الجمـــهوريـــة الجـــزائريـــة الديـمقـراطيـــــــة الشـــعبيـــــة
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعـلـيـم الـعـالـــي و الـبـحـث الـعـلـمي
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
جـامـعـة عمّـــار ثـلـيـــجـــي بالأغـــواط
UNIVERSITE AMAR TELIDJI LAGHOUAT
كـلـيـــــة الـعـلـــــوم
FACULTE DES SCIENCES
DEPARTEMENT DE MATHEMATIQUES ET INFORMATIQUE

Mémoire de MASTER

Domaine : Mathématiques et Informatique
Filière : Informatiques
Option : Réseaux, Systèmes et Applications Réparties

Par:
DAREM Boutheyna
LAROUCI Loubna Yasmine

THEME

Planification de trajectoire pour un robot mobile

Soutenu publiquement le 08-07-2021 devant le jury composé de:

Mlle HAMINI Narjes	*M.A.(A)*	*Présidente*
Mr BOUZIDI Mohammed Redha	*M.A.(A)*	*Examinateur*
Mdm TAABA Kheira	*M.A.(A)*	*Encadreur*

Année Universitaire 2020/2021

Remerciement

Toute la gratitude et le merci à Dieu notre créateur qui nous a donné La force pour effectuer et achever ce travail.

Ainsi nos parents qui nous aident.

Nous tenons à remercier Notre promotrice ***TAABA Kheira*** *Pour avoir accepté de diriger notre travail, pour ses précieux conseils,*

Pour son esprit d'ouverture et sa disponibilité. Grâce à lui,

Notre travail s'est déroulé.

Nous remercions toutes personnes qui nous ont aidés de près ou de loin à la Finalisation de ce travail, nous tenons à leur

Exprimer notre vive gratitude.

Enfin nos remerciements s'adressent aux membres de jury qui nous feront L'honneur de juger notre travail.

...

DEDICACES

Je dédie ce mémoire :

A mes très chers parents pour leur soutient et encouragement durant toutes

Mes années d'études et sans lesquels je n'aurais jamais réussi

A mes chères sœurs, Rihab, Issra, Rawan.

A mon cher frère, Mohamed Islam.

A mon binôme Yasmine.

A toute ma famille.

A tous mes professeurs et enseignants que j'ai eu durant tout mon cursus

Scolaire et qui m'ont permis de réussir dans mes études.

A tous mes amis de l"Université et d'ailleurs

A toute personne ayant contribué à ce travail de près ou de loin.

A tous ceux que j'aime, et tous ceux qui m'aiment

Boutheyna.D

DEDICACES

A Dieu source de toute connaissance A celle qui m'a donné la vie, qui

n'a pas cessé de m'encourager et de prier pour moi, qui

s'est sacrifiée pour mon bonheur et ma réussite, ma

tendre mère.

Au plus cher au monde, mon père,

à me donner l'aide et à

me protéger

Que dieu les gardes et les protège.

Au plus cher au monde, mon mari SalahEddine.

A m'adorable sœur : Amina.

A mes chers frères : Taha, Radouane, Ahmed, Djalel.

A toute ma famille.

A mon binôme Boutheyna.

A tous mes amies

A tous ceux qui me sont chères, A tous ceux qui m'aiment,

A tous ceux que j'aime.

Je dédie ce modeste travail...

Yasmine.L

Résumé

En robotique, la planification automatique de trajectoire sans collision dans différents types d'environnements est le sujet d'actualité . Dans ce mémoire on va faire l'étude des différents types de robots mobiles ainsi que l'étude de la géométrie, de la cinématique et de la dynamique du robot ce qui nous permet de le modéliser par rapport à une référence absolue. Ensuite ,on donne les différentes méthodes permettant de calculer le chemin optimal dans l'environnement considéré ainsi que le contrôleur PurePursuit de suivi de trajectoire. On termine par la simulation des méthodes de planification RRT, Astar et RPM et la méthode de suivi de chemin PurePursuit pour aboutir à des résultats qui justifient l'efficacité et la précision de chaque méthode .

Mots clé : Planification de trajectoire, robot mobile, intelligence artificielle, algorithme RRT, algorithme Astar, navigation autonome.

Abstract

In robotics, automatic collision-free path planning in different types of environments is a hot topic. In this thesis we will study the different types of mobile robots as well as the study of the geometry, kinematics and dynamics of the robot, which allows us to model it in relation to an absolute reference. Then, we give the different methods allowing to calculate the optimal path in the considered environment as well as the PurePursuit trajectory following controller. We finish by simulating the RRT, Astar and RPM planning methods and the PurePursuit path following method to achieve results that justify the efficiency and precision of each method.

Keywords : **Trajectory planning, mobile robot, artificial intelligence, RRT algorithm, Astar algorithm,autonomous navigation.**

Table des matières

Table des figures

Introduction générale

De nos jours, l'émergence de la robotique "intelligente" conduit à rendre les robots de plus en plus autonomes et de plus en plus faciles à piloter. Un système robotique est défini comme un système artificiel, doté de capteurs et d'actionneurs, conçu pour agir sur le monde extérieur. En fonction du domaine d'origine des auteurs, il existe donc diverses définitions du terme robot, mais elles tournent en général autour de celle-ci : Un robot est une machine équipée de capacités de perception, de décision et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception assurée par leurs capteurs avec une trajectoire programmé.

Le développement des moyens informatiques dont disposent les chercheurs en robotique rend de plus en plus facile l'utilisation d'algorithmes puissants pour donner aux robots une capacité d'autonomie.

La robotique comporte deux grands pôles d'intérêt : la robotique de manipulation (robotique industrielle) et la robotique mobile. On regroupe sous l'appellation robots mobiles l'ensemble des robots à base mobile, par opposition notamment aux robots manipulateurs. L'usage veut néanmoins que l'on désigne le plus souvent par ce terme les robots mobiles à roues en raison de leur vaste domaine d'activité et de leurs défis théoriques. Les autres robots mobiles sont en effet le plus souvent désignés par leur type de locomotion, qu'ils soient marcheurs, sous-marins ou aériens.

Dans ce travail, nous nous intéresserons uniquement aux robots mobiles à roues de type unicycle. Ces derniers disposent d'une motorisation électrique et d'un faible encombrement, par conséquent, un asservissement en vitesse est suffisant pour les commander.

L'objectif principale est d'étudier et d'implémenter des algorithmes de planification de trajectoire permettant au robot mobile unicycle de naviguer dans un environnement complexe et cela en évitant les obstacles et de lui appliquer un contrôleur simple et efficace pour suivre la trajectoire souhaitée.

Un des problèmes majeurs de la robotique mobile est la planification de mouvement. Autour de ce problème de planification de mouvement de nombreuses études ont été réalisées dans le but de développer des méthodes générales pour guider les robots. Deux types de méthodes peuvent

être distinguées : celles qui cherche à construire une représentation de l'espace libre (méthodes globales) et celle qui se basent sur des informations locales pour construire incrémentalement une trajectoire (méthodes locales). Chacune de ces méthodes a ses avantages et ses inconvénients.

Dans ce mémoire, on s'intéresse à la planification de trajectoire du robot mobile par les méthodes globales probabilistes RRT et PRM (Probabilistic raodmap) et la méthode heuristique Astar. Après la planification de chemin du robot mobile, on a besoin d'une méthode de commande permettant au robot de suivre ce chemin avec le maximum de précision. Pour ce faire, on a choisi la méthode de poursuite pure (Pure Pursuit algorithm) qui est une méthode de détermination géométrique de la courbure qui conduira le véhicule à un point choisi de la trajectoire. Ainsi, le suivi de trajectoire est le processus qui consiste à déterminer les paramètres de vitesse et de direction à chaque instant afin que le robot suive le chemin désirée. Le mémoire est organisé comme suit :

- Le premier chapitre, est consacré à des généralités sur la robotique mobile, ainsi que la modélisation cinématiques des robots mobiles à roues.
- Le second chapitre donne un état de l'art des méthodes fondamentales de planification de trajectoire et cite leurs avantages et inconvénients permettant aux lecteurs de choisir la méthode de planification appropriée.
- Le troisième chapitre illustre les différentes étapes et les résultats d'implémentation des méthodes de planification RRT, PRM et Astar et les résultats de simulation obtenus dans le même environnement. Les résultats de l'application de l'algorithme de poursuite de trajectoire Pure Pursuit seront également montrés et commentés.

Enfin, une conclusion générale suivie de quelques perspectives termine ce mémoire.

Chapitre I

Généralité sur les robots mobile

I.1 Introduction

La robotique est un domaine de recherche important qui fait appel aux connaissances croisées de plusieurs disciplines. L'objectif est l'automatisation des systèmes mécaniques, en les dotant de capacités de perception, de décision et d'action permettant au robot d'interagir rationnellement avec son environnement sans intervention humaine. La robotique mobile est un domaine dans lequel l'expérience pratique est primordiale. Au début, les robots font leur apparition dans l'industrie grâce aux capacités des manipulateurs : des bras imitant le bras humain et capables d'utiliser différents outils pour accomplir divers tâches. Au fur et à mesure, ces bras gagnent de nouveaux degrés de liberté à l'aide de plates-formes mobiles ; c'est l'apparition des robots mobiles à roues. Les premiers robots autonomes utilisaient la roue mais pour certaines applications où la géométrie de l'environnement sont diffèrent (terrain accidenté, endroits difficiles à atteindre), les recherches on orientés vers d'autres moyens de locomotion, inspirés du monde des animaux. Ainsi, les robots à pattes sont apparus et on connaît déjà des robots hexapodes ou des robots bipèdes. Des problèmes tels que la conception mécanique, la perception de l'environnement, la modélisation de l'espace de travail, la planification de trajectoires sans collision [1]. L'objectif de ce chapitre est de donner un bref exposé sur le domaine de la robotique mobile et ses différents types.

I.2 Définitions

La robotique est l'ensemble des activités de construction et de mise en œuvre des robots,on peut dire aussi que tout dispositif comporte une partie [opérationnelle]qui réalise la tâche et une partie [décisionnelle ou commande]qui contrôle la partie opérationnelle.

Définition d'un robot mobile

Un robot est un dispositif mécanique articulé capable d'imiter certaines fonctions humaines telles que la manipulation d'objets ou la locomotion, dans le but de se substituer à l'homme pour la réalisation de certaines tâches matérielles. Les robots mobiles ont la capacité de se déplacer dans leur environnement et ne sont pas fixés à un seul emplacement physique. Ils peuvent être "autonomes", ce qui signifie qu'ils sont capables de naviguer dans un environnement non contrôlé sans recourir à des dispositifs de guidage physiques ou électromécaniques. Les robots mobiles peuvent également utiliser des dispositifs de guidage leur permettant de parcourir une route de navigation prédéfinie dans un espace relativement contrôlé (AGV – véhicule guidé autonome) [2].

I.3 Historique de la robotique

Les origines de la robotique

Les ancêtres des robots sont les automates. Un automate très évolué fut présenté par Jacques de Vaucanson en 1738 : il représentait un homme jouant d'un instrument de musique à vent. Jacques de Vaucanson créa également un automate représentant un canard mangeant et refoulant sa nourriture après ingestion de cette dernière [2].

a) Les premiers robots

Unimate est le premier robot industriel créé. Il fut intégré aux lignes d'assemblage de Général Motors en 1961. En 1970, le robot lunaire Lunokhod 1 envoyé par l'union soviétique a voyagé sur une distance de 10 Km et a transmis plus de 20 000 images[3].

b) Dates marquantes de la robotique

- 1947 : Premier manipulateur électrique télé opéré.
- 1961 : Premier robot industriel mis en place dans une usine de General Motors. Qui a fait premier robot avec contrôle en effort.
- 1963 : Utilisation de la vision pour commander un robot.
- 1972 : Nissan ouvre la première chaine de production complètement robotisée.
- 1973 : Premier robot mobile à roues.
- 1977 : Premier robot mobile français HILARE au LAAS (CNRS Toulouse).
- 1992 : Mise en place de la compétition annuelle AAAI sur la robotique mobile.
- 1995 : Mise en place de la Roba Cup.
- 1997 : Premier robot mobile extra planétaire sur Mars.
  Depuis 2000 : Exploration :
- 2003 : Projet « Mars Exploration Rover » (Spirit and Opportunity).
- 2009 : Projet « Mars Science Laboratory » succédant au projet Rover, envoi prévu de Curiosity fin 2001.

I.4 Les composants d'un robot mobile

Les composants d'un robot mobile sont un contrôleur, un logiciel de contrôle, des capteurs et des actionneurs[2].

Le contrôleur est généralement un microprocesseur, un microcontrôleur intégré ou un ordinateur personnel (PC).

Le logiciel de contrôle mobile peut être un langage d'assemblage ou des langages de haut niveau tels que C, C ++, Pascal, Fortran ou un logiciel spécial en temps réel.

Les capteurs utilisés dépendent des exigences du robot. Les exigences peuvent être la mise au point, la détection tactile et de proximité, la télémétrie par triangulation, la prévention des collisions, la localisation et d'autres applications spécifiques.

Un actionneur est le constituant d'un système mécanique (exemple : bras, patte, roue motrice. . .) réalisant une action motrice. Il anime les interfaces réalisant les actions de saisies d'objets dans les applications de télémanipulation.

I.5 Structure d'un robot mobile

a) Module de locomotion

Le module locomotion comporte la motricité et l'énergie utilisée dans le déplacement du robot mobile. Les robots mobiles sont en effet le plus souvent désignés par leur type de locomotion, qu'ils soient à roues, marcheurs, sous-marins ou aériens. Différents laboratoire de recherche contribuent jusqu'à présent à la construction des différents robots mobiles selon le type de locomotion[4].

b) Module de perception

Perception en robotique est la capacité du système à recueillir, traiter et mettre en forme des informations utiles au robot pour agir et réagir dans le monde qui l'entoure[4].
Le robot doit acquérir des informations sur l'environnement dans lequel il évolue par l'intermédiaire de ses capteurs, qui peuvent être utilisés directement comme des entrés .

c) Module de décision

Le robot doit définir des séquences d'actions résultant d'un raisonnement appliqué sur un modèle de l'environnement ou répondant de manière réflexe à des stimulis étroitement liés aux capteurs[1].

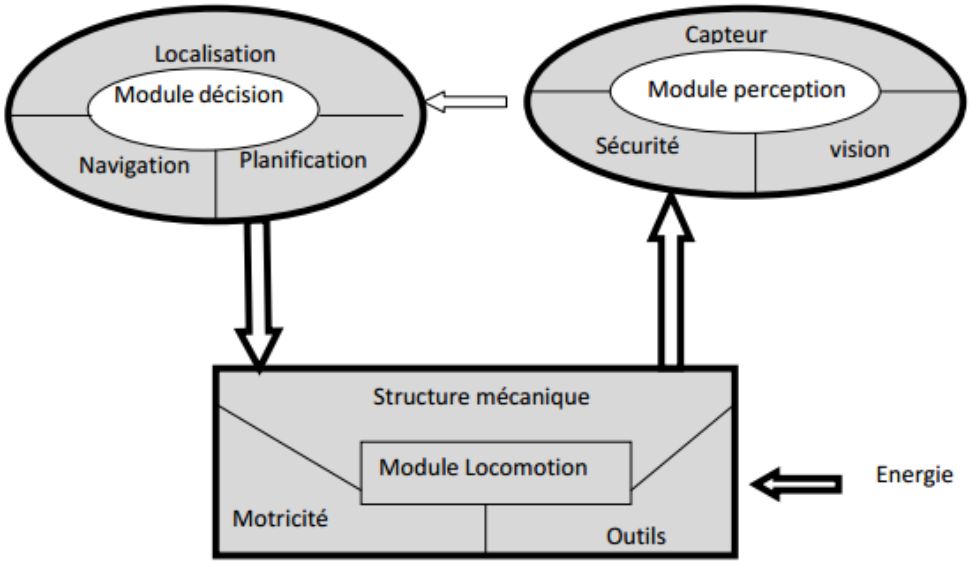


FIGURE I.1 – Structure d'un robot mobile

I.6 Les types de robot mobiles

La caractéristique la plus remarquable d'un robot mobile est évidement son moyen de locomotion. Celui-ci dépend directement du type d'application visé ainsi que de type de terrain dans lequel le robot mobile doit évoluer (environnement d'intérieur, extérieur, libre ou encombré d'obstacles,...). Les robots mobiles sont classés généralement selon le type de locomotion utilisé en quatre groupes distincts ; qu'ils soient : à roues, à chenilles, à pattes ou avec d'autres moyens de locomotions [1]. En effet, le type de locomotion définit deux types de contraintes :

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

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

Selon la cinématique, un robot est dit holonome, s'il peut se déplacer instantanément dans toutes les directions possibles. Il est dit non holonome, si ses déplacements autorisés sont des courbes dont la courbure est bornée.
Il existe plusieurs classes de robots à roues déterminées principalement par la position et le nombre de roues utilisées.

Nous citerons ici les quatre classes principales de robots à roues[5].

I.6.1 Robot unicycle

Un robot de type unicycle est actionné par deux roues indépendantes, il possède éventuellement un seul roue folle pour assurer sa stabilité.
Son centre de rotation est situé sur l'axe reliant les deux roues motrices.
C'est un robot non-holonome, en effet il est impossible de le déplacer dans une direction perpendiculaire aux roues de locomotion. Sa commande peut être très simple, il est en effet assez facile de le déplacer d'un point a un autre par une suite de rotations simples et de lignes droites.

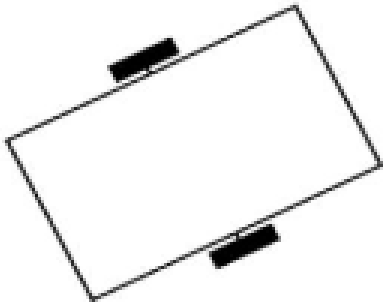


FIGURE I.2 – Robot de type unicycle

I.6.2 Robot tricycle

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

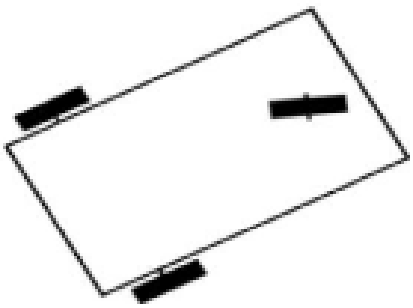


FIGURE I.3 – Robot de type tricycle

I.6.3 Robot voiture

Un robot de type voiture est semblable au tricycle, il est constitué de deux roues fixes placées sur un même axe et de deux roues centrées orientables placées elles aussi sur un même axe.
Le robot de type voiture est cependant plus stable puisqu'il possède un point d'appui supplémentaire. Toutes les autres propriétés du robot voiture sont identiques au robot tricycle le deuxièmes pouvant être ramené au premier en remplaçant les deux roues avant par une seule placée au centre de l'axe et ceci de manière à laisser le centre de rotation inchangé.

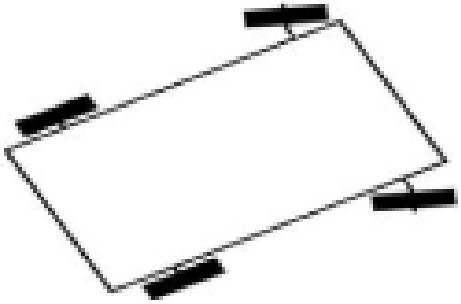


FIGURE I.4 – Robot de type voiture

I.6.4 Robot omnidirectionnel

Un robot omnidirectionnel est un robot qui peut se déplacer librement dans toutes les directions. Il est en général constitué de trois roues décentrées orientables placées en triangle équilatéral.
L'énorme avantage du robot omnidirectionnel est qu'il est holonome puisqu'il peut se déplacer dans toutes les directions. Mais ceci se fait au dépend d'une complexité mécanique bien plus grande [7].

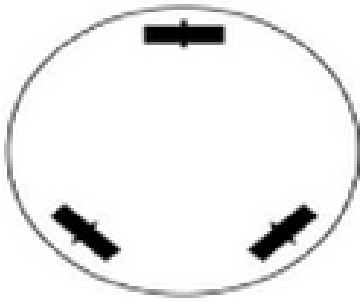


FIGURE I.5 – Robot de type omnidirectionnel

I.7 Modélisation d'un robot mobile

Pour la modélisation du robot mobile, il existe deux parties : la partie de planification et la partie de la commande.

Ici on s'intéresse à la partie commande.

I.7.1 Hypothèse de la modélisation

Généralement pour la commande des robots mobiles, un modèle de commande en vitesse est utilisé plutôt qu'un modèle de commande en couple[6]. 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.

Pour ces raisons, nous ne considérons dans les chapitres suivants que le modèles cinématique 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 indéformables, de rayon notée ci-dessous :
- Chaque contact roue/sol est réduit à un point.
- Les roues roulent sans glisser sur le sol[6].

Nous restreignons notre étude aux robots mobiles de type unicyle qui est le robot le plus simple, le moins coûteux et le plus pratique pour la recherche au niveau des laboratoires. En plus les études et les applications sur ce robot peuvent être généralisées facilement sur les autres types de robots.

I.7.2 Roulement sans glissement

On suppose que ce robot mobile est constitué d'un châssis rigide et de roues indéformables, et il se déplace dans un plan 2D horizontal dans le repère global (O, X, Y).
Le plan de chaque roue est perpendiculaire au sol et le contact entre les roues et le sol est idéal pour rouler sans déraper ni glisser ; la vitesse du centre du robot mobile est perpendiculaire à l'axe des roues. Sous ces hypothèses, la roue a deux contraintes. La première contrainte est que la roue doit rouler dans le sens de mouvement approprié, tandis que la seconde contrainte suppose que la roue ne doit pas glisser orthogonalement par rapport au plan de la roue[9] , comme le montre la figure I.6.

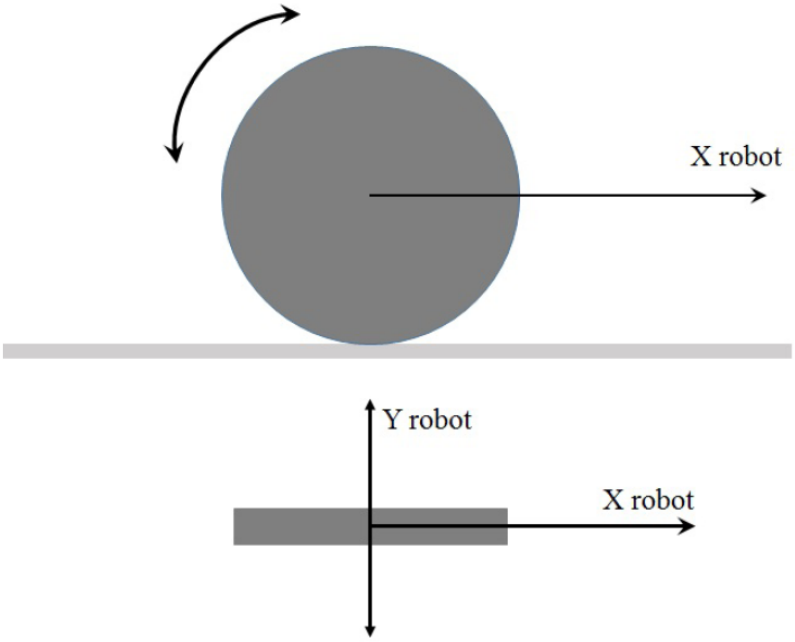


FIGURE I.6 – Roue de roulement du robot mobile

Le mouvement du robot mobile est actionné en modifiant les vitesses angulaires relatives des roues motrices comme illustré sur la Figure I.7. Selon le principe de mouvement de la cinématique à corps rigide, le mouvement d'un robot mobile à deux roues à entraînement différentiel peut être décrit comme

$$v_l = \omega_l r \tag{I.1}$$

$$v_r = \omega_r r \tag{I.2}$$

La vitesse angulaire du robot mobile est donnée par

$$\omega = \frac{v_l v_r}{L} \tag{I.3}$$

La vitesse angulaire du robot mobile est donnée par

$$v = \frac{v_l v_r}{2} \tag{I.4}$$

Le robot mobile se déplacera librement et suivra différentes trajectoires en modifiant les vitesses des deux roues comme illustré à la figure I.7.

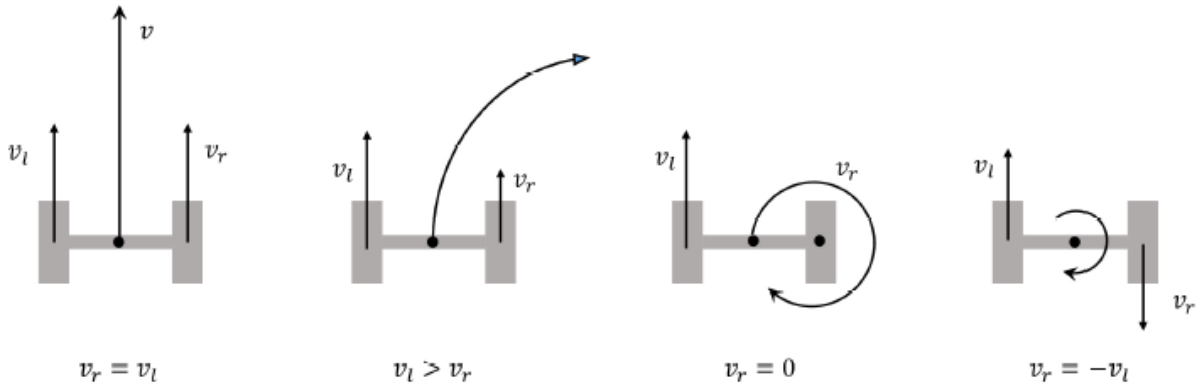


FIGURE I.7 – Les différentes possibilités de mouvement du robot unicycle

I.7.3 Configuration géométrique du robot unicycle

Le robot mobile unicycle considéré est constitué d'une plate-forme équipée de deux roues motrices indépendantes [8]. La configuration opérationnelle du système est donnée dans le référentiel (O, X, Y) par :

$$q = [x, y, \theta]^l \tag{I.5}$$

Où (x, y) représentent la position centrale du robot et θ l'orientation du robot autour de l'axe (O, X, Y) (voir la figure I.8).

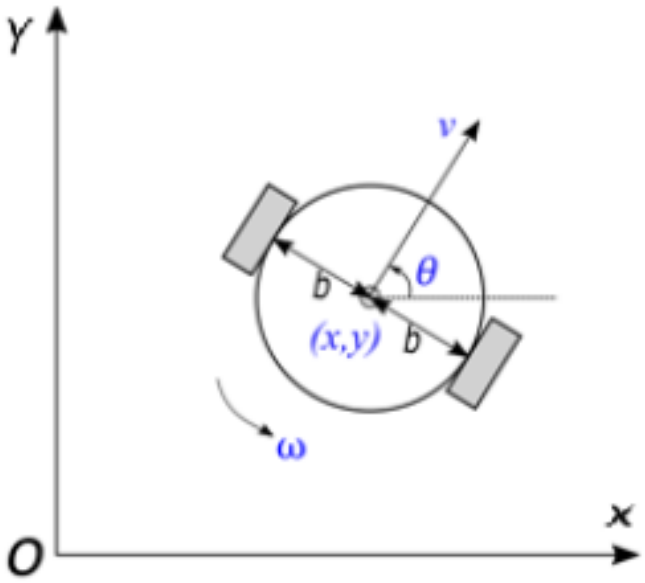


FIGURE I.8 – Représentation géométrique du robot mobile unicycle

I.7.4 Modélisation cinématique du robot mobile unicycle

La modélisation cinématique traite des relations géométriques qui contrôlent le robot mobile sans tenir compte des forces qui l'affectent. Par exemple, il fait référence à l'évolution de la position et des vitesses du système de robot mobile, sans se référer à sa masse et à son couple. Le modèle cinématique du robot est donnée par :

$$\dot{q} = s(q)v \tag{I.6}$$

$$v = [v\omega]^r.s(p) = \begin{pmatrix} cos\theta & 0 & 0 \\ sin\theta & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{I.7}$$

v et ω représentent respectivement la vitesse linéaire et la vitesse angulaire du robot. Les relations I.6 et I.7 sont équivalent à :

$$\dot{x} = vcos\theta \tag{I.8}$$

$$\dot{y} = vsin\theta \tag{I.9}$$

$$\dot{\theta} = \omega \tag{I.10}$$

La contrainte non holonomique signifie que le robot se déplace uniquement dans la direction normale à l'axe des roues motrices sans glissement, cette contrainte est donnée sous la forme d'une équation différentielle :

$$c(q)\dot{q} = 0 \tag{I.11}$$

Où

$$c(q) = [sin\theta \ -cos\theta \ 0] \tag{I.12}$$

I.7.5 Modélisation dynamique du robot

D'autre part, un modèle dynamique du robot mobile est l'étude du mouvement du robot en ce qui concerne sa vitesse, son emplacement et son accélération avec les forces et énergies internes, ou le moment d'inertie au sein du système. Le modèle dynamique utilise les lois de la physique qui régissent plusieurs parties du robot, notamment : la dynamique du robot (équations de mouvement), la dynamique de l'actionneur (caractéristiques électriques et mécaniques des moteurs) et les frictions [8].
Le modèle dynamique du robot unicycle, satisfaisant les contraintes cinématiques I.11, peut être obtenu en utilisant les équations de Lagrange :

$$\frac{\mathrm{d}}{\mathrm{d}t}(\frac{\partial T}{\partial \dot{q}}) - \frac{\partial T}{\partial q} + C^T\lambda = Q \tag{I.13}$$

Où q est le vecteur de coordonnées généralisé, T est le système énergie cinétique, λ est le vecteur multiplicateur de Lagrange et C est donné par I.12. L'équation I.13 peut s'écrire comme suit :

$$M(q)\ddot{q} + v(q,\dot{q}) = E(q)\tau - C^T(q)\lambda \tag{I.14}$$

Où

$$M = \begin{pmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_Z \end{pmatrix} .v = 0 \tag{I.15}$$

$$E = \frac{1}{r} \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & \cos\theta \\ b & -b \end{pmatrix} .\tau = \begin{pmatrix} \tau_r \\ \tau_l \end{pmatrix} \tag{I.16}$$

M étant la matrice d'inertie,$m = m_1 + 2m_2$ où m_1 est la masse de la plate-forme,m_2 est la masse d'une roue,I_2 est le moment d'inertie de la plate-forme autour de (O, Z),r est le rayon des roues,b est la demi-distance entre les deux roues, et τ_r et τ_l sont les couples d'actionneurs des roues droite et gauche, respectivement. On suppose que les moments d'inertie des deux roues sont négligeables par rapport à celle du robot mobile.
En dérivant l'équation I.6, nous obtenons :

$$\ddot{q} = \dot{s}(q)v + s(q)\dot{v} \tag{I.17}$$

En remplaçant $\ddot{q}$ par son expression dans l'équation I.14, et en multipliant par S^T on trouve :

$$\dot{v} = fr \tag{I.18}$$

Où

$$f = (s^T M S)^{-}1 s^T E.\dot{v} = [\dot{v}\dot{\omega}]^T \tag{I.19}$$

Enfin, les couples moteurs peuvent être exprimés comme suit :

$$\begin{pmatrix} \tau_r \\ \tau_l \end{pmatrix} = f^{-}1 \begin{pmatrix} \dot{(v)} \\ \dot{(\omega)} \end{pmatrix} \tag{I.20}$$

Avec

$$f^{-}1 = \begin{pmatrix} \frac{mr}{2} & \frac{rI_Z}{2b} \\ \frac{mr}{2} & -\frac{rI_Z}{2b} \end{pmatrix} \tag{I.21}$$

I.8 Conclusion

Ce chapitre présente des généralités importantes sur la robotique mobile. En premier temps, nous avons vu une petite définition sur la robotique et les robots mobiles et une brève historique sur celles. Ensuite les différents types des robots mobiles à roues. Nous avons ensuite défini les liaisons de non holonomie imposée au mouvement et qui caractérisent le contact roues motrice-sol qui doit se faire sans glissement (roulement sans glissement des roues) permettant l'établissement des lois de mouvement du robot mobile. Enfin, nous avons présenté la formalisation de Lagrange qui permet d'établir le lien entre les positions, vitesses et accélération du robot .Dans le chapitre suivant, nous allons faire un état de l'art des méthodes permettant au robot de planifier sa trajectoire d'un point de départ pour atteindre un point destination, et la commande qui lui permet de suivre le chemin ainsi planifié tout en se localisant à chaque instant de son parcours.

Chapitre II

Méthodes de planification et de suivi de trajectoire

II.1 Introduction

Les robots mobiles sont largement utilisés dans les environnements industriels pour le transport de produits par exemple. Le plus souvent ces tâches sont répétitives et suivent un chemin bien défini parfois même bien matérialisé comme des lignes sur le sol.

Il y a actuellement une forte tendance à élargir les milieux où évoluent les robots à des environnements de bureaux ou à des environnements domestiques (robots de service). Les types d'applications possibles sont innombrables. Cela peut être des tâches de nettoyage et d'entretien ou encore une assistance à une personne handicapée, ou dans des tâches d'exploration et de préhension. Un robot peut également servir de guide pour la visite d'un musée. Il peut être considéré comme une machine équipée de capacités de perception, de décision et d'action qui lui permette d'agir de manière autonome dans son environnement en fonction de la perception qu'il en a.

La robotique mobile vise à rendre un système autonome dans ses déplacements. Un robot, doté de capacités de perception et d'information sur son environnement, doit pouvoir se mouvoir en autonomie, sans se perdre et tout en évitant les obstacles.
La planification automatique de trajectoire des robots sans collision dans des environnements, a été le sujet d'un nombre très important de recherches ces dernières années. Depuis une vingtaine d'années, plusieurs techniques de planification ont été proposées, mais très peu de ces techniques sont efficaces à résoudre de façon satisfaisante le problème de la planification, notamment dans l'industrie[9].

II.2 Problématique

Pour un robot A évoluant dans un environnement W donné, le problème général de planification consiste à déterminer pour A un mouvement lui permettant de se déplacer entre deux configurations données tout en évitant les obstacles et en respectant un certain nombre de contraintes et de critères. Ceux-ci découlent de plusieurs facteurs de natures diverses et dépendent généralement des caractéristiques du robot, de l'environnement et du type de tâche à exécuter. En l'occurrence, les contraintes relatives au robot concernent sa géométrie, sa cinématique, sa dynamique et son architecture pouvant correspondre à un robot mobile, à un système articulé d'objets rigides tel qu'un bras manipulateur, une main à plusieurs doigts ou un véhicule tractant des remorques, ou encore à plusieurs systèmes de robots à coordonner tels que des robots mobiles de type voiture évoluant dans un réseau routier. Les contraintes émanant de l'environnement concernent essentiellement la non-collision aux obstacles fixes encombrant W et la prise en compte d'interactions de contact avec le robot.

II.3 Méthodes de planification de trajectoire

La planification de chemin est la formulation la plus simple du problème de planification de mouvement puis que l'environnement dans lequel évolue le robot est statique.
Ainsi, le problème se réduit à l'aspect géométrique de la solution et par conséquent toute courbe qui assure les contraintes de continuité et de la non-collision est acceptable comme solution et le défi revient, en général, à calculer le plus court chemin entre la configuration initiale et la configuration finale.

Plusieurs méthodes ont été développées et en fonction de la connaissance à priori que le robot dispose de son environnement, deux familles d'approches se présentent :

II.3.1 Approches de planification globales

L'approche globale c'est une méthode utilisé dans le cas d'un environnement partiellement ou complètement connu. Elle utilise un modèle de l'espace libre dont l'espace de configuration permet la recherche exhaustive de la trajectoire de cout minimum au sens de critère donné ou de conclure à la non-exhaustive d'une trajectoire amenant le système de la configuration initiale à la configuration finale. Notons que cette approche est très couteuse en temps de calcul qu'en occupation mémoire, mais son avantage consiste à la garantie d'arrivée au but en suivant un chemin optimal. Une multitude des méthodes ont à ce jour été proposées nous mentionnons cidessous quelques méthodes[10] :

II.3.1.1 Les roadmaps

Cette classe de solutions au problème de planification vise à capturer la topologie de l'espace des configurations libres des obstacles Cfree afin de simplifier le problème à une recherche d'un plus court chemin dans un graphe connectant la configuration initiale à la configuration finale. Ces méthodes se déroulent, donc, en deux phases :
En premier lieu la construction du graphe dans l'espace de recherche. Et ensuite, le parcours de ce graphe dans l'objectif de calculer une solution.
Cette deuxième étape s'appuie sur la théorie des graphes en faisant appel aux algorithmes de recherche de plus court chemin comme Bellman, Dijkstra, A*.
Quant à l'étape de construction du graphe, nous présentons ci-dessous les méthodes les plus répandues[10].

a) Les graphes de visibilité

Les graphes de visibilité sont proposés par Nilsson en 1969, suite à l'idée que le chemin le plus court pour joindre la configuration finale à partir de l'initiale, passe par les sommets des obstacles polygonaux placés dans l'environnement. Ce graphe est construit en reliant avec des droites

tous les sommets des obstacles entre eux, sans que ces droites ne coupent d'autres obstacles(voir figure II.1). En effet, la configuration initiale est reliée par des droites aux sommets visibles des obstacles. Ce processus est réitéré entre les sommets des obstacles et les sommets visibles des obstacles suivants jusqu'à atteindre la configuration finale.Vu que cette méthode autorise le contact entre le robot et les obstacles, elle devient relativement peu employée pour la résolution du problème de planification [9].

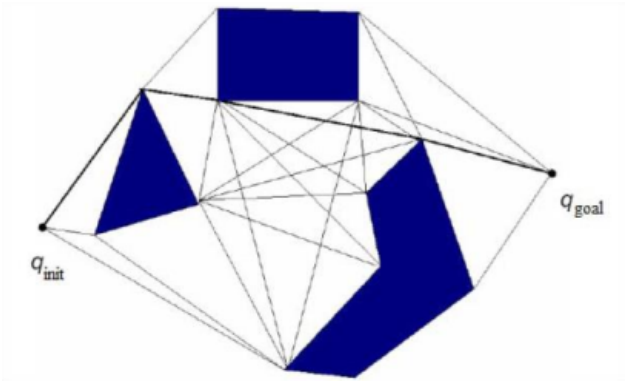


FIGURE II.1 – Trois obstacles, Graphe de visibilité et Résultat de planification

b) Les diagrammes de voronoï

Une deuxième méthode pour construire une roadmap est de se servir des diagrammes de Voronoï qui a proposé par Takahashi and Schilling en 1989, qui contrairement aux graphes de visibilité, évitent le contact avec les obstacles et valident, par conséquence, la distance de sécurité. La construction d'un diagramme de Voronoï se fait en traçant des lignes d'égale distance aux obstacles et au bord de l'environnement exploré. Ensuite le graphe est obtenu en raccordant la configuration initiale au point le plus proche de ce diagramme et de même pour la configuration finale [11]. Cette procédure est illustrée par la figure II.2 .

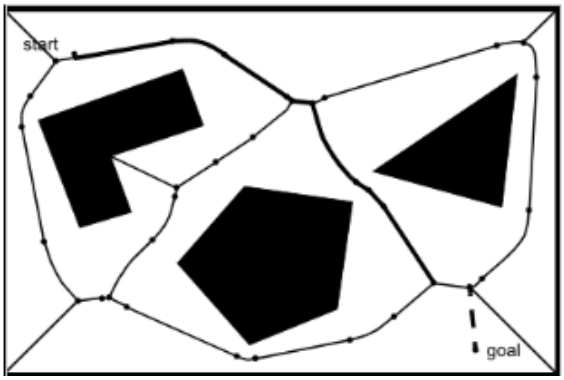
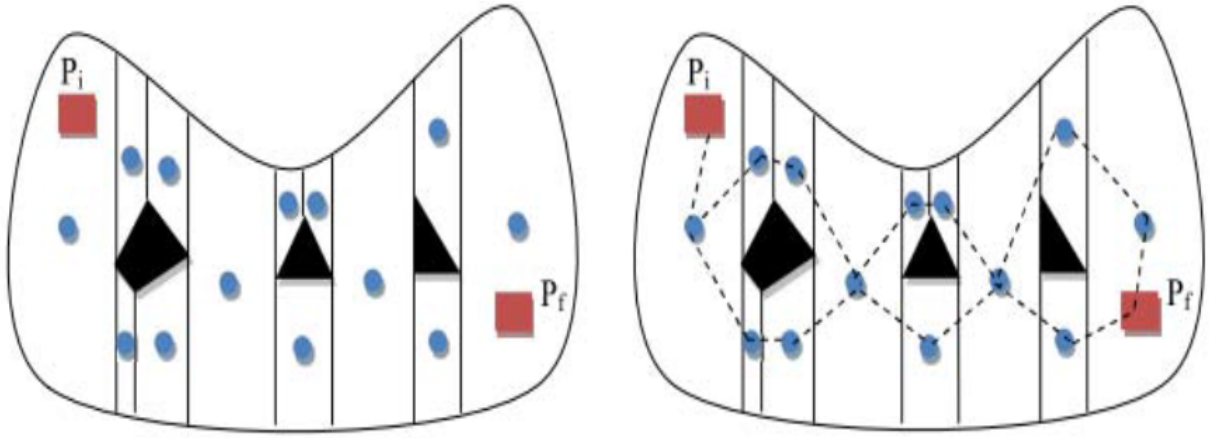


FIGURE II.2 – Planification de chemin à partir d'un diagramme de voronoï

c) La décomposition cellulaire

Les approches par décomposition cellulaire consistent à partitionner l'espace de configurations libres en un ensemble de cellules adjacentes (régions connexes). Il existe différentes techniques de décomposition de l'espace qui se classifient en deux catégories : les méthodes exactes et les méthodes approchées. Les méthodes exactes effectuent une décomposition qui se base sur un recouvrement exact de l'espace des configurations libres. Quant aux méthodes approchées, elles tentent de rapprocher la structure de l'espace libre avec des cellules qui ont une forme simple comme la forme rectangulaire. Ainsi, si la décomposition est exacte, l'union de ces cellules permet de retrouver l'espace des configurations libres. Un graphe de connexion peut alors être construit à partir des barycentres de ces cellules et les configurations initiale et finale du robot (voir figure II.3) [10].



(a) Décomposition en cellules (b) Génération du graphe.

FIGURE II.3 – Construction d'une roadmap par la technique de décomposition cellulaire.

II.3.1.2 Les méthodes heuristiques

On sépare tout d'abord les méthodes d'optimisation en deux grandes familles de méthodes : les méthodes exactes et les méthodes approchées. les méthodes heuristique est l'un des méthodes approchées.
Il existe deux types de méthode heuristique : métaheuristiques et heuristiques dédiées.

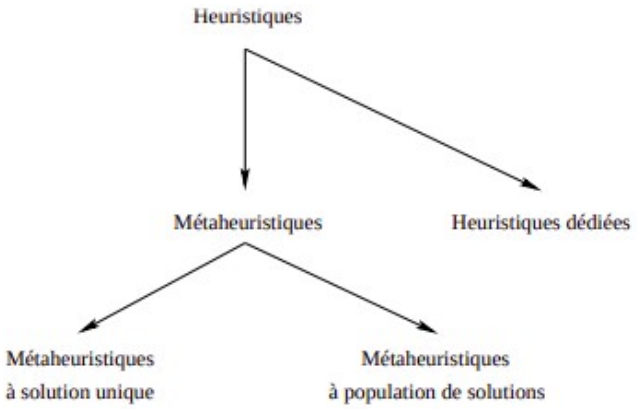


FIGURE II.4 – Hiérarchie des méthodes d'optimisation.

a) Métaheuristique

Une métaheuristique est un algorithme d'optimisation visant à résoudre des problèmes d'optimisation difficiles pour lesquels on ne connaît pas de méthode classique plus efficace. Les métaheuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global. Parmi ces méthodes on a :

Les algorithmes génétiques

Les algorithmes génétiques (AG) ont été développés dans les années 60 par John Holland. Ils tentent de simuler le processus d'évolution naturelle suivant le modèle darwinien dans un environnement donné et ils ont été appliqués à différents problèmes d'optimisation. Cet algorithme a été largement utilisé pour résoudre les problèmes de planification demouvement. Comme illustré à la figure II.5, son principe consisteà représenter des solutions aléatoires sous forme d'une population d'individus et de simuler le processus évolutif par la sélection, la reproduction (croisement) et les mutations. Ainsi, les solutions encodées évoluent pour optimiser la fonction de coût. Par conséquent, la population s'améliore et évolue vers une trajectoire quasi optimale. Une technique commune pour la planification des trajectoires est d'encoder chaque individu comme une trajectoire complète (ensemble de coordonnées euclidiennes 2D ou 3D).

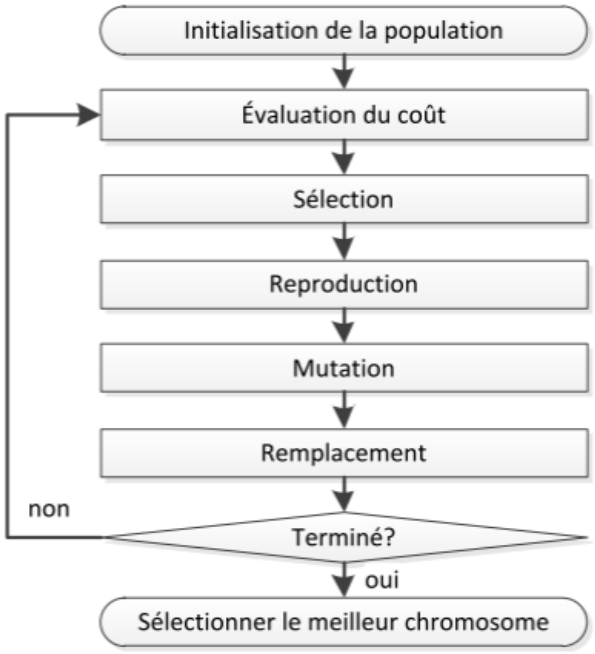


FIGURE II.5 – Schéma général d'un algorithme génétique.

Le recuit simulé

Le Recuit simulé est un algorithme stochastique développé par Kirkpatrick et alen 1983. Cette méthode s'inspire du principe physique du recuit des matériaux solides. En fait, le chauffage du matériau à très haute température, puis son refroidissement lent conduisent à une réorganisation plus ordonnée des atomes constituant le matériau, et l'amenant par conséquent vers un état plus stable (énergie plus faible). Le déroulement de l'algorithme est comme suit :
Une solution possible est encodée dans une particule qui se déplace dans un espace multidimensionnel. Au début du procédé, lorsque la température est élevée, la particule se déplace presque aléatoirement allant même vers une solution avec un plus haut coût.
Afin de favoriser l'exploration. À mesure que le procédé avance et que la température baisse, cet aspect aléatoire diminue et la particule se dirige principalement vers une solution qui minimise la fonction de coût.

Les avantages de l'algorithme du recuit simulé sont la grande aptitude à éviter les minima locaux et à converger vers le minimum global. En plus, cette technique n'exige pas de condition particulière sur les particularités de la fonction coût. Néanmoins, cet algorithme converge vers une solution approchée et non une solution exacte et aussi la vitesse de convergence est plus faible au voisinage du minimum global. Le recuit simulé permet une exploration plus limitée de l'espace de recherche puisqu'il ne se base pas sur l'utilisation d'une population de solutions, contrairement à d'autres algorithmes d'optimisation comme les algorithmes génétiques par exemple. Pour le pro-

blème de planification de trajectoires, le recuit simulé a été le plus souvent utilisé pour améliorer ou compléter un autre algorithme (exemple le gradient de potentiel).

b) Heuristique

La méthode heuristique conçu pour un problème d'optimisation paticulier permettant de trouver des solutions avec un temps de calcul raisonable, comme l'algorithme Astar.

L'algorithme A star

L'algorithme Astar (on l'appelle également Algorithme A*), qui peut trouver le chemin le plus court sans collision à travers un environnement entièrement mappé en utilisant une file d'attente prioritaire. Astar est un algorithme itératif, à chaque itération, A* va tenter de prendre le chemin le plus court pour aller d'une position Initiale à une position Destination. Pour déterminer si un noeud est susceptible de faire partie du chemin solution, il faut pouvoir quantifier sa qualité qui est calculée par l'équation suivante :

$$F = G + H \tag{II.1}$$

Le plus souvent, on calcule la distance entre le point étudié et le dernier point qu'on a jugé comme bon G. Et on calcule aussi la distance entre le point étudié et le point de destination H. La somme de ces deux distances nous donne la qualité du noeud étudié F. Plus un noeud à une qualité faible, meilleur il est [12].

II.3.1.3 Les méthodes probabilistes

Ces méthodes sont basées sur la construction préalable d'un graphe de l'environnement. Elles sont appelées aussi "Probabiliste RoadMap". Contrairement aux deux méthodes précédentes, les noeuds ne sont pas choisis selon la géométrie de l'environnement et des obstacles, mais par un tirage aléatoire dans l'espace libre. Une fois le graphe construit, lorsqu'un problème de planification est posé, on cherche simplement à connecter les configurations initiale et finale [11].
Nous détaillons ici les deux principales méthodes que sont PRM et RRT. Ces deux méthodes se sont imposées au fil du temps comme méthodes de base de la planification de mouvement probabiliste. La façon dont les méthodes probabilistes explorent l'environnement dépend d'abord de la méthode d'échantillonnage aléatoire. L'échantillonnage le plus répandu est uniforme sur l'espace des configurations. Cela donne une probabilité égale pour chaque point d'être choisi. Le succès de ces méthodes est largement dépendant de la manière dont se fait l'échantillonnage et de la fonction qui permet de connecter ces échantillons.

a) La méthode PRM

La méthode PRM (Probabilistic Roadmap Method)a été introduite par Kavraki. L'idée de cette

méthode est d'échantillonner l'espace des configurations pour trouver suffisamment de configurations sans collisions pour permettre de représenter l'espace des configurations libres. L'algorithme détermine si les configurations échantillonnées sont en collision ou non grâce à un détecteur de collision. Celles qui sont dans l'espace libre sont gardées et l'algorithme tente de les relier par une arête au graphe contenant les autres nœuds. Le lien se fait à l'aide d'une méthode locale permettant de relier un nœud à un autre le long d'un espace libre. Le graphe comporte initialement deux nœuds, la configuration initiale et la configuration finale. Lorsque l'on souhaite trouver le chemin entre deux nœuds du graphe, celui-ci est parcouru à l'aide d'un algorithme de type A*, voir figure II.6.
Cette méthode a l'avantage de pouvoir conserver le graphe entre deux requêtes de planification. Cela permet d'effectuer plusieurs planifications successives très rapidement sur le même environnement en gardant le même graphe mais avec des configurations initiales et finales différentes. Son désavantage est celui de toutes les méthodes de planification demouvement probabilistes, il faut que le graphe soit suffisamment représentatif de l'espace libre avant qu'il ne soit utilisable pour des requêtes de planification de mouvement. Cette étape est donc généralement effectuée hors ligne car la génération d'une roadmap est coûteuse en temps ; le graphe de l'environnement obtenu est conservé pour des requêtes effectuées en ligne ultérieurement[13].

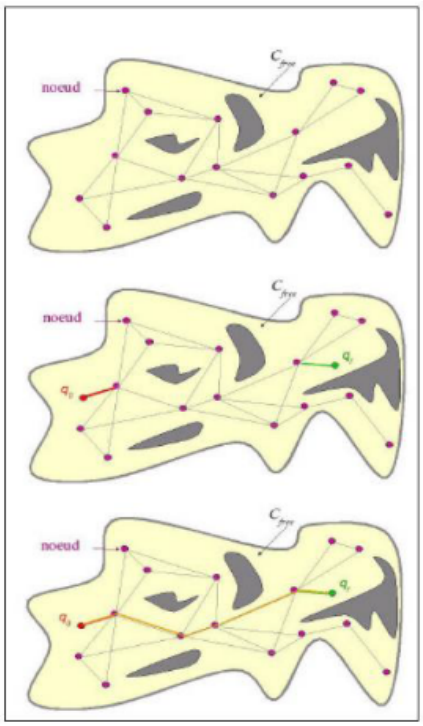


FIGURE II.6 – Planification par PRM.

b) La méthode RRT

L'algorithme RRT, pour Rapidly-exploring Random Trees. Il représente l'un des algorithmes

les plus populaires au cours de ces dernières années. RRT fait partie de la famille des méthodes par diffusion de roadmap. Il permet de construire une roadmap rapidement, à usage unique, dont le but n'est pas de couvrir l'intégralité de l'environnement mais simplement de répondre à une unique requête de planification. Au lieu de construire un graphe, le RRT construit un arbre qui est un graphe ne contenant pas de boucles [13].

Dans notre travail, on s'intéresse aux algorithmes RRT et Astar vu à leur simplicité et efficacité dans les problèmes de planification de trajectoire en robotique mobile. Ils seront détaillés dans la section II.4 et II.5.

II.3.2 Approches de planification locales

Les techniques locales n'utilisent pas de modèle complet de l'espace libre. Elles sont sans mémoire, et ne prennent en compte à un instant donné que l'environnement proche du mobile pour modifier une trajectoire de consigne. Ces méthodes sont attrayantes par leur simplicité, bien sûr elles présentent un inconvénient majeur de ne pas pouvoir suivre le chemin optimal. Le système peut être bloqué par des dispositions concaves d'obstacle. Ces méthodes sont beaucoup moins couteuses et ne utilises pour acquérir des informations nécessaires au fur et à mesure de déplacement les méthodes locales existantes sont[10] :

II.3.2.1 Les champs de potentiel

Initialement proposée par Oussama Khatib en 1985, l'approche par champs de potentiels a comme idée, la création d'un champ de potentiel artificiel qui guidera les déplacements du robot dans son environnement. Cette approche s'appuie sur le fait qu'elle considère le robot comme une particule plongée dans un champ de potentiel où il est attiré par le but et repoussé par les obstacles. De ce fait, le modèle de l'environnement est spécifié par une fonction de potentiel qui détermine les forces qui s'exercent dans le système robotique (Figure II.7). Bien que cette technique se marque par sa simplicité de présentation et de mise en œuvre et son faible coût de calcul, elle est sujette à plusieurs limites notamment le problème des minima locaux qui provoquent l'immobilité du robot. Par ailleurs, les champs de potentiel peuvent engendrer, dans certaines situations, des mouvements oscillatoires (lors de la traversée d'un passage étroit entre les obstacles). De plus, cette technique ne permet pas de détecter un passage entre des obstacles assez proches.
Aussi, le robot ne converge pas exactement vers le but si un obstacle se trouve proche de ce dernier. En dépit de ces limites, plusieurs adaptations de cette méthode ont été présentées. Pour éviter les problèmes liés aux minima locaux, certains proposent de suivre un comportement particulier (suivi des murs ou déplacement aléatoire) dès lors la détection d'une telle situation. D'autres proposent que les champs répulsifs soient calculés non seulement en fonction de la distance aux obstacles mais aussi la vitesse de ces derniers [14].

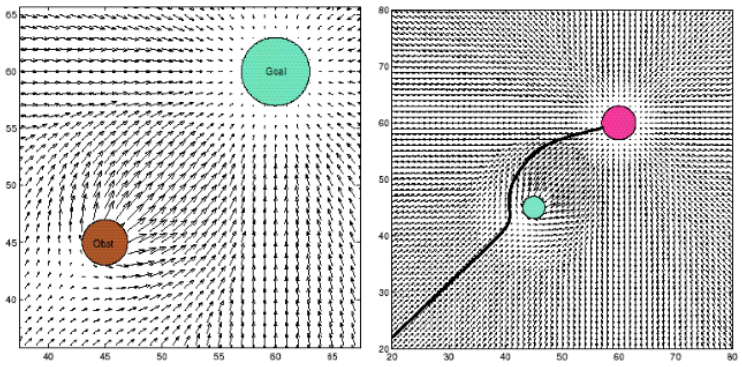


FIGURE II.7 – Vecteur somme de deux champs de potentiel (à gauche) et la trajectoire du robot résultante (à droite).

II.3.2.2 La fenêtre dynamique

Cette technique, proposée par Dieter Fox et al en 1997, est une méthode d'évitement d'obstacles en temps réel qui travaille dans l'espace des commandes du robot. Cette méthode détermine le domaine des vitesses possibles du robot, c'est à dire celles qui n'entraînent pas de collisions. Puis, une fois l'espace de recherche est calculé, les commandes envoyées au robot sont le résultat de la maximisation sur ce domaine d'une certaine fonction de coût, qui peut être la minimisation du temps de parcourt, la maximisation de la vitesse ou encore la minimisation de l'énergie dépensée. Ainsi, à partir de la perception locale de l'environnement, cet algorithme permet de sélectionner un couple (v, ω) de vitesses de translation et de rotation du robot qui résout les différentes contraintes, dont celle d'éviter les obstacles (voir la figure II.8)[15].

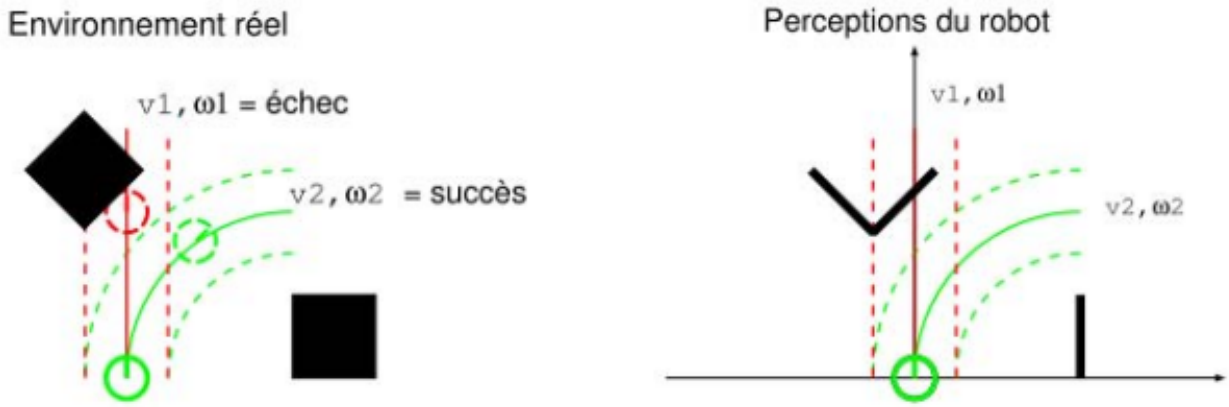


FIGURE II.8 – Évitement d'obstacles pour la méthode de la fenêtre dynamique

Bien que cette méthode permette de prendre en compte les contraintes cinématiques du robot, son implémentation dans un cadre multi-robot s'avère difficile vu le manque de flexibilité. De plus, seul les positions courantes des obstacles étaient prises en compte, mais pas leurs mouvements. Ainsi, la navigation dans un environnement dynamique en utilisant cette approche était fortement

compromise. Cette dernière calcule, à chaque instant, un ensemble de trajectoires évitant les obstacles puis une vérification de collision à court terme peut être opérée.

II.4 L'algorithme RRT

Initialement proposée par Lavalle 1998, qui consiste à construire un arbre, en partant de la position initiale du robot qui est le nœud racine de l'arbre. L'algorithme RRT explore progressivement l'espace des configurations pour trouver un chemin atteignant la position but.
L'objectif de l'algorithme RRT est de construire un arbre, c'est-à-dire un graphe généralement non-orienté et dont les différentes composantes connexes ne contiennent pas de boucles. Chaque nœud du graphe possède donc un seul et unique nœud parent (sauf pour la racine de l'arbre, qui est le premier nœud du graphe, en général la configuration initiale) et plusieurs nœuds fils (sauf pour les feuilles de l'arbre qui n'en ont pas) [16].
Le principe est le suivant : à partir d'une configuration initiale qinit, l'environnement du robot est exploré en choisissant arbitrairement à chaque itération une nouvelle configuration qrand non obstruée. La deuxième étape consiste à déterminer le nœud qnear le plus proche de qrand dans l'arbre existant. L'idée ensuite serait d'essayer d'étendre l'arbre à partir de qnear dans la direction de qrand d'une longueur ". Enfin si cette extension réussit, la nouvelle configuration créée qnew sera ajoutée dans l'arbre. Ce processus sera réitéré jusqu'à ce que la configuration finale qgoal soit atteinte. Ce principe est illustré dans la figure II.9.

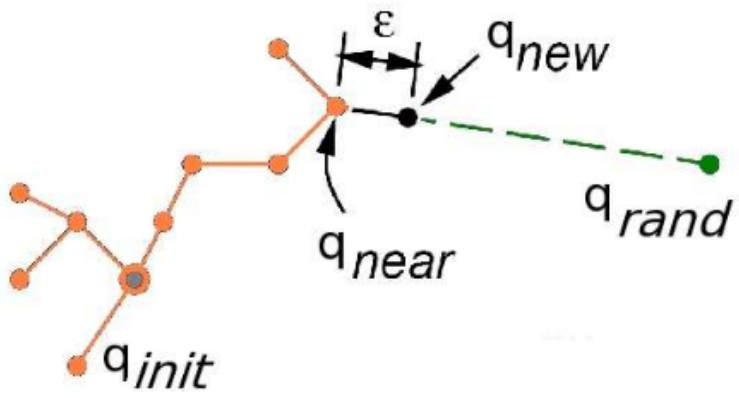


FIGURE II.9 – Illustration du principe d'extension du RRT.

La méthode de planification RRT se compose de deux phases :

- une phase de construction d'un arbre couvrant l'espace libre (voir figure II.10).
- une phase de détermination du chemin entre deux nœuds de l'arbre en utilisant un algorithme de de recherche de plus court chemin comme Bellman, Dijkstra.

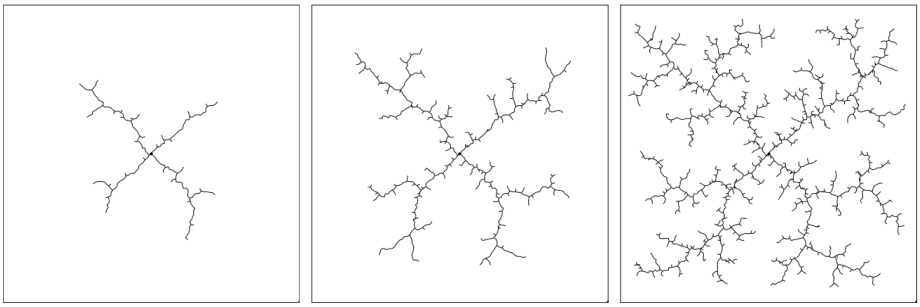


FIGURE II.10 – Evolution d'un arbre couvrant l'espace libre par la méthode RRT

La performance de cette méthode vient du fait qu'elle ne nécessite pas de phase de pré-calcul. Une propriété intéressante inhérente de cette approche est que la croissance des arbres est fortement biaisée en faveur des zones inexplorées de l'espace de configuration, l'exploration se fait rapidement. Malgré ces avantages, le RRT comme toute approche peut avoir des lacunes ; elle produise des cheminsqui sont parfois loins d'être optimaux.

II.5 L'algorithme Astar

L'algorithme de recherche A* (A star) est un algorithme de recherche de chemin dans un graphe entre un nœud initial et un nœud final tous deux donnés. De par sa simplicité il est souvent présenté comme exemple typique d'algorithme utilisé en intelligence artificielle. L'algorithme A* a été créé pour que la première solution trouvée soit l'une des meilleures, c'est pourquoi il est célèbre dans des applications comme les jeux vidéo privilégiant la vitesse de calcul sur l'exactitude des résultats. Cet algorithme a été proposé pour la première fois par Peter E. Hart, Nils Nilsson et Bertram Raphael en 1968. Il s'agit d'une extension de l'algorithme de Dijkstra de 1959 .
A chaque itération, A* va tenter de prendre le chemin le plus court pour aller d'une position source à une position destination. Pour déterminer si un nœud est suceptible de faire partie du chemin solution, il faut pouvoir quantifier sa qualité. Le plus souvent, on utilise la distance en "vol d'oiseau" entre la position du noeud et destination, mais d'autres fonctions peuvent être utilisées [17].
On travaille à partir d'une position de départ, d'une arrivée, d'un terrain, et de deux listes de nœuds (ouverte et fermée), vides à l'origine.

La liste ouverte va contenir tous les nœuds étudiés. Dès que l'algorithme va se pencher sur un nœud, il passera dans la liste ouverte (sauf s'il y est déjà).

La liste fermée contiendra tous les nœuds qui à un moment où à un autre, ont été considérés comme faisant partie du chemin solution.

1. On commence par le nœud de départ (position initiale), on le place dans la liste fermée et

il sera considéré comme étant le nœud courant (on peut placer également les nœuds faisant partie des obstacles dans la liste fermée pour ne pas les traiter par la suite).

2. On regarde quels sont ses nœuds voisins.
3. Pour chaque nœud voisin :
   - Si c'est un obstacle, on passe (on ne traite pas ce nœud, il ne sera ajouté nulle part)
   - S'il est déjà dans la liste fermée, on passe (on a déjà étudié ce trajet)
   - S'il est déjà dans la liste ouvert, On vérifie sa qualité. Si la qualité actuelle est meilleure, c'est que le chemin est plus court en passant par le trajet actuel, donc on met à jour sa qualité dans la liste ouverte, et on met à jour son lien parent (avec le nœud courant qui correspond à un meilleur chemin)
   - Sinon, on ajoute ce nœud à la liste ouverte avec comme parent le nœud courant.
4. On cherche le meilleur nœud de toute la liste ouverte. Si la liste ouverte est vide, il n'y a pas de solution, fin de l'algorithme.
5. On le retire de la liste ouverte et on le met dans la liste fermée.
6. Ce nœud est considéré comme nœud courant. Si ce nœud courant est identique au nœud Destination alors c'est bon, on a trouvé un bon chemin, sinon on réitère en 2.

II.6 Algorithme de suivi de trajectoire

La poursuite pure est un algorithme de suivi de chemin (pure pursuit). Le suivi de trajectoire est le processus qui consiste à déterminer les paramètres de vitesse et de direction à chaque instant afin que le robot suive le chemin désirée. L'algorithme calcule la commande de vitesse angulaire qui déplace le robot de sa position actuelle pour atteindre un point d'anticipation devant le robot (look-ahead point). L'algorithme déplace ensuite le point d'anticipation sur le chemin en fonction de la position actuelle du robot jusqu'au dernier point du chemin. Vous pouvez considérer cela comme le robot pourchassant constamment un point devant lui. La propriété Look Ahead Distance décide à quelle distance le point d'anticipation est placé.

Les vitesses angulaires, linéaires et maximales souhaitées peuvent être spécifiées. Ces propriétés sont déterminées en fonction des spécifications du véhicule. Compte tenu de la pose (position et orientation) du véhicule en entrée. La façon dont le robot utilise les commandes de vitesses linéaires et angulaires dépend du système que vous utilisez. La dernière propriété importante est la Look Ahead Distance, qui indique au robot jusqu'où il doit suivre sur le chemin.

Il est important de comprendre le cadre de coordonnées de référence utilisé par l'algorithme de poursuite pure pour ses entrées et ses sorties. La figure ci-dessous montre le système de coordonnées de référence. Les points de cheminement d'entrée sont des coordonnées [x y], qui sont

utilisées pour calculer les commandes de vitesse du robot. La pose du robot est entrée sous la forme d'une liste de points de pose et d'orientation (thêta) sous la forme [x y thêta]. La valeur thêta est l'orientation angulaire du robot mesurée dans le sens inverse des aiguilles d'une montre en radians à partir de l'axe des x (robot actuellement à 0 radians) [18].

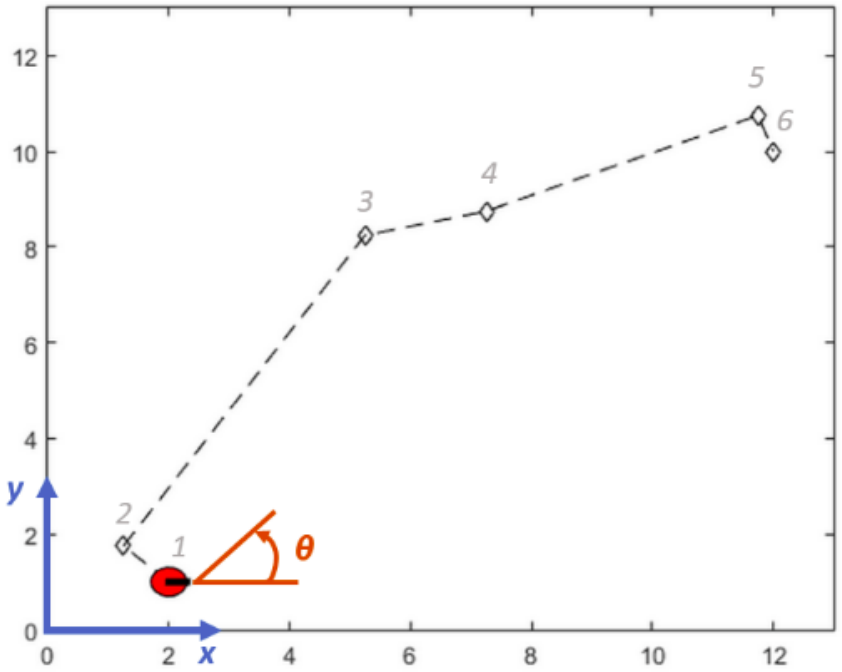


FIGURE II.11 – Illustration du principe de l'algorithme Pure Pursuit.

La propriété LookAheadDistance est la propriété de réglage principale du contrôleur. La distance d'anticipation est à quelle distance le long de la trajectoire le robot doit regarder à partir de l'emplacement actuel pour calculer les commandes de vitesse angulaire. La figure ci-dessous montre le robot et le point d'anticipation. Comme affiché dans cette image, notez que le chemin réel ne correspond pas à la ligne directe entre les points de cheminement(Waypoints).

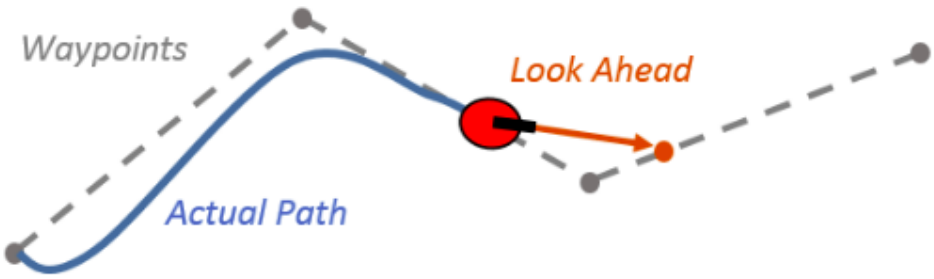


FIGURE II.12 – Exemple de Suivi de chemin par l'algorithme Pure Pursuit

L'effet de la modification de ce paramètre peut changer la façon dont notre robot suit le chemin et il y a deux objectifs principaux : retrouver et maintenir le chemin. Afin de retrouver rapidement le chemin entre les waypoints, une petite Look Ahead Distanceamènera notre robot à se déplacer rapidement vers le chemin. Cependant, comme le montre la figure II.13, le robot dépasse la trajectoire et oscille le long de la trajectoire souhaitée.

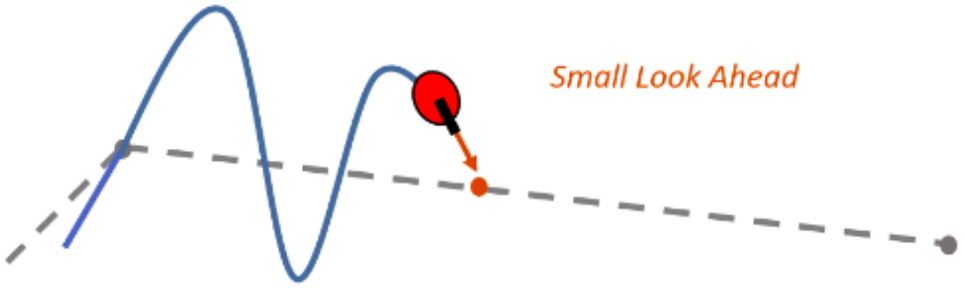


FIGURE II.13 – Exemple de Suivi de chemin par l'algorithme Pure Pursuit pour une petite distance d'anticipation.

Afin de réduire les oscillations le long du chemin, une distance d'anticipation plus grande peut être choisie, cependant, cela peut entraîner des courbures plus importantes près des coins comme la montre la figure ci-dessous.

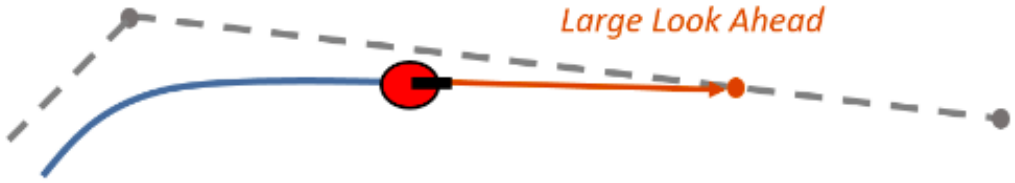


FIGURE II.14 – Exemple de Suivi de chemin par l'algorithme Pure Pursuit pour une grande distance d'anticipation.

La propriété Look Ahead Distance doit être adaptée à votre application et à votre système de robot. Différentes vitesses linéaires et angulaires affecteront également cette réponse et doivent être prises en compte pour le contrôleur de suivi de trajectoire.

II.7 Conclusion

Nous avons présenté dans ce chapitre un état de l'art sur la planification de mouvement pour les robots mobiles. Nous avons également présenté les avantages et les limites de chaque méthode, dont certaines font partie des méthodes globales et des méthodes locales. Ensuite nous avons mis le point et détaillé les deux algorithmes de planification de trajectoire pour un robot mobile, qui sont les algorithmes RRT et Astar pour rechercher un chemin dans un graphe entre un nœud initial de départ et un nœud final comme destination, tous deux données. Ces algorithmes seront mis en œuvre dans le chapitre suivant dans différents environnement.

Chapitre III

Résultats et Interprétations

III.1 Introduction

Le robot mobile a un rôle très important dans la vie moderne de l'humain car il peut réaliser les travaux pénible et répétitif par exemple le nettoyage quotidienne et la livraison commission à domicile. La planification de sa trajectoire est une partie essentielle pour le contrôler . Ceci est plus vrai que jamais dans le contexte automobile et tout particulièrement pour le véhicule autonome. Le rôle d'un planificateur de trajectoire consiste à augmenter l'autonomie du robot pour lui permettre de se mouvoir automatiquement d'une position initiale vers une position finale en évitant les collisions avec les obstacles qui l'entourent dans l'environnement où il opère. Notre objectif dans ce chapitre est d'implémenter quelques algorithmes de planification à savoir l'algorithme RRT, Astar et l'algorithme PRM dans différents types d'environnements et une fois la trajectoire est planifiée nous appliquons au robot un contrôleur de suivi de chemin tel que le PurePursuit qui lui permet de suivre ce chemin avec une grande précision.

III.2 Exemple illustratif

Dans un premier temps, nous avons simulé le cas qui est disponible dans Matlab où il n'y a pas d'obstacles dans l'environnement qui est représenté par une grille 30x30. Le robot doit se déplacer de la position (0,30) à la position (30,0). Un arbre tout d'abord est construit couvrant l'espace libre. Cet arbre, représenté en rouge sur la figure si-dessous, est constitué de nœuds générés aléatoirement de manière incrémentale comme nous l'avons expliqué au chapitre 2. Ensuite, se fait la détermination du chemin entre les deux nœuds Départ et Destination qui font partie eux aussi de l'arbre, et ce en utilisant un algorithme de recherche de plus court chemin tels que Astar ou Dijkstra. Ce chemin est montré en bleu sur la figure III.1.

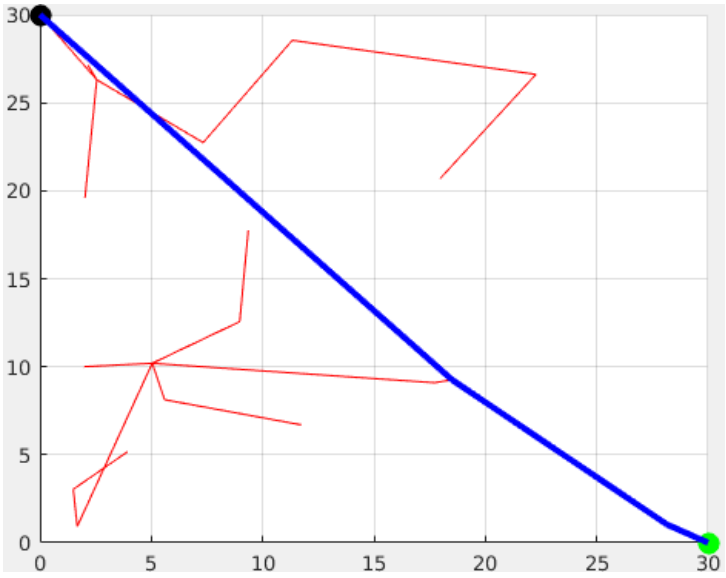
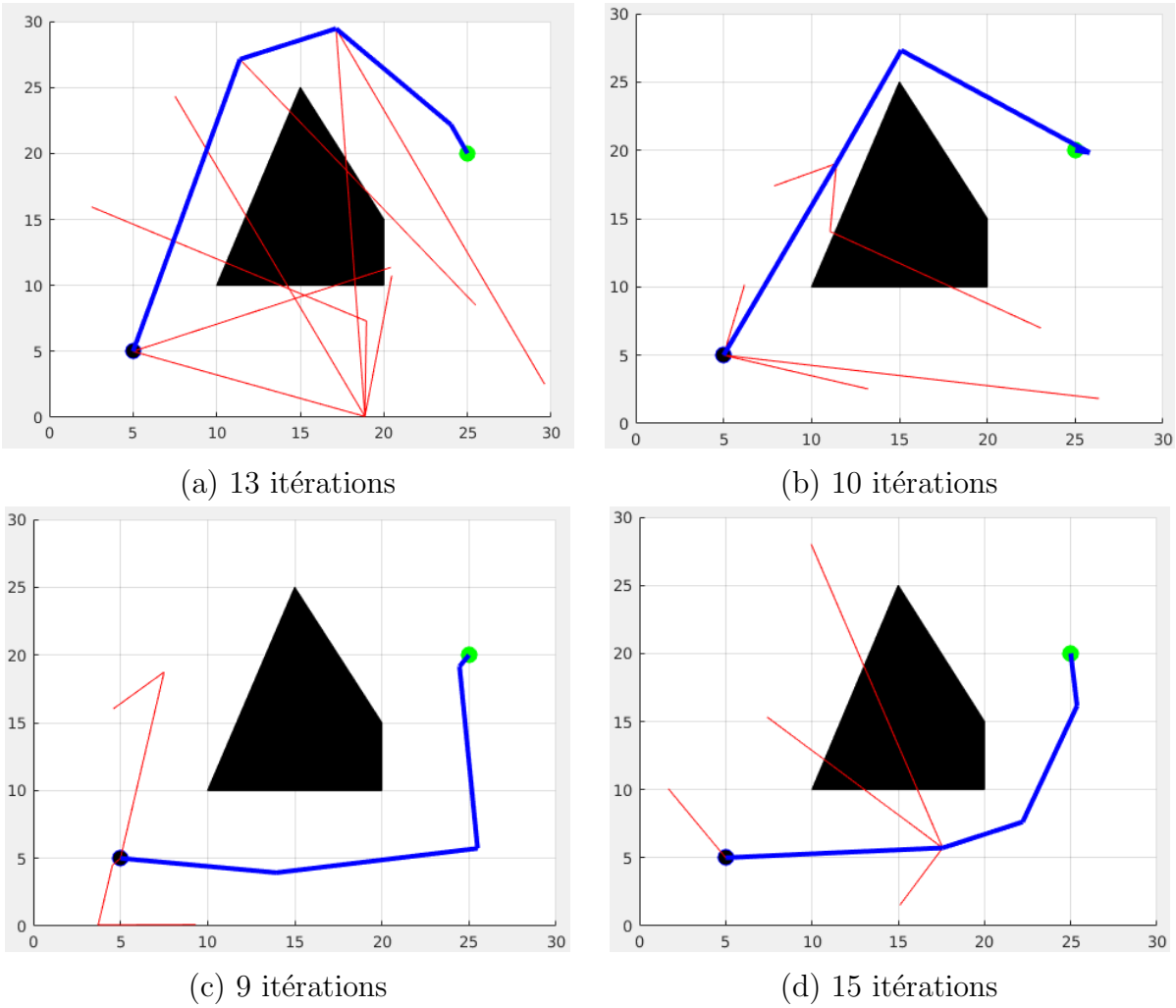


FIGURE III.1 – Résultat de planification de trajectoire par RRT dans un espace libre.

Dans une seconde expérience, on améliore l'algorithme précédent et on prend un exemple d'un environnement contenant un seul obstacle. Dans cet exemple, on désire aller du point de départ (5,5) à un point destination (25,20) en évitant l'obstacle comme c'est montré sur la figure ci-dessous (figure III.2).Il est à noter qu'à chaque fois qu'on refait l'expérience, on obtient un chemin différent vu l'aspect probabiliste de l'algorithme RRT.



(a) 13 itérations

(b) 10 itérations

(c) 9 itérations

(d) 15 itérations

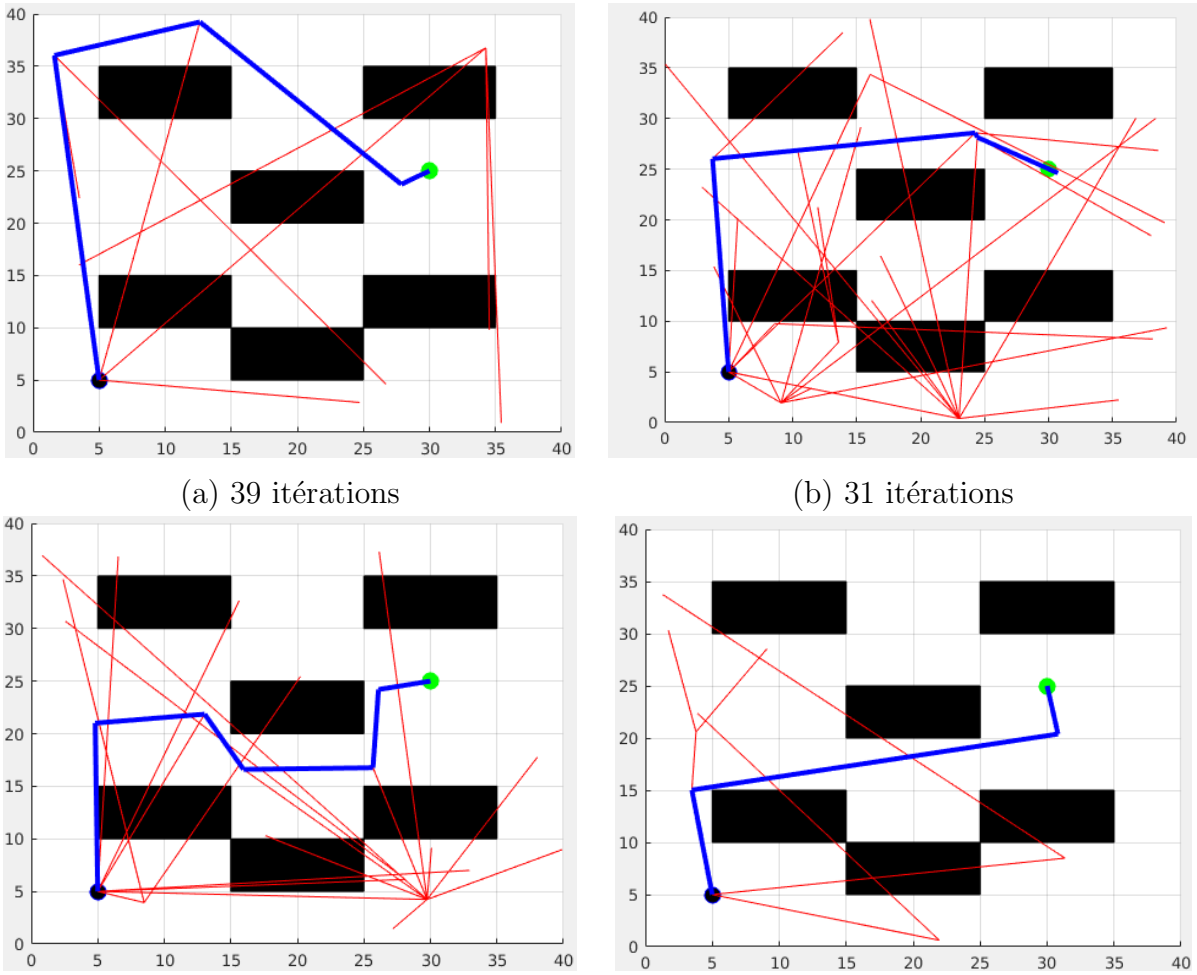
FIGURE III.2 – Environnement à un seul obstacle

III.3 Autres exemples d'application de l'algorithme RRT

Dans ce cas, on va considérer des environnements plus complexes avec un nombre d'obstacles élevé et de formes différentes.

Dans le premier cas, nous avons pris un environnement relativement simple contenant 6 obstacles de forme rectangle. Le robot doit démarrer de la position de départ (5, 5) pour aller à la posi-

tion Destination (30,25) en évitant les obstacles présents dans l'environnement. Les résultats est montré sur la figure III.3. Nous remarquons que la trajectoire ainsi planifiée permet au robot d'atteindre la destination sans collision.



(a) 39 itérations

(b) 31 itérations

(c) 23 itérations

(d) 49 itérations

FIGURE III.3 – Les cas d'un environnement à 6 obstacles de forme rectangle

Dans le deuxième cas, on considère un environnement plus complexe et encombrant avec 15 obstacles de formes compliquées et différentes. Le robot doit démarrer de la position de départ (5, 29) pour aller à la position Destination (18,3) en évitant les obstacles présents dans l'environnement. La figure III.4 illustre le résultat obtenu, nous remarquons que le chemin planifié évite toute collision avec les obstacles ce qui permet au robot d'atteindre la cible sain et malgré la complexité de l'environnement. La figure III.5 montre le cas où la destination est à la position (29, 20), l'algorithme RRT réussit à planifier le chemin vers cette position en évitant les obstacles malgré la longueur du chemin et sa complexité.

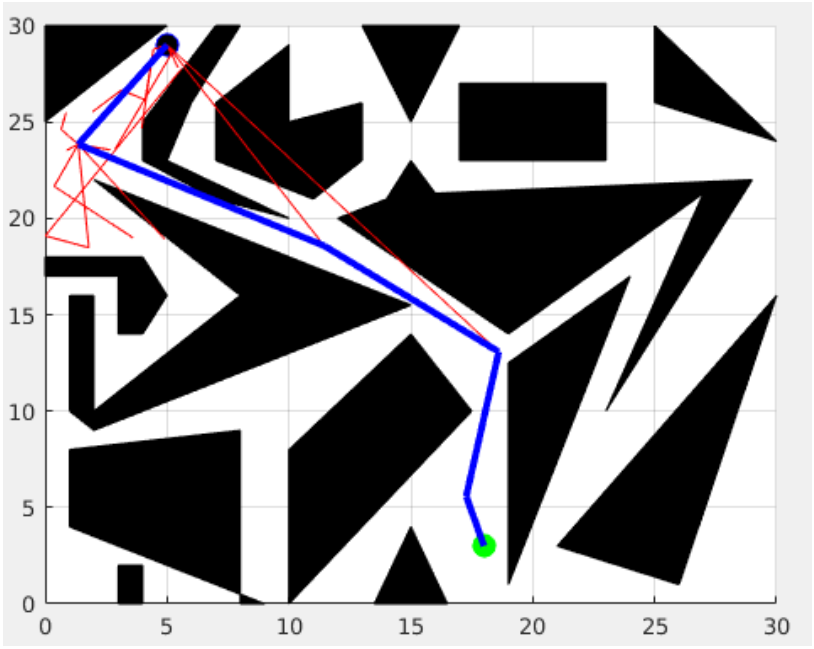


FIGURE III.4 – Cas d'un environnement à 15 obstacles de forme différente

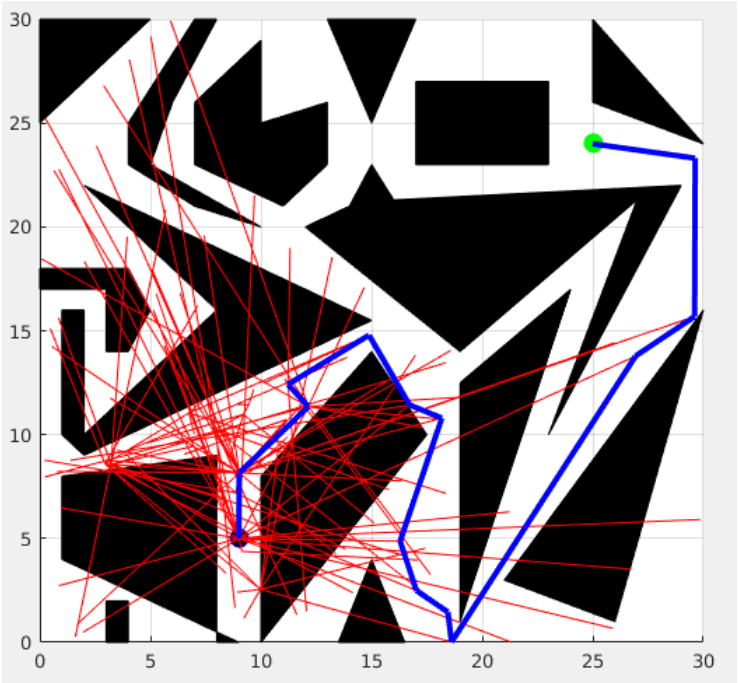


FIGURE III.5 – Résultat de planification de trajectoire par RRT après changement de la destination.

III.4 Résultats de Simulation par Astar

L'algorithme A* est un algorithme de recherche de chemin dans un graphe entre un nœud initial et un nœud final. Il utilise une évaluation heuristique sur chaque nœud pour estimer le meilleur

chemin y passant, et visite ensuite les nœuds par ordre de cette évaluation heuristique. C'est un algorithme simple, ne nécessitant pas de prétraitement.

Premièrement , nous avons simulé le cas où il y a un seul obstacle dans l'environnement. Le robot doit démarrer de la position de départ (5, 5) pour aller à la position Destination (25,20) en évitant l'obstacle. Le résultat est montré sur la figure III.6.

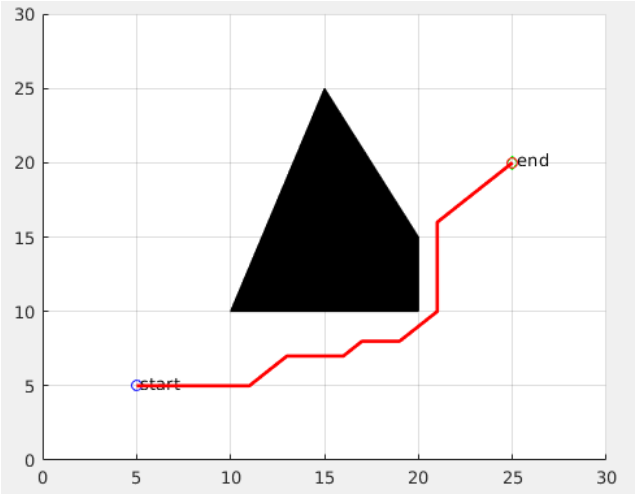


FIGURE III.6 – Environnement à un seul obstacle.

Deuxièmement , nous avons utilisé le même environnement complexe à 6 et 15 obstacles de la section précédente, avec les points de départ (5,5) ; (5,29) et les points de destination (30,25) ; (18,3) successivement. Le résultat est illustré sur les deux figures III.7 et III.8.

FIGURE III.7 – Résultat de planification de trajectoire par Astar dans un environnement à 6 obstacles.

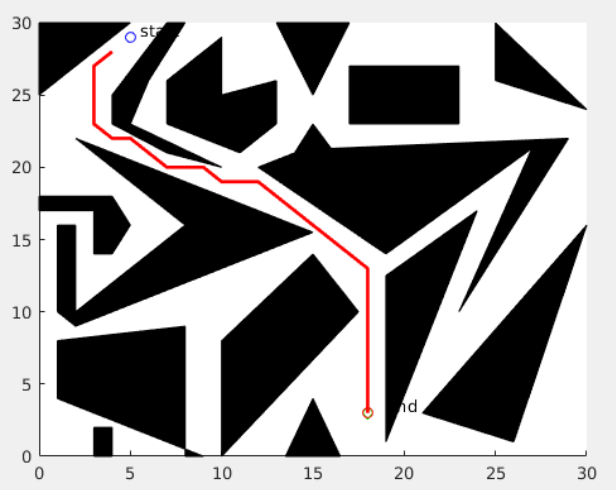


FIGURE III.8 – Résultat de planification de trajectoire par Astar dans un environnement à 15 obstacles.

III.5 Planification par PRM

Dans cette section, on va utiliser le planificateur de chemin Probabilistic Roadmap (PRM), disponible dans Matlab, pour calculer un chemin entre deux emplacements sur une carte donnée. Ce planificateur de chemin PRM construit une feuille de route (Roadmap) dans l'espace libre d'une carte donnée, en utilisant des nœuds générés au hasard dans l'espace libre et en les connectant les uns aux autres. Une fois la feuille de route construite, nous pouvons rechercher un chemin entre un emplacement de départ donné et un emplacement d'arrivée donné sur la carte. Plusieurs types d'environnement sont simulés dans Matlab, entre autres l'environnement simpleMap et l'environnement complexMap. La figure (a) III.9 montre la carte simpleMap. Pour assurer que le robot ne heurte aucun obstacle la carte doit être gonflée de la dimension du robot avant d'appliquer le planificateur de chemin PRM.Nous obtenons ainsi la carte gonflée de la figure (b).

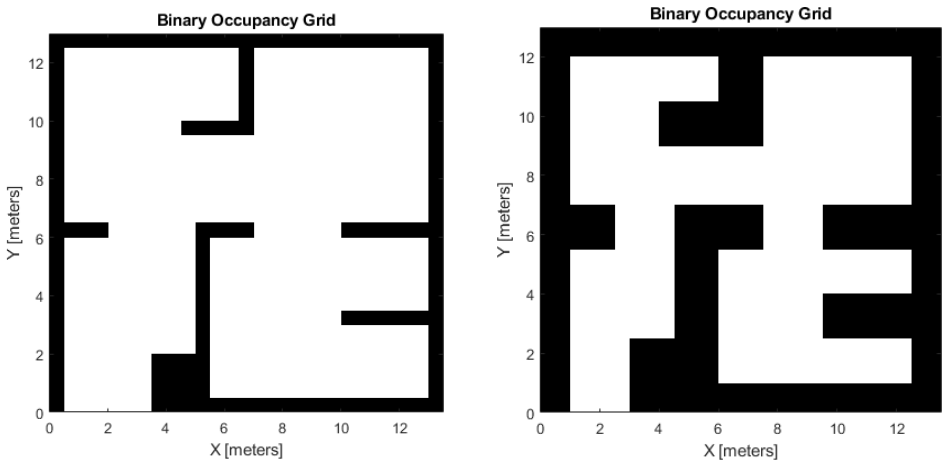


FIGURE III.9 – La carte simpleMap simple et gonflée

Dans le cas où nous considérons que le robot doit démarrer de la position (2,11) vers la destination située à la position (10,2), la trajectoire planifiée par PRM est illustrée sur la figure III.10. Les nœuds bleus représentent les nœuds échantillonnés aléatoirement dans l'espace libre de la carte constituant ainsi la feuille de route (RoadMap).

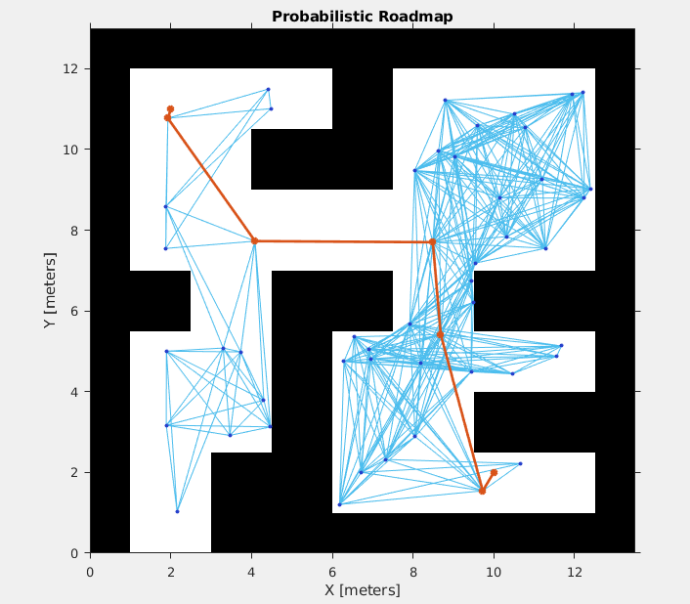


FIGURE III.10 – Trajectoire planifiée par PRM dans le cas de la carte simpleMap

Dans ce cas, l'environnement est complexMap. La figure III.11 montre le chemin calculé pour aller de la position initiale (5,37) à la position finale (45,5). On note que malgré la complexité de la carte, le chemin calculée permet au robot d'atteindre la destination sans collision.

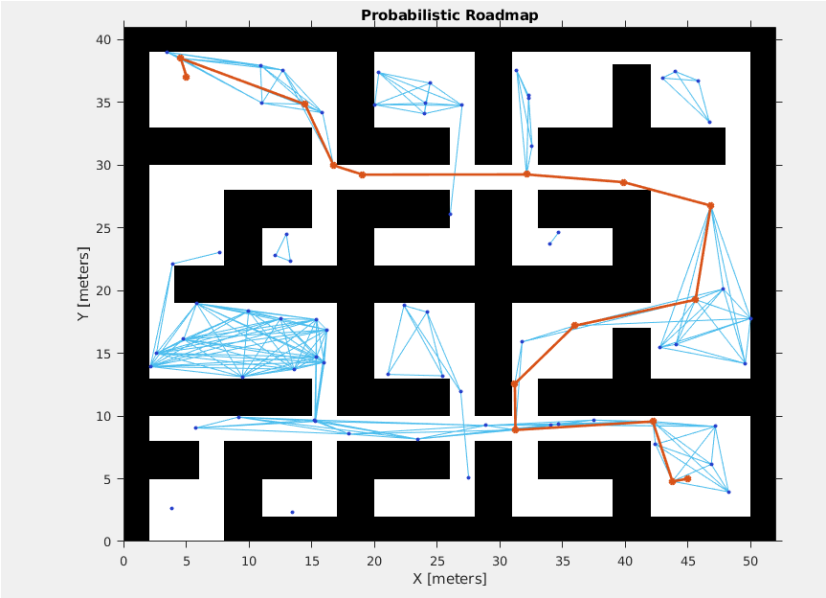


FIGURE III.11 – Trajectoire planifiée par PRM dans le cas de la carte complexMap

III.6 Planification dans un parking

On utilise le planificateur RRT pour permettre à un véhicule robotisé de démarrer d'une posture initiale [1.25 13.5 0] pour se garer dans la position [66.75 43.75 90] ; le résultat est donné à la figure III.12. Ici le parking est gonflé de la dimension du robot afin d'assurer qu'il ne heurte aucun obstacle ou autre véhicule.

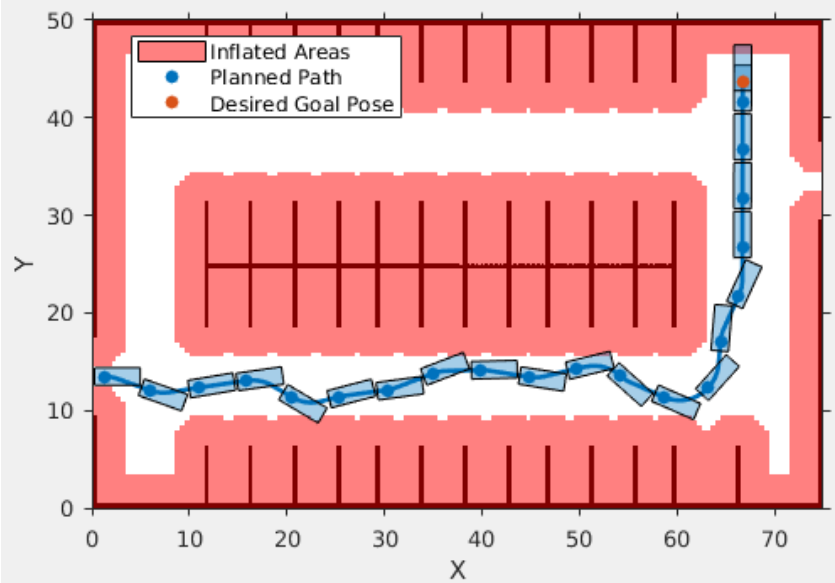


FIGURE III.12 – Trajectoire planifiée par RRT dans un parking

La figure III.13 illustre que le véhicule doit démarrer d'une posture initiale [1.25 13.5 0] pour se garer dans la position [70.5 6.75 -90].

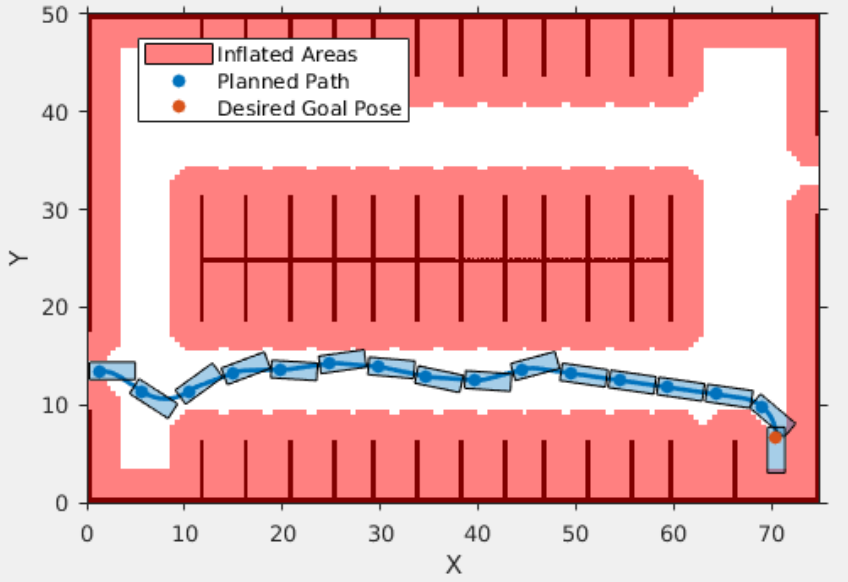


FIGURE III.13 – Trajectoire planifiée par RRT dans un parking après changement de la destination

III.7 Résultats de suivi de trajectoire

Une fois la trajectoire planifiée, le robot doit la suivre avec le maximum de précision afin d'atteindre la destination désirée. On doit utiliser pour cela un contrôleur tel que le contrôleur Pure-Pursuit disponible sur Matlab. Ce dernier calcule à chaque instant les signaux de commande qui vont être utilisés pour entraîner le robot pour suivre la trajectoire souhaitée.

Premièrement , nous avons appliqué le contrôleur PurePursuit dans le cas la planification de la trajectoire par l'algorithme Astar dans l'environnement à 15 obstacles. Nous remarquons que la trajectoire réelle du robot(en bleu) suit parfaitement bien la trajectoire prédéterminée(en rouge) par l'algorithme Astar.

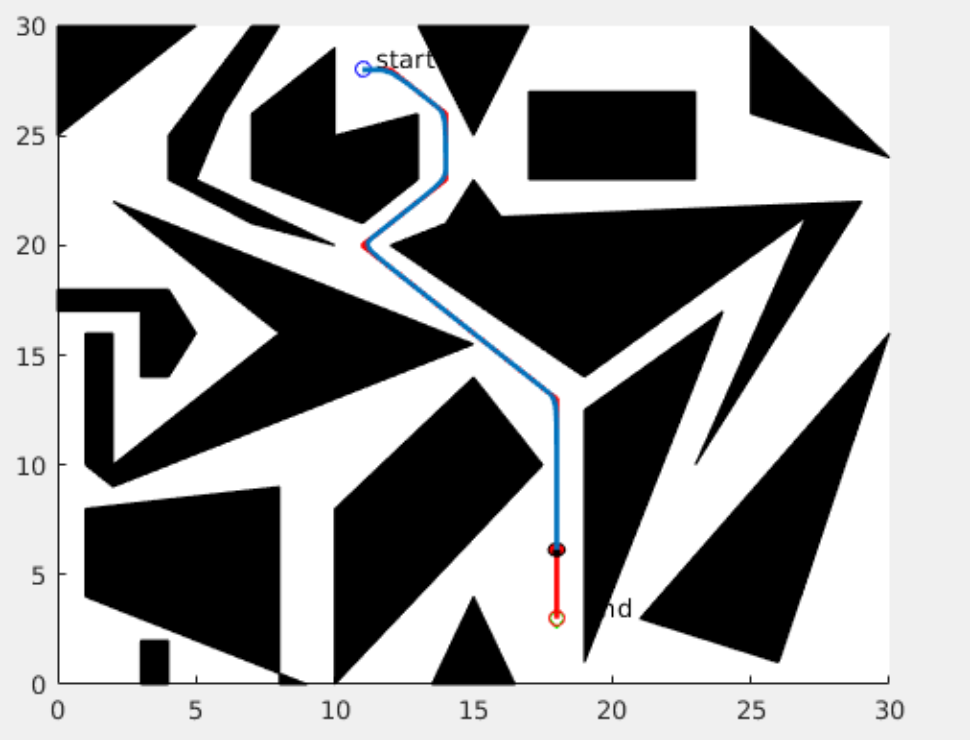


FIGURE III.14 – Suivi de trajectoire par l'algorithme Astar.

Deuxièmement , nous avons appliqué le planificateur PRM pour calculer la trajectoire souhaitée sans collision. On a utilisé l'environnement simpleMap et on cherche à ce que notre robot se déplace de la position initiale (0,30) à la position désirée (30,0). La distance look-ahead est fixée à 0.6 m. La figure III.15 montre les résultats obtenus. Nous remarquons que la trajectoire réelle du robot suit parfaitement bien la trajectoire planifiée par PRM.

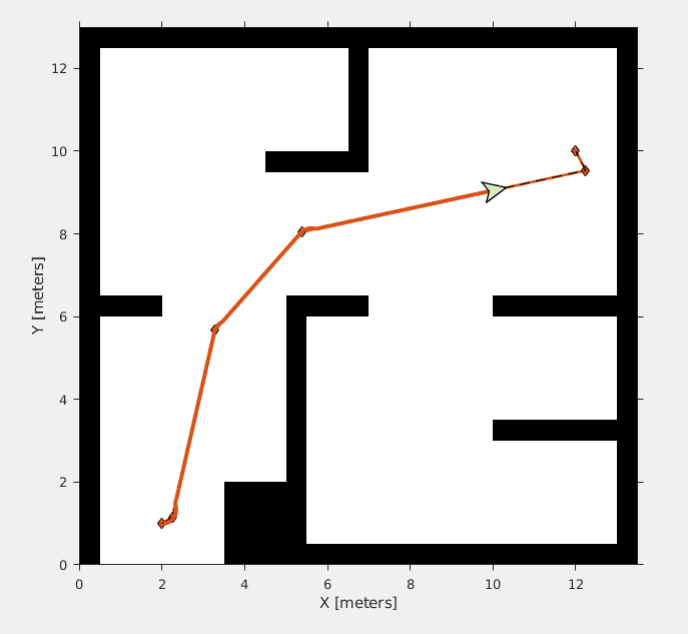


FIGURE III.15 – Suivi de trajectoire par l'algorithme PRM.

Ensuite, pour étudier l'influence de la valeur de la distance look-ahead du contrôleur PurePursuit sur les résultats de commande. La figure III.16 montre la trajectoire réelle du robot quand la valeur de la distance look-ahead est élevée (2 mètres). On remarque que l'erreur de suivi augmente surtout au niveau des sommets de la trajectoire désirée (waypoints). Par contre dans le cas où est de valeur faible (0.04 mètres), la trajectoire réelle présente des oscillations autour de la trajectoire désirée (voir la figure III.17).

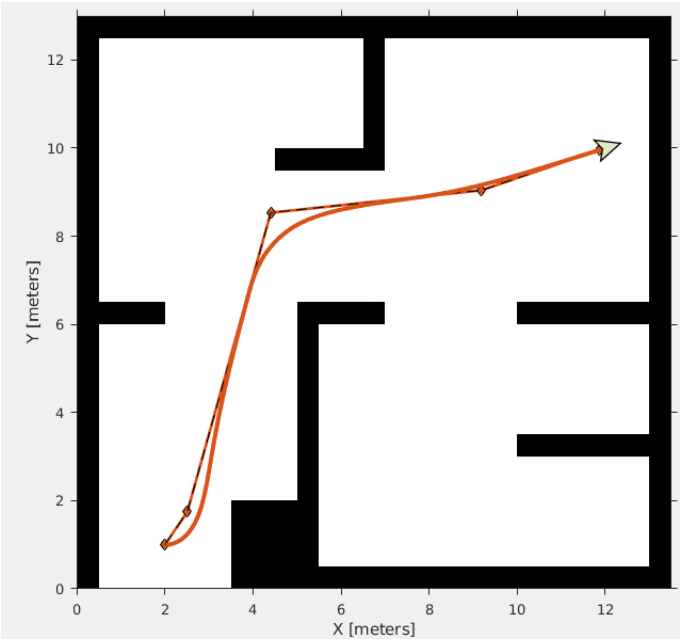


FIGURE III.16 – Résultat pour une distance look-ahead élevée (2 m)

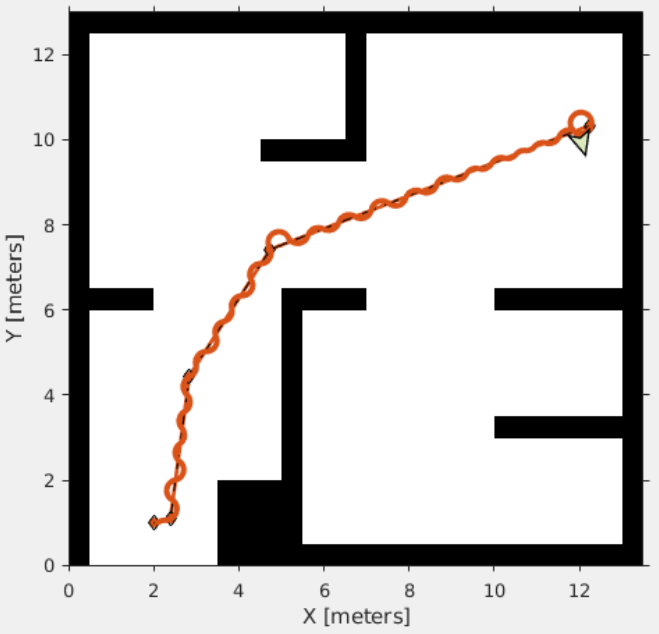


FIGURE III.17 – Résultat pour une distance look-ahead faible (0.04 m)

III.8 Conclusion

Dans ce chapitre, nous avons tout d'abord commencé par donner un exemple illustratif simple de la méthode de planification RRT. Ensuite, nous avons appliqué cette dernière à des environnements plus complexes pour prouver son efficacité. Les résultats de simulations obtenus étaient très concluants. Les deux méthodes Astar et RPM ont été également appliquées à des différents environnements. A la fin, le contrôleur de suivi de chemin Pure Pursuit a été utilisé pour commander le robot on lui permettant ainsi de suivre le chemin désiré. Ses performances ont été évaluées à travers des simulations. Nous avons constaté que la trajectoire réelle suit parfaitement bien la trajectoire désirée.

Conclusion générale

Les travaux présentés dans ce mémoire portent sur la planification de trajectoire sans collision et la commande d'un robot mobile pour lui permettre de gérer ses mouvements en évitement des obstacles prémédités.

Nous avons commencé par des généralités sur la robotique mobile et les techniques de modélisation utilisées, tout en citant les différents types de robots mobiles à roues et ainsi abordé l'étude de la géométrie, de la cinématique et de la dynamique du robot mobile unicycle.

La suite de travail a été consacrée aux différentes méthodes de planification de trajectoire existantes dans la littérature. Nous avons présenté, entre autres, le principe de la méthode heuristique Astar, ensuite nous avons présenté les deux principales méthodes probabilistes, à savoir, les algorithmes PRM et RRT. Nous avons par la suite, mis le point sur les deux algorithmes RRT et Astar permettant de rechercher un chemin entre un nœud initial de départ et un nœud final.

Ces méthodes de planification ont été testées sur différents types d'environnements allant du plus simple au plus compliqué comportant de nombreux obstacles de forme différente, des parkings et des bureaux. Dans ces différents tests, ces méthodes ont prouvé leur efficacité dans le calcul de chemin court et sans collision.

La méthode Pure Pursuit pour le suivi de chemin a été étudiée et appliquée au robot. Ce contrôleur permet de calculer les signaux de commande nécessaires pour le suivi de la trajectoire planifiée avec le maximum de précision. Les résultats de simulation montrent les bonnes performances de ce contrôleur.

Comme perspective, et en se basant sur les méthodes étudiées et implémentées dans ce mémoire, nous proposons d'apporter des améliorations permettant de planifier des trajectoires en temps réel dans des environnements inconnus et/ou comportant des obstacles dynamiques.

Bibliographie

[1] CHERROUN Lakhmissi, Navigation Autonome d'un Robot Mobile par des Techniques Neuro-Floues, Thése de Doctorat Université Mohamed Khider – Biskra, 2014.

[2] ROBOT MOBILE, [en ligne] Disponible sur : https://www.hisour.com/fr/mobile-robot/.

[3] ROBOT, [en ligne] Disponible sur : https://fr.wikipedia.org/wiki/Robot.

[4] BOUFERA Fatma, Contribution Des Outils de l'Intelligence Artificielle dans la Robotique Mobile, Thèse Doctorat Université d'Oran, 2014.

[5] TAKHI Hocine , Conception et réalisation d'un robot mobile à base d'arduino, Mémoire de Master Université Amar Telidji, 2014.

[6] CHEBILI zakaria, Contribution à la Modélisation dynamique et la Commande des robots mobiles non-holonomes à roues, Mémoire de Master Université Oum El Bouaghi, 2018.

[7] BALI Chaher, Réalisation d'un robot mobile avec évitement d'obstacle et trajectoire programmée, Mémoire de Master Université Mohamed Khider Biskra, 2012.

[8] BAYLE Bernard, Cours de Robotique mobile, Télécom Physique Strasbourg 3A, option ISAV, master IRIV.

[9] BOUHALASSA LOUBNA, Planification de trajectoire d'un robot basée sur les

réseaux de neurones et les algorithmes génétiques, Mémoire de magister Université Mohamed BOUDIAF D'ORAN, 2010.

[10] TONKIN Massinissa, OUHARCHAOU Hamidouche, Planification de trajectoire pour un robot mobile Dr. Robot I90, Mémoire de Master Université de TIZI-OUZOU, 2017.

[11] MORBIDI Fabio, cours de Localisation et navigation de robot, Université de Picardie, département EEA, 2017.

[12] Nilsson, N.J, Principles of Artificial Intelligence, Tioga Publishing Company.

[13] BLIN Nassim, Planification interactive de mouvement avec contact, Thèse de Doctorat Unversité de Toulouse, 2017.

[14] Kenzai Abderrahmane, Planification de trajectoires de robots mobiles via des méthodes ensemblistes, Mémoire DEA, Université d'Angers, 2005.

[15] FILLIAT David, Cour de Robotique Mobile, École Nationale Supérieure de Techniques Avancées ParisTech, 2011.

[16] David Flavigne, Planification de mouvement interactive: coopération humain-machine pour la recherche de trajectoire et l'animation, Thèse de Doctorat Université Toulouse, 2010.

[17] Pathfinding: Algorithmes en A*, [en ligne] Disponible sur : https://www.createursdemondes.fr/creation-de-jeux-video/pathfinding-algorithmes-en-a/

[18] Pure Persuit Controller, [en ligne] Disponible sur : https://www.mathworks.com/help/robotics/ug/pure-pursuit-controller.html

Annexe

Exemple illustratif

Afin d'expliquer le déroulement de l'algorithme Astar pour la planification de trajectoire, on prend un exemple simple d'un environnement représenté par une petite grille 7x7 contenant un seul obstacle. Dans cet exemple, on désire aller du point de départ (étoile bleu) à un point destination (cercle vert) en évitant l'obstacle comme c'est montré sur la figure ci-dessous (figure III.1).
La première chose à noter est que la zone de recherche est divisée en une grille carrée (7X7 ici). Simplifier la zone de recherche, comme nous l'avons fait, est la première étape de la recherche de chemin. Les sommets des carrés de la grille sont appelés "nœuds". L'algorithme Astar se base sur deux listes : la liste ouverte et la liste fermée initialement vides.
Dans ce qui suit on va expliquer, étape par étape, le fonctionnement de cet algorithme.

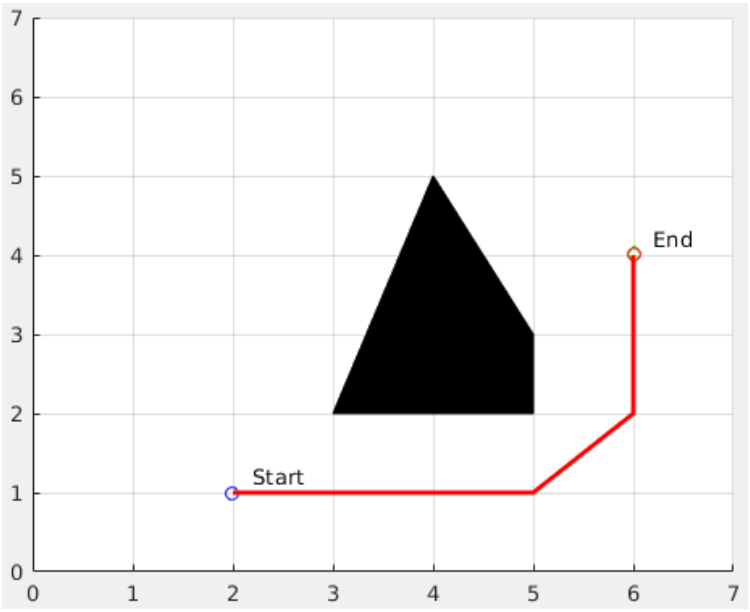


FIGURE 1 – Environnement à un seul obstacle et une grille 7x7

Intialement ,Le point de départ est considéré comme étant le nœud courant car il constitue le début de la trajectoire à planifier. On met les nœuds faisant partie des obstacles ainsi que le nœud de départ dans la liste fermée pour ne pas les traiter par la suite (voir tableau III.1)

Les coordonnées des nœuds faisant partie de la liste fermée (CLOSED liste)							
3	4	4	4	4	5	5	2
2	2	3	4	5	2	3	1

TABLE 1 – Contenu de la liste fermée à l'état initial

a) Itération1 :

Lors de la première itération, le nœud courant est le nœud de départ :

$$\begin{pmatrix} xCourant \\ yCourant \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

On démarre de ce nœud et on passe à l'un de ces voisins. Les nœuds voisins accessibles au nœud courant leurs coûts G, H et F sont calculés et donnés dans le tableau III.2. Ces nœuds sont mis dans la liste ouverte comme c'est montré dans le tableau III.3 avec comme parent le nœud courant. On cherche le meilleur nœud de toute la liste ouverte ayant la fonction coût minimale, ainsi les

Les coordonnées des nœuds voisins		le cout G	Le cout H	le cout F
3	1	1	4.2426	5.2426
2	2	1	4.4721	5.4721
1	2	1.4142	5.3852	6.7994
1	1	1	5.8310	6.8310

TABLE 2 – Les nœuds voisins du nœud courant et leurs fonctions couts à l'itération 1.

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
1	3	1	2	1	1	4.2426	5.2426
1	2	2	2	1	1	4.4721	5.4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310

TABLE 3 – Contenu de la liste ouverte à l'itération 1

coordonnées du nœud choisi comme candidat en ce moment sont :

$$\begin{pmatrix} xCourant \\ yCourant \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

Ce nœud sera considéré comme nœud courant dans l'itération suivante. On retire le nœud choisi de la liste ouverte en mettant 0 dans la case qui lui correspond au niveau de la première colonne (0/1) de la liste ouverte et on le met dans la liste fermée. Ainsi, le nouveau contenu de la liste ouverte est illustré par le tableau 4 et celui de la liste fermée est donné dans le tableau 5 (on y a ajouté les coordonnées du nœud choisi).

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
1	2	2	2	1	1	4.4721	5.4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310

TABLE 4 – Le nouveau contenu de la liste ouverte à l'itération 1

Les coordonnées des nœuds faisant partie de la liste fermée (CLOSED liste)								
3	4	4	4	4	5	5	2	3
2	2	3	4	5	2	3	1	1

TABLE 5 – Le nouveau contenu de la liste fermée à l'itération 1

b) Itération 2 :

Dans cette itération, le nœud courant est celui déterminé à l'itération précédente, soit :

$$\begin{pmatrix} xCourant \\ yCourant \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

Les coordonnées des nœuds voisins		le cout G	Le cout H	le cout F
4	1	2	3.6056	5.6056
2	2	2.4142	4.4721	6.8863

TABLE 6 – Les nœuds voisins du nœud courant et leurs fonctions couts à l'itération 2.

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
1	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
1	4	1	3	1	2	3.6056	5.6056

TABLE 7 – Contenu de la liste ouverte à l'itération 2

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
1	4	1	3	1	2	3.6056	5.6056

TABLE 8 – Le nouveau contenu de la liste ouverte à l'itération 2

Les coordonnées des nœuds faisant partie de la liste fermée (CLOSED liste)									
3	4	4	4	4	5	5	2	3	2
2	2	3	4	5	2	3	1	1	2

TABLE 9 – Le nouveau contenu de la liste fermée à l'itération 2

c) Les itérations 3 à 10 :

Les coordonnées des nœuds voisins		le cout G	Le cout H	le cout F
3	3	2.4142	3.1623	5.5765
2	3	2	4.1231	6.1231
1	3	2.4142	5.0990	7.5132
1	2	2	5.3852	7.3852
1	1	2.4142	5.8310	8.2452

TABLE 10 – Les nœuds voisins du nœud courant et leurs fonctions couts à l'itération 3.

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
1	4	1	3	1	2	3.6056	5.6056
1	3	3	2	2	2.4142	3.1623	5.5765
1	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132

TABLE 11 – Contenu de la liste ouverte à l'itération 3

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
1	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
1	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132

TABLE 12 – Le nouveau contenu de la liste ouverte à l'itération 3

Les coordonnées des nœuds voisins		le cout G	Le cout H	le cout F
3	4	3.4142	3	6.4142
2	4	3.8284	4	7.8284
2	3	3.4142	4.1231	7.5373

TABLE 13 – Les nœuds voisins du nœud courant et leurs fonctions couts à l'itération 4.

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
1	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
1	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
1	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284

TABLE 14 – Contenu de la liste ouverte à l'itération 4

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
1	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
1	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284

TABLE 15 – Le nouveau contenu de la liste ouverte à l'itération 4

Les coordonnées des nœuds voisins		le cout G	Le cout H	le cout F
5	1	3	3.1623	6.1623

TABLE 16 – Les nœuds voisins du nœud courant et leurs fonctions couts à l'itération 5.

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
1	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
1	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284
1	5	1	4	1	3	3.1623	6.1623

TABLE 17 – Contenu de la liste ouverte à l'itération 5

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
0	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
1	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284
1	5	1	4	1	3	3.1623	6.1623

TABLE 18 – Le nouveau contenu de la liste ouverte à l'itération 5

Les coordonnées des nœuds voisins		le cout G	Le cout H	le cout F
3	4	3.4142	3	6.4142
2	4	3	4	7
1	4	3.4142	5	8.4142
1	3	3	5.0990	8.0990
1	2	3.4142	5.3852	8.7994

TABLE 19 – Les nœuds voisins du nœud courant et leurs fonctions couts à l'itération 6.

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
0	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
1	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284
1	5	1	4	1	3	3.1623	6.1623
1	1	4	2	3	3.4142	5	8.4142

TABLE 20 – Contenu de la liste ouverte à l'itération 6

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
0	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
1	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284
0	5	1	4	1	3	3.1623	6.1623
1	1	4	2	3	3.4142	5	8.4142

TABLE 21 – Le nouveau contenu de la liste ouverte à l'itération 6

Les coordonnées des nœuds voisins		le cout G	Le cout H	le cout F
6	2	4.4142	2	6.4142
6	1	4	3	7.0000

TABLE 22 – Les nœuds voisins du nœud courant et leurs fonctions couts à l'itération 7.

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
0	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
1	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284
0	5	1	4	1	3	3.1623	6.1623
1	1	4	2	3	3.4142	5	8.4142
1	6	2	5	1	4.4142	2	6.4142
1	6	1	5	1	4	3	7

TABLE 23 – Contenu de la liste ouverte à l'itération 7

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
0	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
0	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284
0	5	1	4	1	3	3.1623	6.1623
1	1	4	2	3	3.4142	5	8.4142
1	6	2	5	1	4.4142	2	6.4142
1	6	1	5	1	4	3	7

TABLE 24 – Le nouveau contenu de la liste ouverte à l'itération 7

Les coordonnées des nœuds voisins		le cout G	Le cout H	le cout F
3	5	4.4142	3.1623	7.5765
2	5	4.8284	4.1231	8.9515
2	4	4.4142	4	8.4142

TABLE 25 – Les nœuds voisins du nœud courant et leurs fonctions couts à l'itération 8.

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
0	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
0	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284
0	5	1	4	1	3	3.1623	6.1623
1	1	4	2	3	3.4142	5	8.4142
1	6	2	5	1	4.4142	2	6.4142
1	6	1	5	1	4	3	7
1	3	5	3	4	4.4142	3.1623	7.5765
1	2	5	3	4	4.8284	4.1231	8.9515

TABLE 26 – Contenu de la liste ouverte à l'itération 8

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
0	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
0	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284
0	5	1	4	1	3	3.1623	6.1623
1	1	4	2	3	3.4142	5	8.4142
0	6	2	5	1	4.4142	2	6.4142
1	6	1	5	1	4	3	7
1	3	5	3	4	4.4142	3.1623	7.5765
1	2	5	3	4	4.8284	4.1231	8.9515

TABLE 27 – Le nouveau contenu de la liste ouverte à l'itération 8

Les coordonnées des nœuds voisins		le cout G	Le cout H	le cout F
7	3	5.8284	1.4142	7.2426
7	2	5.4142	2.2361	7.6503
7	1	5.8284	3.1623	8.9907
6	3	5.4142	1	6.4142
6	1	5.4142	3	8.4142

TABLE 28 – Les nœuds voisins du nœud courant et leurs fonctions couts à l'itération 9.

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
0	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
0	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284
0	5	1	4	1	3	3.1623	6.1623
1	1	4	2	3	3.4142	5	8.4142
0	6	2	5	1	4.4142	2	6.4142
1	6	1	5	1	4	3	7
1	3	5	3	4	4.4142	3.1623	7.5765
1	2	5	3	4	4.8284	4.1231	8.9515
1	7	3	6	2	5.8284	1.4142	7.2426
1	7	2	6	2	5.4142	2.2361	7.6503
1	7	1	6	2	5.8284	3.1623	8.9907
1	6	3	6	2	5.4142	1	6.4142

TABLE 29 – Contenu de la liste ouverte à l'itération 9

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
0	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
0	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284
0	5	1	4	1	3	3.1623	6.1623
1	1	4	2	3	3.4142	5	8.4142
0	6	2	5	1	4.4142	2	6.4142
1	6	1	5	1	4	3	7
1	3	5	3	4	4.4142	3.1623	7.5765
1	2	5	3	4	4.8284	4.1231	8.9515
1	7	3	6	2	5.8284	1.4142	7.2426
1	7	2	6	2	5.4142	2.2361	7.6503
1	7	1	6	2	5.8284	3.1623	8.9907
0	6	3	6	2	5.4142	1	6.4142

TABLE 30 – Le nouveau contenu de la liste ouverte à l'itération 9

Les coordonnées des nœuds voisins		le cout G	Le cout H	le cout F
7	4	6.8284	1	7.8284
7	3	6.4142	1.4142	7.8284
7	2	6.8284	2.2361	9.0645
6	4	6.4142	0	6.4142
5	4	6.8284	1	7.8284

TABLE 31 – Les nœuds voisins du nœud courant et leurs fonctions couts à l'itération 10.

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
0	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
0	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284
0	5	1	4	1	3	3.1623	6.1623
1	1	4	2	3	3.4142	5	8.4142
0	6	2	5	1	4.4142	2	6.4142
1	6	1	5	1	4	3	7
1	3	5	3	4	4.4142	3.1623	7.5765
1	2	5	3	4	4.8284	4.1231	8.9515
1	7	3	6	2	5.8284	1.4142	7.2426
1	7	2	6	2	5.4142	2.2361	7.6503
1	7	1	6	2	5.8284	3.1623	8.9907
0	6	3	6	2	5.4142	1	6.4142
1	7	4	6	3	6.8284	1	7.8284
1	6	4	6	3	6.4142	0	6.4142
1	5	4	6	3	6.8284	1	7.8284

TABLE 32 – Contenu de la liste ouverte à l'itération 10

(0/1)	X val	Y val	Parent X val	Parent Y val	G	H	F
0	3	1	2	1	1	4.2426	5.2426
0	2	2	2	1	1	4.4721	4721
1	1	2	2	1	1.4142	5.3852	6.7994
1	1	1	2	1	1	5.8310	6.8310
0	4	1	3	1	2	3.6056	5.6056
0	3	3	2	2	2.4142	3.1623	5.5765
0	2	3	2	2	2	4.1231	6.1231
1	1	3	2	2	2.4142	5.0990	7.5132
0	3	4	3	3	3.4142	3	6.4142
1	2	4	3	3	3.8284	4	7.8284
0	5	1	4	1	3	3.1623	6.1623
1	1	4	2	3	3.4142	5	8.4142
0	6	2	5	1	4.4142	2	6.4142
1	6	1	5	1	4	3	7
1	3	5	3	4	4.4142	3.1623	7.5765
1	2	5	3	4	4.8284	4.1231	8.9515
1	7	3	6	2	5.8284	1.4142	7.2426
1	7	2	6	2	5.4142	2.2361	7.6503
1	7	1	6	2	5.8284	3.1623	8.9907
0	6	3	6	2	5.4142	1	6.4142
1	7	4	6	3	6.8284	1	7.8284
0	6	4	6	3	6.4142	0	6.4142
1	5	4	6	3	6.8284	1	7.8284

TABLE 33 – Le nouveau contenu de la liste ouverte à l'itération 10