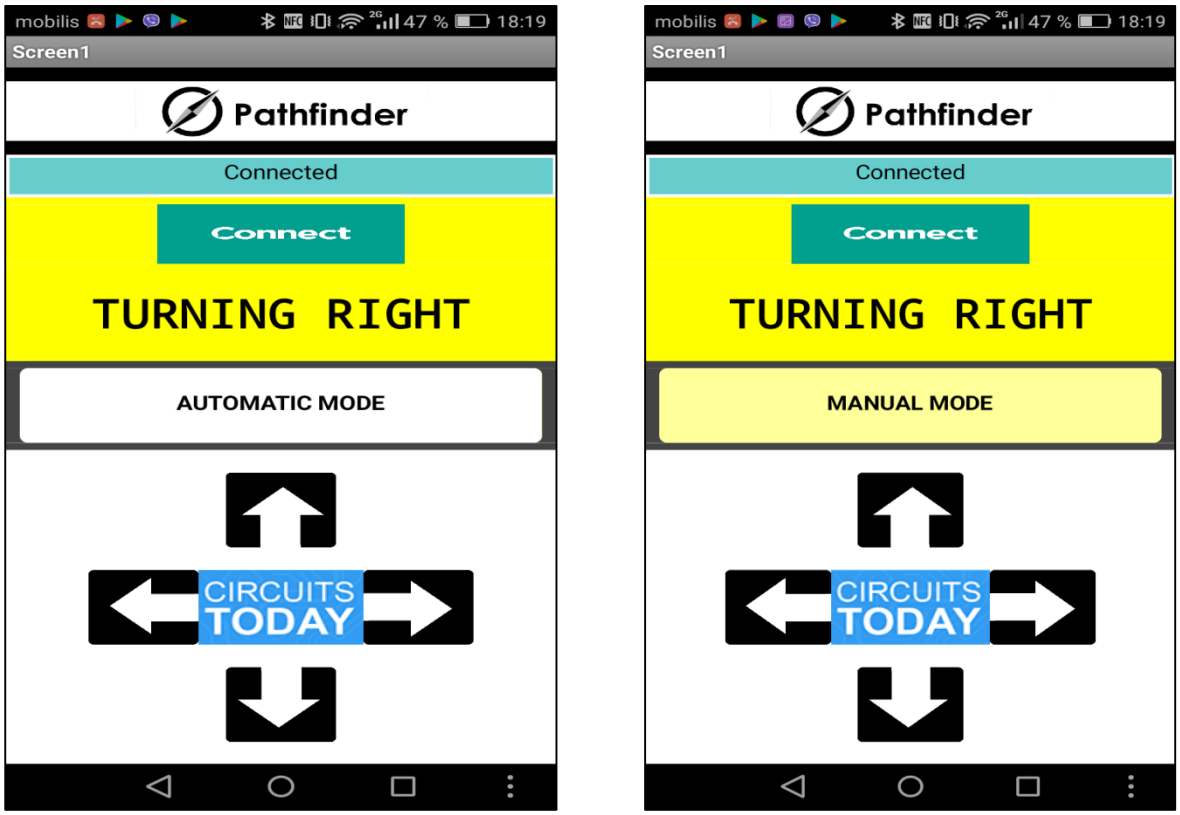


Figure 4.18 : Application de communication entre Android et Arduino via Bluetooth

7. MODE 03 : Localisation (L'odométrie)

Dans cette partie on s'intéresse à la réalisation d'un robot mobile qui sera capable de déplacer d'un point A (point de départ) à un point B (point de cible) d'une manière autonome (sans intervention humaine). Nous avons utilisé deux capteurs de vitesse par fourche optique de type LM393 disposés sur les roues. Ce robot est constitué d'une carte Arduino Méga, deux moteurs à courant continue, un driver moteur (pont H L298N) et une batterie.

7.1. Circuit pour la localisation du robot (L'Odométrie)

Dans ce circuit nous avons intégré les deux capteurs de vitesse LM393 pour calculer le nombre tour qui est parcourus par chaque roue (pour but estimer la position du robot).

Dans cette réalisation (mode 03) nous avons décidé de choisir le châssis du robot 2WD basic (Acrylique), à cause de la disponibilité de l'emplacement pour les deux capteurs de vitesse LM393 sur ce châssis, ce schéma est réalisé sur Fritzing (Fig.4.19).

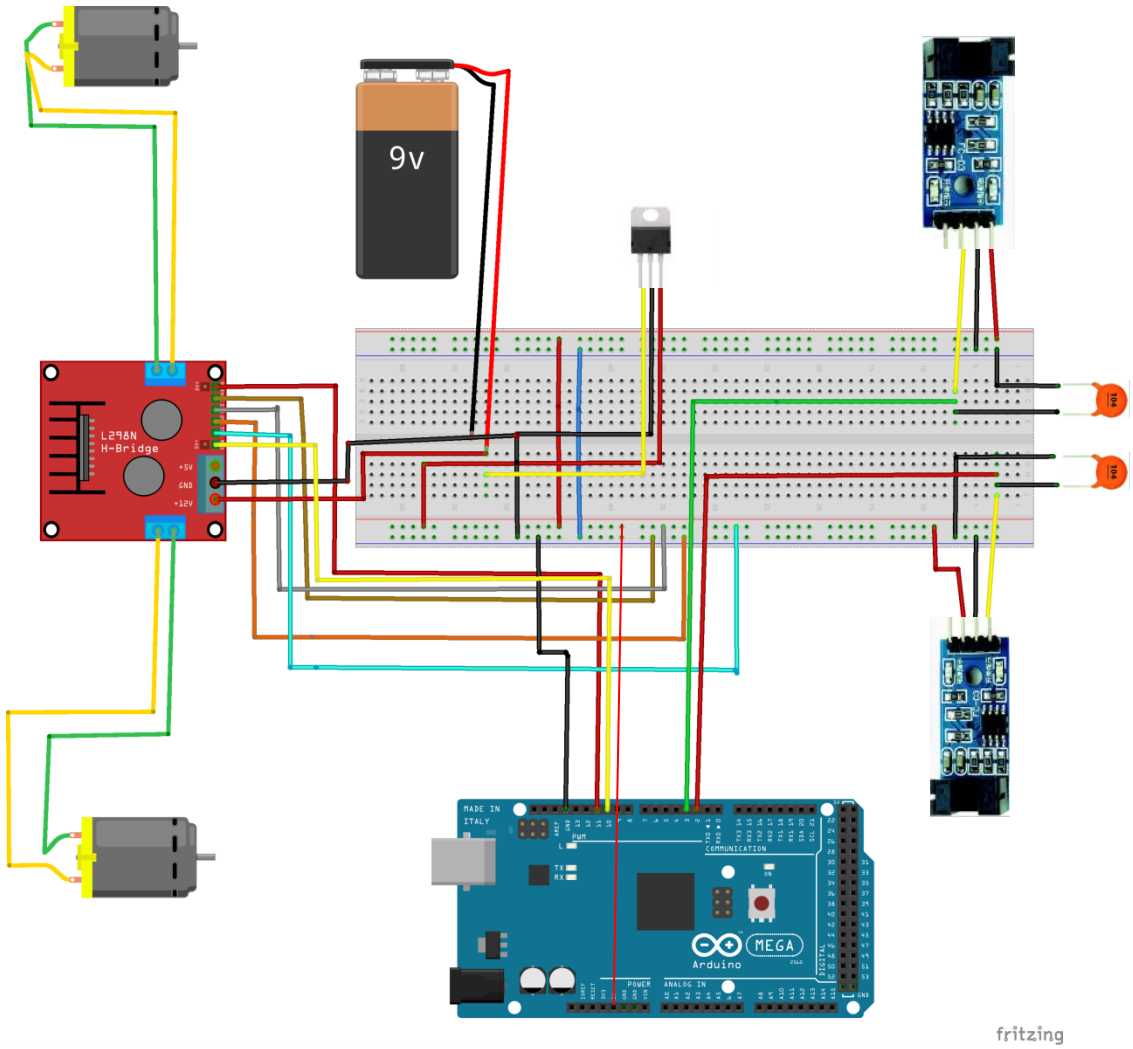


Figure 4.19 : Schéma pour la localisation du robot mobile sur Fritzing (l'odométrie)

7.2. Simulation en temps réal sur MATLAB

Avant l'implémentation du robot sur le terrain nous avons pu faire une simulation en temps réal sur MATLAB et Arduino pour voire les résultats de l'algorithme d'odométrie sur le robot (Fig.4.20).

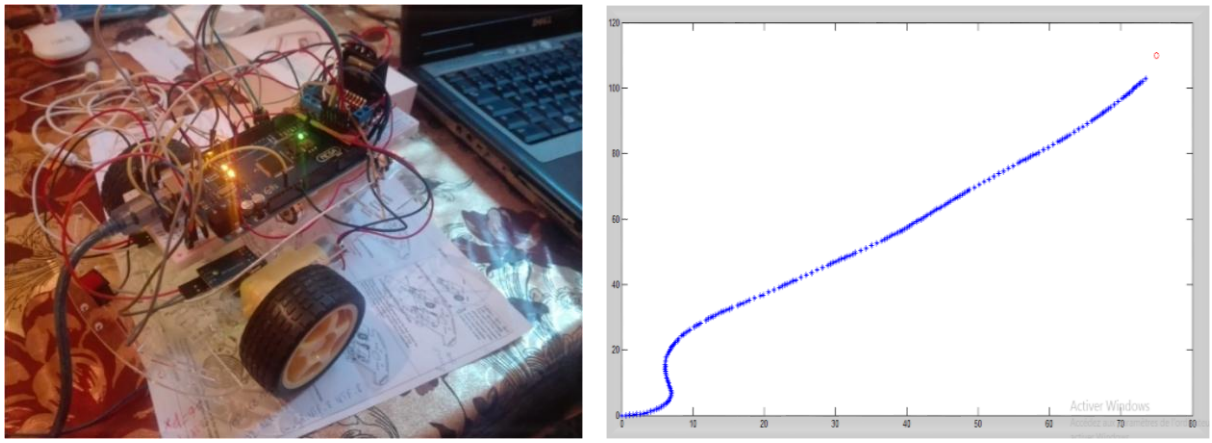


Figure 4.20 : Simulation en temps réal sur le robot en utilisant MATLAB et Arduino

7.3. Implémentation du robot sur un plan

Le résultat que nous avons obtenu peut-être observé sur la figure.4.21, qui représente la plateforme mobile réellement obtenue, et qui est implémenté sur un plan bien défini. Ce rebot se déplace d'un point de départ (0,0) à un point de cible (75,110). Nous obtenons alors un résultat satisfaisant.

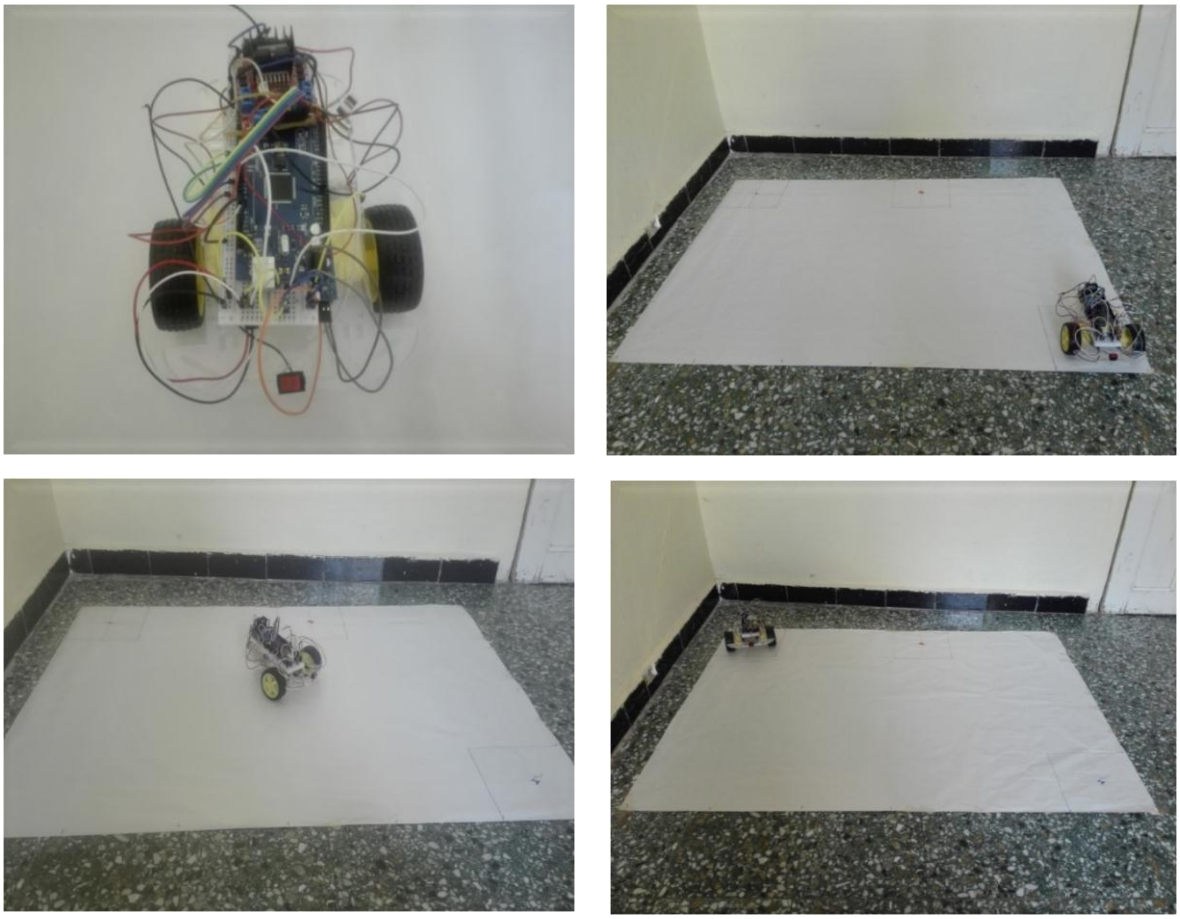


Figure 4.21 : L'implémentation du robot sur un plan

8. MODE 04: Localisation et cartographie simultanées (SLAM)

Dans ce quatrième mode une amélioration possible a été établit sur le robot aspirateur. Nous avons ajouté une caméra kinect de Xbox 360 pour montrer les SLAM sur le robot mobile. Pour ce fait nous avons utilisé les logiciels et les systèmes d'exploitation suivants: ROS, le RTAB-Map et bien aussi Rviz.

8.1. Circuit pour les SLAM

Bien que le Kinect, il dispose également d'un moteur d'inclinaison, d'un réseau de microphones, d'un accéléromètre et d'un meilleur soutien de la communauté des ROS PointClouds peut être converti en LaserScan, un ensemble de données de capteurs de distance avec une certaine plage d'angle. Cela ressemble à des données lidar, mais pas en 360 degrés. Les informations du Laserscan peuvent être utilisées pour le SLAM : construction de cartes, évitement d'obstacles, etc.…

Donc notre réalisation, nous avons utilisé le kinect pour le but de la construction de la Carte. La figure (figure 4.22) montre le montage du Kinect avec un Raspberry pi :

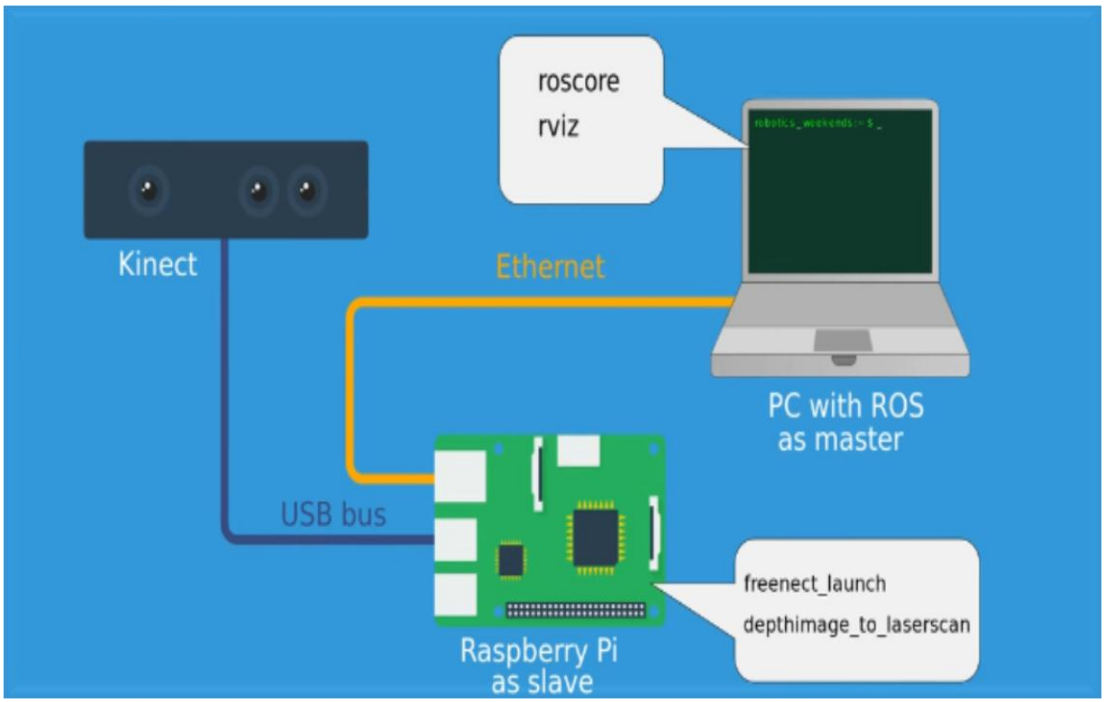


Figure 4.22 : Schéma pour la localisation et la cartographie simultanées sur un Raspberry pi 3 model B (SLAM) [101]

8.2. Implémentation du robot dans un environnement de travail

Le résultat que nous avons obtenu peut-être observé sur la (figure.4.23), qui représente la plateforme mobile réellement obtenue, et l'implémentation réale dans un environnement de travail.

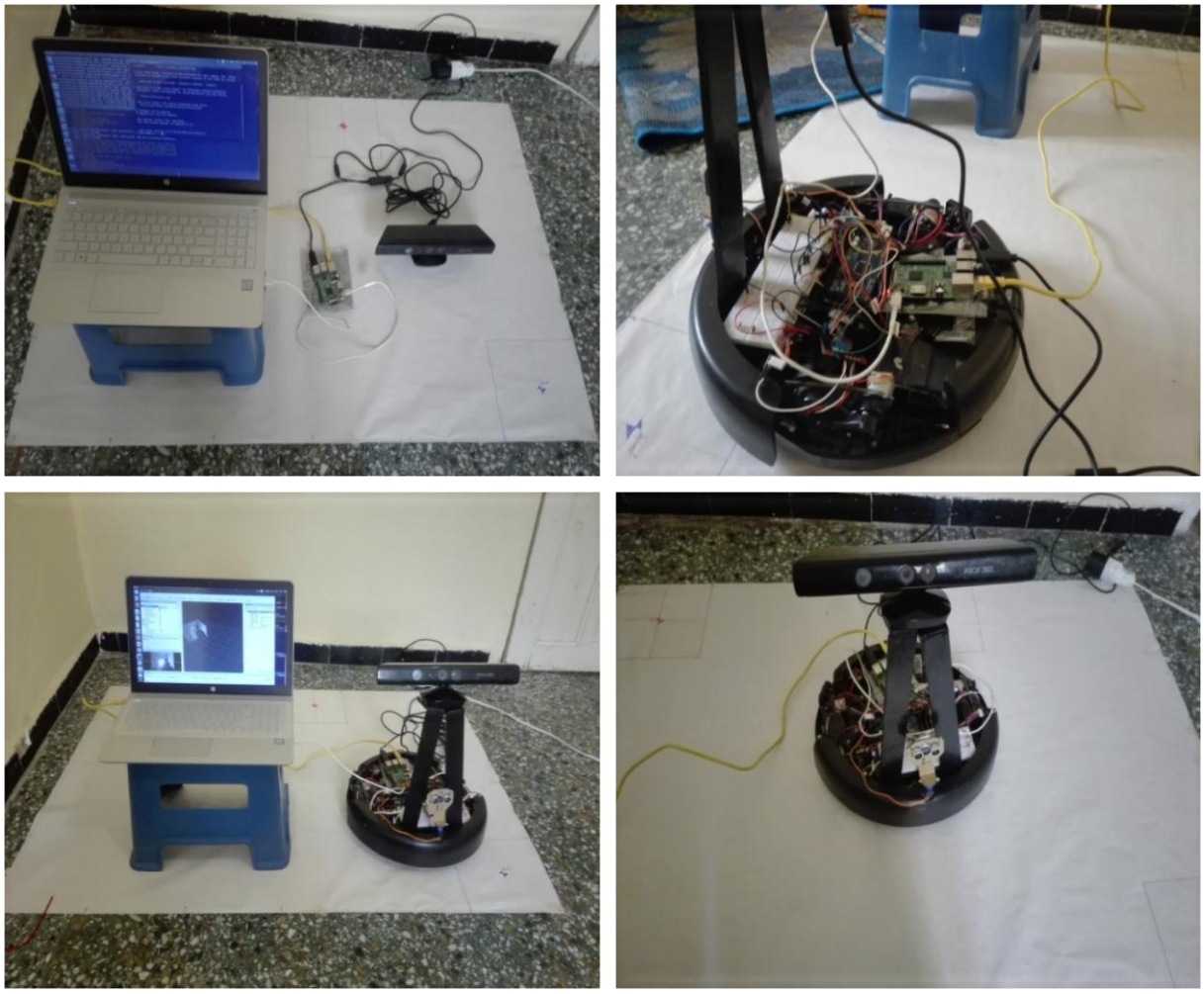


Figure 4.23 : localisation et cartographie simultanées sur le robot mobile en ajoutant le kinect 360

9. Conclusion

Ce chapitre était le noyau de notre projet de fin d'étude. On a commencé par la conception des circuits de chaque mode de notre robot, par la suit nous avons effectué une petite simulation en temps réel sur le robot mobile par le logiciel MATLAB et Arduino en même temps pour le but de montrer l'efficacité de l'algorithme d'odométrie.

Cette simulation nous a permis de voir le comportement de notre robot et la performance de ses capteurs. Finalement nous avons pu réaliser les quatre modes et leurs implémentations en utilisant deux robots mobiles déférents dans leurs designs.

Conclusion Générale

Conclusion Générale

Ce projet nous a permis d'élaborer un large spectre de connaissances acquises lors de notre formation. Nous avons utilisé les approches de l'ingénierie des systèmes en transformant les objectifs du robot et ses exigences en termes de critères de design lors de sa conception pour établir une architecture globale qui répond et accomplit la mission du robot. Nous avons utilisé la mécanique, l'électronique et l'automatique durant l'implémentation des composantes, la réalisation des circuits et la simulation des systèmes du robot.

En cas d'un terrain accidenté il faut utiliser une autre structure robot mobile de genre tous terrains car notre structure (la plateforme expérimentale) navigue uniquement à l'intérieur ou bien sur un terrain plat.

Nous avons contribué à la résolution du problème de la navigation autonome d'un robot mobile dans des environnements de travail encombrés. Nos contributions portent sur le développement de nouvelles méthodes et des modèles mathématiques inspirées de l'odométrie et le filtre de Kalman pour améliorer les performances de navigation en dotant le robot mobile de capacités décisionnelles lui permettant de prendre ses propres décisions. Cela signifie qu'il doit être capable à la fois de percevoir correctement son environnement et également de savoir comment réagir en conséquence suivant le niveau d'autonomie. C'est à lui de planifier son parcours et de déterminer avec quels mouvements va-t-il atteindre son objectif.

Des résultats de simulation affirment l'efficacité de l'approche proposée et la résolution de la problématique proposée. Dans le premier cas de la simulation, nous avons estimé la position du véhicule en utilisant l'odométrie. Puis, des simulations en 3D a était établir sur MATLAB pour montrer l'odométrie, et une planification de trajectoire pour un robot mobile sur un autre logiciel qui s'appelle V-REP PRO EDU. Enfin une simulation sur ROS avec l'utilisation d'une série de requêtes sur ubuntu et des logiciels comme Rviz et RTAB-Map pour montrer la localisation et cartographie simultanées (SLAM).

Le gain majeur de ce travail est d'avoir fait la réalisation et l'étude sur un robot mobile à base différentiel qui est facile à monter et à démonter et peut être utilisé en quatre (04) modes. Le premier mode fait des mouvements en évitant les obstacles pour montrer l'efficacité de la perception du robot sur les premiers résultats de réalisation. Pour le mode RC via Bluetooth peut être contrôlé à distance. Il peut jouer le rôle d'un robot exploiteur dans un milieu

dangereux ou de l'envoyer d’une place à une autre par une application Android .Pour le mode de l’odométrie (La localisation du robot mobile) est généralement utilisé comme un AMR dans les systèmes de production ou de stockage intelligent et même dans un environnement de travail pour envoyer un objet d’une place à une autre sur le robot. Le dernier mode représente une combinaison entre les modes 1 et 3 en ajoutant une caméra kinect pour faire la localisation et la cartographie simultanées (SLAM pour la construction de la carte). Il peut être amélioré en ajoutant des modules comme le moteur DC avec encodeur pour qu’il puisse suivre une trajectoire définie sur un programme.

Notre travail reste très modeste en comparaison avec ce qui peut réellement être accompli avec plus de moyens, mais il a le mérite de pouvoir être un point de départ pour des études plus complexes et plus poussées de systèmes de plus en plus compliqués.

Références Bibliographiques

Références Bibliographiques

[1] : HAMANI MILOUD, Navigation des robots mobiles non-holonomes sous contrôle flou, THÈSE Présentée à la faculté de technologie Département d'Electronique Pour l'obtention du diplôme de Doctorat en Sciences, université Ferhat Abbas ‑SETIF 1,2016.

[2] : Cyril DROCOURT, Localisation et modélisation de l'environnement d'un robot mobile par coopération de deux capteurs omnidirectionnels, Discipline : Robotique, pour l'obtention du grade de Docteur de l'université de technologie de COMPIEGNE, 2002.

[3] : Pierre Tournassoud, Planification de trajectoire et contrôle en robotique, application aux robots mobiles et manipulateurs, Bibliothèque de la faculté de technologie, université de TLEMCEN, 1992.

[4] : Robert Holmberg, Oussama Khatib, Development and Control of a Holonomic Mobile Robot for Mobile Manipulation Tasks, International Journal of Robotics Research, vol. 19, N°11, Stanford University, Stanford, CA, USA, november 2000.

[5] : https://www.mb-s.fr/mir-vs-agv.html

[6] : Stephane Petti and Thierry Fraichard. Partial motion planning framework for reactive planning within dynamic environments. In IEEE Int. Conf. On Robotics and Automation, Barcelona, Spain, 2005.

[7] : Youcef MECHALIKH et Ali MILOUDI, Développement d'algorithmes d'évitement d'obstacles statiques et dynamiques. PFE pour l'obtention du diplôme MASTER en automatique, Université Kasdi Merbah-Ouargla, 2012.

[8] : Ishak *MOUMENE,* Etude comparative de quelques algorithmes d'évitement d'obstacles en robotique mobile**,** En vue de l'obtention du diplôme MASTER en Electronique, Université Abderrahmane Mira de Bejaia, 2013.

[9] : Robotique Mobile – David FILLIAT, École Nationale Supérieure de Techniques Avancées ParisTech.

[10] : REDJRADJ Djillali et KACIMI Fares, Suivi de trajectoires d'un robot mobile par la logique floue, En vue de l'obtention du diplôme MASTER en Electronique option automatique, Université Abderrahmane Mira de Bejaia, 2013/2014.

[11] : T. Takagi et M. Sugeno, fuzzy identification and its application to modeling and control, IEEE transactions on systems. Man and Cybernetics. Pages: 116 – 132, février 1985.

[12] : N. Morette, "Contribution à la navigation de robots mobiles : approche par modèle direct et commande prédictive", Thèse de doctorat de l'Institut Prisme, Equipe Systèmes Robotiques interactifs, Université d'Orléans, France, 2009.

[13] : Ben-Zion Sandler – ROBOTICS Designing the Mechanisms for Automated Machinery – Second edition

[14] : G. Frappier, Système inertiels de navigation pour robots mobiles, Séminaire sur Les robots mobiles, EC2, Paris, 1990.

[15] : ADOUNA Abdellah Charefeddine et YOUSFI Hadj Seddik,Conception, simulation et réalisation d'un robot intelligent , En vue de l'obtention du diplôme MASTER en Génie industriel, faculté de technologie de université de Tlemcen , 2016.

[16] : El-Hadi GUECHI, Suivi de trajectoires d'un robot mobile non holonome : approche par modèle flou de Takagi-Sugeno et prise en compte des retards, thèse de doctorat Université d'Annaba (Algérie), 2010.

[17] : Spong, M., Hutchinson, S. and Vidyasagar, M. (2006). Robot Modeling and Control, John Wiley & Sons, INC, New York.

[18] : Bloch A.M., Baillieul, J., Crouch, P., Marsden, J. (2003). Non-holonomic mechanics and control. Springer.

[19] : Warner, F.W. (1983). Foundations of differentiable manifolds and lie groups. Graduate Texts in Mathematics. Springer-Verlag.

[20] : Nijmeijer H. et Van Der Schaft, A. (1990). Nonlinear dynamical control systems. Springer Verlag, New York.

[21] : Thierry Fraichard, Planification de mouvement pour mobile non-holonome en espace de travail dynamique, Modélisation et simulation. Institut National Polytechnique de Grenoble - INPG, 1992.

[22] : E. Papadopoulos & J. Poulakakist, "Trajectory planning and control for mobile manipulator systems", 8th IEEE Proc. Mediterranean Conf. on Control & Automation, Patras, Greece, pp. 01-06, 2000.

[23] : T. Guesbaya, K. Benmahammed & A. Benali, "Planificateur de trajectoire avec évitement d'obstacle basé sur la méthode des contraintes pour robot mobile", Courrier du Savoir, No. 9, pp. 71-81, Mars 2009.

[24] : A. De Luca, G. Oriolo & P. R. Giordano, "Kinematic control of nonholonomic Mobile manipulators in the presence of steering wheels", Inter. IEEE Conf. on Robotics and Automation,Anchorage,Alaska,USA,pp.1792-1798,2010.

[25] : J. Plaskonka, "Different kinematic path following controllers for a wheeled mobile robot of (2,0) Type", Journal of Intelligent robot Systems, Vol. 77, pp. 481-498, 2013.

[26] : M. Ghanavati, "A new method of mobile robot navigation: Shortness null space", Journal of Advances in Computer Research, Vol. 3, No. 3, pp. 65-74, August 2012.

[27] : B. Chazelle, Voronoi diagrams : A survey of a fundamental geometric data structure. Algorithmic and Geometric Aspects of Robotics, 1987.

[28] : A. Lingas. The power of non-rectilinear holes.In In Proceedings 9th International Colloquium on Automata, 1982.

[29] : N. Vandapel, J. Kuffner & O. Amidi, "Planning 3-d path network in unstructured environments", IEEE Int. Conf. on Robotics and Automation, pp. 4624‑4629, 2005.

[30] : T. Fraichard, "Trajectory planning in a dynamic workspace: a state-time space approach", Advanced Robotics, Vol. 13, No. 1, pp. 75‑94, 1998.

[31] : C.-T. Kim & J.-J. Lee, "Mobile robot navigation using multi-resolution electrostatic potential field", 2005.

[32] : M. Pivtoraiko & A. Kelly, "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces", IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, 2005.

[33] : L. Kravaki, P. Svestka, J-C. Latombe, and M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration space. IEEE Trans. Robotic and Automation, 1996.

[34] : N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In Proc. IEEE Int. Conf. On Robotics & Automation, 1996.

[35] : L. Jaillet and T. Simeon. Path deformation roadmaps : Compact graphs with useful cycles for motion planning. Int. Journalof Robotics Research, 2008.

[36] : S. A. Wilmarth, N. M. Amato, and P. F. Stiller, A probabilistic roadmap planner with sampling on the medial axis of the free space. In Proc. IEEE Int. Conf. on Robotics & Automation, 1999.

[37] : R.Simmons. The curvature-Velocity Method for Obstacle Avaoidance.in Proc. Of the IEEE Int. Conf. On Robotics and Automation. Minneapolis, Minnesota, April 1996.

[38] : Jur P. van den Berg and Mark H. Overmars, Roadmap-based motion planning in dynamic environment. IEEE Transactions on Robotics, 2005.

[39] : Anthony Stentz, "Optimal and efficient path planning for unknown and dynamic environments", International Journal of Robotics and Automation, Vol. 10, pp. 89‑100, 1993.

[40] : M. Likhachev, D. Ferguson, G. Gordon, A. Stentz & S. Thrun, "Anytime dynamic a* : An anytime replanning algorithm", 2005.

[41] : F. Kuhne, W. F. Lages & J. M. G. Da Silva Jr., "Mobile robot Trajectory racking using model predictive control", IEEE Lars, Sao Luis, pp. 1-7, September 2005.

[42] : E. Swere & D. Mulvaney, "Robot navigation using decision trees", Electronic Systems and Control Division Research, UK, pp. 15-17, 2003.

[43] : Dave Ferguson & Anthony Stentz, "Anytime rrts", IEEE- RSJ Int. Conf. on Intelligent Robots and Systems, Beijing, China, Oct. 2006.

[44] : P. Bessiere, J. Ahuactzin, T. El-Ghazali, and E. Mazer. The ariane's clew algorithm : Global planning with local methods. In IEEE-RSJ Int. Conference on Intelligent Robots and Systems, 1993.

[45] : S. Carpin and G. Pillonetto. Robot motion planning using adaptive random walks. IEEE Trans. on Robotics & Automation, 21(1) :129–136, 2005.

[46] : D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. Int. Journal Computational Geometry & Applications, 9 :495–512, 1999.

[47] : Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. The International Journal of Robotics Research, Vol N° 5, 1986.

[48] : Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In IEEE International Conference on Robotics and Automation, April 1991.

[49] : S.S. Ge and Y.J. Cui. Dynamic motion planning for mobil robots using potential field method. Autonomous Robots, page : 207–222, 2002.

[50] : I. Ulrich and J. Borenstein. Vfh+ : Reliable obstacle avoidance for fast mobile robots. pages 1572–1577, 1998.

[51] : I. Ulrich and J. Borenstein. Vfh* : Local obstacle avoidance with look-ahead verification. San Francisco, USA, 2000.

[52] : J. Minguez & L. Montano, "Nearness diagram navigation (nd): A new real time collision avoidance approach", In IEEE RSJ Int. Conf. on Intelligent Robots and Systems, 2000.

[53] : R.Simmons. The curvature-Velocity Method for Obstacle Avaoidance.in Proc. Of the IEEE Int. Conf. On Robotics and Automation. Minneapolis, Minnesota, April 1996.

[54] : N.Y. Ko and R.Simmons.The lane-curvature method for local obstacle avoidance. In Proc. Of th IEEE/RSJ Int. Conf. On Intelligent Robots and Systems (IROS), 1998.

[55] : D. Fox, Burgard W, and S. Thrun. The dynamic window approach to collision avoidance. IEEE Journal on Robotics and Automation, 1997.

[56] : P. Fiorini and Z. Shiller. Motion planning in dynamic environments using the relative velocity paradigm. In IEEE Int. Conf. On Automation & Robotics, 1993.

[57] : Oren Gal, Zvi Shiller, and Elon Rimon. Efficient and safe online motion planning in dynamic environments. In Proc. Of the Intl Conf. On Robotics & Automation, Kobe, Japan, May 2009.

[58] : F. Large, C. Laugier, and Z. Shiller. Navigation among moving obstacles using the nlvo : Principles and applications to intelligent vehicles. Autonomous Robots Journal, September 2005.

[59] : Thierry Fraichard and Hajime Asama. Inevitable collision states a step towards safer robots. In IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, Las Vegas, 2003.

[60] : R. Parthasarathi and T. Fraichard. An inevitable collision state checker for a car-like vehicle. In Proc. of the Intl Conf. On Robotics & Automation, Roma, Italy, April 2007.

[61] : Luis Martinez-Gomez and Thierry Fraichard. An efficient and generic 2d inevitable collision state-checker. In IEEE-RSJ Int.Conf. on Intelligent Robots and Systems, France Nice, 2008.

[62] : Luis Martinez-Gomez and Thierry Fraichard. Collision avoidance in dynamic environments : an ics-based solution and its comparative evaluation. In Proc. of the Intl Conf. on Robotics & Automation, Kobe, Japan, May 2009.

[63] : E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. AIAA Journal of Guidance, Control and Dynamics, 2002..

[64] : Stephane Petti and Thierry Fraichard. Partial motion planning framework for reactive planning within dynamic environments. In IEEE Int. Conf. On Robotics and Automation, Barcelona, Spain, 2005.

[65] : S. Quinlan & O. Khatib, "Elastic bands : Connecting path planning and control", IEEE Int. Conf. on Robotics and Automation, Atlanta, GA (US), May 1993.

[66] : V. Delsart, "Navigation autonome en environnement dynamique : Une approche par déformation de trajectoire", Thèse de doctorat Université de Grenoble, 2010.

[67] : Hichem MAAREF, Mourad OUSSALAH et Claude BARRETI, "Fusion de données capteurs en vue de la localisation absolue d'un robot mobile par une méthode basée sur la théorie des possibilités. Comparaison avec le filtre de Kalman", Article par CEMIF-Groupe Systèmes Complexes 40, rue du Pelvoux 91020 Evry cedex France et K. U. Leuven, PMA, Celestijnenlaan 3000B B-3001 Heverlee, Belgique.

[68] : MEZRAG Moustafa Mohammed El-amine et OUKERIMI Hadjer, Localisation d'un robot mobile à roues par un algorithme basé sur le Filtre de Kalman Etendu, en vue de l'obtention du diplôme MASTER en Génie Electrique, option automatique, UNIVERSITE MOHAMED BOUDIAF - M'SILA, 2017/2018.

[69] : https://www.rs-online.com/designspark/noise-sensor-fusion-and-lost-drones-fr

[70] : PIETTE Ferdinand, "Le filtre de Kalman étendu : principe et exemple". Sciences et Techniques, [Online], Disponible à: http://www.ferdinandpiette.com/blog/2011/05/le-filtre-de-kalman-etendu-principe-et-exemple/, 21 Mai 2011.

[71] **:** MALLET Anthony**, "**Localisation d'un robot mobile autonome en environnements naturels". Thèse de Doctorat de l'Institut National Polytechnique de Toulouse, 2 juillet 2001.

[72] : Handry Khoswanto, Petrus Santoso and Resmana Lim, Chapter 17 Odometry Algorithm with Obstacle Avoidance on Mobile Robot Navigation, Article scientifique, Department of Electrical Engineering, Petra Christian University, Surabaya, Indonesia, Springer Science+Business Media Singapore 2016.

[73] : J. Borenstein et L. Feng. UMBmark - A Method For Measuring, Comparing, and Correcting Dead-reckoning Errors in Mobile Robots. Rapport technique UM-MEAM-94-22, 1994, University of Michigan, 1994

[74] : BAYLE Bernard, "Robotique mobile". Rapport de l'Ecole Nationale Supérieure de Physique de Strasbourg, 2008.

[75] : Oussama El Hamzaoui, "Localisation et Cartographie Simultanées pour un robot mobile équipé d'un laser à balayage : CoreSLAM".thèse pour obtenir le grade de docteur délivré par l'École nationale supérieure des mines de Paris, Spécialité « Informatique temps-réel, robotique et automatique », en 2012.

[76] : R. C.Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," International Journal of Robotics Research, vol. 5, pp. 56_68, 1986.

[77] : H. Durrant-Whyte, "Uncertain geometry in robotics, " IEEE Journal of Robotics and Automation, vol. 4, pp. 23_31, 1988.

[78] : A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera, " Ninth IEEE International Conference on Computer Vision (ICCV'03) – Volume 2, 2003.

[79] : J. Sola, A. e Monin, M. Devy, and T. Vidal-Calleja, "Fusing monocular information in multicamera slam, " IEEE Transactions on Robotics, vol. 24, 2008.

[80] : F. Lu and E. Milios, "Robot pose estimation in unknown environments bu matching 2d range scans, " Journal of Intelligent and Robotic Systems, 1997.

[81] : S. Riisgaard and M. R. Blas, SLAM for Dummies.A Tutorial Approach to Simultaneous Localization and Mapping.

[82] : A. Eliazar and R. Parr, "Dp-slam : Fast, robust simultaneous localization and mapping without predetermined landmarks, "Proceedings of the International Joint Conference on Arti_cial Intelligence, 2003.

[83] : M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, " Fastslam : A factored solution to the simultaneous localization and mapping problem," In Proceedings of the AAAI National Conference on Artificial Intelligence, pp. 593_598, 2002.

[84] : R. Ouellette and K. Hirasawa, "A comparison of slam implementations for indoor mobile robots," Proceedings of the 2007 IEEURSJ, Int. Conference on Intelligent Robots and Systems, 2007.

[85] : T.-D. Vu, "Localisation, mapping avec detection, classi_cation et suivi des objets mobiles, " Ph.D. dissertation, Institut National Polytechnique de Grenoble, 2009.

[86] : F. Lu and E. "Milios, Globally consistent range scan alignment for environment mapping, "Autonomous Robots, vol. 4, pp. 333 _ 349, 1997.

[87] : J. Leonard and H. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot, " IEEE/RSJ International Conference on Intelligent Robots and System, 1991.

[88] : A. Elfes, "Occupancy grids : a probabilistic framework for robot perception and navigation," Ph.D. dissertation, Carnegie Mellon University, 1989.

[89] : T. Einsele and G. Farber, " Real-time self-localization in unknown indoor environments using a panorama laser range finder," IEEE/RSJ International Workshop on Robots ans Systems, pp. 697_703, 1997.

[90] : S. P_ster, S. Roumeliotis, and J. Burdick, "Weighted line fitting algorithms for mobile robot map building and efficient data representation," IEEE International Conference on Robotics and Automation, vol. 1, pp. 1304 _ 1311, 2003.

[91] : D. A. Forsyth and J. Ponce, Computer Vision : A Modern Approach. Prentice Hall, 2011.

[92] : https://www.arduino-france.com/tutoriels/quest-ce-que-arduino/

[93] : BEDADI Mehdi, Etude et réalisation d'un véhicule autonome, En vue de l'obtention du diplôme MASTER en Automatique, Spécialité Automatique et Système, Faculté Sciences de l'Ingéniorat d'université Badji-Mokhtar Annaba, 2019.

[94] : https://www.timetoast.com/timelines/histoire-des-robots.

[95] : http://davidbuckley.net/DB/HistoryMakers.htm.

[96] : https://blog.generationrobots.com/fr/qu-est-ce-que-la-technologie-lidar/

[97] : https://robot-r-us.com.sg/p/2wd-mobile-robot-kit

[98] : https://www.briconline.fr/fonctionnement-de-laspirateur-robot/

[99] : http://projectshopbd.com/product/rs-uk-raspberry-pi-3-model-b-391/

[100] : https://www.tutoriel-arduino.com/utiliser-hc-05-arduino/

[101] : http://roboticsweekends.blogspot.com/2017/12/how-to-connect-kinect-to-raspberry-pi-2.html

[102] : https://www.numerama.com/sciences/149460-un-chien-trouble-par-un-robot-a-quatre-pattes.html

Résumé : Ce travail consiste à concevoir, réaliser et commander un robot mobile à base différentiel et puis implémenter ce robot dans un environnement de travail pour qu'il puisse capable de transporter un objet d'un poste à un autre. De nombreux algorithmes ont été conçus sur un système de navigation robotisé mobile. Certains d'entre eux utilisent un algorithme basé sur un concept plus récent comme la logique floue ou génétique algorithme. Certains s'en tiennent à un algorithme plus ancien comme l'odométrie. À répondre à l'exigence, l'algorithme d'odométrie est choisi spécifiquement. Des logiciels de simulation et des résultats expérimentaux ont permis de montrer l'efficacité de l'approche proposée et la performance de notre conception. Notre objectif était de développer un mode de navigation autonome permettant à un robot mobile de s'adapter dans un environnement de travail.

Mots-Clés : robot mobile, Arduino, Raspberry pi, base différentiel, Bluetooth, Odométrie.

Abstract: This work consists of designing, realizing and controlling a mobile differential-based robot and then implementing this robot in a working environment so that it is capable of transporting an object from one station to another. Many algorithms have been designed on a mobile robotic navigation system. Some of them use an algorithm based on a more recent concept such as fuzzy logic or genetic algorithm. Some stick to an older algorithm such as odometry. A simple, less intensive calculation algorithm for a lightweight telepresence robot is required. To meet the requirement, the odometry algorithm is chosen specifically. Simulation software and experimental results have shown the efficiency of the proposed approach and the performance of our design. On the one hand. Our objective was to develop an autonomous navigation mode allowing a mobile robot to adapt in a working environment.

Keywords: Mobile robot, Arduino, Raspberry pi, differential base, Bluetooth, odometry.

ملخص: يتكون هذا العمل من تصميم وتحقيق والتحكم في ربوت متنقل قائم على التفاضل ثم تنفيذ هذا الروبوت في بيئة عمل بهدف تمكينه من نقل شيء من محطة إلى أخرى. تم تصميم العديد من الخوارزميات على نظام ملاحة آلي متنقل. يستخدم بعضهم خوارزمية تستند إلى مفهوم أحدث مثل المنطق الضبابي أو الخوارزمية الجينية. يلتزم البعض بخوارزمية أقدم مثل قياس المسافات. مطلوب خوارزمية حساب بسيطة وأقل كثافة لربوت التواجد عن بعد الخفيف. لتلبية المتطلبات، يتم اختيار خوارزمية قياس المسافات على وجه التحديد. أظهرت برامج المحاكاة والنتائج التجريبية فعالية النهج المقترح وأداء تصميمنا. كان هدفنا هو تطوير وضع ملاحة مستقل يسمح للروبوت المتحرك بالتكيف في بيئة عمل

الكلمات الرئيسية :أردوينو ، راسبري باي، قاعدة التفاضل، روبوت متنقل ، بلوتوث، التنفيذ، قياس المسافات