REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider – BISKRA

Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie

N°d'ordre : IVA 21/IVA/M2/2023

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Image et Vie Artificielle (IVA)**

Navigation d'un robot en utilisant le LiDAR 2D

Par : AZZOUZ Meriem

Soutenu le 19/06/2023devant le jury composé de :

Akrour Djouher	MCB	Président
Cherif Foudil	Professeur	Rapporteur
Dr. Babahenini Djihane	MAA	Examinateur

Année universitaire 2022-2023

Dédicaces

Je dédie ce travail

A mes parents qui m'a soutenu et encouragé ces années d'études.

Qu'ils trouvent ici le témoignage de ma profonde reconnaissance.

A mes sœurs, ma tante et ceux qui ont partagé avec moi tous les moments D'émotion lors de la réalisation de ce travail. Ils m'ont chaleureusement supporté et encouragé tout au long de mon parcours.

A ma famille qui m'a donné de l'amour et de la vivacité.

A tous mes amis qui m'ont toujours encouragé, et à qui je souhaite plus de succès.

AZZOUZ MERIEM

Remerciements

Je rende mon profondes gratitudes à Dieu tout puissant qui j'ai aidés à réaliser ce modeste travail.

Je exprime mon profondes gratitudes à mon parents pour leurs encouragements, leurs soutiens et pour les sacrifices qu'ils ont enduré.

Je remercie, mon encadreur Monsieur Dr. Cherif Foudil, pour ses précieux conseils et surtout pour sa disponibilité et sa grande aide, qui m'a a suivi et oriente et que a leur attention,

Leur disponibilité et leur enthousiasme pour atteindre le but de mon travail.

Je remercie aussi Mm Ouamane Fatima Zahra pour l'assistance et Appui à la réalisation de mon projet.

Résumé

La prolifération croissante des robots dans divers domaines humains tels que les services et la sécurité s'accélère, et cette tendance devrait se poursuivre à l'avenir. Les détectives cherchent à surmonter de nombreux défis techniques pour atteindre l'autonomie de navigation. La navigation joue un rôle central pour les créatures intelligentes opérant dans des environnements non liés ou dynamiques, qui ont été une source d'intérêt dans notre travail.

Dans ce mémoire, notre objectif principal est de résoudre le défi de la navigation autonome en appliquant notre modèle de navigation à un robot mobile. Nous attachons une grande importance à sa capacité à détecter son environnement extérieur et à assurer la sécurité de la navigation. Pour cela, nous utilisons le simulateur ROS pour développer des robots, fournissant un logiciel de simulation tridimensionnelle (Gazebo) que nous utiliserons pour tester le robot et son environnement, ainsi que (Rviz), où nous avons intégré un capteur (LiDAR bidimensionnel) pour recueillir des informations sur l'environnement externe. Notre robot sera en mesure de reconnaître les phares dans son environnement, de se déplacer et d'éviter les obstacles indépendamment et de se déplacer dans son environnement de son emplacement à son point cible.

ملخص

يتسارع الانتشار المتزايد للروبوتات في مختلف مجالات الانسان كالخدمات والأمن ، ومن المتوقع أن يستمر هذا الاتجاه في المستقبل . حيث تسعى المباحث للتغلب على العديد من التحديات التقنية لتحقيق استقلالية الملاحة.

تلعب الملاحة و أساليب التنقل في المحيط الخارجي دورا محوريا للمخلوقات الذكية و التي تعمل في بيئات غير منتظمة أو ديناميكية ، هذا الذي شكل شاغلا أساسيا في مشروعنا هذا.

في هذا المشروع، هدفنا الرئيسي هو حل تحدي الملاحة المستقلة من خلال تطبيق نموذج الملاحة الخاص بنا على روبوت متنقل. ونعلق أهمية كبيرة على قدرتها على اكتشاف بيئتها الخارجية وضمان سلامة الملاحة. لهذا، نستخدم نظام التشغيل الآلي(ROS)الخاص بتطوير الروبوتات حيث يوفر برنامج للمحاكاة ثلاثية الأبعاد (Gazebo) الذي سنستخدمه لمحاكاة الروبوت وبيئته ، إضافة إلى أداة الرؤية ثنائية الأبعاد (Rviz) ، حيث قد قمنا بدمج مستشعر(LiDAR)ثنائي الأبعاد لجمع المعلومات حول البيئة الخارجية. سيكون ربوتنا قادرًا على التعرف على المنارات في بيئته، والتنقل وتجنب العقبات بشكل مستقل و الانتقال في بيئته من موقعه الى الموقع الهدف .

Abstract

The increasing proliferation of robots in various human domains such as services and security is accelerating, and this trend is expected to continue in the future. Detectives seek to overcome many technical challenges to achieve navigation autonomy. Navigation and ocean navigation methods play a central role for intelligent creatures operating in unrelated or dynamic environments, which has been a key concern in our project.

In this project, our main goal is to solve the challenge of autonomous navigation by applying our navigation model to a mobile robot. We attach great importance to its ability to detect its external environment and to ensure the safety of navigation. For this, we use ROS to develop robots, providing a three-dimensional simulation software (Gazebo) that we will use to test the robot and its environment, as well as (Rviz), where we have integrated a sensor (two-dimensional LiDAR) to gather information about the external environment. Our robot will be able to recognize headlights in its environment move and avoid obstacles independently and move in its environment from its location to its target point.

Sommaire

Table des figures

Introduction générale

Un robot mobile peut être décrit comme un système mécanique capable de naviguer de manière autonome dans son environnement. Pour ce faire, le robot a besoin des composants suivants:

Capteurs : Ceux-ci permettent au robot de percevoir et de recueillir des informations sur son environnement, fournissant un niveau de compréhension ou de familiarité.

-Actionneurs : Ces mécanismes permettent au robot de se déplacer physiquement et d'interagir avec son environnement.

-Intelligence : Le robot a besoin d'algorithmes ou de systèmes de contrôle qui peuvent traiter les données du capteur et générer des commandes appropriées pour que les actionneurs puissent accomplir des tâches spécifiques.

-Connectivité : Le robot doit être connecté à un environnement qui correspond au monde dans lequel il opère et aux tâches qu'il doit accomplir.

Les robots mobiles progressent continuellement depuis les années 2000 et trouvent une application étendue dans les domaines militaire (comme les drones volants), médical et agricole. Ils excellent dans l'exécution de tâches jugées difficiles ou dangereuses pour les humains.

Le LiDAR 2D est un capteur laser qui émet des faisceaux à 180 degrés (ou plus) pour mesurer les distances et détecter les obstacles autour du robot. Dans un environnement statique, où les obstacles restent relativement constants, le LiDAR 2D est utilisé pour créer une carte de l'environnement. Cette carte est ensuite utilisée par l'algorithme de navigation pour planifier des trajectoires sûres et efficaces, en évitant les obstacles détectés par le LiDAR 2D.

La navigation d'un robot dans un environnement statique et dynamique en utilisant un LiDAR 2D trouve des applications dans divers domaines tels que la robotique mobile, les véhicules autonomes, la logistique, la surveillance, etc. Elle permet au robot de naviguer de manière autonome en détectant et en évitant les obstacles, offrant ainsi une navigation sécurisée et efficace dans des environnements complexes et changeants.

L'objectif de ce travail est de développer une approche intégrée qui utilise les données du LiDAR 2D pour permettre au robot de naviguer de manière autonome dans un environnement complexe. La méthode proposée comprend plusieurs étapes, telles que la perception des obstacles à l'aide du LiDAR 2D, la construction d'une carte de l'environnement, la planification d'un chemin global en évitant les obstacles détectés, et la mise en œuvre d'un contrôle en boucle fermée pour la navigation locale en utilisant les informations du LiDAR 2D en tempsréel.

Le mémoire est divisé en quatre chapitres. Dans le chapitre1, nous nous intéressons aux robots mobiles car ils sont au centre de notre étude, en plus de l'explication des capteurset en particulier le LiDAR.

Dans le chapitre 2, nous avons donné une vue d'ensemble des types d'environnements et des approches pour les définir, puis nous avons abordé la navigation. Nous avons commencé par la navigation globale, puis la navigation locale, qui se concentre sur l'évitement des collisions. Nous avons présenté dans le chapitre 3, la conception générale et la conception détaillée de notre projet et dans le chapitre 4, nous avons présenté les étapes d'implémentation et les résultats. Le mémoire est cloturé par une conclusion générale et les travaux.

Chapitre 01

La robotique

1.1 Introduction

Le terme robotique a été employé pour la première fois par Asimov en 1941.C'est l'une des branches les plus importantes du IA. La robotique est la nouvelle révolution dans ce monde numérique a rendu les emplois de l'homme plus facile, est largement utilisé dans les industries manufacturières, laboratoires, contrôle de la circulation, la recherche, pour effectuer des opérations de recherche et dans les missions militaires.

En d'autres termes, ils ont remplacé de nombreux emplois qui sont par ailleurs occupés par des humains. La technologie joue un rôle essentiel pour aider les humains à travailler plus efficacement. Depuis l'automatisation est devenue une partie intégrante des opérations commerciales, nous pouvons prédire que les robots vont bientôt remplacer bon nombre des emplois qui sont maintenant effectués par les humains.

1.2 Robot

1.2.1 Définition

Un robot est un automate machine programmable conçu pour effectuer une tâche spécifique dans un environnement donné. Il est doté de capteurs et d'actionneurs qui lui confèrent des capacités de mobilité et d'adaptation similaires à celles d'un véhicule autonome. Un robot est un agent physique capable de détecter les problèmes dans son environnement (1) .Bien que la compréhension générale du concept de robot soit répandue, il est difficile de donner une définition précise. Selon le Oxford English Dictionnaire, un robot est capable d'accomplir une série complexe d'activités automatiques, en particulier lorsqu'il est programmé par ordinateur. Cette définition souligne l'aspect fondamental de l'automatisation des activités. Contrairement à un dispositif automatisé de base tel qu'une machine à laver, où le processus est prédéfini, un robot exécute un ensemble d'actions plus complexes. Le degré d'autonomie d'un robot détermine le niveau d'implication humaine nécessaire. Il peut être mesuré en termes d'autonomie dans la réception de l'information, le traitement de l'information, la prise de décision et la mise en œuvre de l'action **[2]**.

1.2.2 Classification de Robot

Les robots peuvent être classés en fonction de l'environnement dans lequel ils opèrent, il existe deux grandes catégories de robots sont :

a) Robots fixes

Les robots fixes sont surtout des manipulateurs robotiques industrielles travaillant dans des environnements bien définis et adaptés aux robots le robot exécute une même série d'actions indéfiniment, sans aucune perception de son environnement. Les robots ancrés physiquement à leur place de travail et généralement mis en place pour réaliser une tâche précise ou répétitive **[1].**

b) Robots mobiles

Sont censés se déplacer et effectuer des tâches dans des environnements vastes, mal définis et incertains qui ne sont pas conçus spécifiquement pour les robots. Ils doivent faire face à des situations qui ne sont pas précisément connues à l'avance et qui changent avec le temps **[1].**

Il y a trois environnements principaux pour les robots mobiles qui nécessitent de manière significative. Différents principes de conception parce qu'ils diffèrent dans la mécanique du mouvement : aquatique (exploration sous-marine), terrestre (voitures) ou aérienne (drones) **[3].**

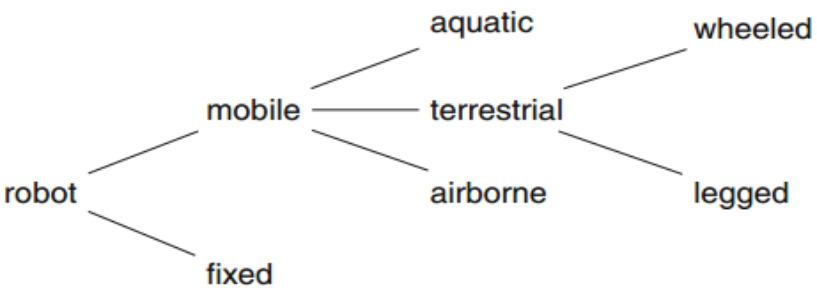


Figure 1.1 : Classement des robots par environnement et mécanisme d'interaction **[3]**.

c) Différence entre les robots fixes et les robots mobiles

-Les deux catégories de robots présentent des environnements de travail distincts, ce qui nécessite des capacités différentes.

-Les robots fixes sont fixés à un support stable au sol, ce qui leur permet de calculer leur position en fonction de leur état interne. En revanche, les robots mobiles doivent s'appuyer sur leur perception de l'environnement pour déterminer leur emplacement.

-Une différence fondamentale se manifeste également au niveau de la programmation.

-La robotique mobile repose sur des règles et peut être considérée comme axée sur des tâches, tandis que les robots industriels classiques sont équipés d'un logiciel explicite composé d'une séquence d'ordres fixes **[3].**

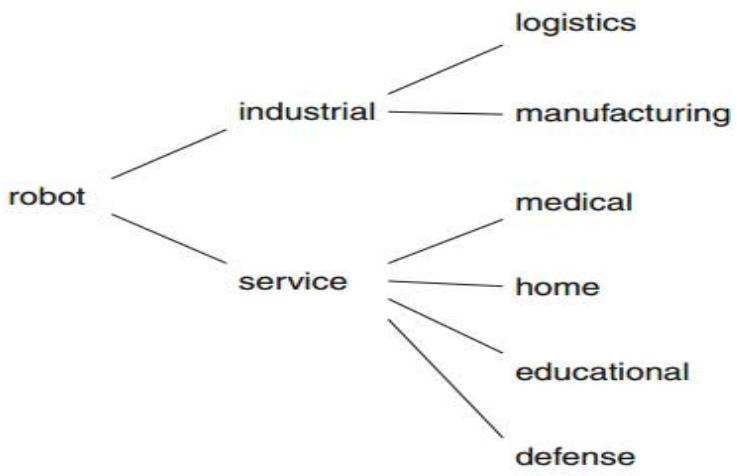


Figure 1.2 : Classification des robots selon le domaine d'utilisation **[3]**.

1.2.3 Robots industriels

Les premiers robots étaient des robots industriels qui remplaçaient des travailleurs humains exécutant des tâches simples et répétitives **[3].** En 1961 le premier robot industriel installé dans une installation de General Motors : UNIMATE (tubes cathodiques de télévision).En 1972, Nissan ouvre la première chaîne de production entièrement automatisée. PUMA (Global Programmable Machine for Assembly) développé par GM en 1978 (toujours en service) **[1]**.

Les chaînes d'assemblage d'usine peuvent fonctionner sans la présence d'humains sauf que l'humain fournit les tâches globales au robot. Les robots industriels travaillent dans un environnement bien défini où le robot doit effectuer des tâches dans un ordre précis agissant sur des objets placés précisément devant lui, Cependant, comme ils fonctionnent dans un environnement personnalisé auquel les humains ne sont pas autorisés pendant que le robot fonctionne, leur conception a été simplifiée **[3].**

Figure 1.3 : À droite Robots sur une chaîne de montage dans une usine automobile, À gauche Robots sur une chaîne de montage dans l'industrie électronique.

Lorsque des robots industriels interagissent avec des personnes, une plus grande flexibilité est nécessaire, ce qui soulève de graves problèmes de sécurité. En particulier, la vitesse du robot doit être réduite et la conception mécanique doit garantir que les pièces mobiles ne mettent pas l'utilisateur en danger **[3].**

L'utilisation de robots industriels dans de nombreux domaines (manipulation de produits chimiques, chaine montage dans une usine automobile,...).

1.2.4 Robots mobiles autonomes (ARM)

Les entreprises agricoles, les entreprises logistiques, les entrepôts et les organismes de soins de santé sont tous à la recherche d'approches novatrices pour accroître l'efficacité opérationnelle, assurer la précision, accélérer et accroître la sécurité. Les ARM (Autonomous Mobile Robots), ou robots mobiles autonomes, sont de plus en plus populaires.

Un robot mobile autonome est un type de robot qui a la capacité de comprendre et de naviguer son environnement par lui-même. Les GAR sont différents des véhicules autonomes guidés (AGV), qui nécessitent souvent la supervision de l'opérateur et reposent sur des voies ou des trajectoires prédéfinies. Des robots mobiles entièrement autonomes effectuent des tâches et prennent des décisions sans avoir besoin d'un opérateur.

Non limités par l'énergie filaire, les RAM utilisent une gamme complexe de capteurs, d'intelligence artificielle, d'apprentissage automatique et de calcul pour la planification des trajectoires afin d'interpréter et de se déplacer dans leur environnement. Donc s'ils rencontrent un obstacle imprévu alors qu'ils naviguent dans leur environnement, comme une boîte tombée ou une foule de personnes, ils utiliseront une technique de navigation comme l'évitement de collision, puis poursuivre leur tâche à réparer.

1.2.5 Robots de service

Différentes définitions ont été données pour décrire les robots de service, un type de robot principalement utilisé en dehors des environnements industriels. Les robots de service, par exemple, sont ceux qui "effectuent des tâches bénéfiques pour l'homme ou l'équipement à l'exclusion des applications d'automatisation industrielle", selon la Fédération internationale de robotique. "Les interfaces autonomes et adaptatives basées sur les systèmes qui interagissent, communiquent et fournissent des services aux clients d'une organisation," selon Wirtz et al **[4].**

Un robot de service peut interagir avec les gens en plus d'avoir des caractéristiques technologiques pour fournir des services. En plus d'améliorer la vision par ordinateur, la

reconnaissance vocale, les capteurs et l'intelligence artificielle, la technologie des robots de service progresse rapidement. Les robots deviennent plus intelligents, plus mobiles et plus abordables en raison des progrès réalisés dans les capteurs, les systèmes de navigation et l'apprentissage automatique **[4].**

Aujourd'hui ce type de robot est devenu un service pour divers domaines (médecine, pharmacie, militaire, orientation, addition, éducation, ...) et tous les groupes de la société (enfants, personnes âgées, handicapés,...).

Figure 1.4: Robots de service**[19].**

1.2.6 Accessoires d'un robot

Un robot est un assemblage complexe de pièces mécaniques et de pièces électroniques, le tout piloté par une intelligence artificielle. Lorsque les robots autonomes sont mobiles, ils possèdent également une source d'énergie embarquée, généralement une batterie d'accumulateurs électrique **[5].**

a) Système de Contrôle

Le contrôleur d'un robot doit traiter l'information et générer des actions appropriées. Les petits appareils ont tendance à traiter avec des environnements moins complexes et plus restreints et à faire des mouvements plus petits et plus lents, donc ils pourraient nécessiter un

système de contrôle plus simple et moins rapide. Néanmoins, en raison des très petites dimensions des microrobots, les processeurs embarqués avec une puissance de calcul suffisante sont encore un défi **[6].**

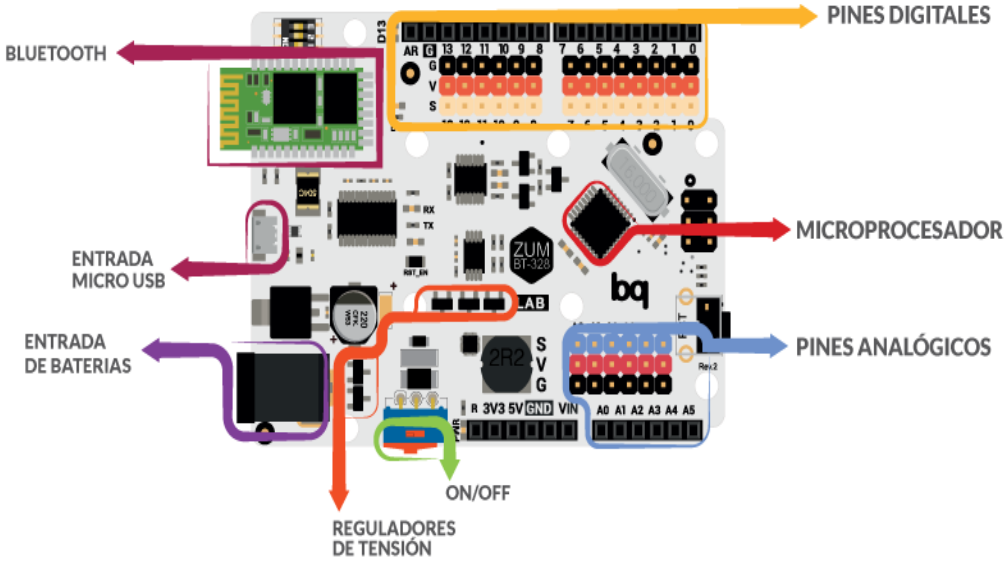


Figure 1.5 : Éléments constitutifs de la carte ZUM **[6].**

Carte de commande

Le cerveau ou la carte de commande fonctionne comme un petit ordinateur. Il s'agit en fait d'un microcontrôleur qui peut être relié aux capteurs et aux actionneurs (muscles, voix, etc.).

Microprocesseur

La puce effectue toutes les opérations nécessaires pour rendre votre carte fonctionnelle avec votre programme. Sont des éléments essentiels d'un robot, car ils permettent l'exécution de logiciels donnant autonomie au robot. Dans un robot, on trouve souvent des modèles à très faible consommation d'énergie, surtout pour les petits robots, qui ne peuvent pas transporter avec eux une importante source d'énergie **[6].**

Entrées numériques

Elles ne peuvent avoir que deux valeurs : 0 ou 1 (tout ou rien, on ou off).

Entrées analogiques

Elles peuvent avoir plusieurs valeurs : 0, 1, 34, 255, etc.

Entrée micro-USB

La connexion entre votre ordinateur et la carte ZUM s'établit via ce port.

Entrée batterie

La carte électronique est alimentée par la batterie ou le bloc d'alimentation pour la faire fonctionner.

Bluetooth

Cela permet à la carte électronique de communiquer sans fil avec un ordinateur, un Smartphone ou un autre appareil.

Bouton marche/arrêt

Il coupe l'alimentation de la carte électronique afin de pouvoir programmer le robot sans risquer qu'il se mette en marche.

Régulateurs de tension

Ils ajustent la tension d'alimentation pour qu'elle soit égale à la tension de la carte électronique : 5V.

b) Capteurs

Pour accomplir sa tâche, un robot devrait être équipé d'autant de capteurs que nécessaire pour percevoir l'environnement environnant. Les robots mobiles doivent être en mesure de détecter les obstacles à une distance suffisante pour les éviter ; ainsi, les capteurs tactiles, les capteurs de distance et/ou les capteurs de proximité.

Les capteurs pour microrobots doivent être réduits en taille et en puissance consommation réduite, ce qui peut être un problème majeur. Par conséquent, les capteurs passifs, qui ne fournissent pas d'énergie à l'environnement, où les capteurs actifs très simples sont les capteurs les plus appropriés pour les microrobots. Les caméras et les microphones sont donc très communs comme capteurs passifs dans les microrobots, avec des jauges de contrainte qui peuvent être facilement réduites. Les capteurs de proximité infrarouges actifs, simples d'utilisation, peu coûteux et disponibles dans des emballages compacts, sont également très adaptés et les détails dans la rubrique **[3].**

c) Actionneurs

Les actionneurs sont l'un des problèmes majeurs dans la conception de robots miniatures. Le choix des principes d'actionnement pour la conception d'un microrobot doit atteindre un compromis dans l'amplitude de mouvement, la force, la fréquence d'actionnement, la consommation d'énergie, la précision du contrôle, la fiabilité du système, la robustesse, la capacité de charge, etc.

Les actionneurs sont l'un des problèmes majeurs dans la conception de robots miniatures. Le choix des principes d'actionnement pour la conception d'un microrobot doit atteindre un compromis dans l'amplitude de mouvements, la force, la fréquence d'actionnement, la consommation d'énergie, la précision du contrôle, la fiabilité du système, la robustesse, la capacité de charge, etc.

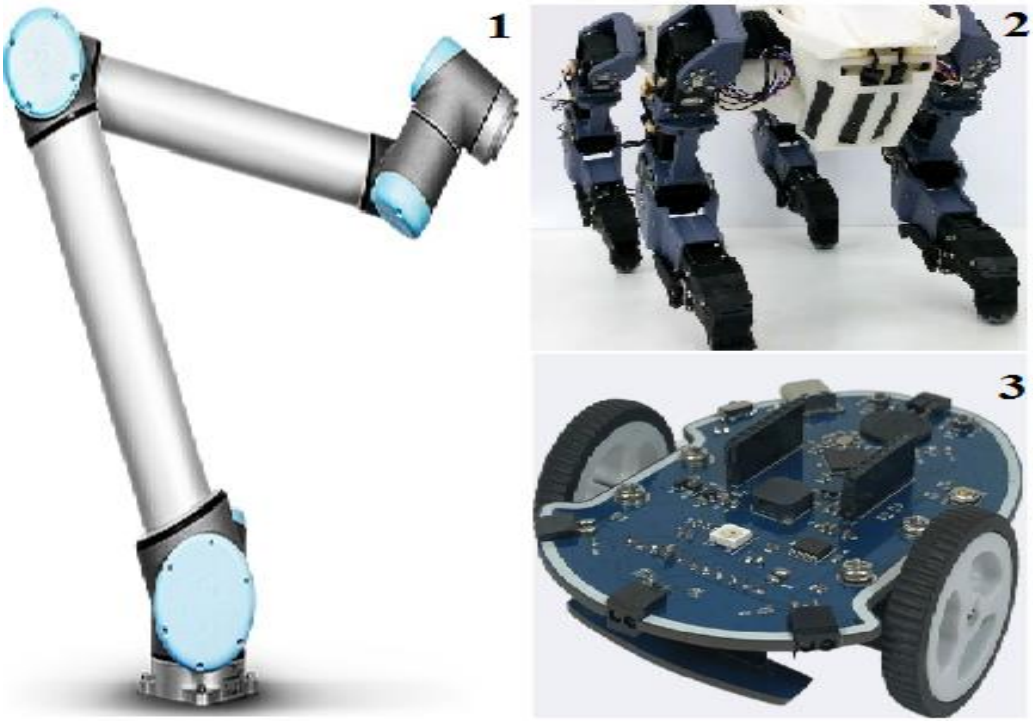


Figure 1.6 : 1 : Bras robotique Robot UR10 UNIVERSAL ROBOTS, 2 : Un robot à 4 pattes capable d'évoluer, 3 : Un robot est composé d'une carte électronique toute simple, d'une batterie au lithium de 3,7 volts, de deux roues et une roue à bille.

1.3 Capteur

1.3.1 Définition.

Un capteur est un composant qui mesure certains aspects de l'environnement. L'ordinateur du robot utilise ces mesures pour contrôler les actions du robot. L'un des capteurs les plus importants en robotique est le capteur de distance qui mesure la distance entre le robot et un objet. En utilisant plusieurs capteurs de distance ou en tournante capteur, l'angle de l'objet par rapport à l'avant du robot peut être mesuré **[3].**

1.3.2 Classification des capteurs

Les capteurs sont classés comme proprioceptifs ou extéroceptifs, et les capteurs extéroceptifs sont classés comme actifs ou passifs **[3].**

a) Capteurs proprioceptif

Un capteur proprioceptif mesure quelque chose d'interne au robot lui-même. L'exemple le plus familier est le compteur de vitesse d'une voiture qui mesure la vitesse de la voiture en comptant les rotations des roues.

b) Capteur extéroceptifs

Un capteur extéroceptif mesure quelque chose d'extérieur au robot comme la distance à un objet. Un capteur actif affecte l'environnement habituellement en émettant de l'énergie : un télémètre sonar sur un sous-marin émet des ondes sonores et utilise le son réfléchi pour déterminer la portée. Un capteur passif ne nuit pas à l'environnement : une caméra capte simplement la lumière réfléchie par un objet. Les robots utilisent invariablement certains capteurs extéroceptifs pour corriger les erreurs qui pourraient résulter des capteurs proprioceptifs ou pour tenir compte des changements dans l'environnement.

1.3.3 Capteurs de distance

Dans la plupart des applications, le robot doit mesurer la distance entre le robot et un objet à l'aide d'un capteur de distance. Les capteurs de distance sont habituellement actifs : ils transmettent un signal et reçoivent ensuite sa réflexion (le cas échéant) d'un objet. Une façon de déterminer la distance est de mesurer le temps qui s'écoule entre l'envoi d'un signal et sa réception (signal parcourt la distance deux fois à l'objet et ensuite réfléchi) **[3].**

S=1/2vt

Avec **v : vitesse de lumière** et **t : temps de vol**

Les capteurs de distance à bas coût reposent sur un autre principe : comme l'intensité d'un signal diminue avec la distance, la mesure de l'intensité d'un signal réfléchi donne. Une indication de la distance du capteur vers un objet. L'inconvénient de cette méthode est que l'intensité du signal reçu est influencée par des facteurs tels que la réflectivité de l'objet.

a) Capteurs de distance à ultrasons

L'échographie est un son dont la fréquence est supérieure à 20000 hertz, plus élevée que la fréquence la plus élevée qui peut être entendue par l'oreille humaine. Il y a deux environnements où le son est meilleur que la vision : la nuit et dans l'eau. (Navires et sous-marins).La vitesse du son dans l'air est d'environ 34.000 cm/s. Si un objet est à une distance de 34 cm d'un robot,

L'avantage des capteurs à ultrasons est qu'ils ne sont pas sensibles aux changements de couleur ou de réflectivité des objets, ni à l'intensité lumineuse dans l'environnement. Ils sont cependant sensibles à la texture : le tissu absorbe une partie du son, tandis que le bois ou le métal reflète presque tout le son. Les capteurs à ultrasons sont relativement bon marché et peuvent fonctionner dans des environnements extérieurs. Leur principal inconvénient est que la mesure de distance est relativement lente, puisque la vitesse de son est beaucoup moins que la vitesse de la lumière.

b) Capteurs de proximité infrarouges

L'œil peut voir la lumière avec des longueurs d'onde d'environ 390 à 700 nm , la lumière infrarouge a des longueurs d'onde entre 700 et 1000 nm , Il est invisible pour l'œil humain et est donc utilisé dans les télécommandes de téléviseurs et d'autres appareils électroniques. Les capteurs de proximité sont des dispositifs simples qui utilisent la lumière

pour détecter la présence d'un objet en mesurant l'intensité de la lumière réfléchie. L'intensité lumineuse diminue avec le carré de la distance de la source et cette relation peut être utilisée pour mesurer la distance approximative par rapport à l'objet. Un objet noir réfléchit moins de lumière qu'un objet blanc placé à la même distance, de sorte qu'un capteur de proximité ne peut pas distinguer un objet noir étroit. C'est la raison pour laquelle ces capteurs sont appelés capteurs de proximité, pas des capteurs de distance et ils sont beaucoup moins chers que les capteurs de distance **[3].**

c) Capteurs optique de distance

La distance peut être calculée à partir d'une mesure du temps écoulé entre l'envoi d'un signal lumineux et sa réception. La lumière peut être de la lumière ordinaire ou d'un laser. La lumière produite par un laser est cohérente (voir ci-dessous). Le plus souvent, les lasers pour la mesure utilisation de la distance lumière infrarouge, bien que la lumière visible peut également être utilisé. Premièrement, les lasers sont plus puissants et peuvent détecter et mesurer la distance aux objets à longue portée. Deuxièmement, un faisceau laser est très focalisé, de sorte qu'une mesure précise de l'angle de l'objet peut être effectuée **[3].**

Supposons qu'une impulsion de lumière est transmise par le robot, réfléchie par un objet et reçue par un capteur sur le robot. La vitesse de la lumière dans l'air est d'environ 300000000 m/s, soit 3 10^8 m/s ou 3 10^10 cm/s en notation scientifique. Si un signal lumineux est dirigé vers un objet à 30 cm du robot, le temps de transmission et de réception du signal est 2*30/3*10^10 =2/10^9 =2*10^-9 s =0.002ms

Il s'agit d'une période très courte, mais elle peut être mesurée par des circuits électroniques.

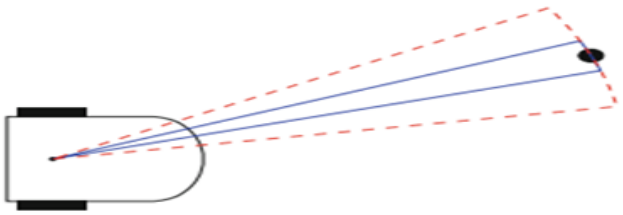


Figure 7 : Largeur de faisceau de lumière laser (solide) et de lumière non lumineuse (pointillée).

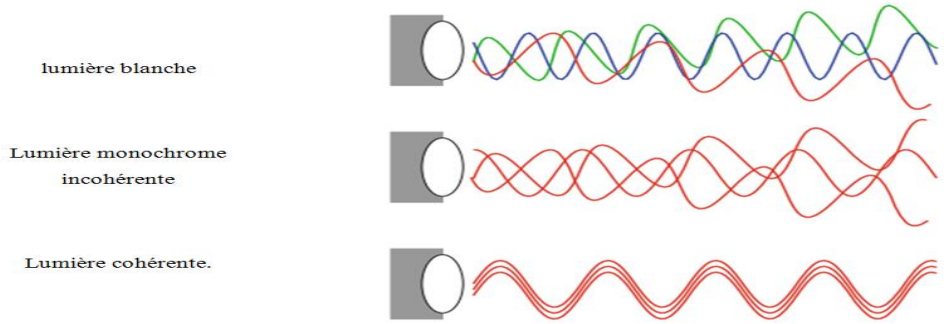


Figure 1.7 : Lumière blanche, monochromatique et cohérente.

d) Capteurs triangulaires

Avant d'expliquer comment un capteur triangulaire fonctionne, nous devons comprendre comment la réflexion de la lumière dépend de l'objet qu'il frappe. Quand un faisceau étroit de lumière comme la lumière cohérente d'un laser frappe une surface brillante comme un miroir, les rayons de lumière rebondissent dans un faisceau étroit. L'angle de réflexion par rapport à la surface de l'objet est le même que l'angle d'incidence. Ceci est appelé réflexion spéculaire (Figure1.8.a).

Lorsque la surface est rugueuse, la réflexion est diffuse (Figure1.8.b) dans toutes les directions car même les zones très rapprochées de la surface ont des angles légèrement différents. La plupart des objets dans un environnement comme les personnes et les murs réfléchissent de façon diffuse, de sorte que pour détecter la lumière laser réfléchie, le détecteur n'a pas besoin d'être placé à un angle précis par rapport à l'émetteur.

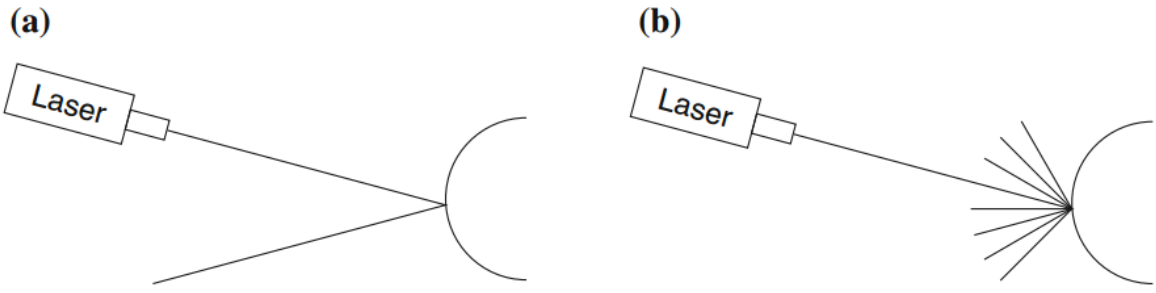


Figure 1.8 : a Réflexion spéculaire, **b** Réflexion diffuse.

La figure 1.9a–b montre un capteur triangulaire simplifié détectant un objet à deux distances. Le capteur se compose d'un émetteur laser et, à distance, d'une lentille qui

concentre la lumière reçue sur un ensemble de capteurs placés à une distance l derrière l'objectif. En supposant que l'objet réfléchit la lumière de façon diffuse, une partie de la lumière sera captée par la lentille et concentrée sur les capteurs. La distance d le long de la matrice de capteurs est inversement proportionnelle à la distance s de l'objet par rapport à l'émetteur laser. Les triangles $\Delta ll'd'$ et $\Delta ss'd$ sont similaires, nous avons donc la formule **:**

s/d = l /d'

Depuis l et d sont fixés par la construction. En mesurant d à partir de l'indice du capteur qui détecte la lumière focalisée, on peut calculer s, la distance de l'objet par rapport à capteur. Le capteur doit être étalonné en mesurant la distance s correspondant à chaque capteur dans le réseau, mais une fois qu'une table est stockée dans l'ordinateur, les distances s peuvent être effectuées par une recherche de table.

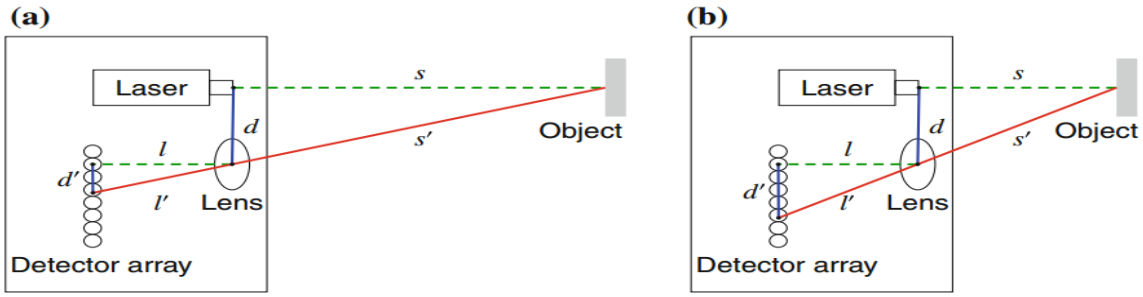


Figure 1.9 : Triangulation d'un objet éloigné, **b** Triangulation d'un objet proche.

1.3.4 Scanners laser

Lorsque des capteurs à ultrasons ou de proximité sont utilisés, un petit nombre de capteurs peuvent être placés autour du robot afin de détecter des objets n'importe où à proximité du robot (Figure. 1.10a). Bien sûr, l'angle de l'objet ne peut pas être mesuré avec précision, mais au moins l'objet sera détecté et le robot peut approcher ou éviter l'objet. Avec un capteur laser, la largeur du faisceau est si petite qu'un grand nombre de lasers (coûteux) seraient nécessaires pour détecter des objets sous n'importe quel angle. Une meilleure conception est de monter un seul capteur laser sur un arbre rotatif pour former un scanner laser (Figure. 1.10b).

Un capteur angulaire peut être utilisé pour déterminer l'angle auquel un objet est détecté. Alternativement, l'ordinateur peut mesurer la période de temps après que le capteur rotatif passe une direction fixe. Une rotation complète de 360 permet à un scanner laser de générer un profil des objets dans l'environnement (Figure.1.11).

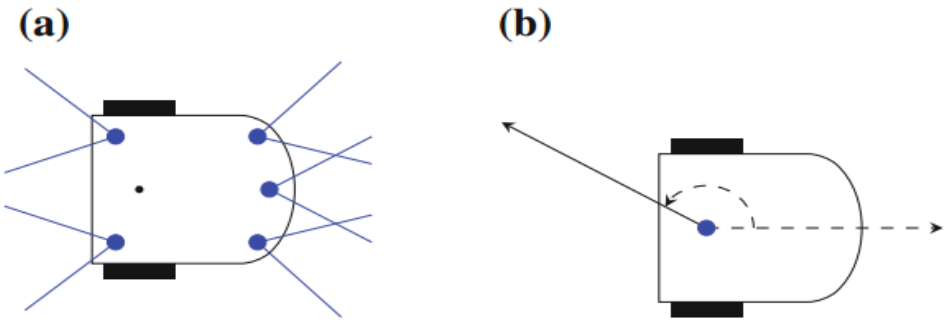


Figure 1.10 : a Cinq capteurs séparés, b Un capteur rotatif

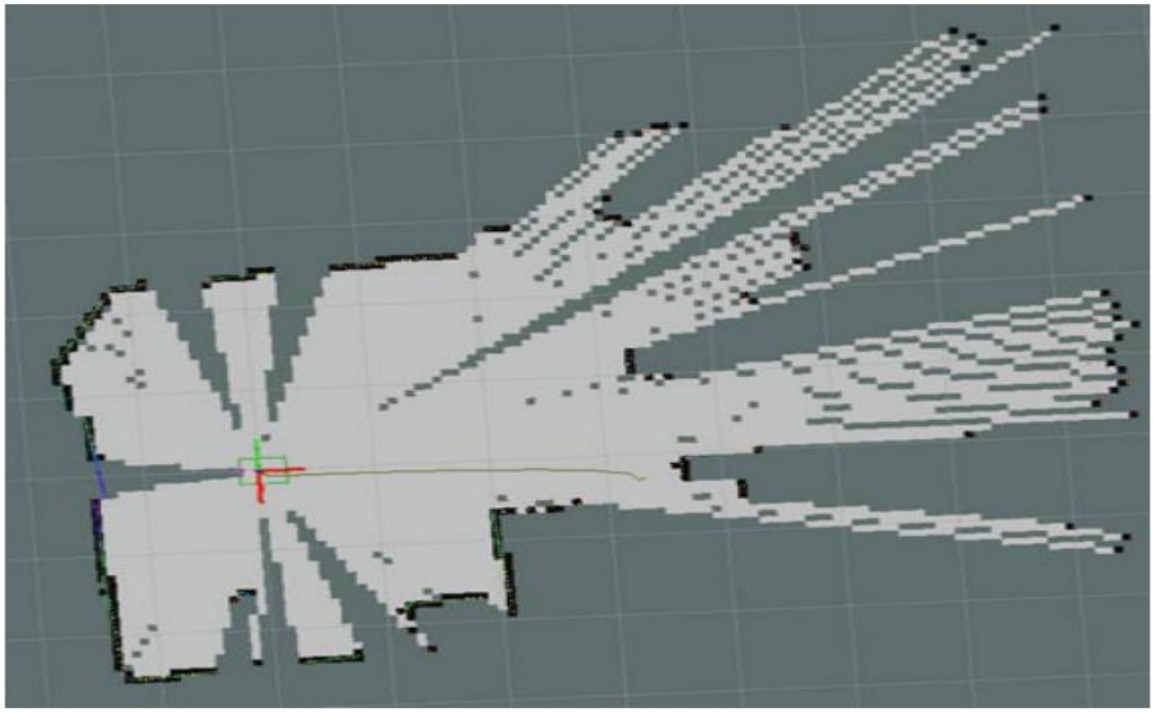


Figure 1.11 : Carte de l'environnement obtenue par un scanner laser.

1.3.5 Caméras

Les appareils photo numériques sont largement utilisés en robotique parce qu'une caméra peut fournir des informations beaucoup plus détaillées que la distance et l'angle d'un objet. Les appareils photo numériques utilisent un composant électronique appelé dispositif à couplage de charge qui détecte les ondes lumineuses et renvoie un tableau d'éléments d'image, ou pixels courts (Figure. 1.12).

Les appareils photo numériques sont caractérisés par le nombre de pixels capturés dans chaque image et par le contenu des pixels. Une petite caméra utilisée dans un robot éducatif contient 192 rangées de 256 pixels chacune pour un total de 49 152 pixels. C'est une très petite image : les capteurs des appareils photo numériques dans les Smartphones enregistrent des images de millions de pixels.

Une caméra peut renvoyer des valeurs pour chaque pixel en noir et blanc (1 bit par pixel), des nuances de gris appelées niveaux de gris (8 bits par pixel) ou en couleur rouge-vert-bleu (RVB) (3 × 8 = 24 bits par pixel). La petite caméra 256 × 192 a donc besoin d'environ 50 kbytes pour une seule image en niveaux de gris ou 150 kbytes pour une image couleur. Comme un robot mobile comme une voiture autonome devra stocker plusieurs images par seconde (les films et la télévision affichent 24 images par seconde), la mémoire nécessaire pour stocker et analyser les images peut être très grande **[3].**

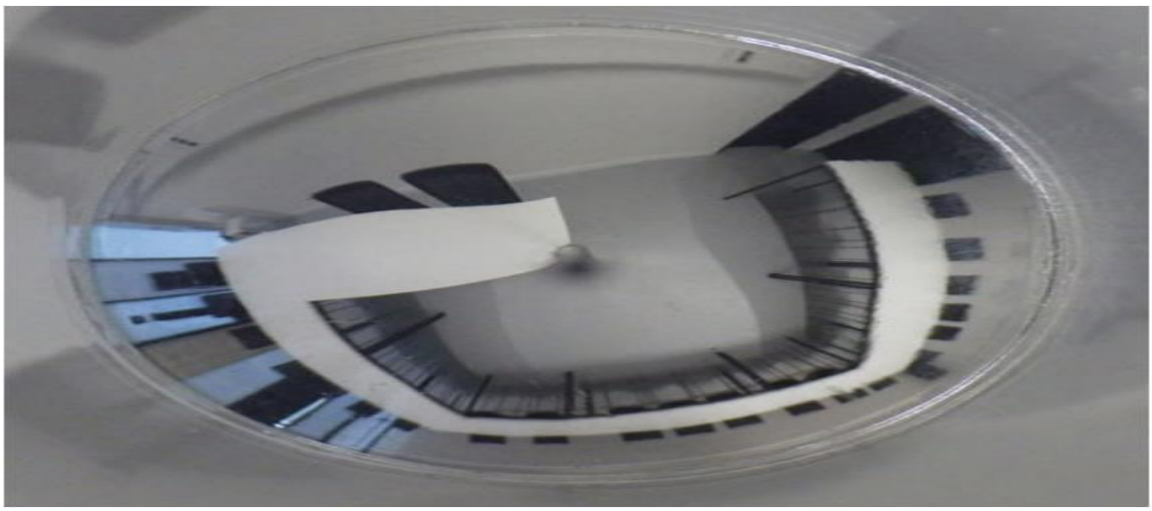


Figure 1.12 : Image capturée par une caméra omnidirectionnelle avec un champ de vision de 360 degrés.

1.3.6 LiDAR

a) Qu'est-ce que le LiDAR exactement ?

Aujourd'hui, les capteurs dont le fonctionnement repose sur la mesure de distance sans contact physique avec un laser sont monnaie courante dans le monde de l'automatisation .Ce phénomène est apparu pour la première fois avec la méthode de mesure TOF .La méthode TOF(Time of Flight), également connu sous le nom de temps de vol ou temps de propagation de la lumière en français, a été largement remplacé par les concepts plus précis de LADAR ou du LIDAR plus répandu .LIDAR (Laser Detection and Ranging) ou LiDAR (Light Detection and Ranging) fait sciemment référence au RADAR familièrement employé et acronyme de Radio Detection and Ranging.

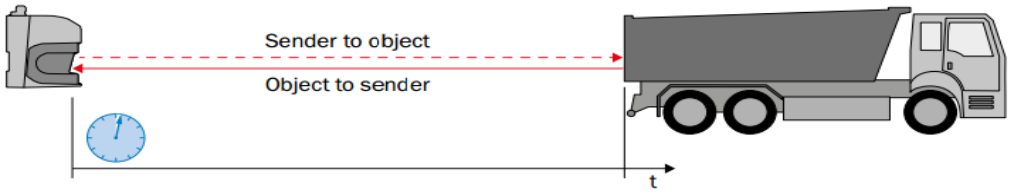


Figure 1.13 : Mesure de la distance par la mesure du temps de propagation de l'impulsion

b) Les capteurs LiDAR en modes 1D, 2D et 3D

La forme la plus simple des capteurs LiDAR se trouve dans les appareils de mesure de distance et dans les capteurs qui fonctionnent comme des systèmes de mesure de distance basés sur des points. Pour une mesure directe de la distance, ils sont dirigés sur une cible naturelle ou sur un réflecteur. Les capteurs monodimensionnels, ou capteurs 1D, sont ceux qui fonctionnent dans une seule dimension (distance).

En déplaçant le faisceau de mesure ou en le faisant tourner dans un plan, on obtient des renseignements sur la distance et l'angle, et donc en deux dimensions. Pour une telle mesure, les capteurs utilisés sont habituellement des scanners laser 2D ou capteurs 2D-LiDAR. Ils déterminent les valeurs de mesure de manière séquentielle avec un intervalle de temps normalement régulier entre les mesures **[8].**

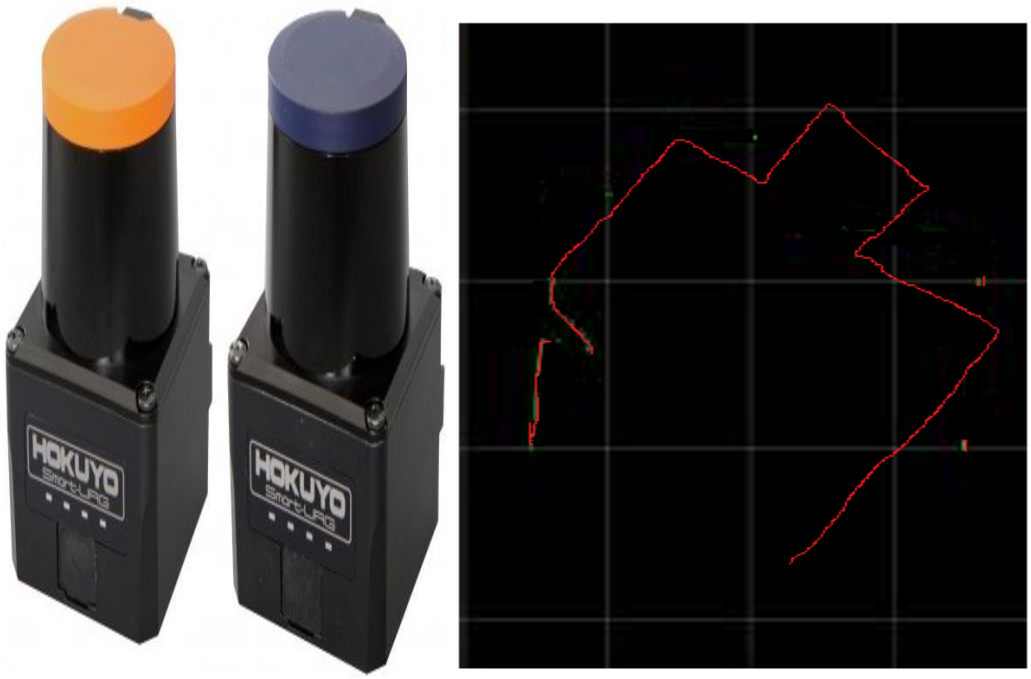


Figure 1.14 : Capteurs 2D-LiDAR Hokuyo UST-10LX.

Les capteurs LiDAR utilisés pour la troisième dimension sont orientés de manière rotative. Cela permet d'obtenir des informations aussi bien sur l'écart et la position sur l'axe x que sur les positions sur les axes y et z. Les mêmes informations de paramètres spatiaux sont obtenues en positionnant plusieurs systèmes émetteurs et récepteurs à divers angles horizontaux dans un capteur tout en utilisant l'équilibrage. Le problème qui se pose concerne plusieurs scanners.

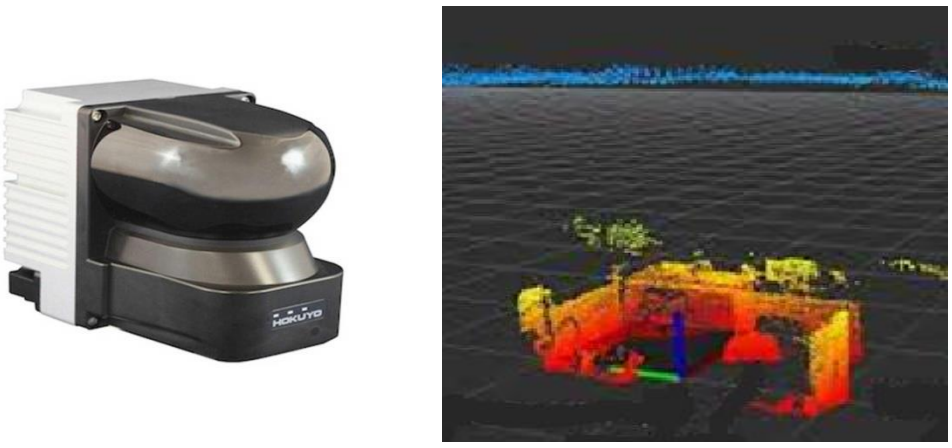


Figure 1.15 : Capteur 3D Hokuyo YVT-35LX-F0.

De manière générale, les capteurs LiDAR sont employés dans des boucles de régulation industrielles traditionnelles. L'utilisation de capteurs de type LiDAR dans des applications liées à l'assurance qualité est expliquée ailleurs.

c) Capteurs de mesure de surface (2D)

Le but du développement des capteurs 2D était de conserver les caractéristiques exceptionnelles de la mesure de surface laser et de les appliquer à un capteur de mesure de surface. La technique utilisée ici pour déplacer un faisceau laser via un miroir rotatif semble simple. Pourtant, la difficulté réside dans les détails. De nombreux capteurs de ballast sont des systèmes de mesure coaxiaux **[8].**

Le faisceau d'émission est situé au milieu du panneau de réception. Il est alors transformé par un miroir tournant. Toutes les caractéristiques décrites qui rendent la mesure laser si exceptionnelle sont conservées, y compris la grande portée et la capacité de même mesurer des objets extrêmement sombres.

Les capteurs LiDAR synchronisent l'ordre séquentiel des impulsions laser avec la fréquence de rotation du moteur et la résolution angulaire souhaitée. Habituellement, la fréquence d'envoi maximale de la source laser et la résolution angulaire souhaitée déterminent la vitesse de rotation du moteur. Lors d'un virage, il n'est pas possible de générer davantage d'impulsions que ne le permet le circuit laser.

Les capteurs de numérisation se distinguent également par la résolution angulaire et la précision angulaire de l'ordre d'envoi par la fréquence de mesure élevée (vitesse de rotation des moteurs). La déviation d'un rayon laser dans un miroir nécessite une haute précision mécanique.

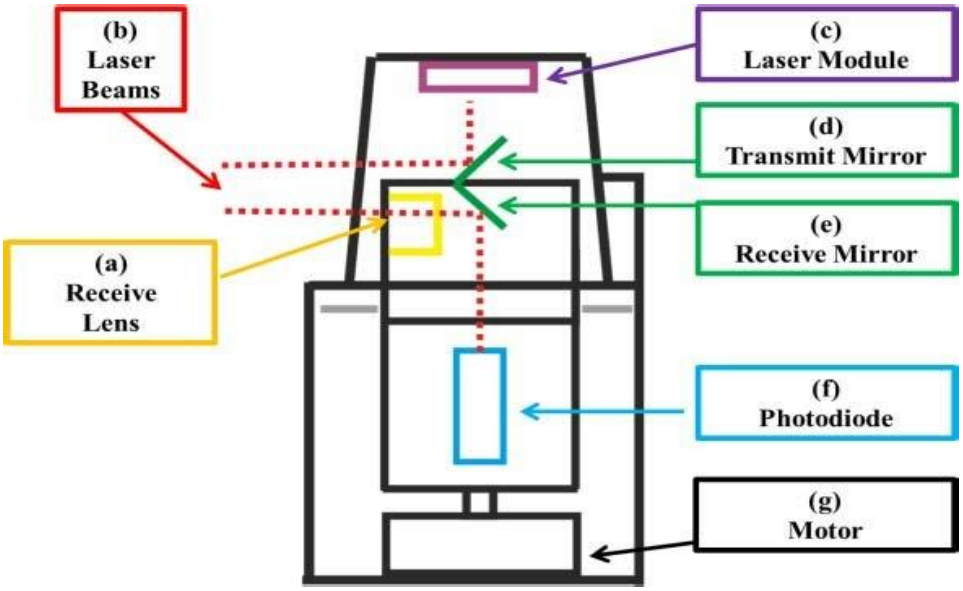


Figure 1.16 : Hokuyo LiDAR structure **[23].**

d) Pourquoi le LiDAR 2D?

Pour le coût d'utilisation de lidar 3d, lidar2d a été utilisé à la place ou modifié pour éliminer ces problèmes **[8].**

-Les capteurs LiDAR 2D sont plus portatifs et efficaces que les capteurs LiDAR 3D.

-Le système peut prédire des objets éloignés sans limites physiques importantes.

-Les systèmes LiDAR 3D modernes sont dotés d'un nuage de points épars, qui peut manquer des objets pendant la numérisation. La profondeur au niveau des pixels du système de l'équipe offre aux utilisateurs une résolution beaucoup plus élevée que le LiDAR 2D

-La précision d'un LiDAR autonome diminue considérablement par mauvais temps comme la neige, la pluie ou la poussière. La fusion des deux capteurs rend le système plus robuste.

e) Avantages du LIDAR

-La collecte de données peut être réalisée rapidement et avec une grande précision.

-Les capteurs LiDAR garantissent une synchronisation précise entre les impulsions laser, la fréquence de rotation du moteur et la résolution angulaire souhaitée.

-Les LiDAR sont rapides et très précis, ce qui en fait un outil idéal pour collecter des données sur de vastes territoires.

-Une fois correctement configuré, un LiDAR est autonome et peut fonctionner de manière indépendante.

f) Inconvénients du LIDAR

-Le coût du LiDAR peut varier en fonction des spécifications requises pour un projet, le rendant potentiellement coûteux.

-Les conditions météorologiques telles que fortes pluies, nuages bas, brouillard, fumée ou présence d'obstacles transparents peuvent rendre les LiDAR inefficaces.

-L'analyse d'un volume important de données collectées peut nécessiter du temps et des ressources supplémentaires.

-Certains LiDAR utilisent des faisceaux laser puissants qui peuvent présenter un risque de dommages pour les yeux humains.

-Il peut être difficile pour un LiDAR de traverser des matériaux extrêmement denses.

1.4 Conclusion

Maintenant, il a été confirmé qu'il est impossible pour l'homme de se passer de robots de toutes sortes : service industriel et mobile, où la robotique est devenue un domaine qui est constamment en cours de développement et d'amélioration et d'essayer de résoudre automatiquement tous les problèmes humains. Nous avons également découvert les types de capteurs, en particulier LiDAR, sur lesquels nous nous appuierons dans les prochains chapitres.

Le chapitre suivant contient une approche des robots mobiles en particulier et une tentative d'approfondir la compréhension de leur conception et programmation pour la mission de navigation dans un environnement réel.

Chapitre 02

Navigation d’un robot

2.1 Introduction

Dès le début des années 1950, le domaine de l'intelligence artificielle s'est intéressé à la création et au développement de robots, en particulier de robots mobiles, programmables et physiques, afin d'effectuer de nombreuses tâches humaines et de se faciliter la vie.

La navigation autonome d'un robot est la tâche fondamentale de toutes les tâches sur lesquelles le robot est programmé est sa capacité à naviguer et à explorer l'environnement. La stratégie de navigation a été classifiée sur la base des informations préalables sur l'environnement nécessaires à la planification des trajectoires. Il est généralement classé comme la navigation globale et la navigation locale. Dans la navigation globale, le robot mobile doit exiger l'information préalable de l'environnement, de la position de l'obstacle et de la position de l'objectif, tandis que dans la navigation locale, le robot mobile n'exige pas l'information préalable de l'environnement. La stratégie globale de navigation traite d'un environnement complètement connu. La stratégie de navigation locale traite de l'environnement inconnu et partiellement connu. L'algorithme de planification de trajectoire pour un environnement connu est basé sur une approche classique telle que CD, RA et APF. Ces algorithmes sont traditionnels et ont une intelligence limitée. Les approches de navigation locales sont appelées approches réactives, car elles sont plus intelligentes et sont capables de mapping, de contrôler et d'exécuter un plan de façon autonome.

2.2 L'Environnement de Robot Mobile

2.2.1 Environnement statique

La planification de parcours est la formulation la plus simple du problème de planification de mouvement puisque l'environnement dans lequel le robot opère est statique. Le problème peut être réduit à la composante géométrique de la solution, de sorte que toute courbe qui satisfait aux exigences de continuité et de non-collision est une solution réalisable, de manière générale, calculer le chemin le plus court de la configuration initiale à la configuration finale. Plusieurs méthodes ont été mises au point et selon la connaissance préalable que le robot a son environnement.

2.2.2 Environnement dynamique

Dans certains problèmes, l'espace de travail peut être dynamique. Contrairement à la planification de trajectoire, les positions des obstacles changent avec le temps, faisant du temps un paramètre important pour évaluer l'efficacité du mouvement du robot. Par conséquent, la planification de trajectoire doit respecter les contraintes dynamiques imposées par l'environnement (obstacles mobiles) et le système robotique considéré (sa dynamique).

2.2.3 Approche classique de représentation d'environnement

2.2.4 Approche de roadmaps (AR)

L'AR est également connu sous le nom d'approche routière. C'est la façon de passer d'un endroit à un autre et la connexion entre les espaces clairs est représentée par une série de courbes unidimensionnelles. Une fois roadmaps elle est utilisée comme un ensemble d'itinéraires uniformes où le planificateur recherchera la meilleure configuration. Les nœuds ici sont cruciaux pour fournir au robot le chemin requis. Le trajet le plus rapide entre la position de départ du robot et sa position de destination est déterminé à l'aide de l'AR, la roadmap est créée à l'aide des graphes de visibilité et de Voronoi.

a) Graphes de visibilité

Suivant l'idée que le chemin le plus court pour relier la configuration finale depuis le début passe au - dessus des barrières polygonales placées dans l'environnement, Nilsson **[9]** a proposé des graphes de visibilité en 1969. Ce graphe est construit en reliant le sommet de chaque obstacle par une droite, sans que ces droites coupent d'autres obstacles. En fait, le tracé de base relie les sommets visibles des obstacles à droite, Ce processus est répété entre les sommets des obstacles et les sommets visibles des obstacles suivants jusqu'à ce que la configuration finale soit atteinte. Puisque cette méthode permet un contact entre le robot et l'obstacle, il devient relativement rarement utilisé pour résoudre des problèmes de planification **[11].**

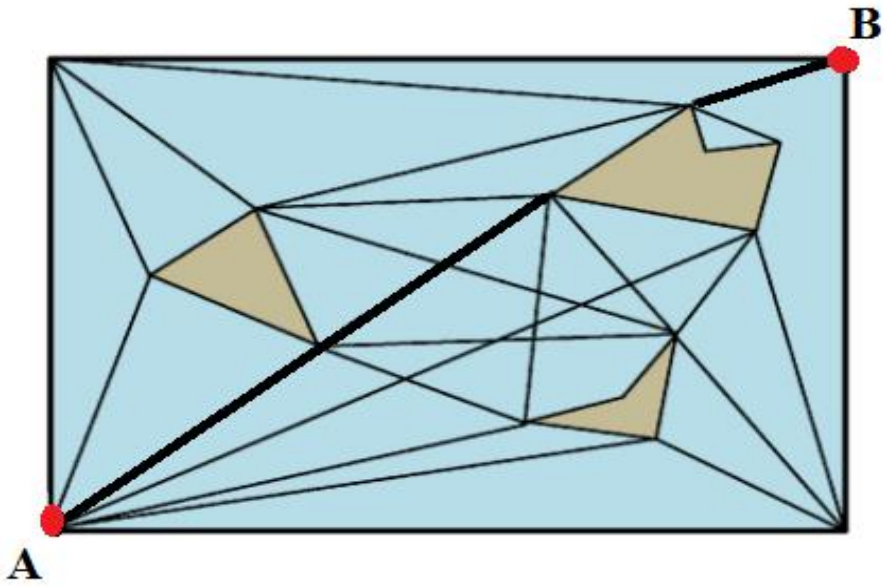


Figure 2.1 : Graphes de visibilité

b) Les diagrammes de Voronoï

Une autre méthode de construction de roadmap consiste à utiliser les diagrammes de Voronoï **[10].** Contrairement aux graphes de visibilité, ces diagrammes évitent tout contact avec les obstacles et garantissent ainsi une distance de sécurité. La construction d'un diagramme de Voronoï implique le tracé de lignes d'égale distance par rapport aux obstacles et aux bords de l'environnement exploré. Ensuite, le graphe de la roadmap est obtenu en reliant la configuration initiale au point le plus proche de ce diagramme, de même que pour la configuration finale. Cette procédure est illustrée dans la figure 2.2.

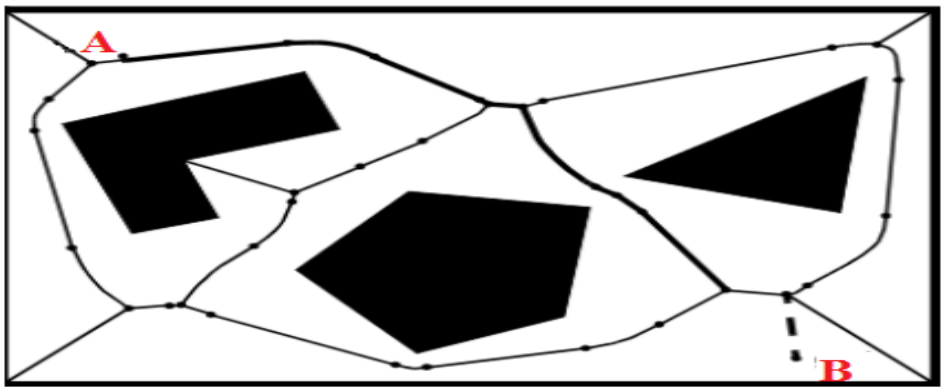


Figure 2.2 : Planification de chemine à l'aide de diagrammes de Voronoï

2.2.5 Approche de décomposition cellulaire (DC)

Cette approche divise la région en grilles (cellules) qui ne se chevauchent pas et utilise un graphe de connexion pour traverser d'une cellule à l'autre pour atteindre l'objectif. Lors de la traversée, des cellules pures (cellules sans obstacles) sont prises en compte pour la planification du chemin de la position initiale à la position cible. Les cellules endommagées (cellules contenant des obstacles) présentes dans le chemin sont ensuite scindées en deux nouvelles cellules pour obtenir une cellule pure, et cette cellule pure est ajoutée à la séquence tout en déterminant la distance la plus courte entre la position initiale et la position cible du meilleur chemin. Les méthodes DC sont classées comme adaptatives, approximatives et exactes **[12].**

2.1.1.1 La décomposition exacte des cellules

Les cellules n'ont pas une forme et une taille mais peuvent être déterminées par la carte de l'environnement, la forme et de l'emplacement de l'obstacle à l'intérieur. Cette méthode utilise la grille de différentes façons. Initialement, l'espace libre disponible dans l'environnement est décomposé en petits éléments (trapézoïdal et triangulaire) suivi d'un nombre à chaque élément. Chaque élément dans l'environnement agit comme un nœud pour un graphique de connectivité. Les nœuds adjacents sont alors autorisés à se joindre dans l'espace de configuration et un chemin dans ce graphique se compare à un détournement dans l'espace libre, qui est décrit par la succession de cellules rayées. Un chemin dans ce diagramme relie à un réseau dans l'espace libre, qui est souligné par la succession des cellules rayées. Ce canal est ensuite converti en voie libre en reliant la disposition sous-jacente à la conception objective par les points médians des points de croisement des cellules adjacentes dans le canal **[12].**

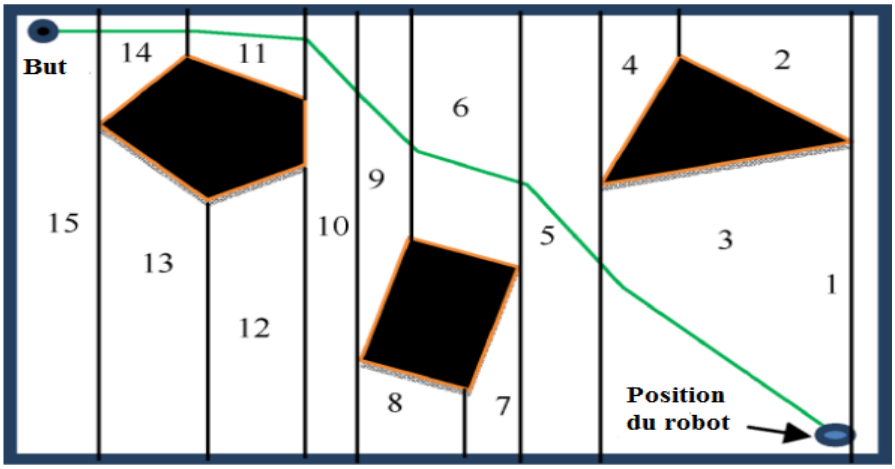


Figure 2.3 : La décomposition exacte des cellules

a) La décomposition cellulaire approximative

La décomposition approximative des cellules divise un espace de configuration en cellules discrètes de formes simples et régulières - comme des rectangles et des carrés (ou d'autres équivalents multidimensionnels). En plus de simplifier le calcul des cellules, Lors de la recherche de chemin dans la zone de recherche, le centre de chaque cellule est supposé être un nœud. La figure 2.4 montre la connexion de 4 et 8 le système de nœuds et le robot doivent se déplacer en diagonale entre eux **[12][18].** Cette méthode prend également en charge la décomposition hiérarchique de l'espace.

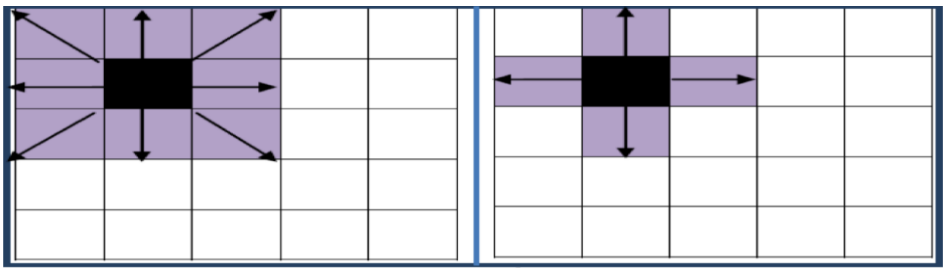


Figure 2.4 : 4 et 8 grilles connectées.

b) Décomposition simple

Un espace de configuration en 2 dimensions peut être divisé en une grille de cellules rectangulaires, qui peuvent être marquées comme occupées ou libres. Un algorithme de recherche peut alors trouver une séquence de cellules libres pour connecter le nœud de départ au nœud final. Cette méthode est plus efficace mais moins exhaustive que la décomposition cellulaire exacte. Cependant, elle peut perdre de son exhaustivité si certaines cellules englobant le chemin sont considérées comme occupées en raison de la présence d'obstacles. Cette approche n'est pas pratique car elle peut classer une cellule comme occupée même si une petite portion de l'espace est réellement occupée **[12].**

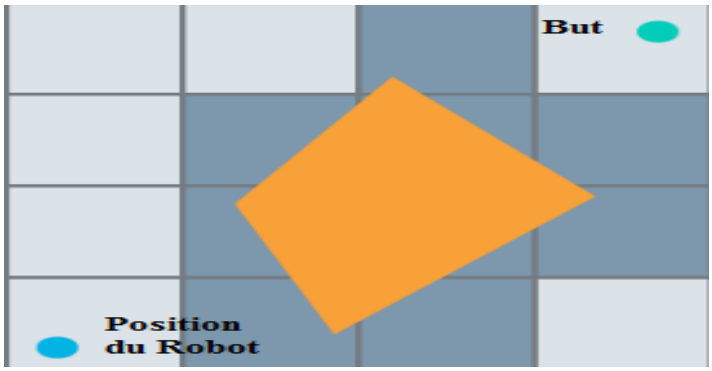


Figure 2.5 : La décomposition cellulaire approximative (Grille régulière)

c) Décomposition itérative (QuadTree)

Il existe une autre méthode pour diviser un espace en cellules simples. Au lieu de décomposer immédiatement l'espace en petites cellules de taille égale, la méthode décompose récursivement un espace en quatre quadrants, utilisé pour représenter les cellules qui sont quelque peu occupées par un obstacle, mais qui contiennent également de l'espace libre. Si une séquence de cellules libres ne peut être trouvée du début à la fin, les cellules mixtes seront décomposées en quatre autres quadrants. Grâce à ce processus, plus de cellules libres émergeront, révélant finalement un chemin si l'on existe. L'implémentation en 2 dimensions de cette méthode est appelée décomposition de QuadTree **[12].**

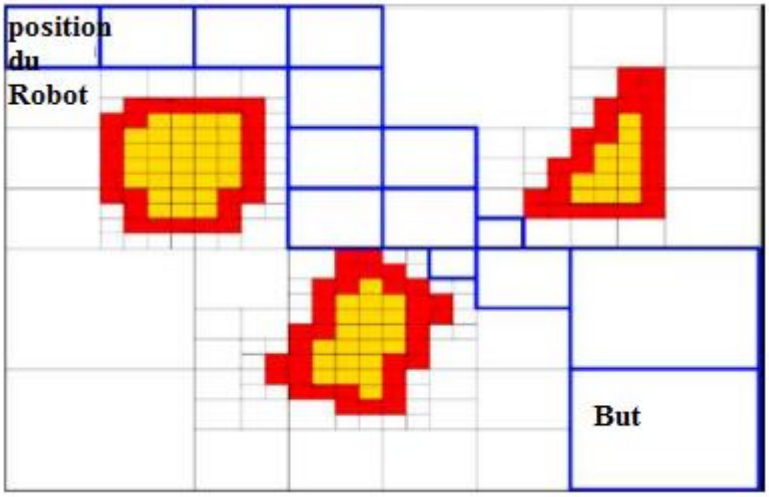
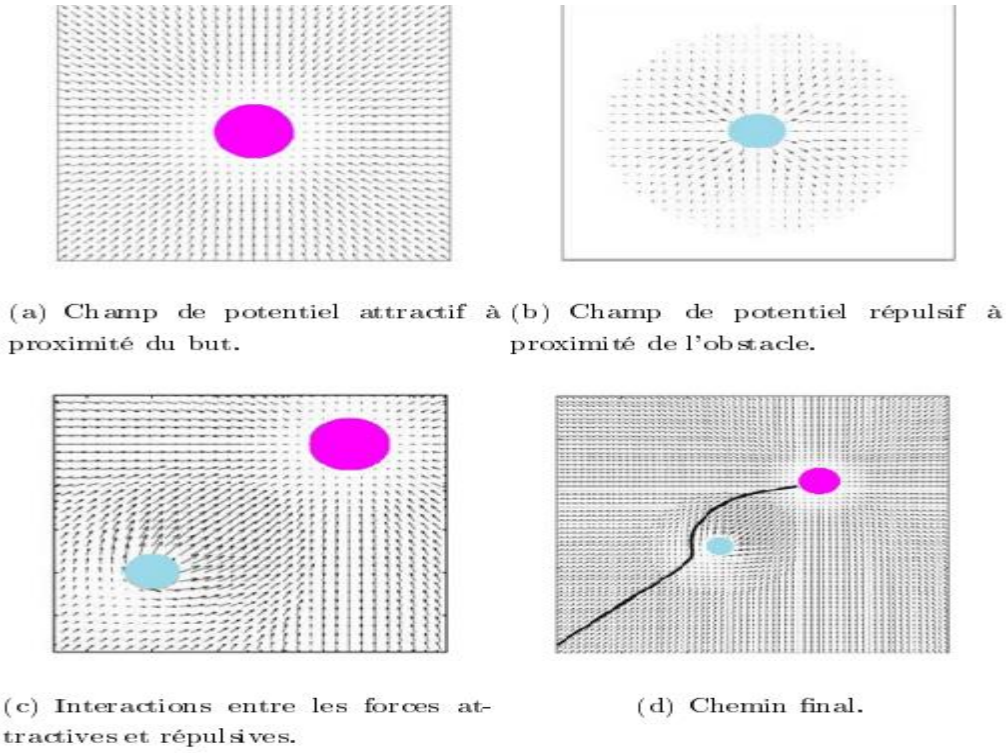


Figure 2.6 : La décomposition cellulaire approximative (QuadTree).

2.4.3 Approche du champ de potentiel artificiel (APF)

L'approche du champ de potentiel d'Oussama Khatib **[20]** consiste à créer un champ artificiel qui guide le mouvement du robot. Le robot est attiré par les objectifs et repoussé par les obstacles, traité comme une particule dans ce champ. Bien que simple et peu coûteuse en calcul, cette approche présente des limitations, notamment le problème de prémisse et le mouvement oscillatoire dans des passages étroits. Elle ne détecte pas efficacement les passages entre obstacles proches. Des adaptations ont été proposées, telles que le suivi de murs ou les déplacements aléatoires pour éviter les minima locaux, et la prise en compte de la vitesse des obstacles. Malgré ses limites, cette approche offre une solution intuitive pour guider les robots, nécessitant des ajustements pour des environnements plus complexes **[13].**



(a) Champ de potentiel attractif à proximité du but.

(b) Champ de potentiel répulsif à proximité de l'obstacle.

(c) Interactions entre les forces attractives et répulsives.

(d) Chemin final.

Figure 2.7 : Approche du champ de potentiel artificiel (APF) **[11].**

2.5 Cartographique

Nous avons vu qu'un robot peut utiliser sa capacité à détecter les obstacles pour se localiser, sur la position des obstacles ou d'autres informations sur l'environnement. Cette information est normalement fournie par une carte. Il est relativement facile de construire une carte des environnements industriels tels que les usines, puisque les machines sont ancrées dans des emplacements fixes. En outre, les robots seraient trop difficiles à utiliser si les clients devaient construire des cartes de leurs appartements et les changer chaque fois qu'un meuble est déplacé. Il va sans dire qu'il est impossible de construire à l'avance des cartes de lieux inaccessibles comme le fond de l'océan.

La solution est de demander au robot de construire sa propre carte de l'environnement. Pour construire une carte, il faut une localisation pour que le robot sache où elle se trouve, mais la localisation a besoin d'une carte, qui a besoin… Pour surmonter ce problème, les robots utilisent des algorithmes de localisation et de cartographie (SLAM) "*simultaneous*

localization and mapping"simultanés. Pour exécuter des robots SLAM utiliser des informations connues pour être valides dans des parties inexplorées de l'environnement, affiner l'information pendant l'exploration.

2.3 Navigation globale

La navigation globale est la capacité de déterminer la localisation et trouver le chemine depuis la position actuelle A de robot vers le but B, cela nécessite de perception par le robot son positionnement dans l'environnement statique à l'aide des capteurs , reconnaître les informations du monde extérieur , puis construire son chemin correctement dans toutes ses étapes, puis le mettre en œuvre une à l'aide des algorithmes de parcours de chemin et donner l'ordre à son moteur de se diriger vers la destination qu'il effectue avec les roues.

L'utilisation des algorithmes de parcours de graphe c'est une étape nécessaire à l'objectif de résolution le problème de programmation le robot sur faire mouvement autogestionnaire et prenez une décision sur le plus court et sûr trajectoire vers la destination voici la planification de chemin.

La complexité de la solution est liée aux caractéristiques des composés de l'environnement dans lequel le robot se déplace en particulier en ce qui concerne de densité et type d'obstacles, dans le cas d'obstacles fixes, le chemin peut être trouvé sans introduire le facteur temps de l'emplacement actuel du robot et de l'obstacle, alors que dans le cas d'un environnement changeant, trouver le bon chemin deviendra plus compliqué que le premier parce que l'environnement a tendance à être mis à jour à chaque unité de temps.

Représentation du problème le plus court du robot vers sa destination avec l'arbre de recherche, les cellules disponibles du robot sont représentées sous la forme des nœuds et la relation avec une orientation.

2.4 Planification globale de chemin

2.4.1 Algorithme de DIJKSTRA

L'algorithme de Dijkstra, développé par le scientifique néerlandais Edsger Dijkstra en 1956 et publié en 1959, est utilisé pour résoudre le problème du plus court chemin à partir d'une seule source dans un graphe. Il génère un arbre du plus court chemin. L'algorithme a diverses applications importantes dans notre vie quotidienne. Par exemple, il aide à déterminer des itinéraires efficaces pour les données transmises sur Internet, en veillant à ce qu'elles soient acheminées par des chemins courts et moins encombrés. Une application courante de l'algorithme de Dijkstra est Google Maps, qui utilise l'algorithme pour spécifier le chemin le plus court entre un emplacement spécifique et un restaurant ou un aéroport. L'algorithme de Dijkstra fait partie des algorithmes gloutons, qui sont efficaces pour résoudre certains problèmes d'optimisation mais peuvent fournir des solutions sous-optimales dans d'autres cas **[3].**

Les Étapes :

Initialiser la distance du nœud source à zéro et attribuer une valeur infinie à la distance de tous les autres nœuds.

Rechercher le nœud avec la plus petite valeur de d(x) et le marquer comme permanent s'il n'existe pas de nœud temporaire ou si sa valeur d(x) est infinie.

Pour chaque nœud temporaire étiqueté en tant que sommet y adjacent à x, comparer la somme de la distance du nœud étiqueté x et le poids du lien (x, y) avec la distance du nœud étiqueté y. Si la somme est plus petite, mettre à jour la distance du nœud étiqueté y.

Avantages et Inconvénients

L'algorithme de Dijkstra arrête de visiter les nœuds restants non désirés une fois que le nœud de destination prévu est atteint. Il est efficace pour trouver le plus court chemin dans un graphe. L'un des inconvénients les plus importants est que l'algorithme de Dijkstra consomme une quantité considérable de mémoire CPU lorsqu'il est exécuté sur un grand nombre de

nœuds. Il ne peut pas gérer les arêtes négatives car il fonctionne uniquement avec des graphes ayant des poids positifs

Algorithme 1 : Algorithme de Dijkstra

```
pour v ∈ V faire
    dist[v]=INFINI;
    pred[v]=AUCUN;
dist[s]=0;
S ← ∅;
F ← V;
tant que F ≠ ∅ faire
    u ← ExtraireMinimum(F)(=ArgMin_{v∈F} d[v]);
    S ← S ∪ {u};
    pour chaque v ∈ Adj[u] faire
        si dist[v] > dist[u] + l(u,v) alors
            dist[v] ← dist[u] + l(u,v);
            pred[v] ← u
```

Figure 2.8 : Algorithme de **Dijkstra [13]**.

2.4.2 Algorithme de A*

L'algorithme A* est un algorithme de recherche graphique. Il est utilisé pour trouver un chemin d'un nœud de départ à un nœud cible .L'algorithme a été développé en 1968 par Hart et al. A* combine l'information utilisée par l'algorithme de Dijkstra avec l'information utilisée par "Best-First-Search". Par rapport à Dijkstra, A* obtient de meilleures performances en utilisant heuristique. C'est une approximation de la distance entre l'emplacement actuel et l'emplacement cible **[14].**

Dans le processus de recherche de l'itinéraire le plus court, chaque cellule de la grille est évaluée en fonction d'une fonction d'évaluation donnée par : **f(n) = g(n) + h(n) ,** f (n) est la fonction d'évaluation, où h(n) est le coût heuristique du chemin minimum pour atteindre le nœud de but g. En outre, le g(n) se réfère au coût cumulé à partir du point de départ s jusqu'au point actuel n. (La figure 2.9) montre comment l'estimation est déterminée **[14].**

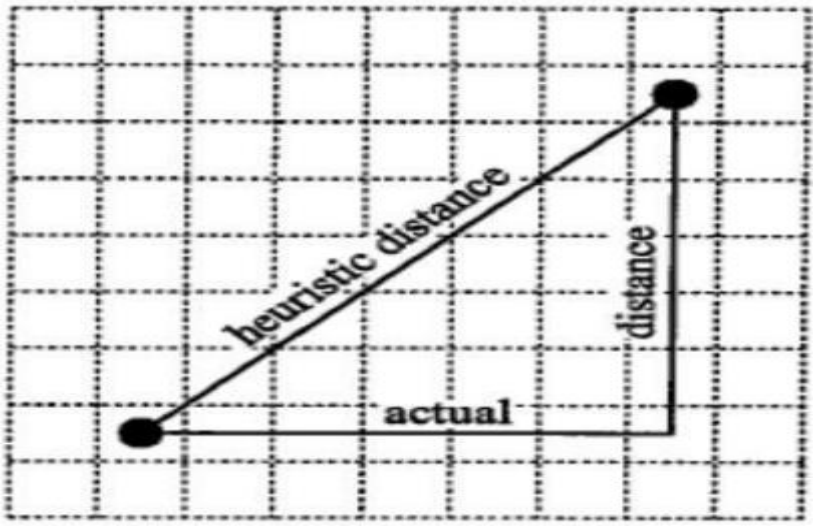


Figure 2.9: Montre l'heuristique A* h(n) (en utilisant la distance euclidienne).

Selon la structure des cellules de la grille illustrée à **la figure 2.10**. La diagonale les bords ont un coût c de 1.4, tandis que les bords horizontaux et verticaux ont un coût c de 1. Par exemple, cx1, x5 = 1.4 et cx1, x2 = 1. Si x9 et x4 sont des obstacles, alorscx1, x9 = ∞ et cx1, x4 = ∞.

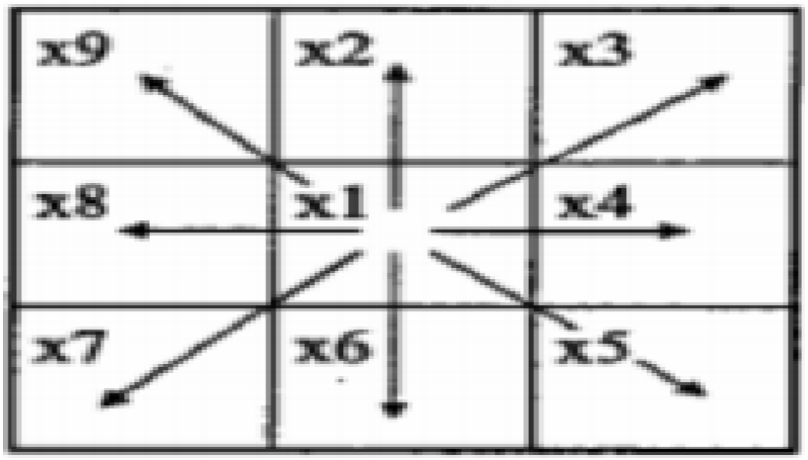


Figure 2.10: Montre les déplacements possibles entre les cellules de la grille.

Exemple de planification du chemin avec l'algorithme A*

Nous démontrons l'algorithme A en utilisant comme fonction heuristique le nombre de pas de la cellule de but G à la cellule (x, y) sans tenir compte des obstacles. Cette fonction

peut être précomputée et reste disponible tout au long de l'exécution de l'algorithme. Pour la carte quadrillée de la figure.2.11, la fonction heuristique est représentée à la figure 2.12.a. Dans les diagrammes, nous garderons la trace des valeurs des trois fonctions f, g, h en les affichant dans différents coins de chaque cellule.

La figure 2.12 b montre la carte quadrillée après deux étapes de l'algorithme A. Les cellules (3, 1) et (3, 0) reçoivent le même coût f : l'une est plus proche de S (par le nombre de pas comptés) et l'autre est plus proche de G (par l'heuristique), mais les deux ont le même coût de 7.L'algorithme doit maintenir une structure de données des cellules ouvertes, les cellules qui n'ont pas encore été étendues. Nous utilisons la notation (r, c, v), où r et c sont la ligne et la colonne de la cellule et v est la valeur f de la cellule. Chaque fois qu'une cellule ouverte est étendue, elle est supprimée de la liste et les nouvelles cellules sont ajoutées. La liste est ordonnée de sorte que les cellules ayant les valeurs les plus faibles apparaissent en premier, ce qui permet de décider facilement quelle cellule développer ensuite. Les trois premières listes .

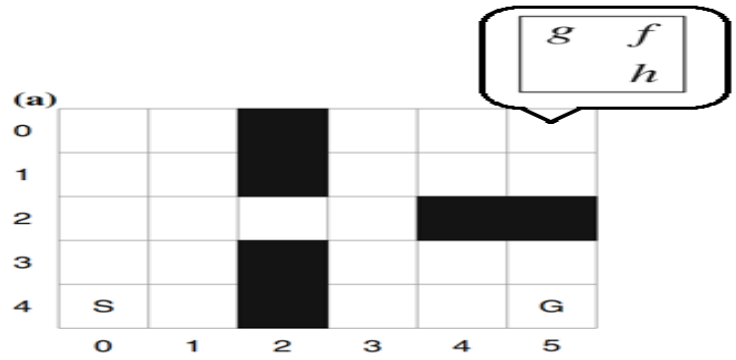


Figure 2.11 : Carte quadrillée après deux étapes de l'algorithme A*.

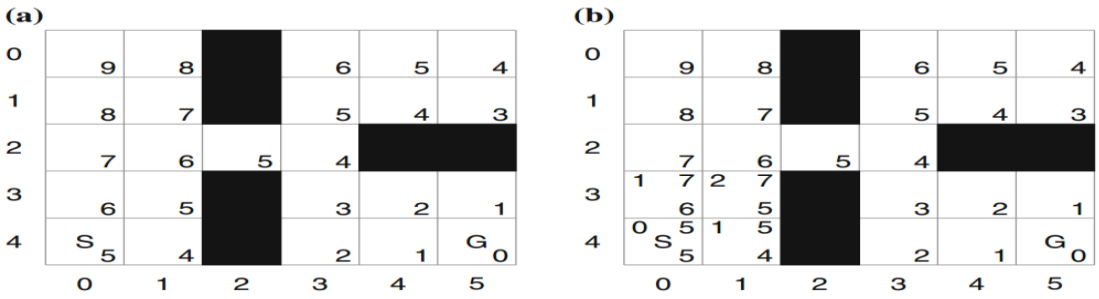


Figure 2.12 : a) Fonction heuristique. b) Les deux premières itérations de l'algorithme A*.

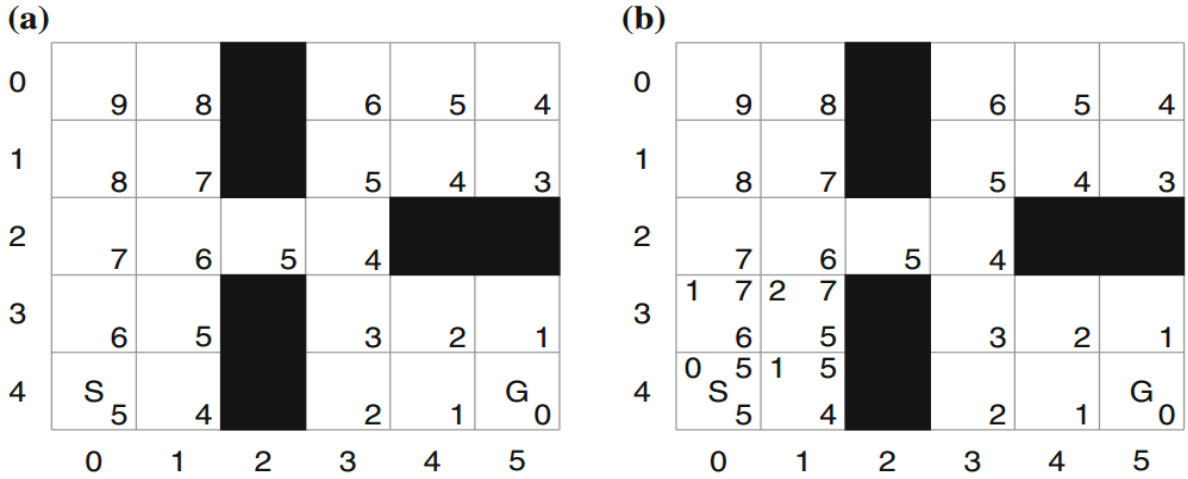


Figure 2.13 : a) L'algorithme A* après 6 étapes. b) L'algorithme A* atteint la cellule de but et trouve un chemin le plus court.

(4, 0, 5)(4, 1, 5), (3, 0, 7) (3, 0, 7), (3, 1, 7).

La figure 2.13.a montre la carte quadrillée après six étapes. On peut le voir en regardant les valeurs g dans le coin supérieur gauche de chaque cellule. La liste actuelle des cellules ouvertes est :***(3, 3, 9), (1, 0, 11), (1, 1, 11), (1, 3, 11).***

L'algorithme A*choisi de développer la cellule (3, 3, 9) avec le plus bas f. Les autres cellules de la liste ont une valeur f de 11 et sont ignorées pour le moment. En continuant (Figure. 2.13.b), la cellule de but est atteinte avec la valeur 9 de f et un chemin le plus court en gris est affiché. La dernière liste avant d'atteindre le but est :***(3, 5, 9) ; (4, 4, 9), (1, 0, 11), (1, 1, 11), (1, 3, 11).***

Peu importe lequel des nœuds avec la valeur 9 est choisi : dans les deux cas, l'algorithme atteint la cellule de but (4, 5, 9). Toutes les cellules en haut à droite de la grille ne sont pas explorées parce que la cellule (1, 3) a la valeur f 11 et ne sera jamais la plus petite valeur. l'algorithme A* explorait seulement 17 cellules.

L'algorithme A* est complet et optimal. En dehors de cela, la complexité de temps de cet algorithme est O $(nlogn)$, il peut être utilisé pour résoudre des problèmes très complexes. Toutefois, la précision de l'algorithme heuristique utilisé pour calculer la valeur $h(n)$ dépend dans une large mesure de la vitesse d'exécution de A*.

2.4.3 Algorithme Rapid-Exploring Random Tree (RRT)

Initialement proposée par Lavalle **[7] ,** Rapid-Exploring Random Tree (RRT) c'est un Arbre aléatoire d'exploration rapide, a été créé par Un espace haute dimension non confus convient à cet algorithme. L'idée principale est d'explorer la partie inexplorée en échantillonnant les points et de « tirer » graduellement l'arbre de recherche près d'eux

Comment fonctionne un RRT standard?

Tout d'abord, un point de départ, s et un point de but, g tentatives de se connecter directement. S'il n'y a pas de collision, le chemin est trouvé. Sinon, un nouveau point doit être créé au hasard dans la zone spécifique, répétez cette étape si le nouveau point est dans l'obstacle. Ensuite, le nouveau point essaie de se connecter au point le plus proche de l'arbre. En outre, un nouveau point est créé au hasard si le nouveau point ne peut pas être connecté au point de l'arbre. D'autre part, le nouveau nœud devient le nœud de l'arborescence si le nouveau nœud est correctement connecté. Enfin, vérifiez si le nœud peut se connecter au nœud cible. Si une collision se produit, un nouveau point sera créé au hasard. Si les nœuds peuvent se connecter, cela signifie que le chemin a été trouvé **[13].**

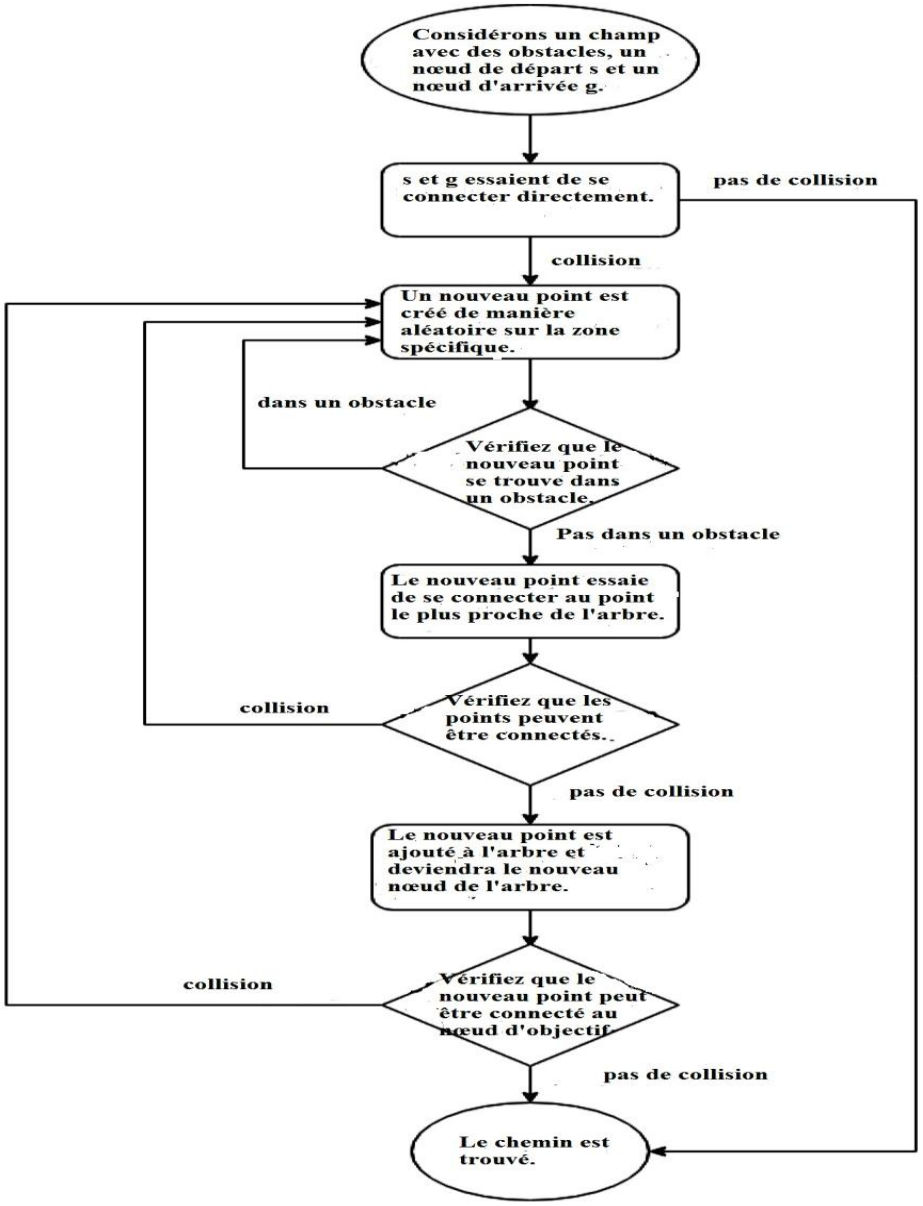


Figure 2.14 : Organigramme de l'algorithme RRT **[13].**

2.5 Navigation Locale

La navigation locale concerne l'identification des conditions dynamiques dans un environnement statique ou dynamique et l'établissement de relations de position entre les différents éléments **[12].**

Dans ces conditions, il n'est pas possible de déterminer un mouvement complet jusqu'au but avant que le robot ne commence à se déplacer. La planification correspond alors à gérer dynamiquement, les informations relatives à l'environnement. En effet, le principe est de calculer uniquement le mouvement à appliquer au prochain pas temporel et c'est à partir des données capteurs recueillis par le système robotique à chaque instant. Par conséquent, la représentation de l'environnement est construite au fur et à mesure du déplacement

du robot **[11].**

2.6 Planification de chemin locale

2.6.1 Évitement des obstacles dynamique

La navigation pour les véhicules autonomes peut être divisée en deux tâches : la tâche la plus importante est de trouver un chemin entre un lieu de départ et un lieu de but. Avant le développement des systèmes informatiques modernes, pour trouver un moyen, vous avez dû étudier une carte ou demander des directions. Il existe maintenant des applications pour Smartphones qui peuvent prendre un point de départ et un point de destination et calculer un chemin entre les deux emplacements. Lorsque l'application reçoit des données sur l'état du trafic en temps réel, elle peut suggérer le moyen de se rendre à votre destination dans les plus brefs délais. Les chemins peuvent être calculés hors ligne, si vous avez un système GPS qui peut déterminer votre emplacement actuel, l'itinéraire peut être trouvé et mis à jour en temps réel pour refléter les conditions changeantes. Trouve en temps réel et mis à jour pour tenir compte de l'évolution des conditions. Une voiture autonome doit également effectuer la tâche de niveau inférieur d'adapter son comportement à l'environnement : s'arrêter pour un piéton dans un passage pour piétons, tourner aux intersections, éviter les obstacles dans la route, et ainsi de suite. Bien qu'il soit possible de trouver un sentier de niveau élevé une fois avant le voyage (ou toutes les quelques minutes), la tâche de niveau bas consistant à éviter les obstacles doit être effectuée fréquemment, parce que le robot ne sait jamais le mouvement futur de l'obstacle dynamique **[11].**

2.6.2 Approche de fenêtre dynamique

Cette méthode, proposée par Dieter Fox et al **[15] ,** est une approche en temps réel pour l'évitement d'obstacles qui opère dans l'espace de contrôle du robot. Elle détermine les plages de vitesses possibles du robot, c'est-à-dire celles qui ne conduisent pas à des collisions. Une fois que l'espace de recherche est calculé, les commandes envoyées au robot résultent de la maximisation d'une fonction de coût spécifique dans cette zone. Cette fonction peut viser à minimiser le temps de parcours, à maximiser la vitesse ou encore à minimiser l'énergie dépensée. Ainsi, en se basant sur la perception locale de l'environnement, cet algorithme permet de sélectionner un couple (v, w) de vitesses de translation et de rotation du robot qui satisfait les différentes contraintes, y compris l'évitement des obstacles.

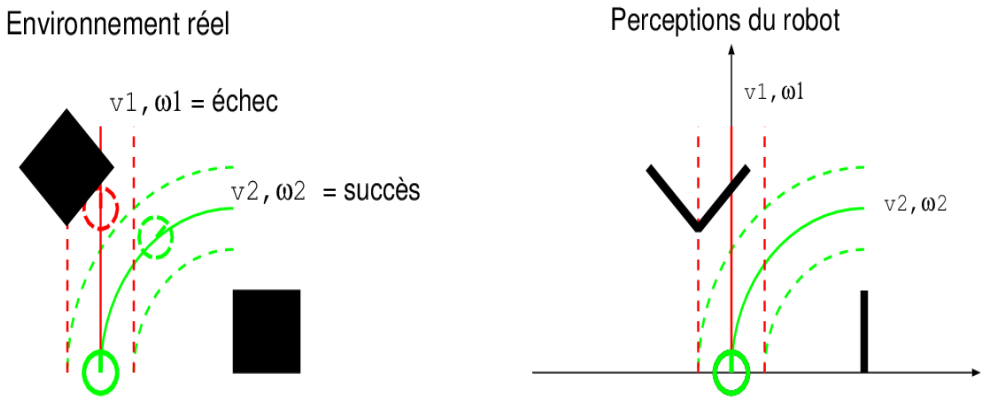


Figure 2.15 : Évitement d'obstacles pour l'approche de fenêtre dynamique.

Cependant, bien que cette méthode prenne en compte les contraintes cinématiques du robot, son implémentation dans un contexte multi-robot est difficile en raison de son manque de flexibilité. De plus, seules les positions actuelles des obstacles étaient prises en compte, sans tenir compte de leurs mouvements. Par conséquent, il était très difficile de naviguer dans un environnement dynamique en utilisant cette approche **[11].**

2.7 Conclusion

Nous avons déjà connu les aspects les plus importants de la navigation par robot mobile, ce sont des catégories intégrées de navigation globale et de navigation locale et les types d'environnements dans lesquels il peut se déplacer et explorer. Nous avons également appris les méthodes de numérisation des environnements afin que l'ordinateur puisse les lire et les traiter pour des simulations. La difficulté de cela varie selon l'environnement et la nature des objets et des obstacles qui y sont présent.

Chapitre 03

Système de navigation

3.1 Introduction

Dans le domaine de la robotique mobile, le LiDAR bidimensionnel joue un rôle essentiel en offrant une solution abordable et fiable pour la perception de l'environnement. Grâce à la technologie LiDAR, un robot mobile peut scanner son environnement en utilisant des lasers pour mesurer les distances et générer une représentation précise de celui-ci en deux dimensions. Cette capacité de cartographie et de perception de l'environnement permet au robot de se déplacer de manière autonome, en évitant les obstacles et en planifiant des trajets efficaces. En outre, l'utilisation d'un LiDAR bidimensionnel permet de réduire les coûts de développement et de production par rapport à un LiDAR tridimensionnel, tout en maintenant un niveau élevé de précision et de performance. Ainsi, réaliser un robot mobile utilisant un LiDAR bidimensionnel offre une solution pratique et économique pour une variété d'applications allant de la robotique domestique à l'automatisation industrielle

Après avoir étudié dans le chapitre précedent la navigation des robots mobiles,dans ce chapitre, l'objectif de notre projet peut se résumer en :

-Construire un environnement robot simulé pour un environnement réel

-Faire la navigation globale dans un environnement avec des obstacles fixes.

-Réaliser la navigation locale dans un environnement avec des obstacles dynamiques.

3.2 Conception générale de notre système de navigation

Il y a quatre étapes clés dans la mise en œuvre de la navigation d'un robot mobile utilisant un LiDAR bidimensionnel :

Création de l'environnement

Dans cette phase, nous utilisons le logiciel ROS attaché à l'émulateur Gazebo et l'outil de vision Rviz pour créer un environnement virtuel dans lequel le robot va évoluer. Cela implique la modélisation des objets, des murs, des obstacles et des caractéristiques de l'environnement, afin de reproduire fidèlement le monde réel dans lequel le robot sera déployé.

Cartographie

Une fois que l'environnement est créé, le robot commence à explorer et à cartographier l'environnement à l'aide du LiDAR bidimensionnel. Le LiDAR envoie des faisceaux laser et mesure les distances pour créer une carte précise en deux dimensions de l'environnement. Cette carte permet au robot de comprendre la disposition de l'environnement, y compris la position des obstacles et des structures, ce qui est essentiel pour la navigation autonome.

Localisation

Après avoir construit la carte de l'environnement, le robot doit être capable de se localiser précisément à l'intérieur de cet environnement. La localisation implique de déterminer en permanence la position et l'orientation du robot par rapport à la carte créée. Cela peut être réalisé en utilisant des techniques telles que l'odométrie, qui mesure les déplacements du robot, ou en combinant les données du LiDAR avec d'autres capteurs tels que des caméras ou des capteurs inertiels.

Planification de trajectoire et évitement d'obstacles

Une fois que le robot est correctement localisé, il peut planifier une trajectoire pour atteindre sa destination souhaitée. La planification de trajectoire consiste à trouver le meilleur chemin à suivre en tenant compte de la carte de l'environnement et de la position actuelle du robot. De plus, le robot doit être capable de détecter et d'éviter les obstacles sur son chemin en utilisant les informations fournies par le LiDAR bidimensionnel. L'approche de fenêtre dynamique est souvent utilisée pour prendre des décisions de navigation en temps réel et éviter les collisions avec les obstacles imprévus. En combinant ces différentes phases, la création de l'environnement, la cartographie, la localisation et la planification de trajectoire

avec l'évitement d'obstacles, on peut réaliser un robot mobile utilisant un LiDAR bidimensionnel capable de naviguer de manière autonome et sûre dans son environnement.

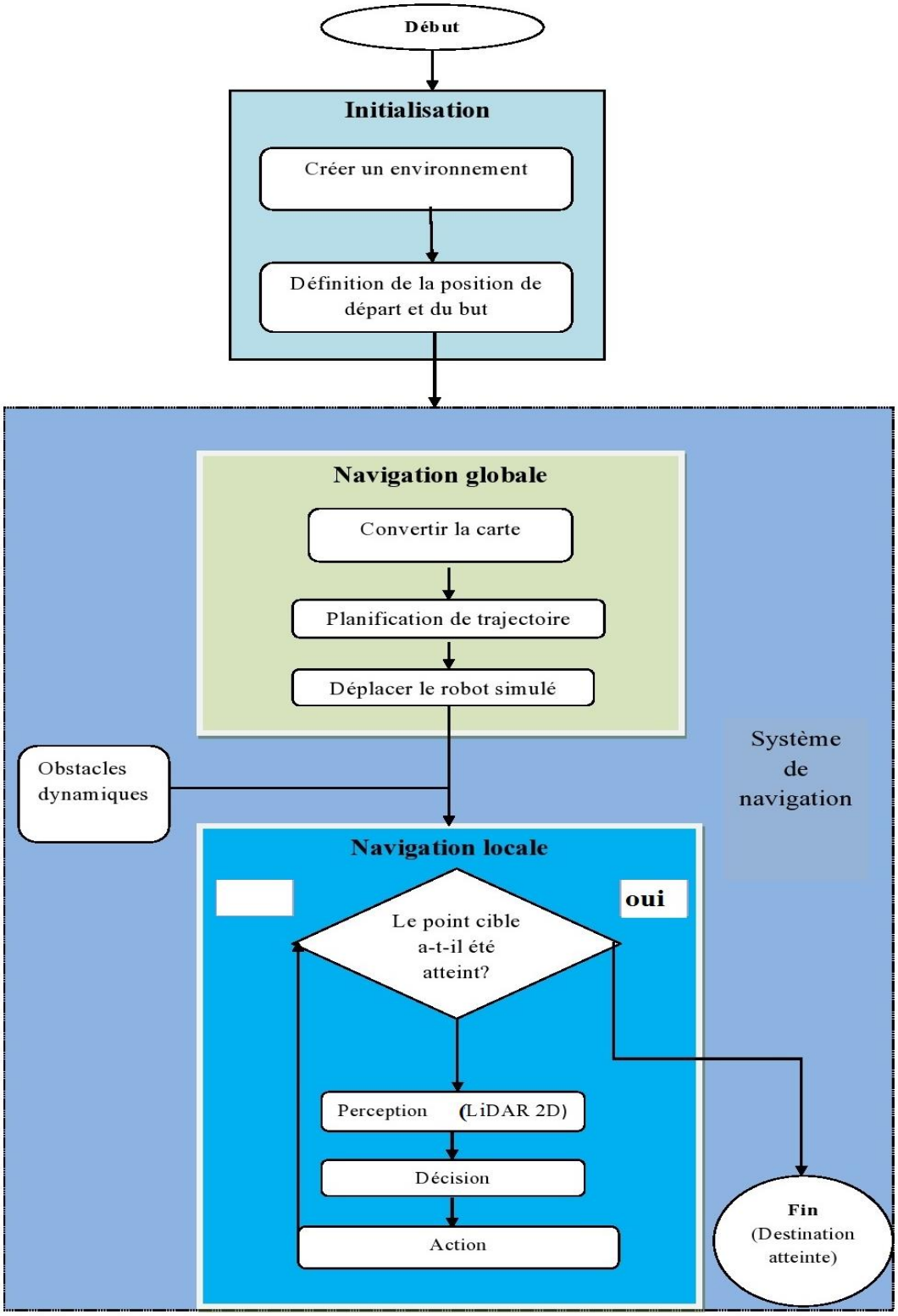


Figure 3.1 : Organigramme de système de navigation.

3.3 Notions de base

Pile de navigation

La pile de navigation (en anglais : navigation stack) fait référence à une gamme de composants logiciels qui permettent la navigation autonome de robots mobiles. Il offre des fonctionnalités (cartographie, localisation, planification des chemin et évitement des obstacles) qui consistent en plusieurs nœuds travaillant ensemble pour atteindre une navigation indépendante.

le paquet move_base :

L'élément le plus important dans la pile de navigation, est responsable du mouvement physique du robot dans l'environnement, la base de déplacement peut comprendre des moteurs, des capteurs de positionnement (comme des encodeurs) et d'autres éléments mécaniques nécessaires pour permettre au robot de se déplacer..où les ordres planificateur globale (Il en va de même pourde planificateur locale)relatifs à l'itinéraire calculé sont appliqués du point de départ du robot au point cible Ils sont convertis en commandes physiques en roues de robot.

3.4 Conception détaillée

3.4.1 Création d'environemment

a) Robot et LiDAR 2D

Pour ajouter le robot à l'environnement , nous avons utilisé dans notre recherche un robot mobile qui simulé une voiture mobile .

Le robot est composé d'une boîte reliée à quatre roues et le capteur Hokuyo LIDAR 2D a été placé dans sa face supérieure et une caméra dans sa face avant ,conçu avec le logiciel de gazebo qui est un fichier. gazebo.

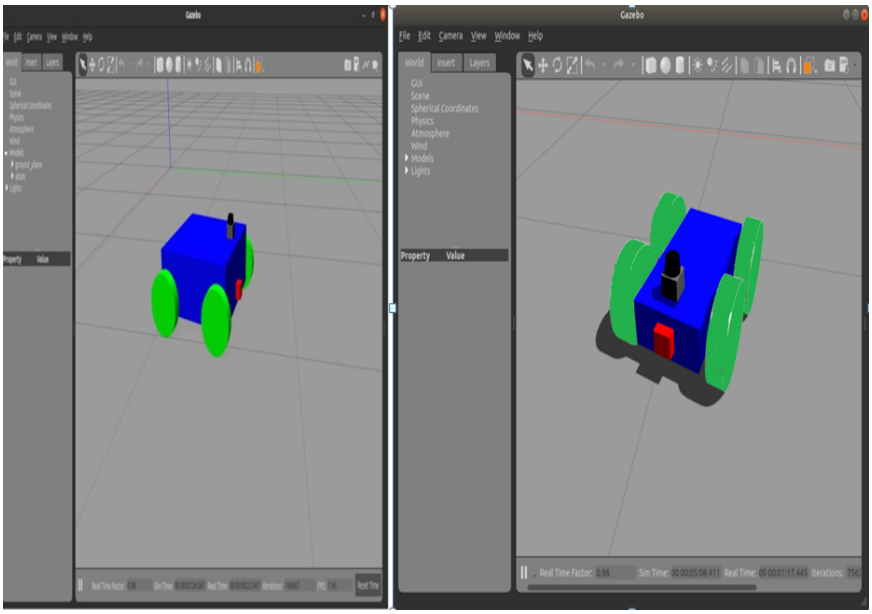


Figure 3.2 : Le modèle de robot dans notre projet.

b) Hokuyo LiDAR 2D

C'est un capteur bidimensionnel, mesure de portée jusqu'à 5,6 mètres, fournit un mécanisme de balayage et de conserver des mesures de distance à différents angles dans le champ de vision 180 degrés, largement utilisé dans les robots (cartographie des cartes, éviter les obstacles).

Figure 3.3 : LiDAR 2D Hokuyo**.**

c) Cartographie(mapping) avec LiDAR 2D

Pour obtenir une carte, nous appliquons le scane en utilisant le gmapping , il fonctionne sur un environnement inconnu, réalisant une localisation et une cartographie simultanées (SLAM). Il n a créé une carte de grille d'occupation 2D à l'aide de la pose du robot et des données de LiDAR .

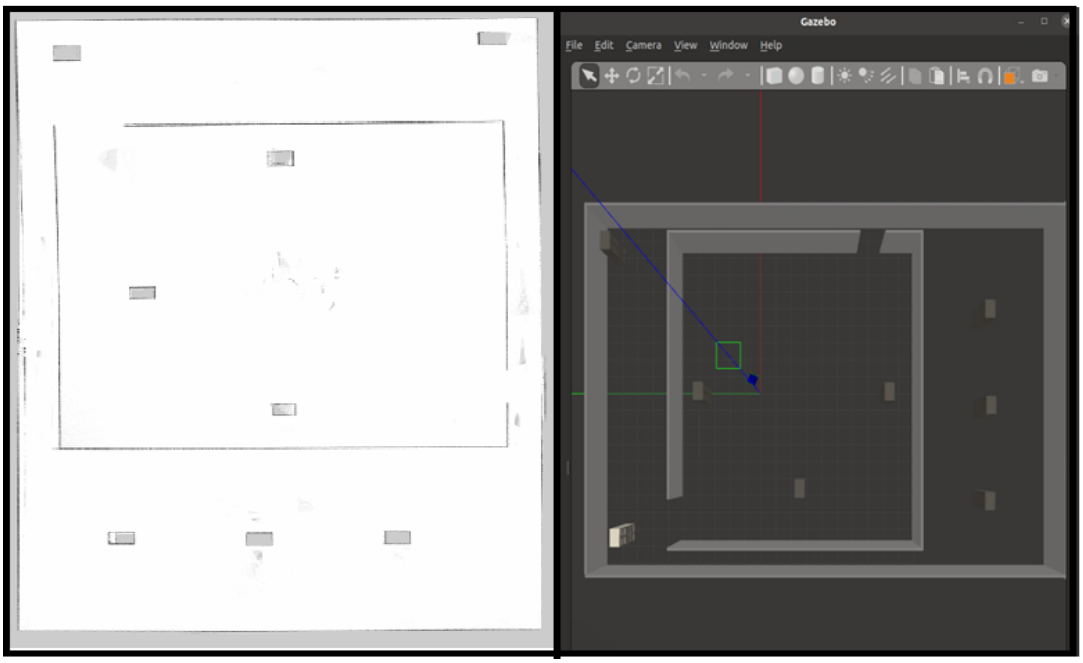


Figure 3.4:Cartographique d'environnement.

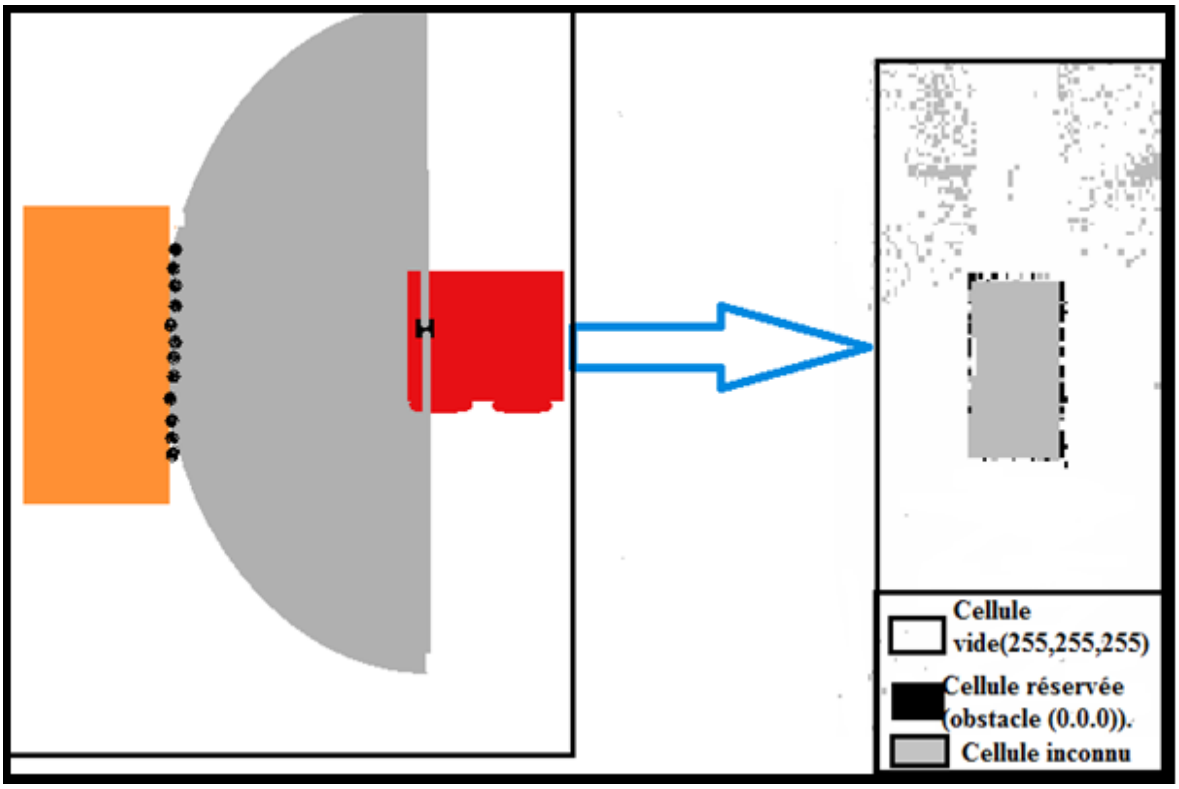


Figure 3.5 : Numérisation de la présence des obstacles

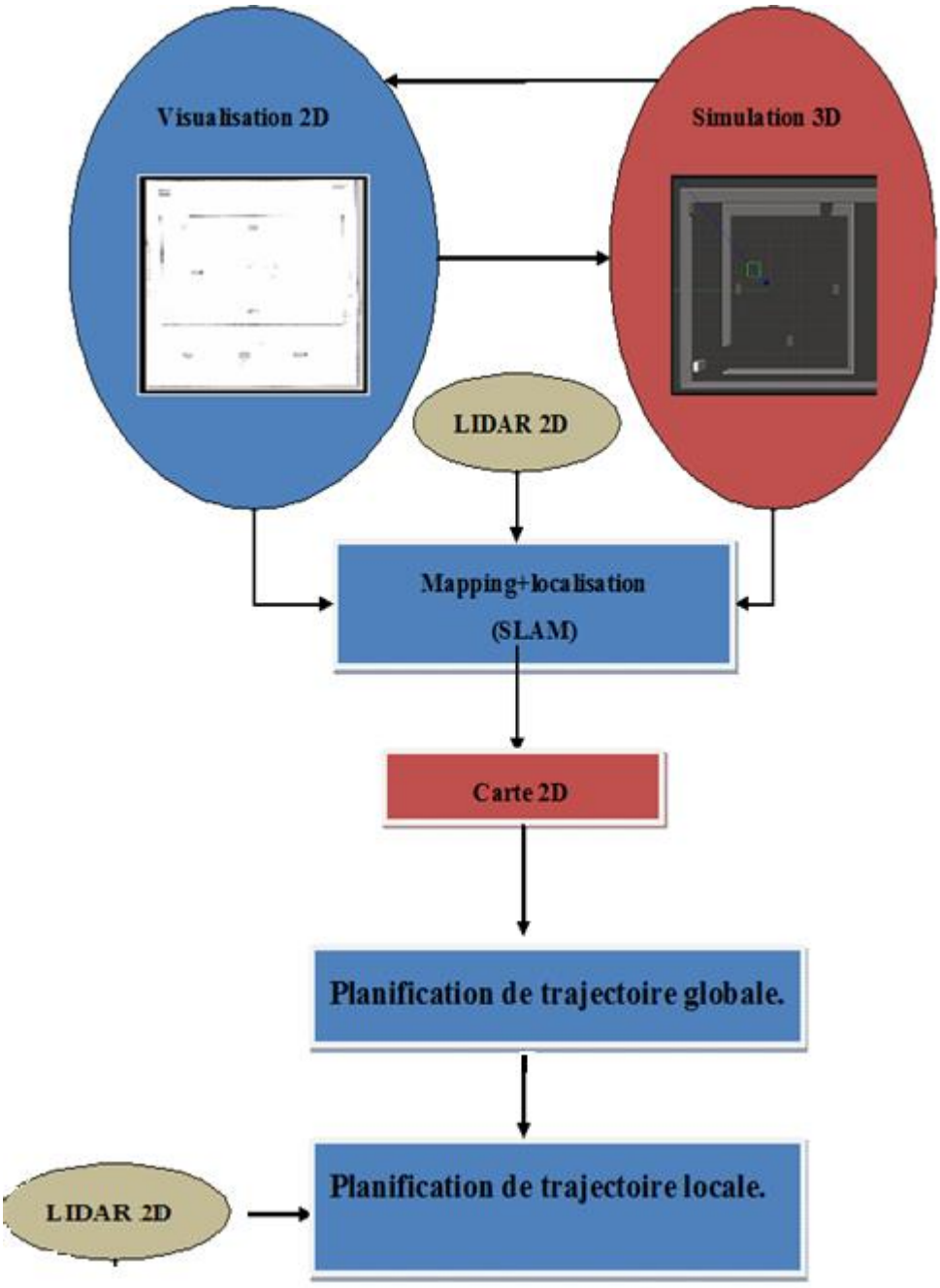


Figure 3.6 : Shéma de Cohérence des tâches et interventions de LiDAR 2D.

3.4.2 Navigation globale

Comme nous avons expliqué la navigation globale dans le chapitre précédent, il est de trouver le chemin du robot du point de départ au point cible en s'appuyant sur une carte static ,et pour y parvenir, il faut des étapes suivantes.

1-Réception du point cible indiqué par le client dans la carte précédemment obtenue.

2-Convertir la carte de l'environnement (map)en une carte bidimensionnelle (costmap2d) en forme de grille de coûts calculables contenant le coût de chaque cellule, par le paquet "map_server".

3-Envoyer les informations de costmap2d au planificateur global défini sur l'algorithme d'application A *, interfère avec ce paquet move_base ,

Où l'algorithme A* a été sélectionné pour ses améliorants liés à la faible complexité et de trouver le chemin optimal en peu de temps.

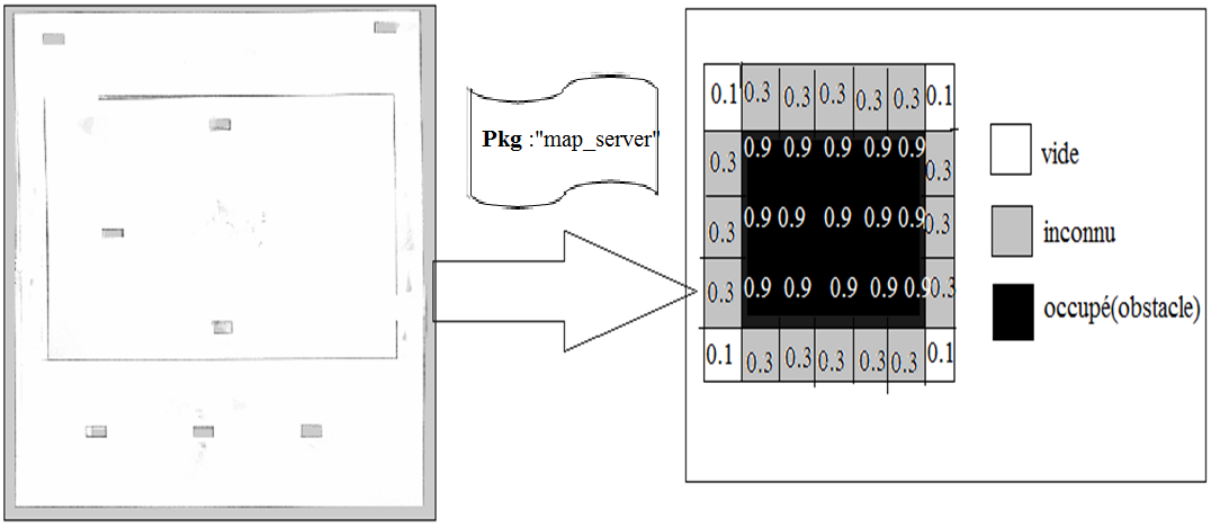


Figure 3.7 : Acquisition d'informations calculables avec l'algorithme A* à partir d'une carte d'environnement static.

Après avoir terminé le calcul de la piste pour mon monde, nous allons directement à l'outil de vision rviz où nous voyons la construction de la voie globale et un marquage en forme de ligne c'est le trajectoire globale .

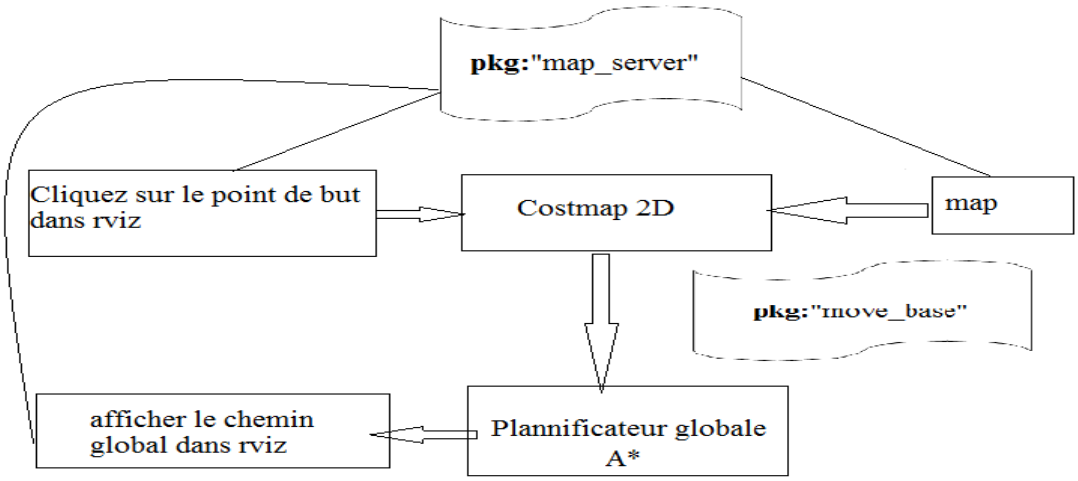


Figure 3.8 : Lancement de processus de navigation globale.

3.4.3 Navigation locale avec LiDAR 2D

La boucle de décision avec le LiDAR 2D dans un contexte de navigation robotique consiste à prendre des décisions en temps réel en se basant sur les informations perçues par le capteur LiDAR 2D. Cette boucle de décision permet au robot de réagir aux changements de l'environnement et d'ajuster sa trajectoire en conséquence

Perception

Dans le cercle de décision utilisant le LiDAR 2D, la première étape est la perception de l'environnement grâce aux données fournies par le capteur LIDAR 2D. Ces données permettent de détecter et de mesurer les distances des obstacles environnants, offrant une vision détaillée du terrain.

Décision

Une fois les données du LiDAR 2D traitées, le système prend des décisions en fonction des informations perçues. Par exemple, il peut décider de modifier la trajectoire du robot pour éviter une collision avec un obstacle détecté. Les décisions sont prises en temps réel et sont basées sur l'analyse des données du LiDAR 2D, afin de garantir une navigation sécurisée.

Action

La dernière étape du cercle de décision est la mise en œuvre des décisions prises. Le système agit en conséquence pour exécuter les actions nécessaires, telles que le contrôle des moteurs pour ajuster la vitesse, la direction ou l'accélération du robot. Les actions sont continuellement adaptées en fonction des nouvelles informations perçues par le LiDAR 2D, afin d'assurer une navigation précise et réactive.

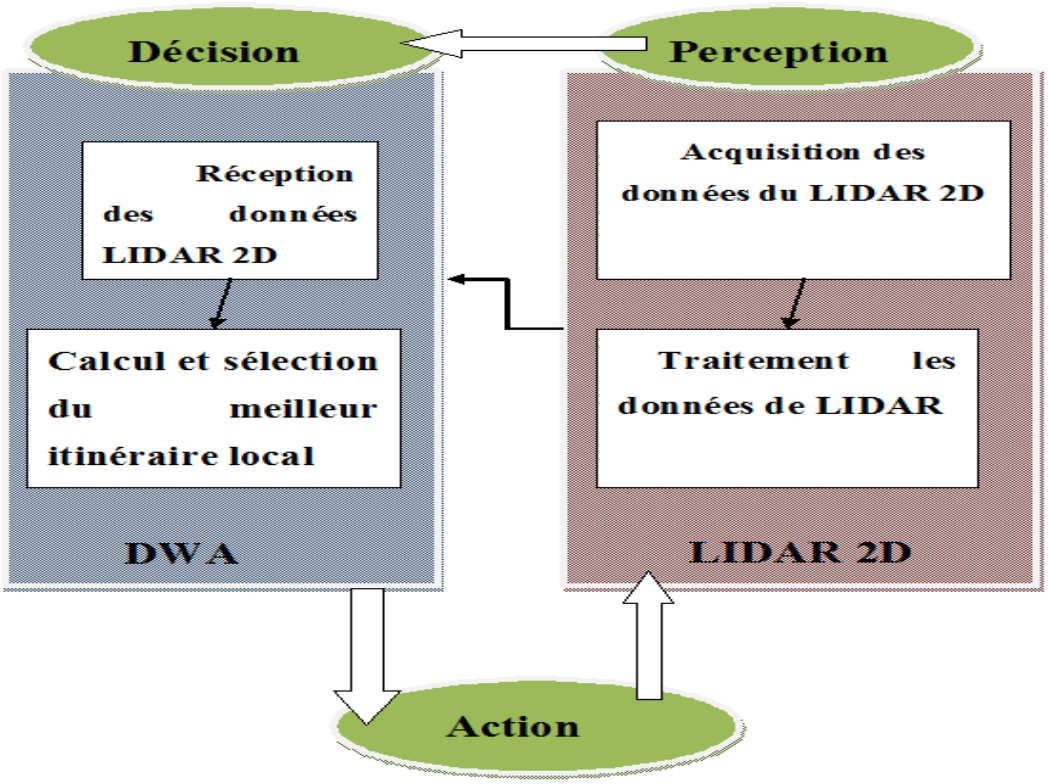


Figure 3.9 : La boucle de décision avec le LiDAR 2D.

a) Phase de Préparation et utilisation des donnéesLiDAR 2D

Lancer de faisceaux par LiDAR 2D

Le processus commence par le lancement de faisceaux laser par le capteur LiDAR 2D. Ces faisceaux sont envoyés dans différentes directions pour balayer l'environnement autour du capteur.

Production de données laser (angle, distance)

Le LiDAR 2D mesure le temps nécessaire aux faisceaux laser pour rebondir sur les objets et revenir au capteur. En utilisant cette information, il produit des données laser qui contiennent à la fois l'angle et la distance des objets détectés.

Obtenir un nuage de points (x, y)

Les données laser sont ensuite traitées pour obtenir un nuage de points en coordonnées cartésiennes (x, y). Chaque point représente un objet détecté dans l'environnement, avec sa position horizontale (x) et verticale (y) par rapport au capteur.

Filtration des points (x, y) avec z = 0

Pour éliminer les points du sol ou des surfaces planes, une étape de filtration est effectuée en fixant la valeur z à zéro. Cela permet de supprimer les points qui ne sont pas pertinents pour la navigation.

Conversion des points en données laser (angle, distance)

Les points filtrés en coordonnées (x, y) sont ensuite convertis en données laser, en récupérant l'angle et la distance correspondants. Cela permet de retrouver les informations originales des faisceaux laser émis par le LiDAR 2D.

Envoi des données laser aux paramètres de DWA

Enfin, les données laser obtenues sont transmises aux paramètres de Dynamic Window Approach (DWA), un algorithme de planification de trajectoire. Ces données permettent à DWA de calculer la meilleure trajectoire en tenant compte des obstacles détectés

et des caractéristiques de l'environnement, pour assurer une navigation sûre et efficace du robot mobile.

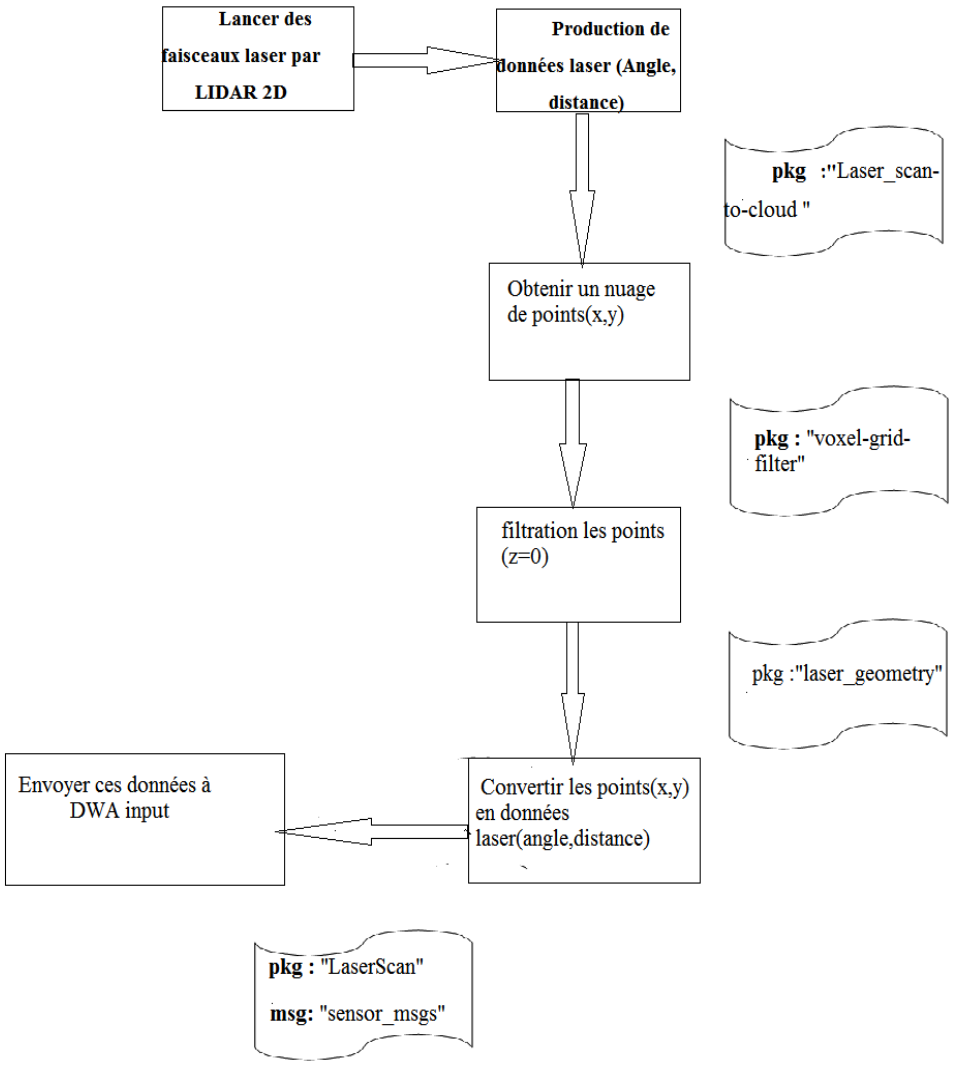


Figure 3.10: Utilisation des données LiDAR.

3.4.4 Phase d'évitement d'obstacle dynamique avec l'approche de fenêtre dynamique (Dynamic Window Approach)

L'algorithme DWA est un algorithme de planification de mouvement couramment utilisé en robotique qui permet à un robot de naviguer dans un environnement dynamique en évaluant différentes trajectoires de mouvement. Il prend en compte la cinématique du robot, les données du capteur pour calculer les meilleures commandes de mouvement possibles.

L'algorithme DWA utilise généralement les données des capteurs, comme les balayages LiDAR 2D, pour percevoir l'environnement environnant.

Les Étapes de l'algorithme DWA

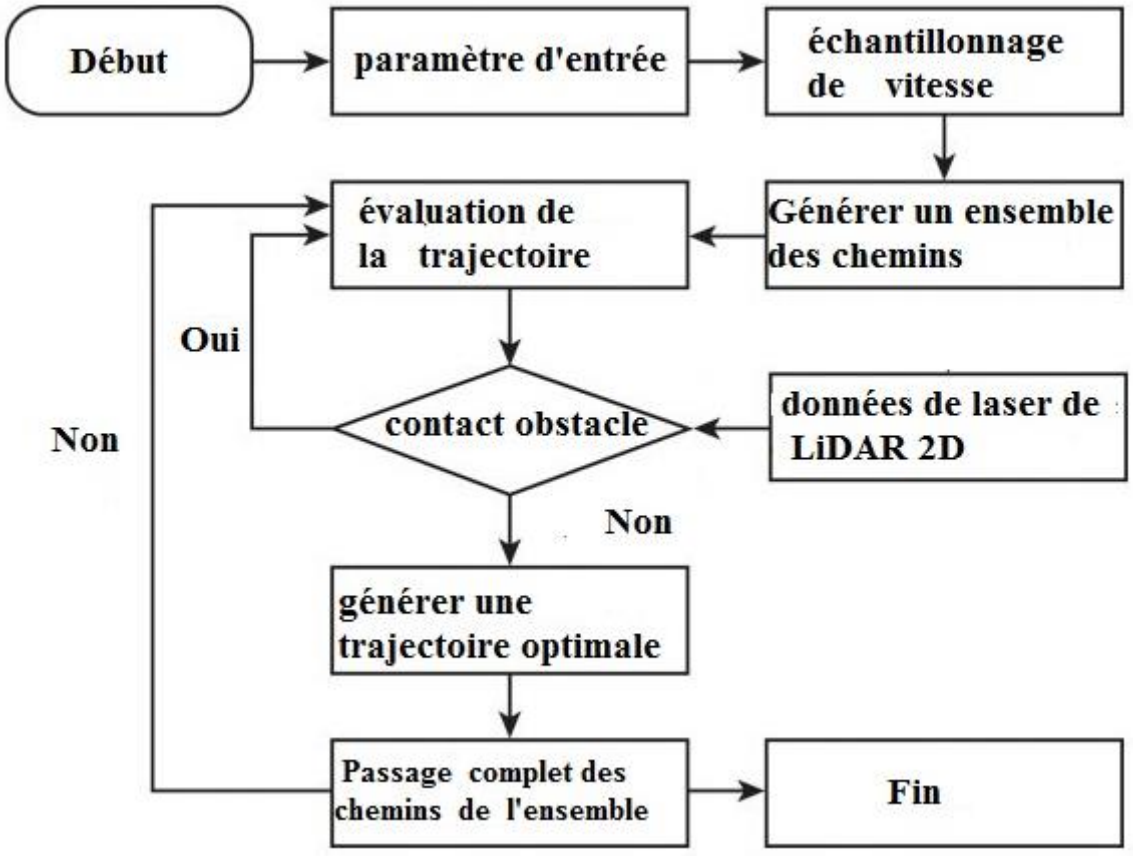


Figure 3.11: Organigramme de l'implémentation de DWA utilisant LiDAR2D.

La planification de trajectoire locale avec Dynamic Window Approach (DWA) comprend les étapes suivantes :

Paramètres d'entrée

Le premier pas consiste à définir les paramètres d'entrée, tels que la configuration actuelle du robot, sa vitesse maximale, son accélération maximale, etc. Ces paramètres sont utilisés pour calculer les vitesses possibles du robot dans l'espace de contrôle.

Échantillonnage de vitesses

Un ensemble de vitesses potentielles est échantillonné à partir des limites de vitesse et d'accélération définies. Cela crée un ensemble de vélocités possibles que le robot peut atteindre.

Génération d'un ensemble de chemins

À partir des vitesses échantillonnées, des chemins sont générés en combinant les positions successives du robot sur une période de temps spécifiée. Cela crée un ensemble de chemins candidats pour la planification de trajectoire.

Évaluation de la trajectoire

Chaque trajectoire candidate est évaluée en fonction de différents critères, tels que la distance par rapport aux obstacles, la proximité aux limites de l'environnement, la vitesse, etc. L'objectif est de trouver la trajectoire la plus sûre et la plus optimale possible.

Collision avec les obstacles (données de LiDAR 2D)

À l'aide des données du capteur LiDAR 2D, la présence d'obstacles est détectée sur chaque trajectoire. Les trajectoires qui entrent en collision avec les obstacles sont éliminées.

Génération d'une trajectoire optimale

Parmi les trajectoires restantes, celle qui présente la meilleure évaluation est choisie comme trajectoire optimale. Cette trajectoire est généralement celle qui minimise la distance aux obstacles, maximise la vitesse ou répond à d'autres objectifs spécifiques.

Passage complet des chemins de l'ensemble

Une fois la trajectoire optimale sélectionnée, le robot suit cette trajectoire en ajustant continuellement sa vitesse et son orientation jusqu'au point cible

Une fois que le robot a suivi la trajectoire jusqu'à sa destination , le processus de planification de trajectoire locale avec DWA est terminé.

.

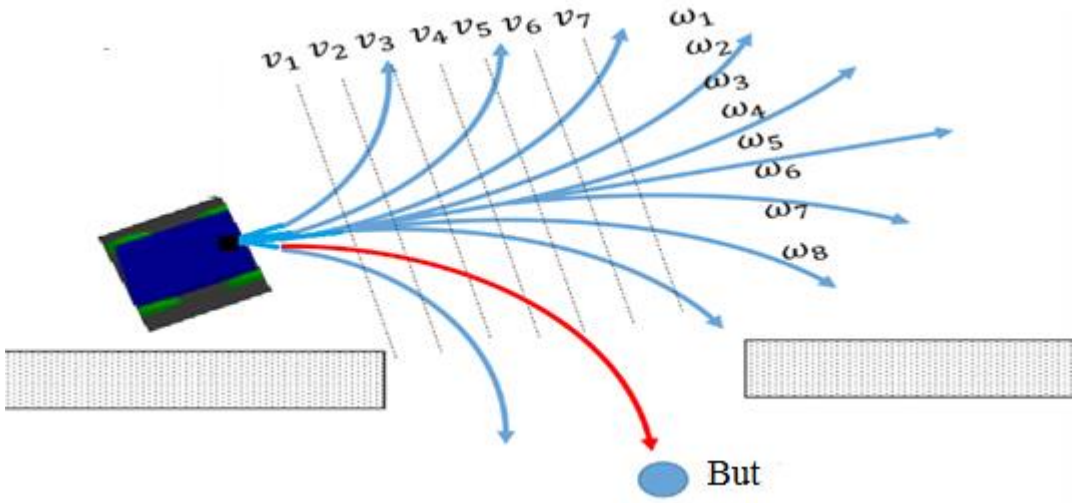


Figure 3.12:Contrôle de la vitesse dans la fenêtre dynamique.

3.5 Conclusion

En combinant la planification globale de piste avec le LiDAR 2D pour la planification locale, les robots peuvent naviguer de manière sûre et efficace dans des environnements dynamiques et imprévisibles. La planification globale fournit une vision d'ensemble du chemin à suivre, tandis que la planification locale utilise les informations en temps réel du LiDAR 2D inclus dans les calculs de fenêtre dynamique pour réagir et s'adapter aux changements survenant dans l'environnement. Cette relation entre les deux niveaux de planification permet une navigation autonome et fiable dans des conditions réelles et variées.

Chapitre 04

Implémentation et résultats

4.1 Introduction

Pour mettre en œuvre les étapes mentionnées dans le chapitre précédent, qui consistent à appliquer un système de navigation sur notre robot, nous avons besoin d'un ensemble de programmes spécialement développés à cet effet.

4.2 Logiciels de développement

ROS (Robot operating System) est un méta-système d'exploitation et un logiciel libre **[16]** .Il fournit les services contenant l'abstraction matérielle, la gestion des paquets, la mise en œuvre de la fonction couramment utilisée, le passage de messages entre les processus et ainsi de suite. En outre, c'est une plate-forme logicielle de robot et il fournit une variété d'environnements de développement dédiés au développement d'applications de robot. Le ROS a 3 caractéristiques principales :

La première caractéristique est la réutilisation du programme. Les utilisateurs peuvent se concentrer sur les fonctionnalités qu'ils veulent développer sans avoir à se soucier des autres fonctions. C'est parce qu'ils peuvent télécharger le paquet correspondant de ROS. En outre, ils peuvent partager leurs propres programmes afin que d'autres puissent les réutiliser.

La deuxième caractéristique est que ROS est un programme de communication. Chaque programme et fonction est programmé sous la forme des plus petites unités de processus exécutables, et chaque processus fonctionne indépendamment et échange des données systématiquement. Par conséquent, ceci est très utile pour trouver des erreurs, parce que les programmes qui sont divisés en fonctions minimales peuvent être débogués séparément.

Troisièmement, le ROS soutient une variété d'outils de développement.

Le ROS fournit outil de visualisation 2D, outil de visualisation 3D, simulateur 3D, outil de débogage et ainsi de suite. Ces outils logiciels nécessaires au développement de robots, qui tirent pleinement parti de la commodité du développement.

Figure 4.1 : Robot Operating System.

4.2.1 Gazebo

Gazebo est un simulateur dynamique 3D capable de simuler avec précision et efficacité des populations de robots dans des environnements intérieurs et extérieurs complexes. Bien que similaire aux moteurs de jeu, Gazebo offre la simulation physique à un degré de fidélité beaucoup plus élevé, une suite de capteurs, et des interfaces pour les utilisateurs et les programmes **[17].**

Figure 4.2 : Gazebo.

4.2.2 RVIZ(Visualization Tool ROS)

Est un logiciel de visualisation 2D pour robots, capteurs et algorithmes. Il vous permet de voir la perception du robot de son monde (réel ou simulé).Le but de rviz est de vous permettre de visualiser l'état d'un robot. Il utilise des données de capteurs pour essayer de créer une représentation précise de ce qui se passe dans l'environnement du robot **[6].**

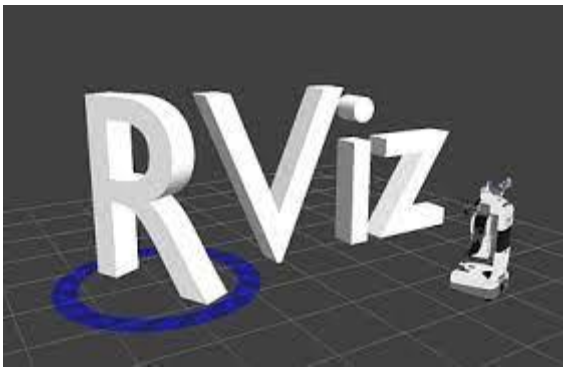


Figure 4.3: Rviz.

4.2.3 Langues de programmation

ROS traite de nombreux langages de programmation, dont les plus importants sont Python et c++, ainsi que des profils de conception xml avec différents formats.

Ceci est dû aux fonctionnalités supérieures et à la disponibilité d'un grand nombre de bibliothèques en particulier le langage Python

ROS utilise Python et C++ pour offrir une flexibilité maximale aux développeurs, en leur permettant de tirer parti des avantages de chaque langage pour différentes parties du développement robotique. Python est souvent préféré pour les tâches de haut niveau, le prototypage rapide et les interactions avec l'utilisateur, tandis que C++ est utilisé pour les parties critiques en termes de performances et l'accès direct au matériel. Cette approche hybride rend ROS polyvalent et adaptable à une grande variété d'applications robotiques.

Figure 4.4 : Les deux langues disponibles en ROS.

4.3 Les résultats

4.3.1 Les paramètres de LiDAR

```
<!-- hokuyo -->
<gazebo reference="hokuyo">
  <sensor type="ray" name="head_hokuyo_sensor">
    <pose>0 0 0 0 0 0</pose>
    <visualize>false</visualize>
    <update_rate>40</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>720</samples>
          <resolution>1</resolution>
          <min_angle>-1.570796</min_angle>
          <max_angle>1.570796</max_angle>
        </horizontal>
      </scan>
      <range>
        <min>0.10</min>
        <max>30.0</max>
        <resolution>0.01</resolution>
      </range>
      <noise>
        <type>gaussian</type>
        <!-- Noise parameters based on published spec for Hokuyo laser
             achieving "+-30mm" accuracy at range < 10m.  A mean of 0.0m and
             stddev of 0.01m will put 99.7% of samples within 0.03m of the true
             reading. -->
        <mean>0.0</mean>
        <stddev>0.01</stddev>
      </noise>
    </ray>
    <plugin name="gazebo_ros_head_hokuyo_controller" filename="libgazebo_ros_laser.so">
      <topicName>/scan</topicName>
      <frameName>hokuyo</frameName>
      <robotNamespace>/atom</robotNamespace>
    </plugin>
  </sensor>
</gazebo>
```

Figure 4.5 : Le fichier de Conception et paramètres de Lidar 2D .gazebo.

Les propriétés Lidar sont incluses dans la conception du robot, ces propriétés comprennent touts les caractéristique physiques de Hokuyo lidar 2d avec le nome de topic(/scan) pour appel dans Rviz .

4.3.2 Cartographige (mapping)

La cartographie est utilisée pour obtenir une carte de l'environnement que nous utilisons pour ce paquet gmapping,(Figure 4.6) nous notons lors de l'exécution de l'ordre ouvrir la fenêtre gazebo et rviz, puis avec les clefs du clavier (figure 4.7), nous pouvons contrôler le robot pour démarrer le processus de cartographie en utilisant lidar bidimensionnel que nous pouvons obtenir, après cela nous appelons le serveur pour enregistrer la carte sous le nom 'map' (Figure)

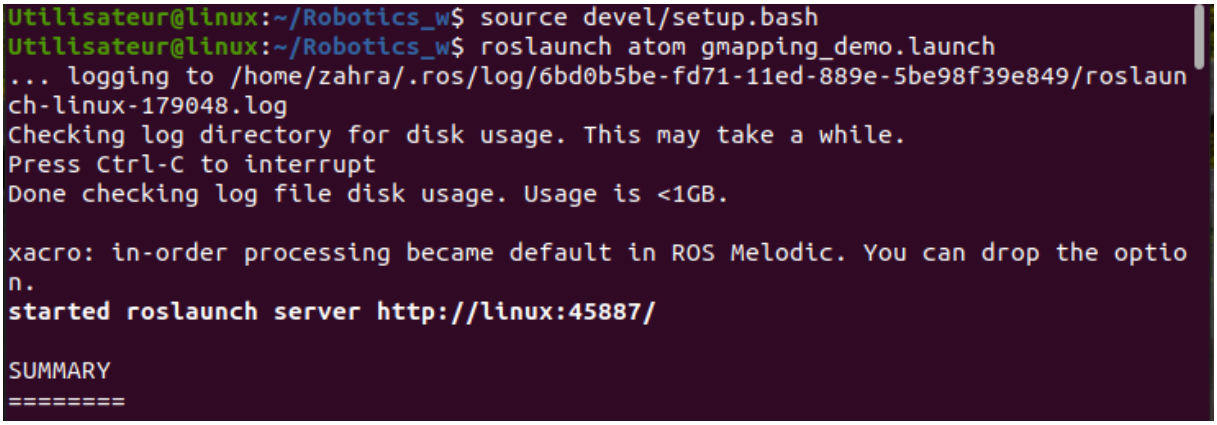


Figure 4.6 : Lancement du nœud gmapping.

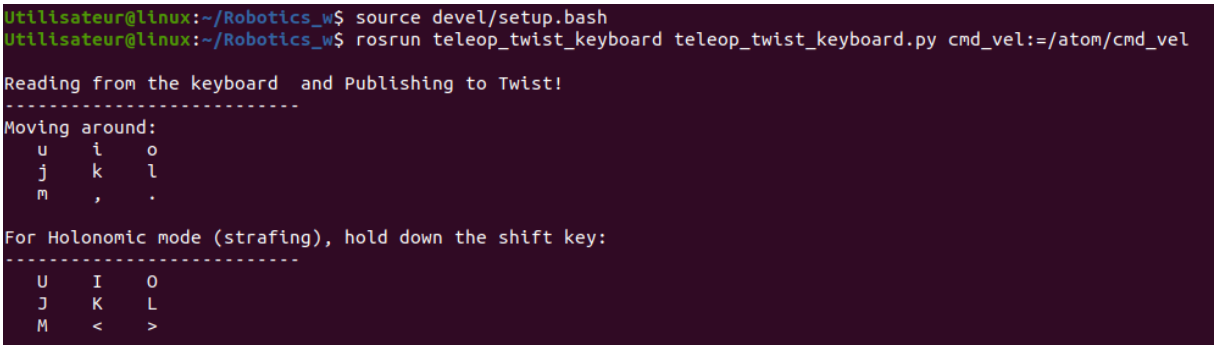


Figure 4.7 : Lancement du noeud de contrôle robot avec clefs de clavier.

```
Utilisateur@linux:~/Robotics_w$ rosrun map_server map_saver -f map
[ INFO] [1685290053.423077537]: Waiting for the map
[ INFO] [1685290053.703118825, 107.436000000]: Received a 1728 X 2784 map @ 0.0
10 m/pix
[ INFO] [1685290053.703177781, 107.436000000]: Writing map occupancy data to ma
p.pgm
[ INFO] [1685290053.885743415, 107.590000000]: Writing map occupancy data to ma
p.yaml
[ INFO] [1685290053.888148418, 107.593000000]: Done

Utilisateur@linux:~/Robotics_w$
```

Figure 4.8 : Enregistrer la carte par map_server.

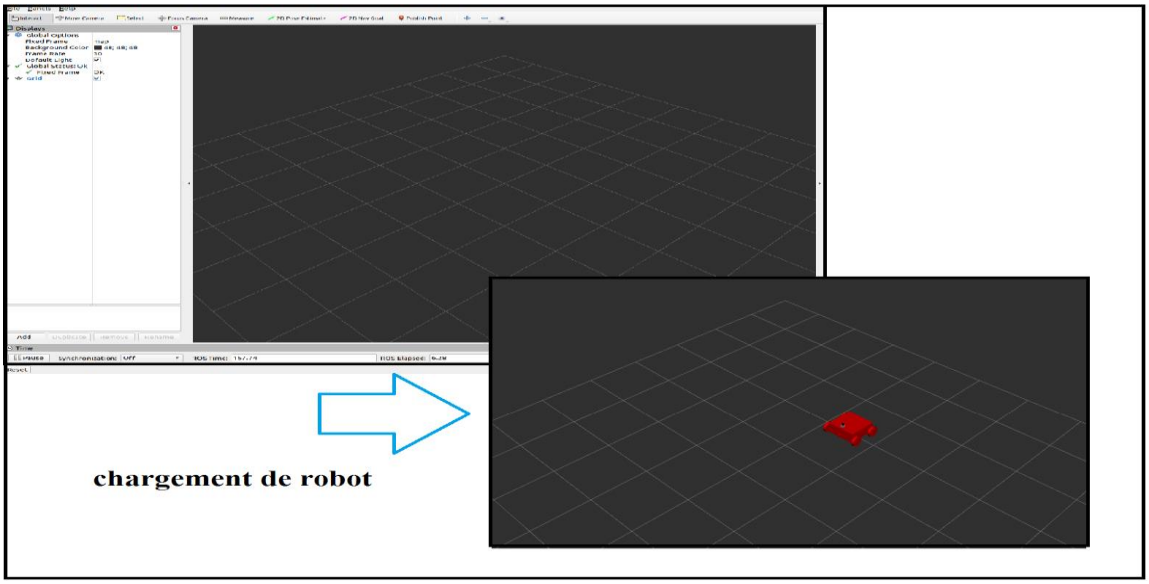


Figure 4.9 : Lancement la fenetre de Rviz et télécharger le modele de robot.

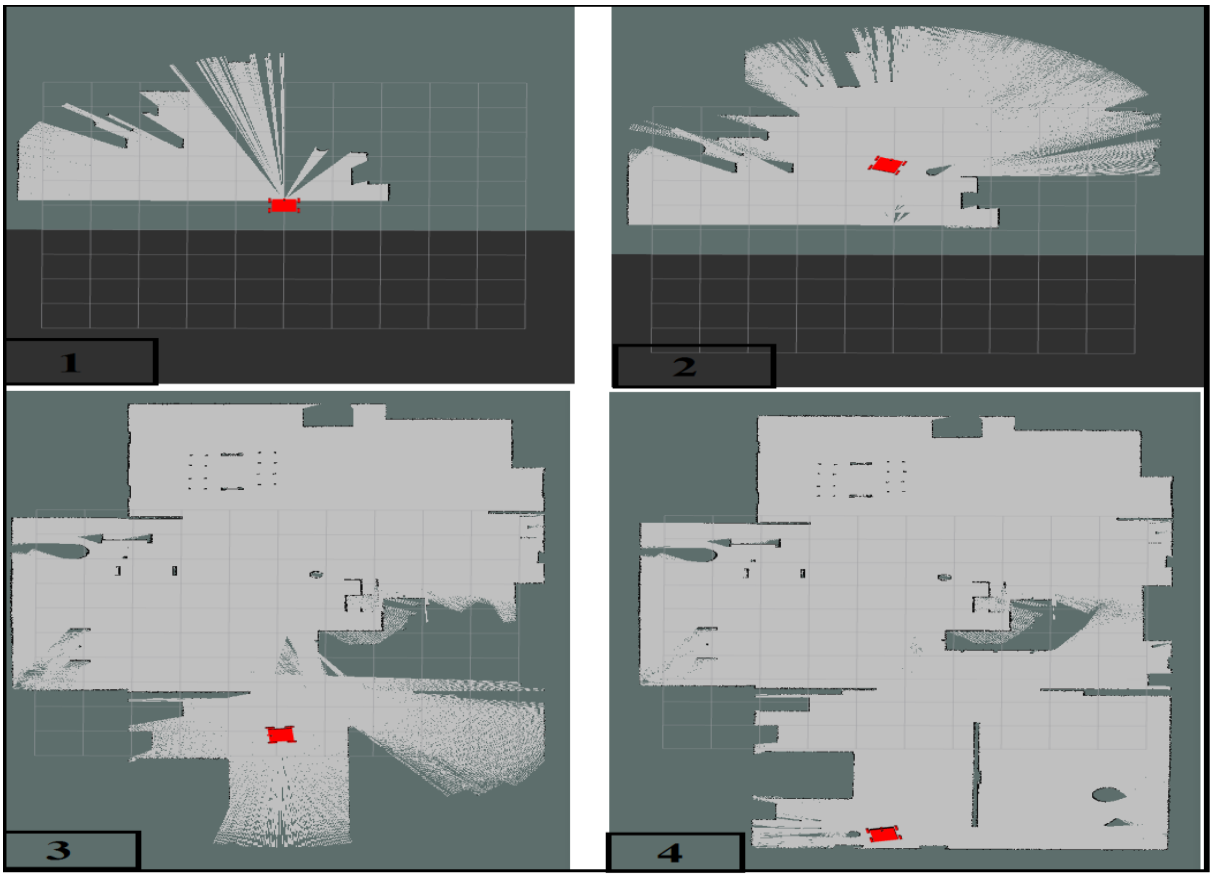


Figure 4.10 : Étapes du processus de cartographie (mapping) de l'environnement du robot.

4.3.3 Obtenir la carte d' environnement

```
image: map.pgm
resolution: 0.010000
origin: [-11.880000, -23.400000, 0.000000]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

Figure 4.11: montre le format du fichier YAML de la carte.

Dans ce cas, le fichier map.png est le chemin d'accès au fichier image. La résolution de la carte est de 0,01 (m/pixel). En d'autres termes, 1 pixel représente 0,01 mètre. Selon l'origine (x, y, lacet), l'axe des x est égal à -11.88, l'axe des y est égal à -23.40, et le lacet est

une rotation dans le sens antihoraire est égal à 0. . La négation de 0 signifie que la sémantique blanc/noir libre/occupation ne sera pas inversée. Le occupied_thresh de 0,65 signifie que si la probabilité d'occupation du pixel est supérieure à 0,65, le pixel est considéré comme entièrement occupé. Inversement, le seuil de 0.196 signifie que si la probabilité d'occupation d'un pixel est inférieure à 0,196, le pixel est considéré comme complètement libre.

4.3.4 Navigation globale

Après avoir obtenu la carte, nous lançons l'ordre qui permet l'ouverture du Gazebo de la fenêtre(Figure 4.9), puis nous commandons l'ouverture de la fenêtre rviz(4.10), puis nous portons le robot en sélectionnant le 'robot_description' et portons la carte en sélectionnant la 'map'.

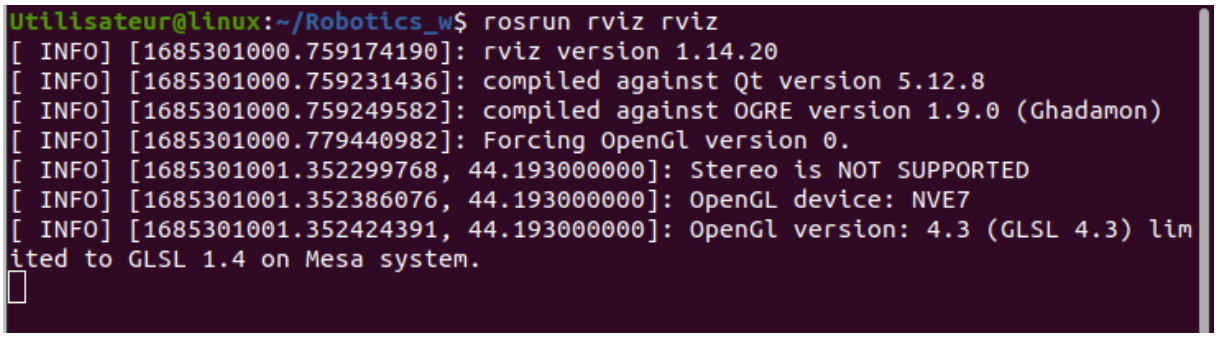


Figure 4.12 : Ouvrir la fenêtre rviz .

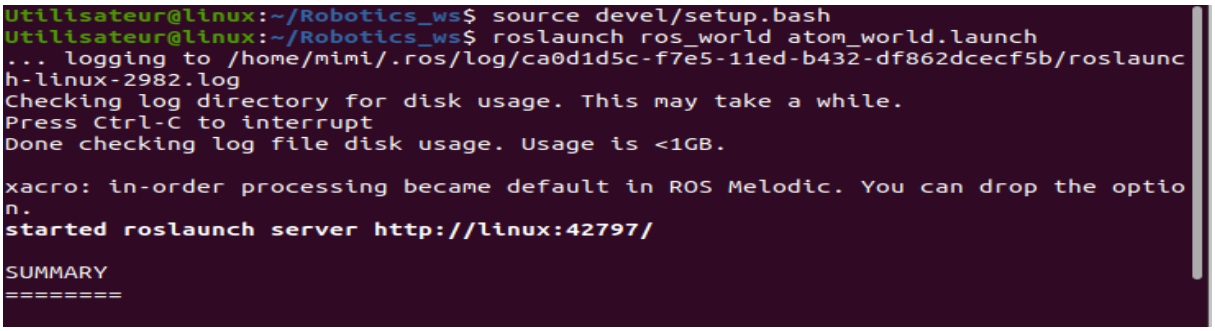


Figure 4.13 : Ouvrir l'environnement dans Gazebo.

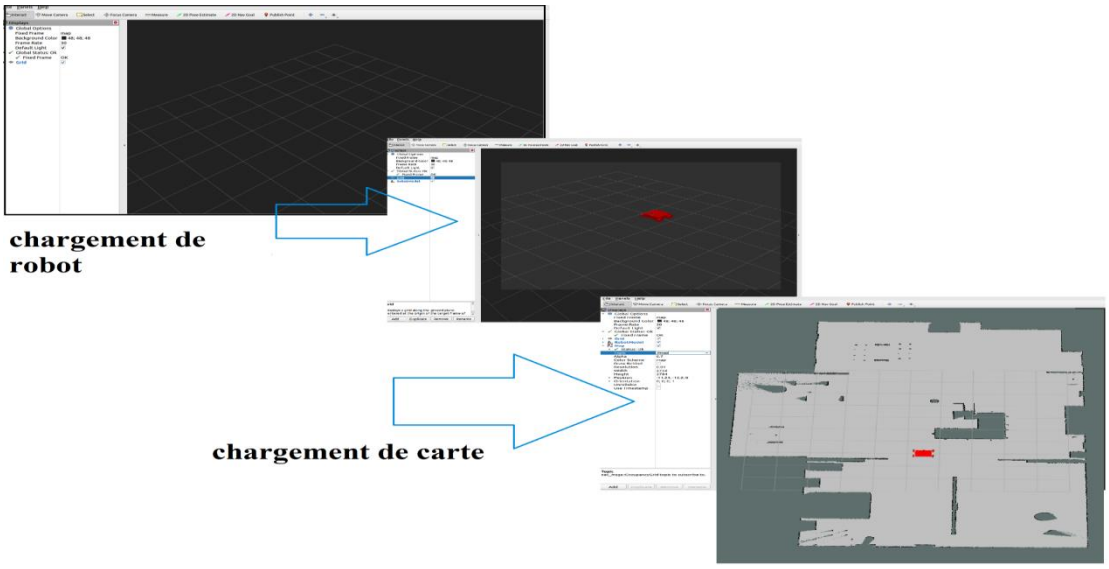


Figure 4.14 :L'ajout de la carte de l'environnement.

Maintenant, nous voulons calculer l'itinéraire, aller à "add" et ajouter "path" puis modifier "topic" **/move_base/NavfnROS/plan**, nous appelons ce "globalpath " pour faciliter l'accès à la mise en œuvre de la navigation globale à l'aide de l'algorithme A*.

Tout d'abord, appuyez sur le bouton 2d_nav_goal et sélectionnez la position que nous voulons que le robot atteigne.

On note d'abord l'émergence d'un point bleu est le point de départ et le second rouge, qui est la cible, le robot commence à calculer la route (prend du temps), est un line avec le couleur vert puis se déplace à travers elle,mais comment cela arrive-t-il ?

Nous allons maintenant lier la relation entre les éléments les plus importants afin de comprendre exactement comment déplacer le robot au point qui a été introduit, en général, nous allons plonger dans les détails de la navigation globale.Initialement, notre inclusion de position cible est égale à la requête **SimpleActionClient**, à recevoir directement par **SimpleActionServer** avec le type de message... Lorsque le noeud implémente cette requête move_base, le noeud **move_base** appartient au **paquet move_base** et puisqu'il est le paquet

```
shutdown_costmaps: false
max_planning_retries: 0
recovery_behavior_enabled: false
controller_frequency: 5.0
controller_patience: 3.0

planner_frequency: 0.0
planner_patience: 5.0

oscillation_timeout: 0.0
oscillation_distance: 0.2


base_global_planner: "navfn/NavfnROS"
```

Figure 4.15 : Les paramétres de fichier Move_base_params.yaml.

responsable du déplacement du robot de son emplacement actuel vers le site cible, nous observerons déjà sa transition.

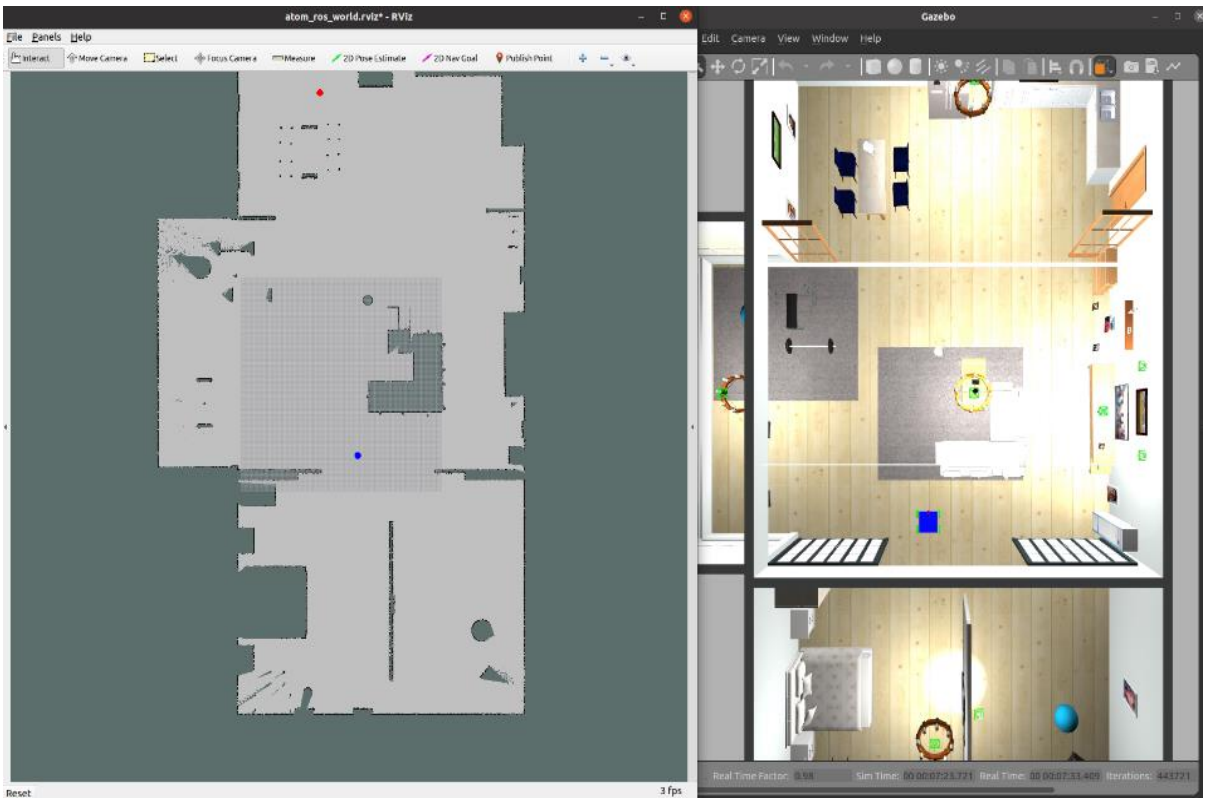


Figure 4.16 Disposition de la planification de chemin global (le point bleu est la position principale du robot et le point rouge est la cible)

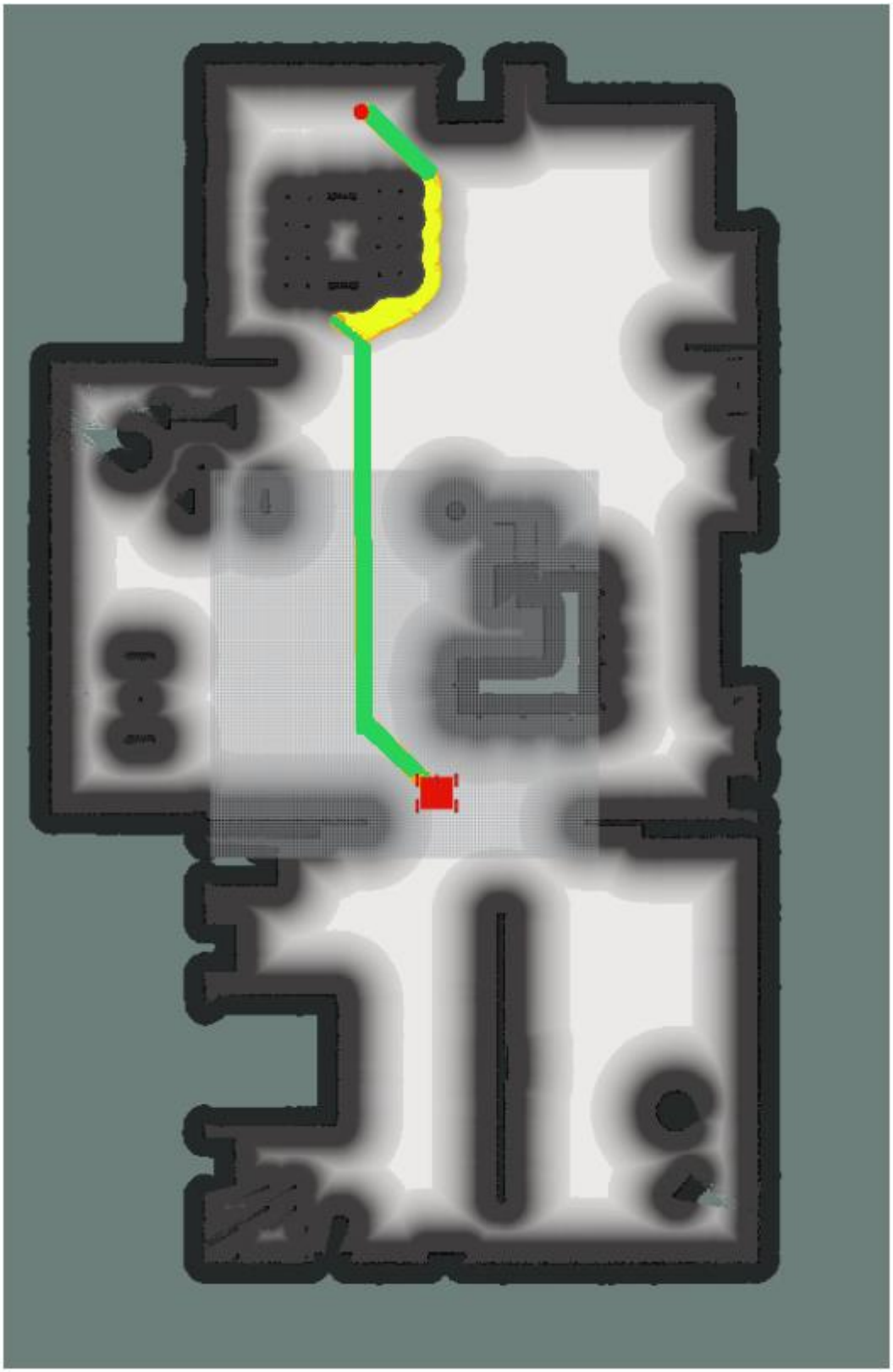


Figure 4.17 : Trouver le chemin global.

4.3.5 Navigation locale avec LiDAR 2D

Maintenant, nous rétablissons les mêmes étapes afin de déterminer le plannificateur local, nous rétablissons les mêmes étapes avec l'étiquette "path" puis modifier "topic" **/move_base/DWAPlannerROS/local_plan**, nous appelons ce "locale path "qui permet de faciliter l'accès au planificateur local **DWA** .out d'abord, appuyez sur le bouton **2d_nav_goal** et sélectionnez la position que nous voulons que le robot atteigne.

```
max_planning_retries: 0
recovery_behavior_enabled: false
controller_frequency: 5.0
controller_patience: 3.0
planner_frequency: 0.0
planner_patience: 5.0

oscillation_timeout: 0.0
oscillation_distance: 0.2

base_local_planner: "dwa_local_planner/DWAPlannerROS"
```

Figure 4.18 : Ajout de dwa_local_planner dans le fichier move_base.ymal

Les paramètres de planificateur local DWA

```
DWAplannerros:
# Robot Configuration Parameters
  max_vel_x: 0.26
  min_vel_x: -0.26
  max_vel_y: 0.0
  min_vel_y: 0.0
# The velocity when robot is moving in a straight line
  max_vel_trans:  0.26
  min_vel_trans:  0.13
  max_vel_theta: 1.82
  min_vel_theta: 0.9
  acc_lim_x: 2.5
  acc_lim_y: 0.0
  acc_lim_theta: 3.2
# Goal Tolerance Parametes
  xy_goal_tolerance: 0.05
  yaw_goal_tolerance: 0.17
  latch_xy_goal_tolerance: false
# Forward Simulation Parameters
  sim_time: 2.0
  vx_samples: 20
  vy_samples: 0
  vth_samples: 40
  controller_frequency: 10.0
# Trajectory Scoring Parameters
  path_distance_bias: 32.0
  goal_distance_bias: 20.0
  occdist_scale: 0.02
  forward_point_distance: 0.325
  stop_time_buffer: 0.2
  scaling_speed: 0.25
  max_scaling_factor: 0.2
# Oscillation Prevention Parameters
  oscillation_reset_dist: 0.05
# Debugging
  publish_traj_pc : true
  publish_cost_grid_pc: true
```

Figure 4.19 : Le fichier Ymal de paramètres d'approche de fenêtre dynamique.

Les paramètres de planificateur local sont définis comme un fichier Ymal, les paramètres les plus importants utilisés comme seuils ou limites dans l'algorithme de planificateur local DWA sont les suivants :

max_vel_x et max_vel_y : Ces paramètres définissent les vitesses de translation maximales dans les directions x et y, respectivement. Ces valeurs servent de seuils pour les vitesses maximales réalisables du robot.

max_vel_trans et min_vel_trans : Ces paramètres définissent les vitesses de translation maximale et minimale.

max_vel_trans : représente la vitesse maximale lorsque le robot se déplace en ligne droite, tandis que **min_vel_trans** est la vitesse minimale lorsqu'il y a une vitesse de rotation négligeable. Ces valeurs définissent les seuils supérieur et inférieur pour la vitesse de translation du robot.

max_vel_theta et min_vel_theta : Ces paramètres définissent les vitesses de rotation maximale et minimale. Le max_vel_theta représente la vitesse angulaire maximale, tandis que le min_vel_theta est la vitesse angulaire minimale lorsqu'il y a une vitesse de translation négligeable. Ces valeurs définissent les seuils supérieur et inférieur pour la vitesse de rotation du robot.

acc_lim_x, acc_lim_y et acc_lim_theta : Ces paramètres définissent les limites d'accélération dans les directions x, y et theta (angulaire), respectivement. Ils représentent des seuils pour les valeurs d'accélération maximales autorisées.

occdist_scale : est un facteur de pondération qui détermine dans quelle mesure le contrôleur doit éviter les obstacles. En augmentant la valeur d'occdist_scale, vous augmentez efficacement le coût associé à être proche des obstacles, encourageant le robot à maintenir une plus grande distance des obstacles.

Ces paramètres de seuil aident à fixer des limites et des contraintes sur le mouvement du robot pour assurer des trajectoires sûres et réalisables. Les valeurs spécifiques de ces seuils

peuvent être ajustées en fonction des capacités du robot et des exigences de l'environnement dans lequel il fonctionne.

```
bool DWAPlannerROS::dwaComputeVelocityCommands(geometry_msgs::PoseStamped &goal_pose, geometry_msgs::Twist& cmd_vel,float Laser_distence ,float laser_angle) {
  if(! isInitialized()){
    ROS_ERROR("This planner has not been initialized, please call initialize() before using this planner");
    return false;
  }

  geometry_msgs::PoseStamped robot_vel;
  odom_helper_.getRobotVel(robot_vel);

  geometry_msgs::PoseStamped drive_cmds;
  base_local_planner::Trajectory path = dp_->findBestPath(goal_pose, robot_vel, drive_cmds);
   cmd_vel.linear.x = drive_cmds.pose.position.x;
   cmd_vel.linear.y = drive_cmds.pose.position.y;
   cmd_vel.angular.z = tf2::getYaw(drive_cmds.pose.orientation);

  std::vector<geometry_msgs::PoseStamped> local_plan;
  if(path.cost_ < 0) {
    ROS_DEBUG_NAMED("dwa_local_planner",
        "The dwa local planner failed to find a valid plan, cost functions discarded all candidates. This can mean there is an obstacle too close to the robot.");
    local_plan.clear();
    publishLocalPlan(local_plan);
    return false;
  }

  ROS_DEBUG_NAMED("dwa_local_planner", "A valid velocity command of (%.2f, %.2f, %.2f) was found for this cycle.",
                  cmd_vel.linear.x, cmd_vel.linear.y, cmd_vel.angular.z);

  for(unsigned int i = 0; i < path.getPointsSize(); ++i) {
    double p_x, p_y, p_th;
    path.getPoint(i, p_x, p_y, p_th);

    geometry_msgs::PoseStamped p;
    p.header.stamp = ros::Time::now();
    p.pose.position.x = p_x;
    p.pose.position.y = p_y;
    p.pose.position.z = 0.0;
    tf2::Quaternion q;
    q.setRPY(0, 0, p_th);
    tf2::convert(q, p.pose.orientation);
    local_plan.push_back(p);
  }


  publishLocalPlan(local_plan);
  return true;
}
```

Figure 4.20 : la fonction dwaComputeVelocityCommands .

A partir de code de base dans le ROS avec l'ajout de certains paramètres liés à la réception d'informations laser .La fonction **dwaComputeVelocityCommands ()**, cette fonction est essentielle pour générer des commandes de vitesse adaptées à une navigation sûre et efficace des robots en utilisant l'approche DWA pour la navigation local. Elle calcule les commandes de vitesse en fonction de l'état actuel du robot et de l'environnement en utilisant la technique d'échantillonnage de fenêtre dynamique. La fonction prend en compte la position du robot et une variable de sortie `cmd_vel` pour stocker les commandes de vitesse calculées. Elle vérifie l'état d'initialisation, récupère la vitesse du robot, calcule la trajectoire en utilisant dp_->findBestPath () et extrait les commandes de conduite. Si une trajectoire valide est trouvée, elle enregistre un message de débogage et construit un plan local. Le plan local est ensuite publié sur un visualiseur. Les commandes de vitesse calculées sont renvoyées dans cmd_vel, ce qui permet de contrôler le mouvement du robot.

```
void DWAPlannerROS::publishLocalPlan(std::vector<geometry_msgs::PoseStamped>& path) {
  base_local_planner::publishPlan(path, l_plan_pub_);
}
```

Figure 4.21 :base_ local_plan.

Cette fonction est appelée pour publier les informations du plan local à un visualiseur. Elle prend en entrée un vecteur de pose-stamped (`local_plan`) représentant les points de la trajectoire locale.

Figure 4.22 : Appel au schéma local et insertion de la fenêtre dynamique

On va d'abord mettre trois exemples en mettant une chaise devant le robot et dans le deuxième exemple deux chaises et ensuite lui demander de se diriger vers l'avant. Dans le troisième exemple, nous allons mettre un cabinet et commander le robot dans la direction de gauche et combiner la planification locale et globale des itinéraires, nous allons observer l'évitement des obstacles dynamiques à chaque fois.

a) Exemple 1

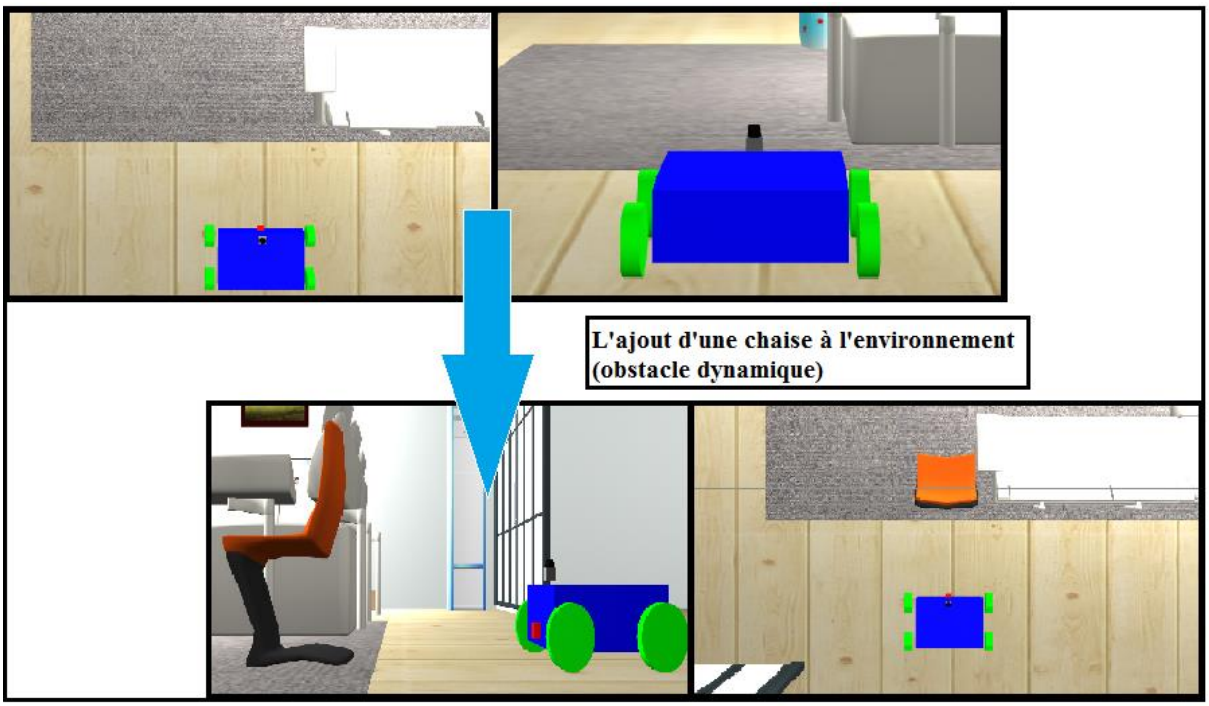


Figure 4.23 : Exemple 1 ajoute un obstacle dynamique.

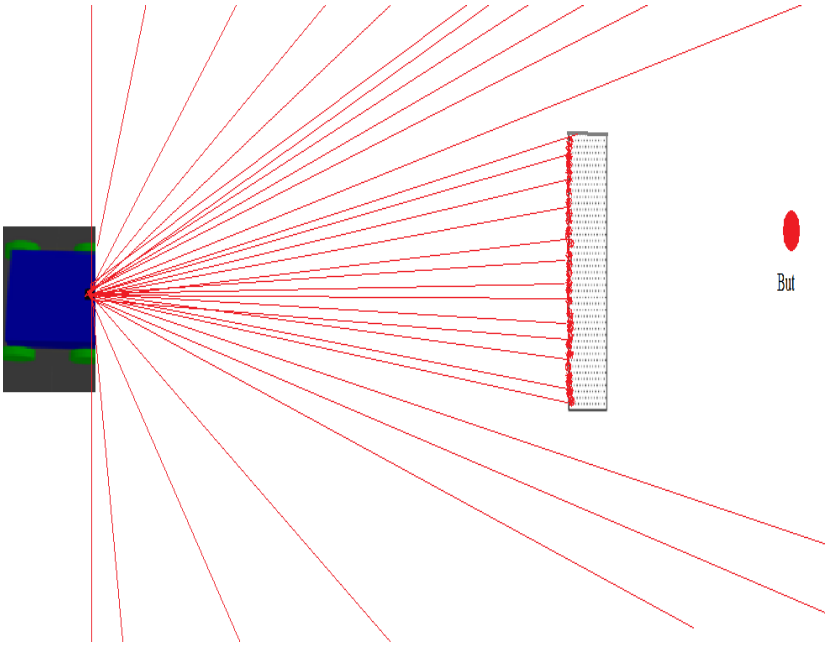


Figure 4.24: L'etape de détection d'obstacle dynamique.

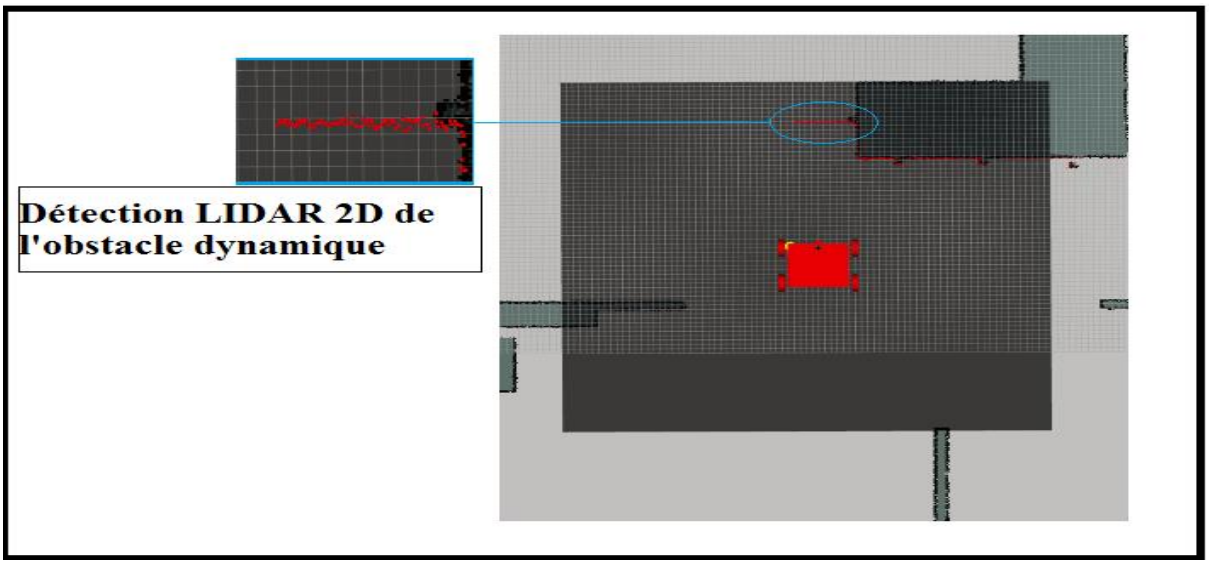


Figure 4.25 : Détecter la position de l'obstacle par le LiDAR 2D.

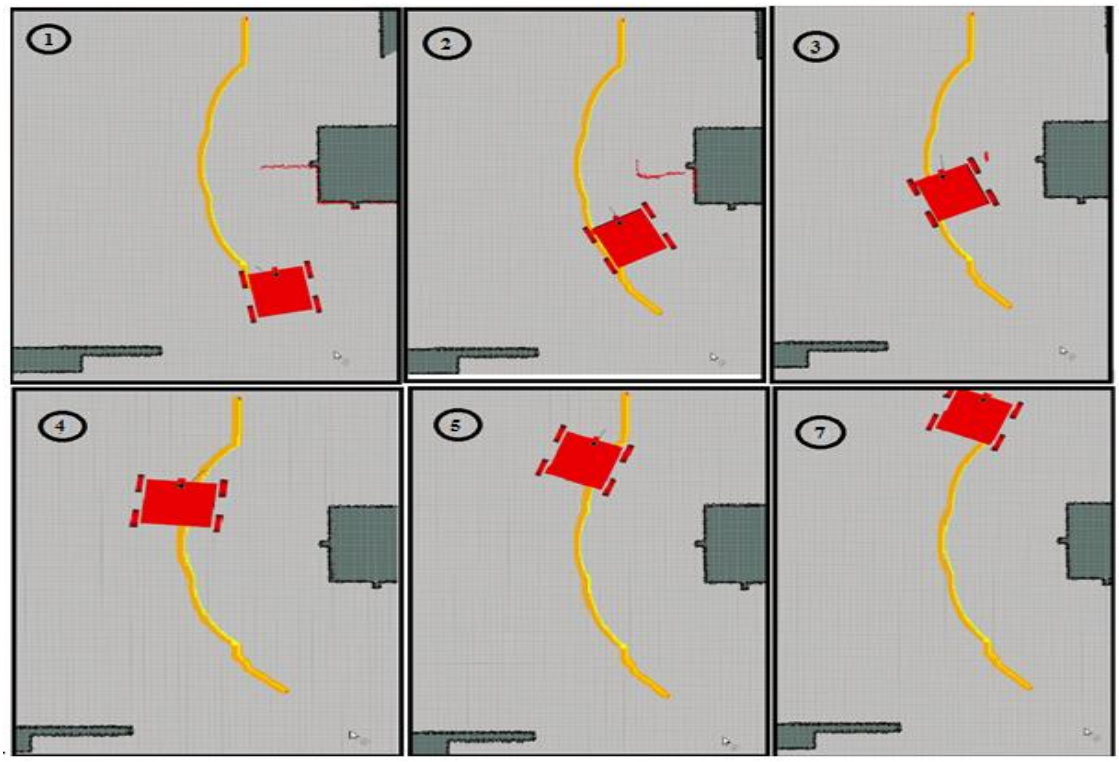


Figure 4.26 : Navigation et surmonter les obstacles dynamiques en implémentant les données LiDAR 2D de l'exemple 1.

b) Exemple 2

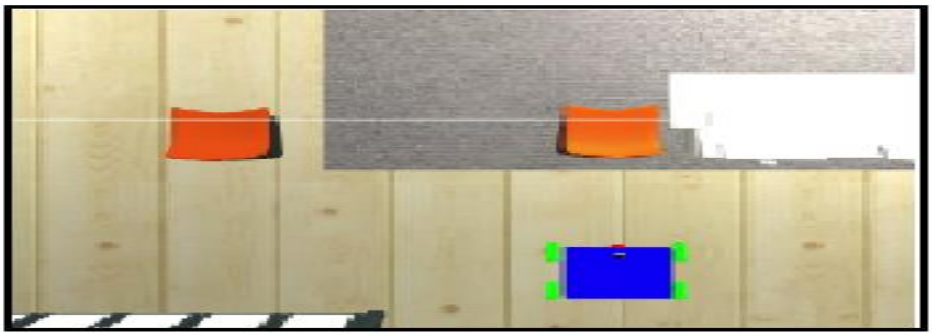


Figure 4.27 : Exemple 2 Ajouter deux obstacles dynamiques (deux chaises).

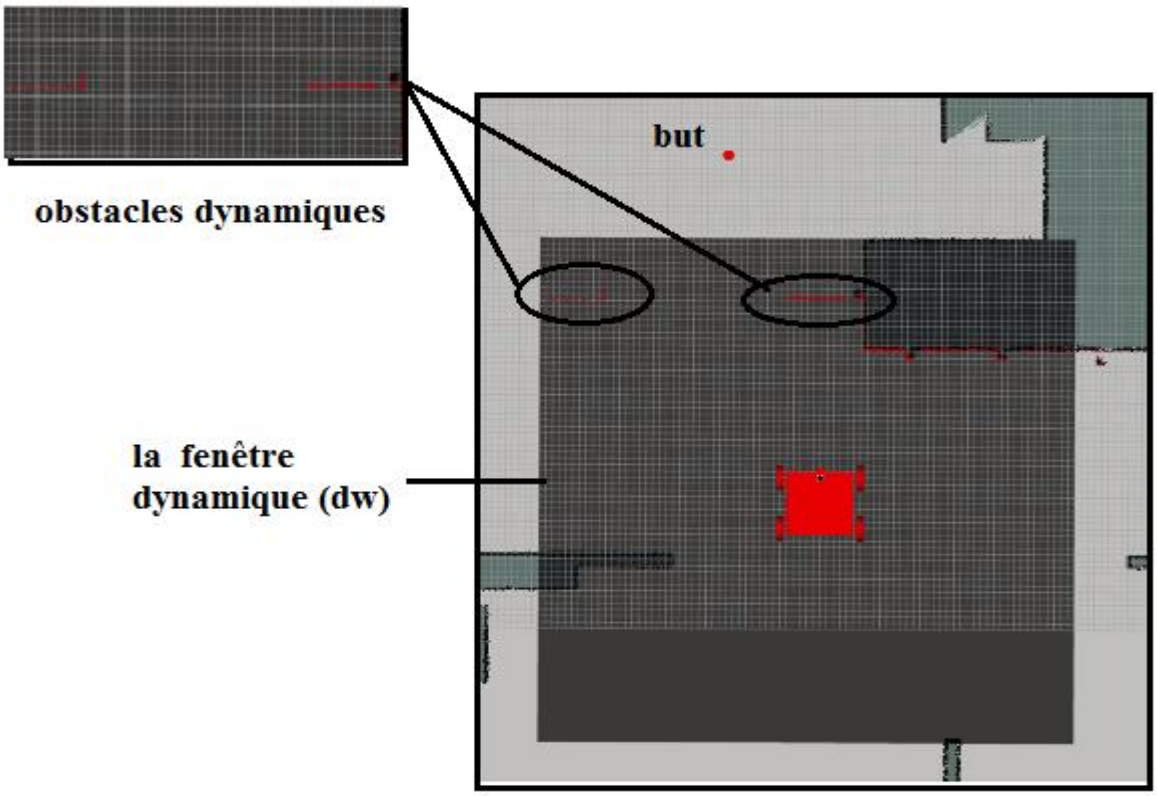


Figure 4.28: Détecter la position de l'obstacles par le LiDAR 2D.

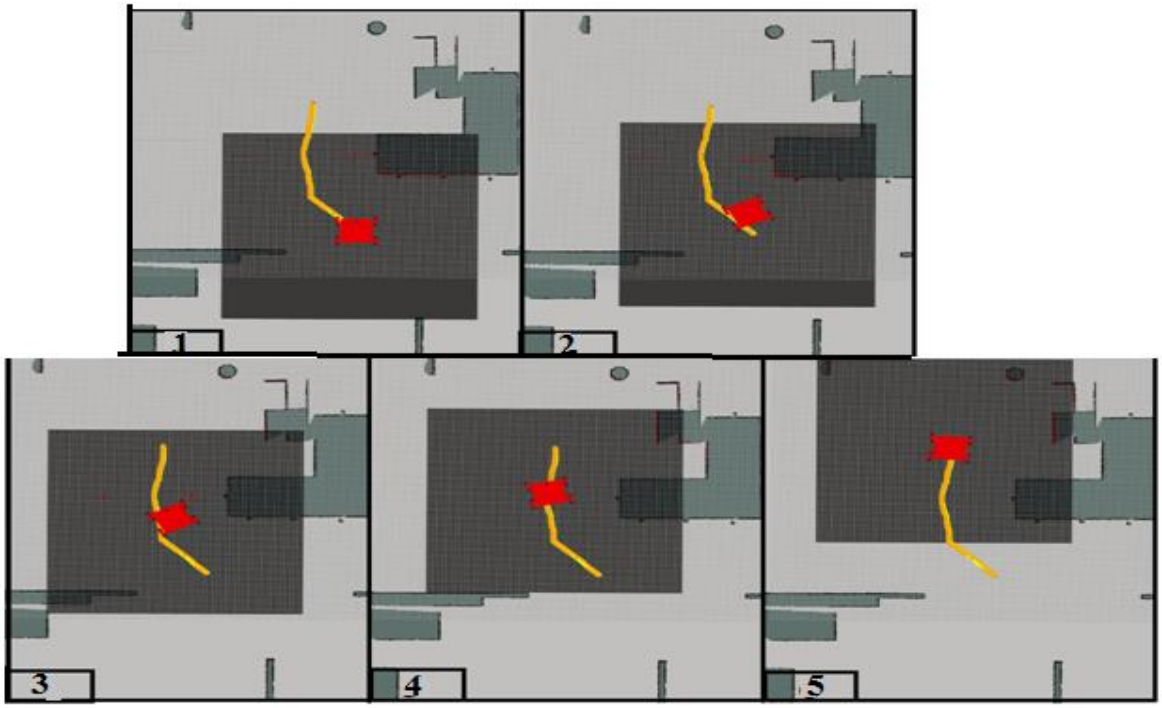


Figure 4.29 : Planification de chemin globale et phase de navigation avec LiDAR 2D de l'exemple 2.

c) Exemple 3

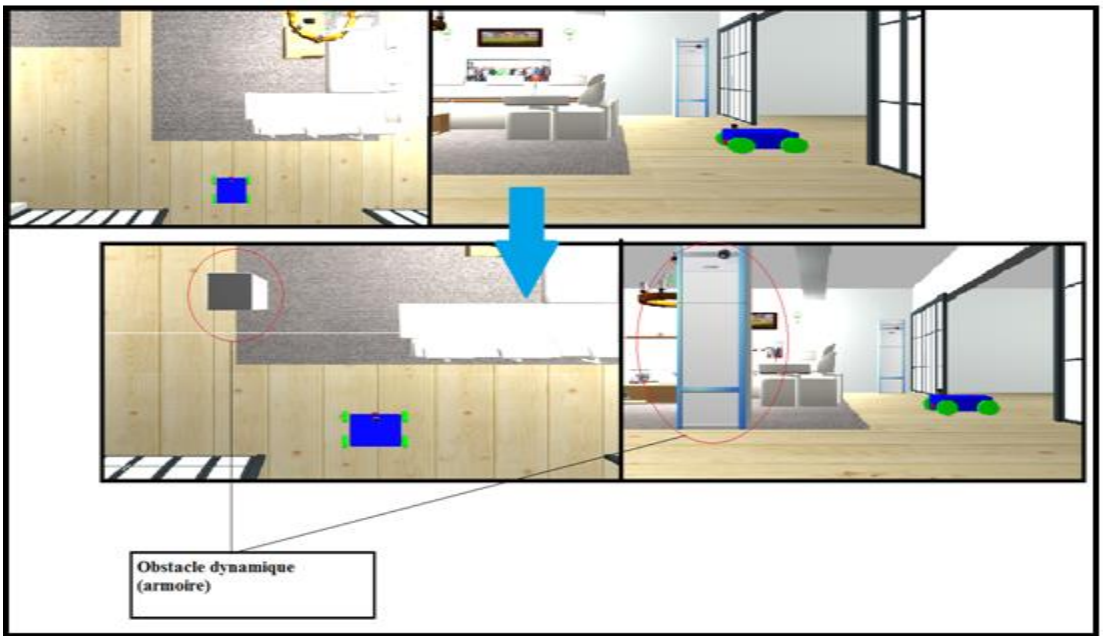


Figure 4.30 Exemple 3 Ajouter un obstacle dynamique (armoire).

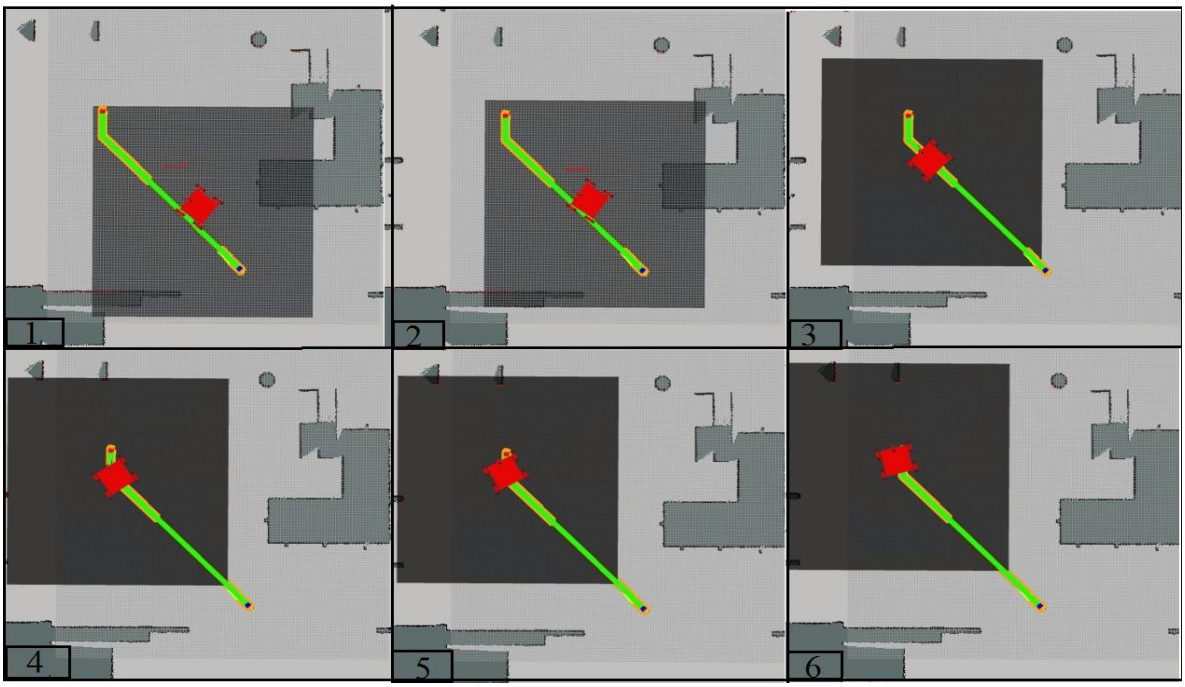


Figure 4.31 : Planification de chemin globale et phase de navigation avec LiDAR 2D de l'exemple 3.

4.4 Conclusion

En bref, dans cette partie, nous avons pu mettre en évidence les étapes les plus importantes de la mobilité des robots dans l'environnement et la façon dont il a trouvé un chemin sûr, même avec des obstacles soudains. Ce dernier a été grâce aux données lidar 2D responsables du transfert de données environnementales, où nous avons appliqué DWA qui a travaillé à planifier le chemin local sans heurter d'obstacles.

C'est grâce au puissant logiciel ROS qui a permis le traitement d'une énorme quantité de fichiers avec différentes variétés et l'équilibre de la synchronisation entre les simulations 3D et les méthodes de robot de suivi dans l'espace 2D.

Conclusion générale

Dans ce travail, nous avons exprimé notre intérêt pour la navigation d'un robot mobile effectuée, pour une navigation sécuritaire utilisant le LiDAR bidimensionnel. Il s'agissait d'une simulation de performance réelle dans le cadre de laquelle nous avons utilisé ROS, un système de contrôle très puissant, qui nous a fourni les plateformes de simulation tridimensionnelle que nous voulons et un système à deuxoutils de vision dimensionnelle pour télécharger les cartes environnementales que nous avons conçues pour simuler où nous les avons obtenues de la cartographie.

Au départ, nous avons travaillé sur la navigation globale, dans laquelle nous utilisons la carte constante de l'environnement, qui est la base pour compléter la navigation locale et surmonter les obstacles.

LiDAR 2D a grandement contribué à la réalisation et à la précision de la navigation en détectant les conséquences soudaines dans l'environnement et en travaillant à les éviter en utilisant dwa

En fait, nous avons eu des problèmes qui sont plus importants que l'aspect informatique dont nous avons besoin. La durée du processus de recherche du chemin a été très longue car il nécessite la puissance de cvalacul, bien que nous ayons pu couvrir ce problème et obtenir des résultats satisfaisants. Nous espérons nous débarrasser de cet obstacle pour réaliser d'autres projets.

Notre vision et notre ambition pour les futurs projets après ce projet sont :

Integration du LiDAR la voiture autonome existante dans notre laboratoire pour concevoir et implémenter des applications réelles à notre université. Fabrication d'un véhicule mobile contenant LiDAR bidimensionnel qui peut surmonter les obstacles fixes et dynamiques et les exploiter pour le service (envoi de marchandises au sein de l'université, voiture pour vérifier la sécurité...),navigation avec utilisation d'un LiDAR tridimensionnel

Bibliographie

[1]. Matignon, La¨etitia. *Introduction `a la robotique.* Caen, France : GREYC-CNRS,Universit´e de Caen, France, 2011/2012.

[2]. Filliat, D. *Robotique mobile1.* Paris : Ecole nationale supérieurde techniques avancées Paris Tech, 2013.

[3]. Mordechai, Ben-Ari et Francesco, Mondada. *Elements of Robotics.* Cham, Switzerland : Springer Cham, 2018. ISBN 978-3-319-62532-4 ISBN 978-3-319-62533-1.

[4]. Juan M, Corchado, Stefanos, Kollias et Javid, Taheri. *Service Robots:A Systematic Literature Review.* Western Illinois : School of Computer Sciences,College of Business and Technology, Western Illinois University,Macomb,USA, 2021.

[5]. Robot. *Ticno-science.net.* [En ligne] 16 03 2023. [Citation : 10 06 2023.] https://www.techno-science.net/glossaire-definition/Robot-page-2.html.

[6]. Addison. What is the Difference Between rviz and Gazebo? *Automatic Addison.* [En ligne] 24 06 2020. [Citation : 10 06 2023.] https://automaticaddison.com/what-is-the-difference-between-rviz-and-gazebo/#:~:text=So%20What%20is%20the%20Difference,you%20what%20is%20really%20happening.%E2%80%9D.

[7]. M, Steven and La, Valle. *Rapidly-Exploring Random Tree: A newTool for path planning.* Iowa USA : Iowa state university, 1998.

[8]. Weber, Harald. *FONCTIONNEMENT ET VARIANTES DES CAPTEURS LiDAR.* Waldkirch / Allemagne : Product Unit Ranging LiDAR sensors,SICK AG, 2018.

[9]. Nilsson, Nils J. *A MOBILE AUTOMATON: AN APPLICATION.* Menlo Park, California : Stanford Research Institute, 1969.

[10]. Takahashi, O et Schilling, R. *Motion planning in a plane using generalized voronoi diagrams.* . s.l. : IEEE Transactions on Robotics, 1989.

[11]. JALEL, SAWSSEN. *Optimisation de la navigation robotique.* Toulouse : Institut National Polytechnique de Toulouse (INP Toulouse), 2016.

[12]. B. K, Patle, et al. *A Review: On Path Planning Strategies for Navigation of.* India : Defence Technology, 2019.

[13]. *COMPARISON OF PATH PLANNING IN SIMULATED ROBOT.* **YU'NG, CH'NG CHEE.** s.l. : Bachelor of Computer Science (HONS) Faculty of Information and Communication Technology (Kampar Campus), UTAR, 2020.

[14]. *Intelligence Artificielle Algorithme A*.* **N.J.Nilsson.** s.l. : springer verlag, 1982.

[15]. Dieter, Fox, Wolfram, Burgard et Sebastian, Thrun. *The Dynamic Window Approach to Collision Avoidance.* Bonn,Germany : IEEE Journal on Robotics and Automation, 1997.

[16]. Dattalo, Amanda. ROS/Introduction. *wiki.ros.org.* [En ligne] 08 08 2018. [Citation : 10 06 2023.] http://wiki.ros.org/ROS/Introduction.

[17]. Why Gazebo? *classic.gazebosim.* [En ligne] 30 01 2019. [Citation : 10 06 2023.] https://classic.gazebosim.org/.

[18]. Ali, GAGUI. *Navigation Réactive d'humain Virtuel Par Planification Locale.* Biskra : Université de Biskra, 2013.

[19]. Christine, Kerdellant. USINENOUVELLE.com. *La robotique industrielle fait ses mues.* [En ligne] La société IPD, 06 Mai 2022. https://www.usinenouvelle.com/article/la-robotique-industrielle-fait-ses-mues.N2001767.

[20]. Khatib, O. *Real-time obstacle avoidance for manipulators and.* s.l. : In Robotics and Automation. Proceedings. 1985 IEEE International, 1985.

[21]. Choset, H, et al. *Principles of Robot Motion:Theory,Algorithms, and Implementations.* Cambridge : MIT Press, 2005.

[22] Qu’est-ce qu’un robot ? Apprendre à connaître les capteurs et les actionneurs. *diwo.* [En ligne] 09 01 2016. [Citation : 10 06 2023.] http://diwo.bq.com/fr/qu-est-ce-qu-un-robot-apprendre-a-connaitre-les-capteurs-et-les-actionneurs/.

[23]. *Quantifying and Improving Laser Range Data When Scanning Industrial Materials.* **Charles, N. MacLeod, Rahul, Summan et Stephen, Gareth Pierce.** Strathclyde : IEEE Sensors Journal, 11 2016.