الجمهـورية الجــزائرية الديمقـراطية الشــعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التعليـم الــعالي والبحث العلمــي
MINISTERE DE L'ENSEIGNEMENT SUPERIEURET DE LA RECHERCHE SCIENTIFIQUE
جامعة 20 أوت 1955 -سكيكدة-
UNIVERSITÉ 20 Aout 1955 – Skikda –

Faculté de Sciences

Département de L'INFORMATIQUE

Mémoire présenté en vue de l'obtention d'un Diplôme de Master 2

Spécialité : Réseaux et Systèmes Distribués (RSD)

THEME :

Réalisation d'un outil de cartographie automatique de l'environnement

Réalisé par :

- BAKKOUCHE Azzedine
- BOUZETTOUTA ABDESSLEM

Encadré par :

BOUTINE Rachid

Année universitaire : 2021-2022

Remerciements

On tient à la fin de ce travail à remercier ALLAH le tout-puissant de nous avoir donné la foi et de nous avoir permis d'en arriver là malgré les circonstances difficiles auxquelles nous avons été confrontés.

En préambule à ce mémoire, je souhaitais adresser mes remerciements les plus sincères aux professeurs qui nous ont enseigné durant notre parcours académique et aux personnes qui nous ont apporté leurs aides et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Je tiens à remercier également notre encadreur,
Mr. BOUTINE RACHID, d'avoir accepté de m'encadrer, proposer mon thème et m'avoir guidé tout au long de mon projet de master par ses précieuses directives, pour sa confiance et sa gentillesse, sa disponibilité ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu me consacrer.

J'exprime également mes gratitudes à nos examinateurs, qui vont nous honorer en évaluant ce travail.

Par la même occasion, Je tiens à remercier les responsables administratifs pour nous avoir accueillies chaleureusement au sein de leur structure.

AZZOU & ISSLAM

Dédicaces

A mes très chers parents

Pour tout l'amour dont vous m'avez entouré, pour tout ce que vous avez

fait pour moi.

Je ferai de mon mieux pour rester un sujet de fierté à vos yeux avec

l'espoir de ne jamais vous décevoir.

Que ce modeste travail, soit l'exaucement de vos vœux tant formulés et de

vos prières quotidiennes.

A mes très cher Sœur

A mes très chers amis

En souvenir de nos éclats de rire, des bons moments et nuits blanches.

En souvenir de tout ce qu'on a vécu ensemble.

J'espère de tout mon cœur

Que notre amitié durera

éternellement.

AZZOU

Dédicace

Je remercie tout d'abord ALLAH le tout puissant de m'avoir donné la santé la patience, la

Puissance et la volonté pour réaliser ce travail

Je dédie ce modeste travail de fin d'étude a mes chers parents ;

Qui ont sacrifié leur vie pour ma réussite et m'ont éclairé le chemin par leur

conseil judicieux

Et dédie aussi ce travail à mes sœurs

A mes frères

A toute la famille

Et à tous ceux qui me sont chers

ISSLAM

ملخص

يستخدم RTAB MAP على نطاق واسع في تطوير الروبوتات المستقلة الجديدة. لديها العديد من التطبيقات مثل: السيارات ذاتية القيادة، وروبوتات التنظيف المستقلة، والتفتيش، والاستكشاف، إلخ. في الوقت الحاضر، هناك العديد من أطر SLAM مفتوحة المصدر التي تساعد في البحث عن تقنيات جديدة. ومع ذلك، فإن قابلية إعادة الاستخدام والقدرة على التكيف لمعظم الأطر محدودة للغاية.

لذلك، يتطلب تنفيذ أجهزة استشعار وخوارزميات جهدًا كبيرًا، حتى إضافة أشهر إلى كل تطوير جديد.

في هذا المشروع، يتم تحليل مشكلات نمطية في خريطة RTAB لإيجاد حل ممكن. المساهمة الرئيسية لهذا المشروع هي تصميم وتنفيذ إطار SLAM المعياري، القادر على العمل مع RTAB-MAP وتحسين قابلية إعادة استخدام SLAM

يتيح استخدام إطار SLAM المعياري مع RTAB-MAP تنفيذًا أسرع بكثير لأجهزة الاستشعار والخوارزميات الجديدة مع الاستمرار في استخدام معظم الوظائف في معالجة المستشعرات المتوفرة في RTAB-MAP.

الكلمات الرئيسية: خريطةRTAB ، SLAM، مستشعرات ...

Résumé

RTAB MAP est largement utilisé dans le développement de nouveaux robots autonomes. Il a de nombreuses applications comme : les voitures autonomes, les robots de nettoyage autonomes, l'inspection, l'exploration, etc. De nos jours, il existe plusieurs Framework SLAM open source qui contribuent à la recherche de nouvelles technologies. Cependant, la réutilisabilité et l'adaptabilité de la plupart des Framework sont très limitées.

Par conséquent, la mise en œuvre de nouveaux capteurs et algorithmes nécessite un important effort, ajoutant même des mois à chaque nouveau développement.

Dans ce projet, les problèmes de modularité dans RTAB-MAP sont analysés pour trouver une solution possible. La principale contribution de ce projet est la conception et la mise en œuvre d'un cadre SLAM modulaire, capable de travailler avec RTAB-MAP et améliorer la réutilisabilité du SLAM

L'utilisation du cadre SLAM modulaire avec RTAB-MAP permet une mise en œuvre beaucoup plus rapide de nouveaux capteurs et algorithmes tout en étant capable d'utiliser la plupart des fonctionnalités dans le traitement des capteurs disponibles dans RTAB-MAP.

Mots clé : *RTAB MAP, SLAM, Les capteurs ...*

Summary

RTAB MAP is widely used in the development of new autonomous robots. It has many applications like: autonomous cars, autonomous cleaning robots, inspection, exploration, etc. Nowadays, there are several open source SLAM frameworks that help in researching new technologies. However, the reusability and adaptability of most frameworks is very limited.

Therefore, implementing new sensors and algorithms requires significant effort, even adding months to each new development.

In this project, modularity issues in RTAB-MAP are analyzed to find a possible solution. The main contribution of this project is the design and implementation of a modular SLAM framework, able to work with RTAB-MAP and improve the reusability of the SLAM

Using the modular SLAM framework with RTAB-MAP allows much faster implementation of new sensors and algorithms while still being able to use most of the functionality in sensor processing available in RTAB-MAP.

Keywords*: RTAB MAP, SLAM, Sensors…*

Table des matières

Liste Des Figures

Liste Des Tableaux

Liste des abréviations

Introduction Général

1. SLAM = Simultaneous Localization And Mapping
2. GPS = Global Positioning System

Chapitre I : L'état De L'art

I.1. 3D = 3 Dimensions

I.2. RUR = Rossum's Universal Robot

I.3. MIT = Massachusetts Institute of Technology

I.4. 2D = 2 Dimensions

I.5. = SIG = Système d'Informations Géographiques

I.6. 3G = 3ème Génération

I.7. ONERA = Office National d'Etudes et de Recherches Aérospatiales

I.8. ToF = Time of Flight Mass Spectrometry

I.9. Lidar = Light Detection And Ranging ou Laser Imaging Detection And Ranging

I.10. RPLIDAR Slamtec A3M1 = un télémètre laser outdoor et indoor

I.11. RTAB-MAP = Real-Time Appearance-Based Mapping

I.12. EKF = Extended Kalman Filter

Chapitre II : Conception Et Description De Notre Système

II.1. ORB-SLAM = is a versatile and accurate SLAM solution for Monocular, Stereo and RGB-D cameras.

II.2. RVB = Rouge-Vert-Bleu

II.3. ID = Identity Document

II.4. VO = Odométrie Visuelle

II.5. LO = Odométrie Lidar

II.6. F2F = Frame To Frame

II.7. F2M = Frame To MAP

II.8. S2S = Scan To Scan S2M

II.9. S2M = Scan To MAP

II.10. ROS = Robot Operating System

II.11. ICP = Iterative Closest Point

II.12. WM = mémoire de travail

II.13. MLT = Mémoire à Long Terme

II.14. SM = Sensory Memory

II.15. BoVW = Bag of Visual Words

Chapitre III : L'implémentation

III.1. MSI = Micro-Star International

III.2. GPU = Graphics Processing Unit

III.3. SDK = Software Development Kit

III.4. LED = Light-Emitting Diode

III.5. API = Application Programming Interface

III.6. PC = Portable Computer

III.7. LTS = Long Term Support

III.8. RVIZ = a ROS graphical interface that allows you to visualize a lot of information

III.9. IOS = est un système d'exploitation mobile

III.10. SIFT = Scale-Invariant Feature Transform

Introduction Générale

RTAB-MAP est une approche populaire du SLAM basée sur les graphes qui met en œuvre une fermeture de boucle visuelle.

RTAB-MAP est une bibliothèque open source implémentant la détection de fermeture de boucle avec une gestion de la mémoire approche, limitant la taille de la carte afin que les détections de fermeture de boucle soient toujours traitées dans un délai fixé, répondant ainsi aux exigences en ligne pour des projets à long terme et à grande échelle cartographie de l'environnement. RTAB-MAP a depuis été étendu à une approche SLAM complète basée sur des graphes à utiliser dans diverses configurations et applications.

La cartographie et localisation simultanée (SLAM) est une approche utilisée en robotique mobile lorsqu'un système de localisation externe comme le GPS n'est pas accessible pour estimer la position précise du robot dans l'environnement.

L'approche de gestion présentée dans cette mémoire a pour but de réaliser un outil de cartographie automatique de l'environnement satisfaisant la contrainte de traitement pour un fonctionnement meilleur à grande échelle et à long terme. Le temps de traitement, c.-à-d. le temps requis pour le robot traiter une image acquise ou le capteur capter, est le critère utilisé pour limiter le nombre d'endroits conservés dans la mémoire de travail du robot.

Afin de bien mener notre étude, il sera partagé comme suit :

1. Une introduction générale définissant le contexte de ce travail. Ensuite, j'ai posé le problème et les contributions apportées, suivie d'une structuration de mémoire.
2. Après je découpe l'architecture conceptuelle de ce système en trois chapitres majeurs :

Chapitre 1 : C'est le chapitre de l'état de l'art de notre projet, en expliquent les robots : concept de base, Quelques Plateformes robotiques pour la localisation et la cartographie, Les capteurs utilisés par les robots de cartographie.

Chapitre 2 : le but de ce chapitre est d'expliquer une description et un modèle de travail choisi, ensuite la conception du notre système.

Chapitre 3 : Ce chapitre est l'incarnation théorique de mon projet car il explique en détail l'implémentation du notre projet.

3. Le mémoire se termine par une conclusion et aussi l'indication de quelques perspectives possibles pour les activités de recherche futures.

Chapitre 1

L’Etat De L’Art

1. Introduction

L'objet de la robotique est l'automatisation de systèmes mécaniques. En dotant le système de capacités de perception, d'action et de décision, l'objectif est de lui permettre d'interagir rationnellement avec son environnement, et de façon autonome. La robotique est un domaine de recherche qui se situe au carrefour de l'intelligence artificielle, de l'automatique, de l'informatique et de la perception par ordinateur, cette interdisciplinarité est à l'origine d'une certaine complexité.

Le domaine de la vision 3D évolue de manière exponentielle, notamment grâce à son côté attractif et l'apparition de nouvelles technologies. Dans la vision tridimensionnelle, l'utilisation des caméras 3D est très appréciée pour des applications dans plusieurs domaines tel que, la robotique, l'architecture, l'industrie, la vision par ordinateur, …etc.

Dans notre travail, on s'intéresse à un nouveau capteur proposé par Microsoft nommé « Kinect ». C'est un dispositif principalement utilisé pour la détection de mouvement et qui a été initialement développé pour la console de jeu Xbox 360 par Microsoft.

Une Kinect version Windows est sortie plus tard. La description de la Kinect est donnée dans ce chapitre

2. Aperçu historique

En 1921, Karel Capek introduisit le terme "robot" dans sa pièce de théâtre "R.U.R." (Rossum's Universal Robot), du tchèque "robota" signifiant travail forcé, corvée. Il raconte l'histoire d'un savant appelé Rossum, ayant réussi à mettre au point des créatures semblables physiquement à des êtres humains, que son fils exploita au sein de son entreprise. Le terme "robotique" fut lui amené par l'écrivain Isaac Asimov, qui proposa les trois lois de la robotique suivantes [1] :

1. Loi 1 : Un robot ne peut blesser un être humain ni, par son inaction, permettre qu'un humain soit blessé.

2. Loi 2 : Un robot doit obéir aux ordres donnés par les êtres humains, sauf si de tels ordres sont en contradiction avec la Première Loi.

3. Loi 3 : Un robot doit protéger sa propre existence aussi longtemps qu'une telle protection n'est pas en contradiction avec la Première et/ou la Deuxième Loi.

La Tortue construite par Grey Walter dans les années 1950 (Figure I.1), est l'un des premiers robots mobiles autonomes. Grey Walter n'utilise que quelques composants analogiques, dont des tubes à vide, mais son robot est capable de se diriger vers une lumière qui marque un but, de s'arrêter face à des obstacles et de recharger ses batteries lorsqu'il arrive

dans sa niche. Toutes ces fonctions sont réalisées dans un environnement entièrement préparé, mais restent des fonctions de base qui sont toujours des sujets de recherche et de développement technologiques pour les rendre de plus en plus génériques et robustes [2].

Figure I.1 : La tortue de Grey Walter (Machina Speculatrix ou Elsie) avec l'illustration [3].

Dans les années 60, les recherches en électronique vont conduire, avec l'apparition du transistor, à des robots plus complexes mais qui vont réaliser des tâches similaires. Ainsi le robot "Beast" (Figure I.2) de l'université John Hopkins est capable de se déplacer au centre des couloirs en utilisant des capteurs ultrason, de chercher des prises électriques (noires sur des murs blancs) en utilisant des photo-diodes et de s'y recharger.

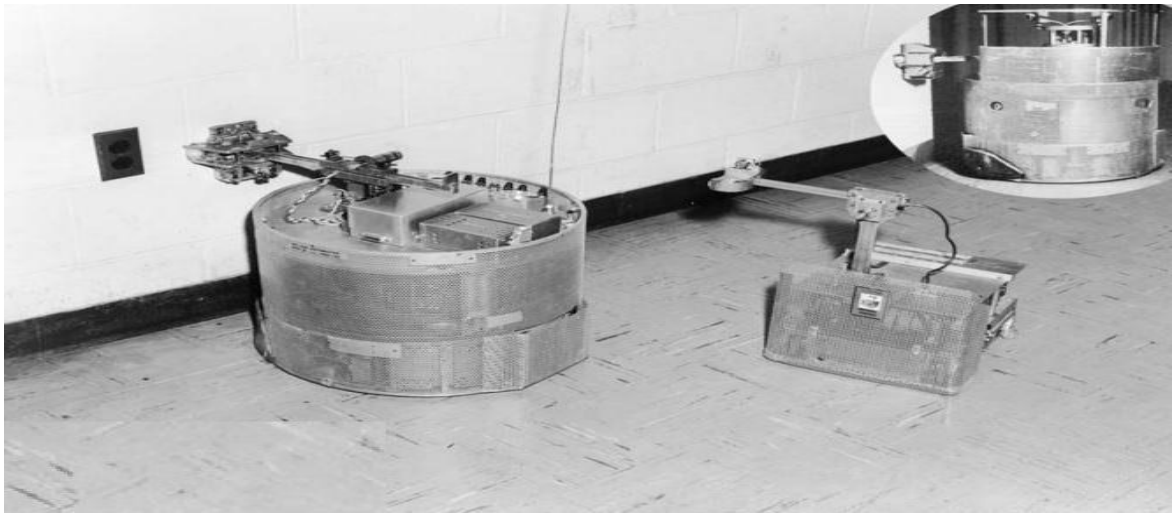


Figure I.2 : Robot « Beast » de l'université John Hopkins.

Les premiers liens entre la recherche en intelligence artificielle et la robotique apparaissent à Stanford en 1969 avec Shakey (Figure.I.3). Ce robot utilise des télémètres à ultrason et une caméra et sert de plate-forme pour la recherche en intelligence artificielle.

Figure I.3 : Shakey de Stanford : Uneplate-forme pour les recherches en

Le Stanford Cart date de la fin des années 1970 (Figure.I.4), avec notamment les premières utilisations de la stéréo-vision pour la détection d'obstacles et la modélisation de l'environnement.

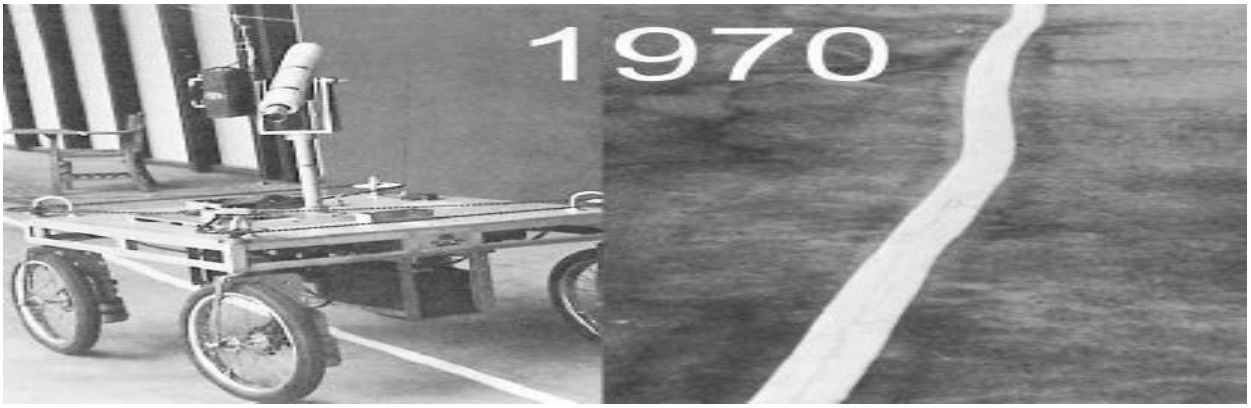
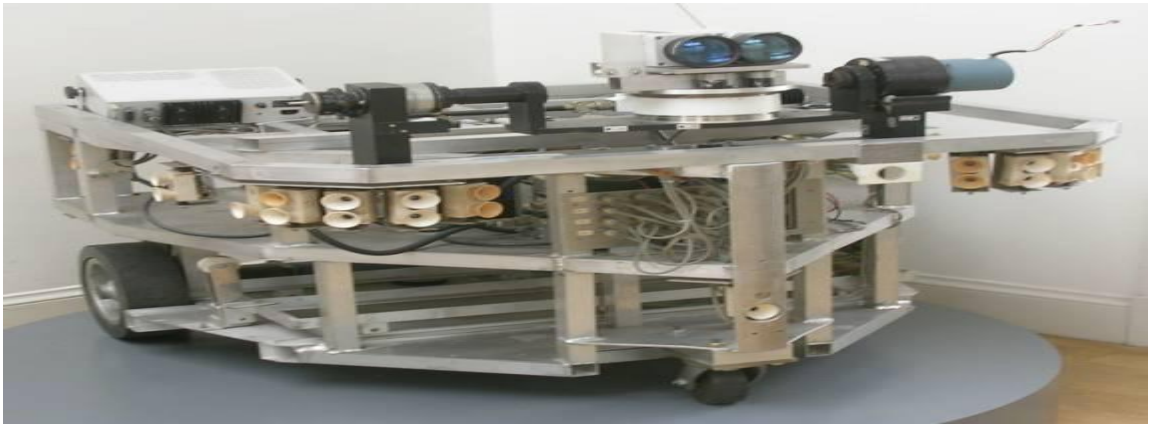


Figure I.4 : Le Stanford Cart de la fin des années 1970 [3].

En France, le robot Hilaire était le premier robot construit au LAAS à Toulouse



(Figure I.5)

Figure I.5 : Robot Hilare du LAAS 1977 [4].

Une étape importante est à signaler au début des années 1990 avec la mise en avant de la robotique réactive, représentée par Genghis (Figure.I.6), développé par Rodney Brooks au MIT. Cette nouvelle approche de la robotique, qui met la perception au centre de la problématique, a permis de passer de gros robots très lents à de petits robots beaucoup plus réactifs et adaptés à leur environnement.

Figure I.6 : Genghis de Rodney Brooks 1990 [4].

3. Concepts de base des robots

3.1. Définition d'un robot

Un robot est un système mécanique composé de corps mobiles reliés par des actionneurs qui lui donnent des capacités de mouvement dans l'espace physique. Les structures mécaniques utilisés sont de types très divers, qu'il s'agisse de robots manipulateurs constitués d'un bras terminé par un outil ou un organe de préhension (pince, main multi-doigts, etc.), ou encore de robots mobiles avec des principes de locomotion adaptés à divers environnements (roues, chenilles, pattes, etc.) [5].

Enfin il existe également de nombreux autres types de robots mobiles (robots marins, sous-marins, drones volants, micro et nano robots), généralement l'étude de ce type de robot se fait dans des thématiques spécifiques avec des problèmes particuliers à l'application visée [6].

3.2. Boucle de contrôle

Un robot mobile est commandé par une boucle de contrôle, comme illustré à la (figure 1.7). De façon itérative, celle boucle fait une lecture des données reçues par les capteurs, les interprète, calcule les commandes motrices et les envoie aux actionneurs. Typiquement, cette boucle est exécutée environ dix (10) fois par seconde ; la fréquence peut varier selon les types de capteurs et d'actionneurs utilisés. La boucle de contrôle n'est pas unique ; selon l'architecture utilisée, elle peut être décomposée en plusieurs sous boucles de contrôle agencées de manières différentes.

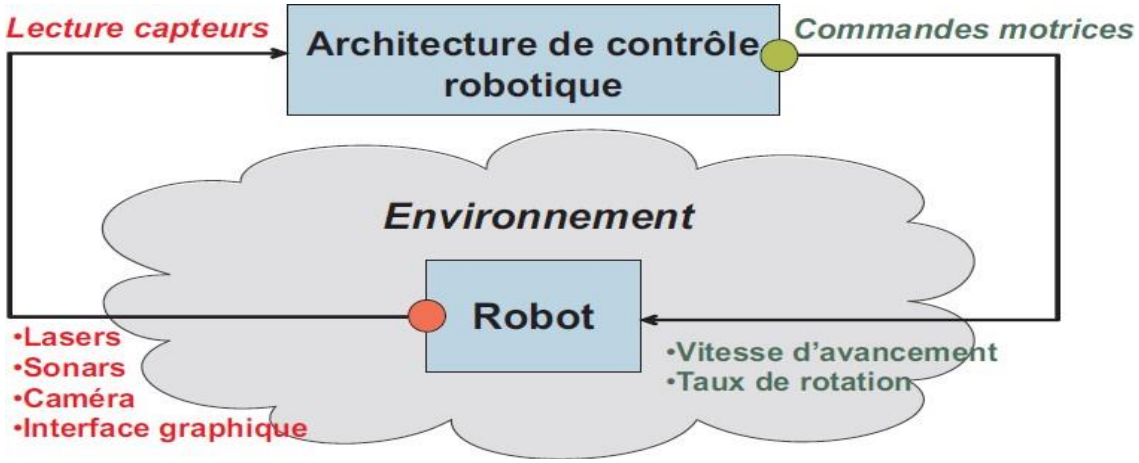


Figure I.7 : Boucle de contrôle.

3.3. Perception

La notion de perception en robotique mobile est relative à la capacité du robot à recueillir, traiter et mettre en forme des informations qui lui sont utiles pour agir et réagir dans l'environnement qui l'entoure. Elle est donc la faculté de détecter et/ou appréhender

l'environnement proche ou éloigné du robot. Alors que pour des tâches de manipulation on peut considérer que l'environnement du robot est relativement structuré, ce n'est plus le cas lorsqu'il s'agit de naviguer de manière autonome dans des lieux très partiellement connus. Aussi, pour extraire les informations utiles à l'accomplissement de sa tâche, il est nécessaire que le robot dispose de nombreux capteurs mesurant aussi bien son état interne que l'état de l'environnement dans lequel il évolue. Le choix des capteurs dépend bien évidemment de l'application envisagée. La perception est nécessaire pour la sécurité du robot, la modélisation de l'environnement et l'évitement et le contournement d'obstacles.

3.4. Raisonnement et décision

Ce niveau consiste l'intelligence du robot. À l'instar du home, le raisonnement du robot lui permet de décider d'une action appropriée à une situation donnée, compte tenu d'une mission à réaliser. Plusieurs tâches élémentaires peuvent être exécutées en parallèle pour synthétiser un comportement. La façon dont elles interagissent est définie par l'architecture décisionnelle du robot. Cette décision est ensuite transmise au niveau fonctionnel pour opérer les différentes parties qui satisfait cette demande.

3.5. Classification des robots

Aujourd'hui, le marché commercial de la robotique mobile est toujours relativement restreint. Mais, il existe de nombreuses perspectives de développement qui en feront probablement un domaine important dans le futur. Les applications des robots peuvent se trouver dans de nombreuses activités "ennuyeuses, salissantes ou dangereuses", mais également pour des applications ludiques ou de service, comme l'assistance aux personnes âgées ou handicapées [7]. Parmi les domaines concernés, citons :

- La robotique de service (hôpital, bureaux) ;
- La robotique de loisir (Aibo, robot 'compagnon') ;
- La robotique industrielle (entrepôts, récolte de productions agricoles,mines) ;
- La robotique en environnement dangereux (spatial, industriel, militaire).
- Les robots mobiles : Un robot mobile est un robot capable de naviguer dans son environnement de façon indépendante et n'est pas fixé à un seul emplacement physique. Alors que la portée et la précision de navigation requise varie en fonction de la taille du robot et du type de sa tâche [6]. Les robots mobiles sont classés selon plusieurs critères (degré d'autonomie, système de locomotion, énergie utilisée, etc.). On peut citer quelques types :

- Véhicule télécommandé par un opérateur qui lui impose chaque tâche élémentaire à réaliser ;
- Véhicule télécommandé au sens de la tâche à réaliser. Le véhicule contrôle automatiquement ses actions ;
- Véhicule semi-autonome réalisant sans l'aide de l'opérateur des tâches prédéfinies ;
- Véhicule autonome qui réalise des tâches semi- définies.

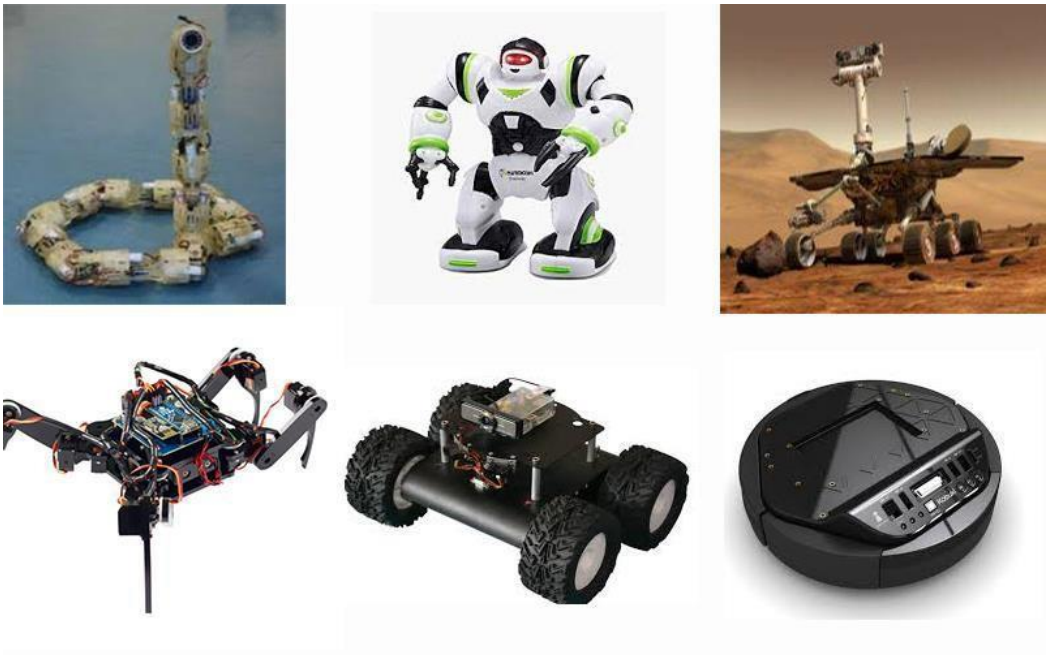


Figure I.8 : Robots mobiles

3.6. Thématiques de la robotique mobile

La communauté des chercheurs dans le domaine de la robotique à dégage 4 grands axes de travail autour desquels s'articulent les recherches actuelles en robotique mobile [9]:

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

- ***Contrôle et commande des véhicules*** : cet axe regroupe les thématiques liées à la planification de chemin, la génération de trajectoires, et la commande des robots de manière générale. Une

prise en compte de plus en plus poussée des contraintes et de la dynamique des robots est nécessaire, pour adapter au mieux les robots à leur environnement. La bonne gestion des obstacles et la prise en compte des incertitudes de mesures sont également des points clés de cette thématique,

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

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

4. Quelques Plateformes robotiques pour la localisation et la cartographie

4.1. Voiture autonome pour la cartographie d'environnement 3D

Pour renforcer les actions de Google MAPs et de Google Earth, le géant du Web a lancé le Google Street View. La voiture Google Street View est équipée de pylônes supportant de nombreuses caméras. Ces dernières enregistrent toutes les images sur leur passage à 360°. Un logiciel traite ensuite les enregistrements pour donner l'effet de continuité. En outre, cette auto détecte les signaux Wi-Fi et 3G pour en obtenir une liste. Des méthodes spécifiques ont été adoptées afin de favoriser la transition entre les captures et le floutage. Elles sont indispensables pour cacher les visages et les plaques d'immatriculation. En Allemagne, certains propriétaires ont même demandé de flouter le numéro de leur immeuble [8].

Figure I.9 : voitures Google Street View.

4.2. Drone autonome pour la cartographie d'environnement 3D

Le drone doté d'une vision 3D et de capacités de calcul temps-réel construit son environnement sous forme d'un modèle 3D, dans lequel il évolue en y évaluant visuellement sa position. Il peut prendre des décisions et élaborer lui-même les commandes de navigation, en fonction des objectifs de sa mission [10].

A partir de composants du marché drone léger quadri-rotor, caméras miniatures, ressources de calcul embarquées – les chercheurs de l'ONERA se sont lancés dans la course internationale à l'autonomie de la robotique aérienne et se retrouvent au meilleur niveau mondial avec un savoir-faire essentiellement concentré dans le logiciel : reconstitution 3D, navigation, commande...

Figure I.10 : Photographie aérienne par drone

Les secteurs d'application sont très variés : surveillance et intervention sur les infrastructures SNCF, dans les transformateurs EDF, pour les militaires, le bâtiment, les ouvrages d'art, l'archéologie…

4.3. Sous-marine autonome pour la cartographied'environnement 3D

Ce type de robot sous-marin pourrait servir dans divers domaines, à commencer par des travaux de décontamination de zones polluées par des matériaux radioactifs. Il pourrait aussi être employé pour l'exploration minière à la recherche de métaux rares ou d'hydrate de méthane. Les chercheurs citent également des tâches de maintenance d'infrastructures sous-marines telles que les câbles de télécommunications ainsi que des missions de surveillance environnementale, d'observation de la faune marine ou encore de récupération de débris et déchets. Nous pouvons envisager des applications autres que sous-marines [14].

Figure I.11 : Sous-marine avec un robot.

5. Les capteurs utilisés par les robots de cartographie

Les capteurs ont pour fonction d'acquérir des données provenant de l'environnement. Les capteurs typiquement installés sur un robot mobile (voir Figure I.1) sont des sonars à ultrasons, un dispositif à balayage laser, des encodeurs de roues (odomètres), une ou deux caméras optiques et des microphones. Les types d'informations perçues ainsi que leurs précisions varient beaucoup d'un capteur à l'autre : on peut citer deux types de capteurs [13].

Les capteurs proprioceptifs, et Les capteurs extéroceptifs :

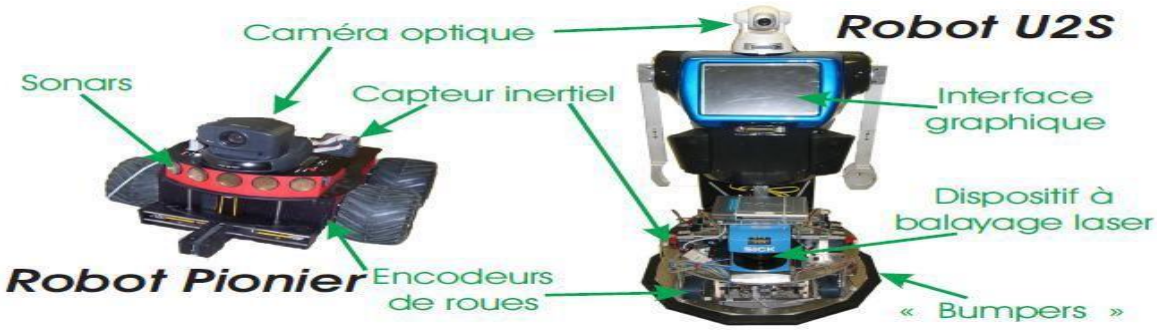


Figure I.12 : Les capteurs proprioceptifs, et Les capteurs extéroceptifs

5.1 Capteurs proprioceptifs

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

On peut regrouper les capteurs proprioceptifs en deux familles :

- Les capteurs de déplacement qui comprennent les odomètres, les accéléromètres et les adars Doppler. Cette catégorie permet de mesurer des déplacements élémentaires, des variations de vitesse ou d'accélération sur des trajectoires rectilignes ou curvilignes.
- Les capteurs d'attitude, qui mesurent deux types de données : les angles de cap et les angles de roulis et de tangage. Ils sont principalement constitués par les gyroscopes, les gyromètres, les gyrocompas, les capteurs inertiels composites, les inclinomètres et les magnétomètres. Ces capteurs sont en majorité de type inertiel [11]. Par exemple :

5.1.1. Les odomètres

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

5.1.2. Les accéléromètres

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

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

5.1.3. Les Radars Doppler

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

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

5.1.4. Les Gyroscope

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

5.2 Capteurs extéroceptifs

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

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

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

5.2.1. Télémètres

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

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

5.2.2. Lidar

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

le poids, la distance, etc. [12]

L'acronyme LiDAR signifié Light Detection and Ranging. Il s'agit d'une méthode de calcul qui permet de déterminer la distance entre le capteur et l'obstacle visé.

Un LiDAR utilise un faisceau laser pour la détection, l'analyse et le suivi. La technologie Lidar est une technologie de télédétection qui mesure la distance entre le capteur et une cible. La lumière est émise par le LiDAR et se dirige vers sa cible. Elle est réfléchie sur sa surface et revient à sa source. Comme la vitesse de la lumière est une valeur constante, le LiDAR est capable de calculer la distance le séparant de la cible [12].

Figure I.13 : Le RPLIDAR A3M1 de Slamtec [12].

5.2.3. Kinect

La Kinect, initialement connu sous le nom de code « Project Natal », a été conçue par PrimeSense, une société Israélienne indépendante de Microsoft.

Figure I.14 : Dispositif Kinect.

6. Travaux connexes

Plusieurs auteurs ont remarqué l'importance d'un cadre SLAM et RTAB MAP modulaire et ont travaillé dans ce but. Les travaux pertinents à ce projet peuvent être classés en : analyses qui se concentrent sur la structure générale du problème d'estimation, algorithmes SLAM et RTAB-MAP sur les différents paradigmes et propositions de cadres modulaires : Initialement, les solutions SLAM classiques étaient basées sur des filtres, et certains auteurs se sont concentrés sur la mise en œuvre de différents types de capteurs dans des approches basées sur des filtres. Lyne et al. (2013) est un exemple de création d'un cadre basé sur un EKF itéré se concentrant sur la fusion de capteurs d'un système de capteurs hétérogène. Cette combinaison de capteurs est obtenue en utilisant une extension du vecteur d'état en ajoutant de nouveaux états et biais, en fonction du nombre et du type de capteurs utilisés. Une autre tentative utilisant une approche basée sur des filtres, utilisant cette fois des filtres à particules, est donnée dans Tim Browning (2016). Dans cette étude, l'auteur crée une structure basée sur des interfaces qui connectent cinq blocs fonctionnels présents dans chaque système SLAM : robot, environnement, fonctionnalité, capteur et spécifique au monde [14].

Jeroen Minnema (2020) est une approche plus récente qui a tenté de trouver des similitudes entre les trois SLAM les plus populaires : EKF SLAM, PF SLAM et SLAM basé sur des graphes.

Les travaux de Minnema constituent une avancée par rapport aux travaux de Mohamed A. Abdelhady (2017), où une analyse approfondie du problème SLAM est donnée, et identifie trois éléments nécessaires à la résolution du problème SLAM : la mesure du capteur, un modèle (modèle de mesure ou modèle de mouvement avec son Jacobien) et un modèle de bruit. Dans le travail de Minnema, un cadre modulaire pour le problème SLAM est également proposé.

Le Framework consiste en un "Back-End" qui utilise ces trois entrées en fonction du type de capteur utilisé pour résoudre le problème SLAM. Les différents capteurs sont classés en idiot étique, allo thétique absolu et allo thétique relatif.

La différence entre idiot étique et allo thétique est que les capteurs idiots étiques utilisent des données intrinsèques de la plate-forme comme des encodeurs, qui estiment de manière incrémentielle la position du robot, tandis que les capteurs allothétiques relatifs utilisent des caractéristiques externes ou des points de repère pour estimer la position.

La différence entre allo thétique absolu et relatif est le cadre de coordonnées utilisé, c'est-à-dire qu'un capteur allothétique absolu pourrait être un système GPS. Ces définitions de capteurs permettent le même traitement pour le même type de capteurs, ce qui améliore la réutilisabilité, mais cela implique que chaque capteur et chaque technique doivent être classés dans ces définitions afin d'être utilisés [14].

Dans Joost van Smoorenburg (2020), il est étudié comment implémenter la conception de Minnema dans RTAB-MAP, en analysant quels aspects de l'algorithme RTAB-MAP doivent être modifiés afin de fonctionner avec le cadre de Minnema. Les travaux de Smoorenburg donnent un premier aperçu de la manière dont la mise en oeuvre des algorithmes SLAM populaires n'est pas adaptée à la modularité et de la manière dont les composants de RTAB-MAP pourraient être MAPpés dans un cadre modulaire.

Mark te Brake (2021) se penche également sur la structure fonctionnelle des algorithmes populaires de pointe. Dans son travail, le flux de données de deux algorithmes SLAM basés sur des graphes visuels, RTAB-MAP et ORB-SLAM2, est analysé. Cette étude tente de trouver une structure commune entre les deux implémentations qui pourrait également être généralisée à d'autres solutions. À la suite de cette analyse, le travail de Mark propose également un ensemble de composants qui pourraient être utilisés sur un cadre modulaire.

Néanmoins, la conception proposée est limitée à un algorithme SLAM visuel et il n'est pas indiqué comment ces composants s'interfaceraient avec différents types de capteurs comme

les capteurs de distance. Au cours des deux dernières décennies, l'approche SLAM basée sur les graphes est devenue très populaire et plusieurs auteurs ont mis en œuvre des solutions utilisant ce paradigme, améliorant ses performances. Grisetti et al. (2010) donne un aperçu du système SLAM basé sur les graphes, en distinguant grossièrement les algorithmes chargés de créer le graphe (appelés "Front-End") et les méthodes qui se concentrent sur l'optimisation du graphe (ou "Back-End"). Les deux blocs sont connectés et échangent les informations nécessaires. Notez que le Back-End a besoin du pose-graph pour l'optimiser, et le Front-End profite des poses optimisées pour obtenir de meilleures estimations [14].

7. Conclusion

La Kinect est un appareil très récent qui offre encore de nombreuses opportunités pour les années à venir. Ainsi, dans ce premier chapitre, nous avons présenté une brève étude sur la Kinect. L'étude de ses caractéristiques prouve que c'est un appareil fiable. Son capteur RGB-D trouve un intérêt par rapport à une caméra classique qui se situe dans son image de profondeur, qui reste plus facile à analyser qu’une image couleur classique.

Le potentiel de la Kinect ne s’arrête pas à certains jeux et de nombreux projets utilisent déjà la Kinect, spécifiquement dans le domaine du traitement d’image et vision par ordinateur.

Dans le prochain chapitre, nous passerons à la conception et la modélisation utilisé dans ce projet.

Chapitre 02

Conception Et Description De Notre Système

1. Introduction

RTAB-MAP signifie cartographie basée sur l'apparence en temps réel. Il s'agit d'un approche SLAM basée sur un graphe basée sur la détection de fermeture de boucle basée sur l'apparence. Il vérifie la probabilité qu'une image provienne de l'emplacement précédent. Le SLAM graphiques a une meilleure précision que FAST SLAM. Ici, nous avons utilisé RTAB-MAP avec Kinect, qui est un capteur de vision stéréo avec caméra de profondeur pour déterminer instantanément la profondeur, ce qui permet d'économiser le calcul d'estimation. SLAM basé sur l'apparence signifie que l'algorithme utilisera les données obtenues à partir de capteurs de vision pour localiser la position du robot et cartographier simultanément le robot dans l'environnement. Les fermetures de boucle sont utilisées pour déterminer si le robot a déjà vu le cadre particulier avant. La fermeture de la boucle locale dépend de l'odométrie visuelle tandis que la fermeture de la boucle globale est indépendante de la pose estimée qui est l'odométrie visuelle. Ainsi, lorsque le robot se déplace, la carte s'agrandit et le nombre d'images comparées augmente à son tour. Il crée ainsi des cartes denses contrairement à ORB SLAM+.

Dans ce qui suit, nous allons décrire RTAB-MAP comme une solution open source, ainsi que les problèmes de modularité RTAB-MAP avec les motifs de conception, nous terminons par un organigramme de notre méthode basée sur le RTAB-MAP.

2. RTAB-MAP : Une solution SLAM open-source

RTAB-MAP (Real-Time Appearance-Based MAPING) : est une approche populaire du SLAM (Simultaneous Localization and MAPING) basée sur les graphes qui met en œuvre une fermeture de boucle visuelle.

RTAB-MAP est notre bibliothèque open source implémentant la détection de fermeture de boucle avec une gestion de la mémoire approche, limitant la taille de la carte afin que les détections de fermeture de boucle soient toujours traitées dans un délai fixé, répondant ainsi aux exigences en ligne pour des projets à long terme et à grande échelle cartographie de l'environnement. RTAB-MAP a depuis été étendu à une approche SLAM complète basée sur des graphes à utiliser dansdiverses configurations et applications.

En conséquence, RTAB-MAP a évolué dans une bibliothèque C++ autonome multiplateforme et un package ROS 3, piloté par des pratiques conditions.

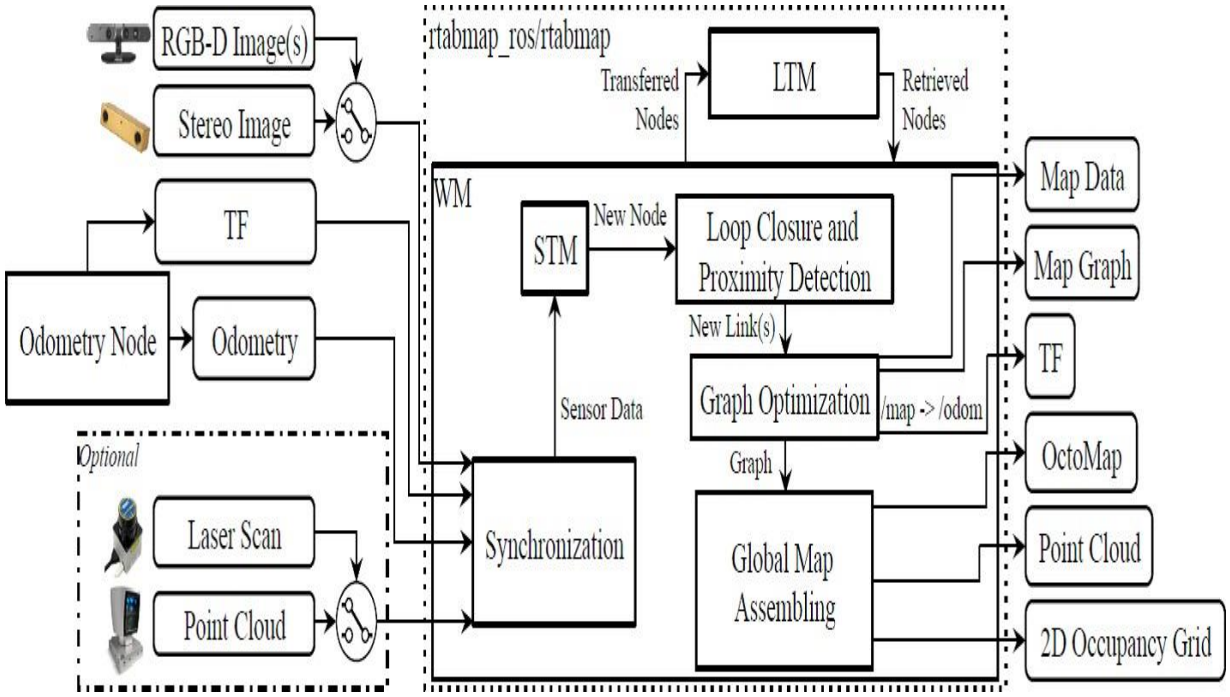


Figure II.1 : Schéma fonctionnel du nœud ROS RTAB-MAP [15]

La figure II.1 illustre les différents blocs fonctionnels de RTAB-MAP, tels qu'ils sont présentés dans le document suivant ; le travail de l'auteur. Ici, il est clair que l'algorithme a besoin d'au moins une source d'odométrie et une entrée d'image RVB-D ou stéréo. L'entrée du capteur laser sert comme entrée optionnelle de données de distance qui peuvent être soit un scan laser 2D soit un nuage de points 3D et peuvent être utilisés pour former des grilles d'occupation 2D ou 3D respectivement. RTAB-MAP comprend également une odométrie dans le cas où aucune odométrie externe n'est fournie. Le bloc principal de RTAB-MAP contient les éléments suivants système de gestion de la mémoire et les blocs de fermeture de boucle, de détection de proximité, le graphe et les modules d'assemblage de cartes globales. RTAB-MAP est capable de donner plusieurs sorties en temps réel : la carte générée dans différents formats (Octo MAP, grille d'occupation et 3D), la correction de l'odométrie sous forme de Tf (format ROS pour la transformation) et les données générées graphiquement avec et sans les informations du capteur. Le schema générale de RTAB-MAP [15].

2.1. Nœuds et liens dans RTAB-MAP

Les nœuds sont des instants déterminés dans le temps où les données du capteur sont reçues, les capteurs montés dans le robot déterminent la fréquence de création de nouveaux nœuds. Dans RTAB-MAP, toutes les données d'entrée sont synchronisées et stockées dans le même nœud, qui sera désormais appelé signatures [19], chaque signature peut stocker (en fonction de la donnée disponible du capteur) :

- ID : Horodatage unique.

- Poids : Importance de la signature. Utilisé pour le système de gestion de la mémoire.
- BoW : Mots visuels utilisés pour la détection de la fermeture des boucles et la mise à jour des poids. Ceci est également connu sous le nom comme signature de l'image.
- Grille d'occupation.
- Données du capteur.
- Pose : Entrée de l'odométrie.
    - Image RVB : Celle qui est utilisée pour obtenir les caractéristiques.
    - Image de profondeur : Utilisée pour trouver la position 3D des mots visuels.
    - Scanner laser : Utilisé pour les transformations de fermeture de boucle et les affinements de l'odométrie, et par le module de détection de proximité.

Toutes les signatures sont interconnectées par des "liens", qui représentent le corps rigide. La transformation entre les signatures et l'incertitude de la mesure comme une informationmatrice. Ces liens peuvent être stockés en 3 Dof (x, y, μ) ou 6 Dof (x, y, z, pitch, yaw, roll), selon :

Si l'espace est représenté en 2D ou en 3D, notez que les coordonnées euclidiennes sont utilisées. Tous lesliens contiennent les mêmes informations, mais RTAB-MAP a une convention d'appellation qui dépend de la nature du lien.

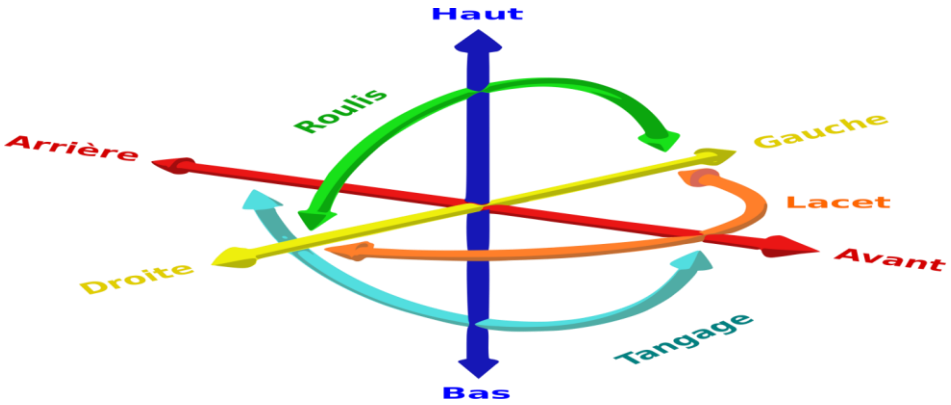


Figure II.2 : Les six degrés de liberté : x, y, z, pitch, yaw, roll.

Le module qui a fourni ces liens. Les types sont :

- Lien avec le voisin : Créé entre une nouvelle signature et la précédente.
- Lien de proximité : Ajouté lorsque deux signatures proches sont alignées ensemble à l'aide de données LiDAR.et l'alignement du scan.
- Lien de fermeture de boucle : Ajouté lorsqu'une fermeture de boucle est détectée entre la nouvelle signature et un dans le tas.
- Contrainte de repère : Ce lien n'est pas explicitement défini dans la documentation officielle de RTAB-MAP, mais il est présent dans les nouvelles fonctionnalités de l'algorithme. Il ne s'agit peut-être pas d'un lien dans la définition formelle que nous

avons, mais la contrainte qui relie une signature à un point de repère identifiable, dont la détection est possible dans les dernières versions de RTAB-MAP depuis 2019.

Notez que les données GPS ne sont pas stockées sous forme de lien. La raison en est que les données GPS est seulement utilisé comme une estimation préalable d'un nœud. Tous les liens sont utilisés comme contraintes pour l'optimisation du graphe.

A chaque fois qu'une fermeture de boucle ou un lien de proximité est créé, le graphe est optimisé pour réduire la dérive de l'odométrie. L'information du pose-graph est extraite du réseau de signatures et de transmis à l'optimiseur.

2.2.1. Nœud d'odométrie

RTAB-MAP peut mettre en œuvre un nœud d'odométrie qui est capable de calculer l'odométrie visuelle (VO) ou l'odométrie Lidar (LO) dans le cas où il n'y a pas de source externe d'odométrie, c'est-à-dire l'odométrie des roues, ou une autre source de VO ou LO[20], calcul de F2F (Frame To Frame) et F2M (Frame To MAP) dans le cas de VO et S2S (Scan to Scan) et S2M (Scan To MAP) dans le cas de LO.Odométrie visuelle

Pour l'approche de l'odométrie visuelle, la transformée entre le cadre de coordonnées de la caméra et le cadre du robot et une source d'images stéréo ou RVB-D sont nécessaires, L'image d'entrée est utilisée pour obtenir des caractéristiques visuelles, Celles-ci seront ensuite soumises à une étape de comparaison des caractéristiques, Obtenue dans l'image courante avec les images clés précédentes ou la carte de caractéristiques stockée, F2F et F2M respectivement, Une fois les caractéristiques correspondantes, une prédiction de mouvement est calculée

La figure II.3 présente l'odométrie visuelle de RTAB-MAP en utilisant deux couleurs pour différencier F2F (vert) et F2M (rouge). Il peut utiliser des caméras RVB-D ou stéréo comme entrées. tf est requis savoir où est placée la caméra sur le robot pour transformer l'odométrie de sortie dans le cadre de base du robot (par exemple, /base_link). Si la caméra est sur la tête du robot et la tête tourne, cela n'influence pas l'odométrie de la base du robot tant que tf entre le corps du robot et la tête du robot est également mis à jour.

Le processus fonctionne comme suit :

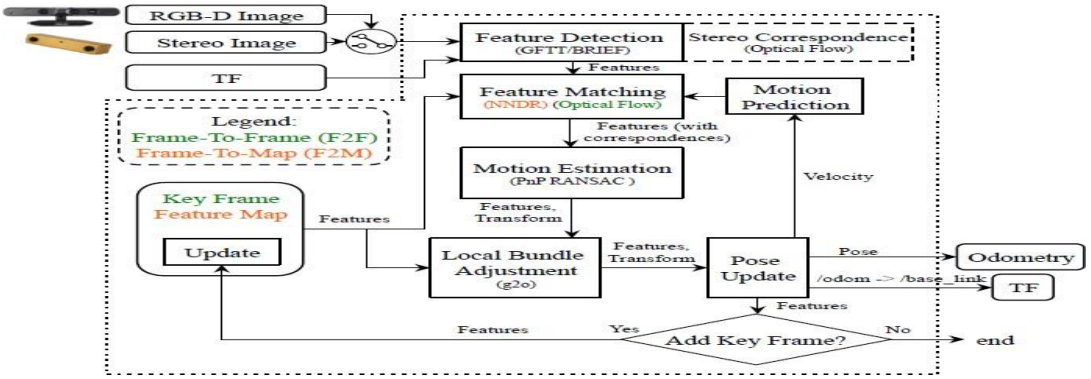


Figure II.3 : Schéma fonctionnel des nœuds ROS rgbd_odometry et stereo_odometry

* **TF** définit la position de la caméra par rapport à la base du robot et comme sortie pour publier la transformée d'odométrie de la base du robot.

2.2.1.1. Odométrie Lidar

L'odométrie Lidar est très similaire à celle présentée dans l'odométrie visuelle. Dans ce cas les données du capteur laser doivent être fournies en entrée ainsi que la transformation entre les coordonnées Lidar et le cadre de base du robot. En outre, l'odométrie externe comme l'odométrie des roues pourrait être ajouté pour améliorer la prédiction de mouvement, car l'odométrie Lidar peut perdre sa trace si elle n'est pas suffisamment de caractéristiques sont détectées, La première étape consiste à filtrer les données des nuages de points, puis à procéder à l'analyse de l'image avec enregistrement ICP du nuage de points actuel dans l'image clé précédente ou dans la carte, S2Sou S2M respectivement. Dans ce cas, la carte est un nuage de points formé par tous les scans enregistrés de chaque image clé. Après l'enregistrement ICP, la pose suivante de l'odométrie est obtenue et donnée comme une sous la forme d'une transformation actualisée et d'une pose avec incertitude. Comme pour la VO.

Si le rapport de correspondance de l'image actuelle est inférieur à un seuil prédéfini, l'approche de la nouvelle trame est considérée comme une trame clé.

Figure II.4 : capteur lidar 3D RS-LiDAR-32

La figure II.4 fournit le schéma fonctionnel de l'odométrie lidar, utilisant également deux couleurs pour différencier entre S2S (vert) et S2M (rouge). En utilisant une terminologie similaire à l'odométrie visuelle, une image clé fait référence à un nuage de points ou à un balayage laser. L'entrée de balayage laser est 2D comme le point l'entrée cloud peut être en 2D ou en 3D. Les scans laser peuvent présenter des distorsions de mouvement lorsque le robot se déplace pendant le scan. On suppose ici que ces distorsions sont corrigées avant de transmettre le scan à RTAB-MAP. Notez que si la fréquence de rotation du scanner laser est élevée par rapport à la vitesse du robot, les balayages laser auraient une très faible distorsion de mouvement et ainsi la correction peut être ignorée sans perte significative de précision d'enregistrement.

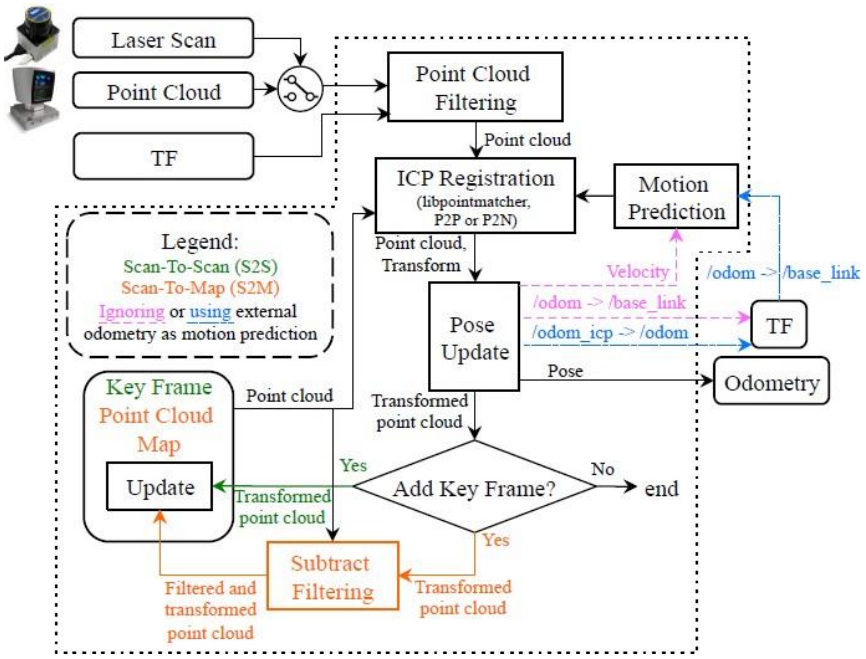


Figure II.5 : Schéma fonctionnel du nœud icp_odometry ROS.

* **TF** définit la position du lidar par rapport à la base du robot et en sortie pour publier le transformée d'odométrie de la base du robot.

Comme mentionné ci-dessus, RTAB-MAP est un système SLAM basé sur les graphes. Cela signifie que le système recrée l'environnement en utilisant des nœuds et des liens.

2.2. Gestion de la mémoire

Le système de gestion de la mémoire fonctionne au-dessus du noyau de RTAB-MAP. Ce système est en charge de gérer la manière dont les données de pose et de capteur sont stockées et de choisir celle qui sera utilisée pour exécuter les tâches suivantes ; la détection de proximité et la détection de fermeture de boucle. Le système de gestion de la mémoire est divisé dans trois mémoires principales : La mémoire à court terme (STM), la mémoire de travail (WM) et la mémoire à long terme (MLT). [21] les deux principaux objectifs de cette distinction sont les suivants est de séparer les données récemment obtenues de celles utilisées pour la fermeture de la boucle et de maintenir un faible niveau de sécurité nombre de signatures lors de la détection de fermeture de boucle pour rester dans les contraintes de temps réel.

- STM : Il peut être considéré comme un tampon de taille fixe qui traite et stocke les signatures les plus récentes, de sorte qu'ils n'affectent pas la détection de la fermeture de la boucle. C'est ici que la grille d'occupation est calculée et que toutes les informations des signatures sont assemblées.

Il existe une mémoire préalable appelé SensoryMemory (SM) dans lequel l'extraction et la réduction des caractéristiques est effectuées avant d'entrer dans la STM. Les caractéristiques extraites sont quantifiées en mots visuels à l'aide d'un BoW incrémentiel (BoW)[15].

- WM : Cette mémoire contient toutes les signatures qui sont candidates à une fermeture de boucle. Ce site est généralement stocké dans la mémoire dite RAM, de sorte que seules les signatures qui sont qui ne sont pas candidats au *transfert* restent dans le WM pour réduire l'utilisation de la mémoire et le traitement temps.
- LTM : Cette mémoire stocke tout le reste des signatures qui ne sont ni récentes ni candidates pour la fermeture de la boucle. D'après Labbe et Michaud (2013) les signatures sont stockées dans une base de données contenant le lien, l'ID des signatures et leur signature

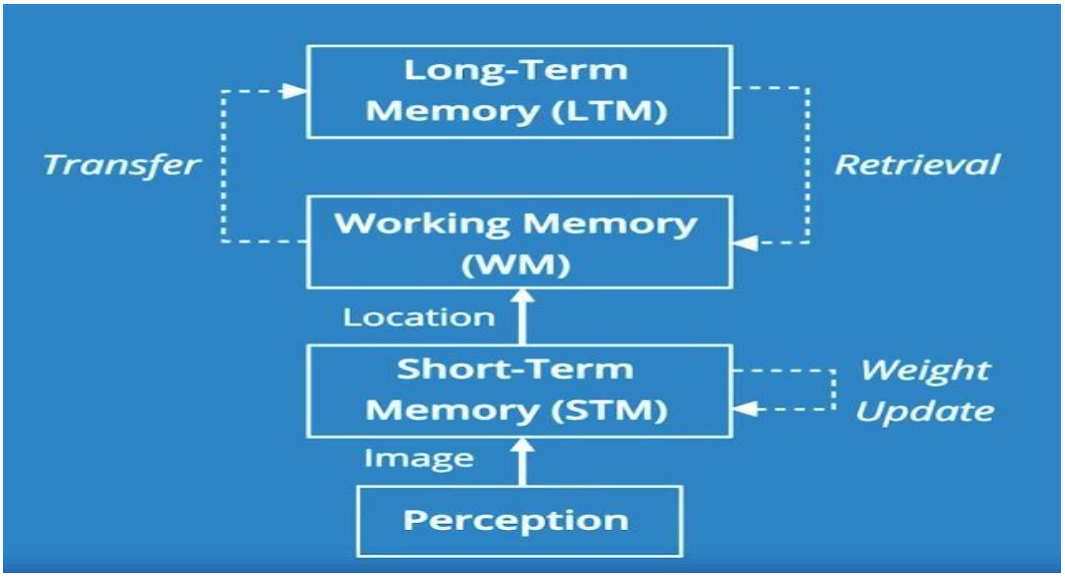


Figure II.6 : Modèle de gestion de la mémoire.

Les mémoires sont gérées par trois méthodes principales qui décident du stockage d'une signature dans une mémoire ou une autre :

- Répétition : Cette méthode s'ajoute à la STM, réduisant ainsi le nombre de signatures, qui entrent dans le WM et en mettant à jour les pondérations. Cette méthode utilise les signatures pour déterminer si deux signatures sont trop similaires. Si c'est le cas, il fusionne les données et met à jour le poids de la signature.
- Récupération : C'est l'une des méthodes qui opèrent entre la MM et la MLT. Une fois qu'une hypothèse de fermeture de boucle est acceptée, les signatures voisines dont la boucle est plus élevée, les probabilités de fermeture sont récupérées de LTM to WM. Cette méthode permet au système de mettre à jour le WM avec des signatures qui sont candidates à une fermeture de boucle [22].

- Transfert : A l'inverse de la méthode "Retrieval", celle-ci envoie les signatures les moins significatives. À la GLT pour un stockage à long terme. Cette méthode contient les critères permettant d'évaluer quelles signatures à transférer, qui se base sur deux heuristiques [19] : les signatures plus anciennes avec moins de poids ont la priorité pour être transférées à LTM et les signatures qui sont utilisés dans la planification de la trajectoire doivent rester dans WM.

2.3. Détection de la fermeture de la boucle

La détection de la fermeture de boucle dans RTAB-MAP est basée sur l'approche communément connue du BoVW. Dans cette méthodologie, les caractéristiques de chaque image-clé détectée sont combinées dans l'image-clé. BoW, et chaque image-clé est quantifiée en une représentation utilisant les mots visuels résultants.

Un filtre bayésien discret est utilisé pour garder la trace des fermetures de boucle en estimant la probabilité que l'emplacement actuel a avec les signatures disponibles dans WM. Si la probabilité est plus élevée que d'un certain seuil, cette hypothèse est considérée comme un candidat à la fermeture de boucle et le pose graph est mis à jour en conséquence.

2.4. Optimisation

Dans RTAB-MAP, le modèle du monde à l'intérieur de WM est optimisé pour représenter avec précision la trajectoire réalisée par le robot. Pour réaliser l'optimisation, les informations de pose sont extraites des signatures et introduites dans un optimiseur non-linéaire. Actuellement, RTAB-MAP implémente plusieurs des optimiseurs les plus populaires comme g2o, gtsam et TORO, L'optimisation est réalisée comme une optimisation de la pose.

2.5. Dépendances dans l'architecture de RTAB-MAP

L'implémentation de RTAB-MAP a beaucoup de dépendances entre les modules, ce qui affecte sévèrement sa réutilisabilité. Ils ont fait une analyse approfondie de ces dépendances, et il a été d'une grande aide pour comprendre l'architecture deRTAB-MAP pour ce projet dans la section, je reproduis certains de ses chiffres et paraphrase les dépendances qu'il a trouvées, telles qu'elles sont été utilisé plus tard dans l'analyse [21].

La figure II.7 montre les dépendances du module de création de signature décrit dans le tableau ci-dessous.

Il est possible de voir que ce module a besoin de données d'apparence et d'odométrie, le module de création de signatures s'interface également avec les modules d'optimisation de la

pose et du graphique et de transfert. Pour notifier les changements dans la mémoire et le STM pour raconter les signatures nouvellement créées.

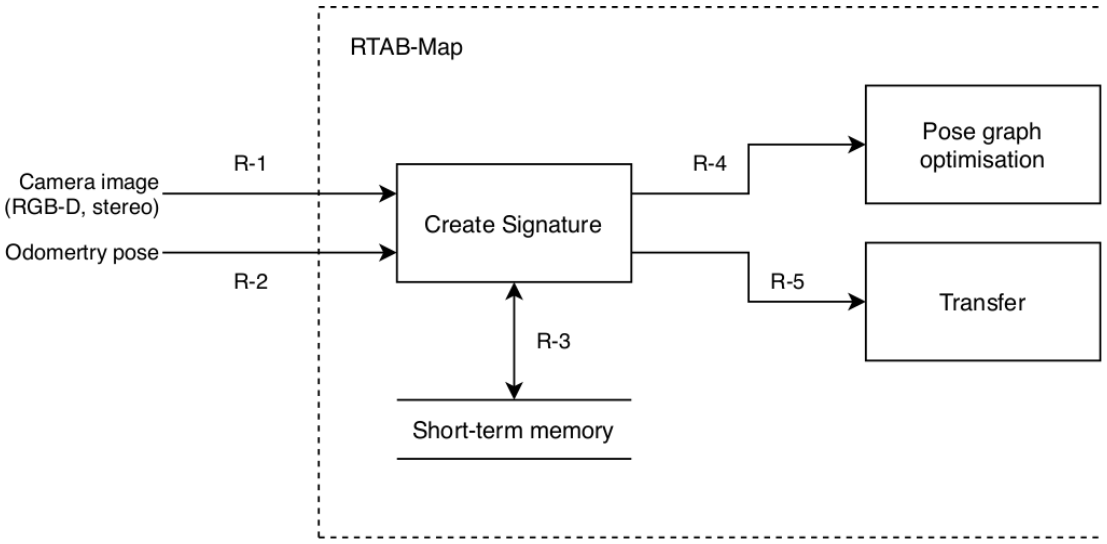


Figure II.7 : Dépendances présentes dans le module de création de signature [15].

Dépendance	**Direction**	**Description**
R-1	In	Les données visuelles.
R-2	In	La pose d'odométrie.
R-3	In	Les données de la signature précédente pour le couplage avec la nouvelle.
R-3	Out	Stockage de la nouvelle signature.
R-4	Out	Flag : Les données de stockage ont été modifiées.
R-5	Out	Numéro de sortie de la signature nouvellement ajoutée.

Tableau II.1 : Description des dépendances présentes dans le module de création de signature.

Le module de répétition envoie un type d'information similaire à l'optimisation du graphe de pose et à l'optimisation de l'image, modules de transfert notifiant la modification des Signatures en stockage. Ces modules permettent également échange des informations avec le STM pour recueillir les informations des deux Signatures les plus récentes ou supprimer celle qui considère qu'il n'y a pas d'information nouvelle suffisante. Ces dépendances sont présentées à la figure II.8 et décrits au tableau II.2. Dans cette figure, on peut également voir la dépendance entre le STMet le WMen déplaçant la signature la plus ancienne après la Répétition.

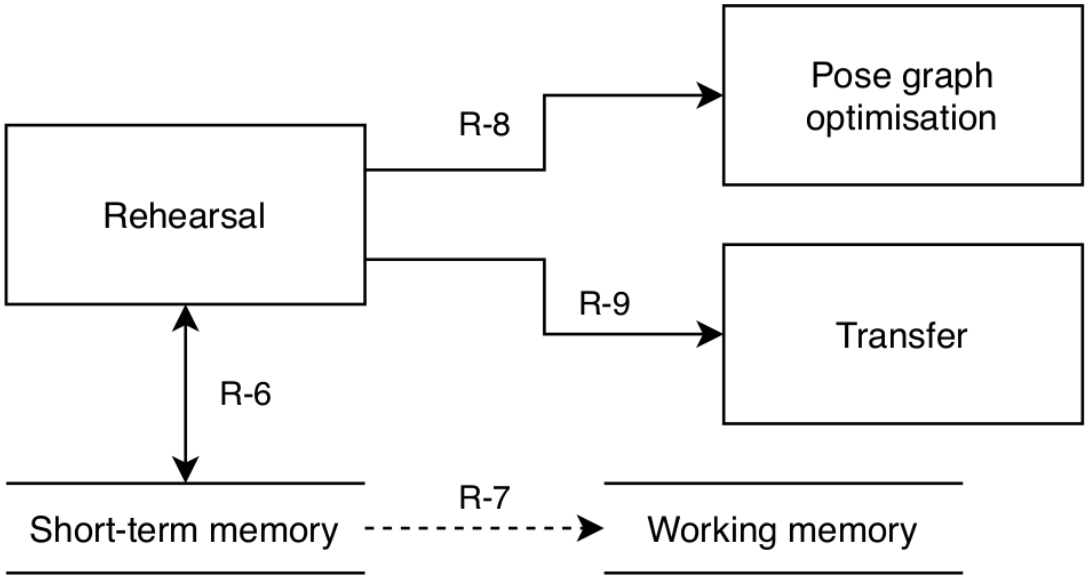


Figure II.8 : Dépendances présentes dans le module Répétition [15].

Dépendance	**Direction**	**Description**
R-6	In	Données de la plus récente et de la précédente signature.
R-6	Out	Supprimer la signature la plus récente et mettre à jour son poids si nécessaire.
R-7	-	Déplacer la signature la plus ancienne vers la mémoire de travail après la répétition.
R-8	Out	Flag : Les données de stockage ont été modifiées.
R-9	Out	Numéro de sortie de la signature supprimée.

Tableau II.2 : Description des dépendances présentes dans le module Répétition.

La figure II.9 montre les dépendances du module de filtre de Bayes, décrites dans le tableau II.3.

Ce module récupère les données de signature du STM et du WM en recherchant une fermeture de boucle, et notifier les modules de récupération et de génération de liens de fermeture de boucle lorsqu'une fermeture de boucle est détectée.

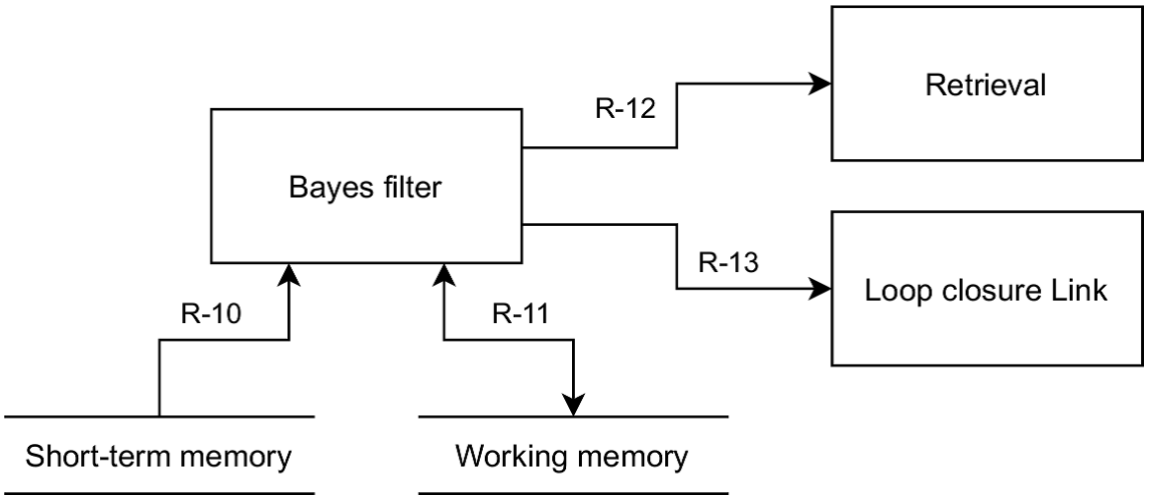


Figure II.9 : Dépendances présentes dans le module de filtre de Bayes [15].

Dépendance	**Direction**	**Description**
R-10	In	Les données d'apparence de la plus récente signature.
R-11	In	Les données d'apparence de toutes les signatures dans WM.
R-11	Out	Virtual Signature Storage.
R-12	Out	Cette signature est candidate à une fermeture en boucle
R-13	Out	Cette signature forme une boucle fermée avec la signature actuelle

Tableau II.3 : Description des dépendances présentes dans le module du filtre de Bayes.

Les dépendances du module de recherche sont illustrées à la figure 2.9 et décrites dans la section "Dépendances".

Tableau II.4 Le module de récupération reçoit des informations du module de filtrage de Bayes, du STM et du module d'analyse des données.

Le WM pour sélectionner les signatures qui seront récupérées dans le LTM. Une fois qu'une signature est récupérée, elle est supprimée de la MLT et stocké dans la MW. Ensuite, l'optimisation du graphe de pose et les modules de transfert sont notifiés de la récupération, ce qui déclenche une éventuelle optimisation du graphe et interdisant le transfert des Signatures qui ont été récemment récupérées.

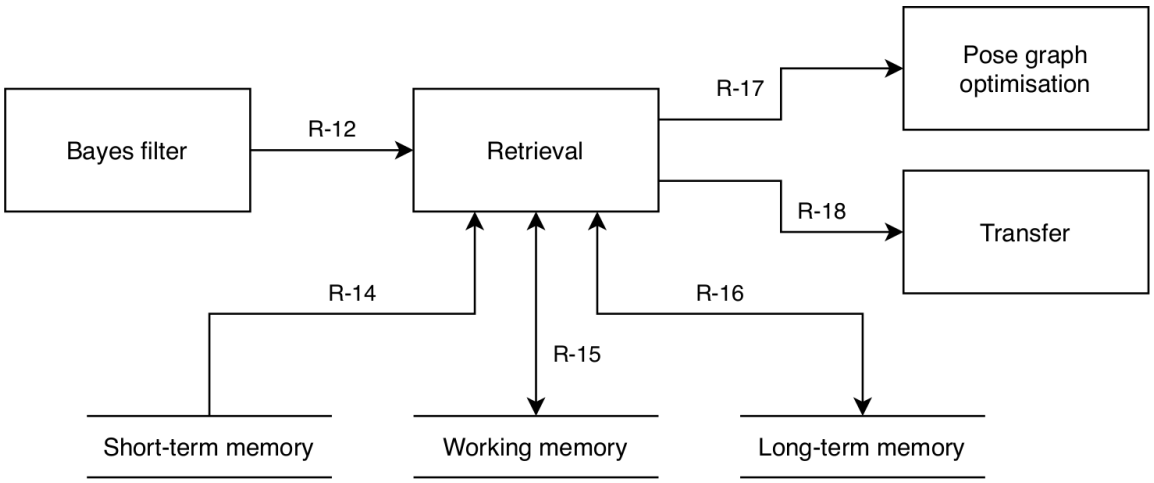


Figure II.10 : Dépendances présentes dans le module de récupération

Dépendance	**Direction**	**Description**
R-12	In	Cette signature est un candidat pour une fermeture en boucle.
R-14	In	Les données des signatures en STM pour la sélection d'une récupération possible.
R-15	In	Data of Signatures in WM : pour sélectionner une récupération possible.
R-15	Out	Stocker les signatures récupérées dans le MLT.
R-16	In	Les données des signatures en MLT pour sélectionner une récupération possible.
R-16	Out	Supprimer les signatures récupérées dans le MLT.
R-17	Out	Flag : Les données de stockage ont été modifiées.
R-18	Out	Montant des signatures récupérées. Liste des Signatures qui ne peut pas être transféré à LTM.

Tableau II.4 : Description des dépendances présentes dans le module de recherche.

Le module de génération de liens de fermeture de boucle reçoit l'ID candidat du module de filtre de Bayes, récupère les données de pose de la signature correspondante à partir du WM et de la signature actuelle de la STM. Une fois que le lien de fermeture de la boucle est calculé, les signatures concernées sont mises à jour, avec le nouveau lien et un drapeau est envoyé au module d'optimisation pose-graphe pour déclencher l'optimisation.

Ces dépendances sont visibles à la figure II.10 et décrites au tableau II.5.

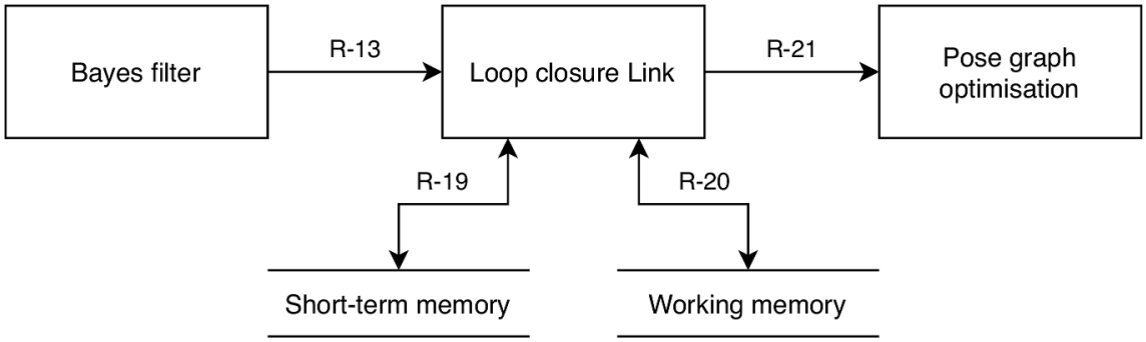


Figure II.11 : Dépendances présentes dans le module de fermeture de boucle

Dépendance	**Direction**	**Description**
R-13	In	This Signature forme une boucle avec la signature actuelle.
R-19	In	Données de signature actuelles pour le calcul de la contrainte de fermeture de boucle.
R-19	Out	Mémoriser la fermeture de la boucle Lien.
R-20	In	Signature candidat à la fermeture de boucle pour l'informatique la contrainte de fermeture de la boucle.
R-20	Out	Mémoriser la fermeture de la boucle Lien.
R-21	Out	Flag : Les données de stockage ont été modifiées.

Tableau II.5 : Description des dépendances présentes dans le module de fermeture de boucle.

Les dépendances du module d'optimisation du graphe de pose sont illustrées à la figure II.12 et décrites dans le tableau ci-dessous.

Dans le tableau II.6 le module d'optimisation de pose-graphes reçoit des drapeaux de la signature, Les modules de génération, de répétition, de récupération et de génération de liens de fermeture de boucle pour déclencher l'optimisation, et les informations de pose-graphes provenant du STM et du WM. Une fois le graphe optimisé,

Le module d'optimisation des poses et des graphes envoie les informations sur les poses optimisées à l'unité de gestion de l'information.

STM et le WM pour mettre à jour les Signatures.

Dépendance	Direction	Description
R-4	In	Flag : Les données de stockage ont été modifiées.
R-8	In	Flag : Les données de stockage ont été modifiées.
R-17	In	Drapeau : Les données de stockage ont été modifiées.
R-21	In	Drapeau : Les données de stockage ont été modifiées.
R-22	In	Données de Signatures en STM pour construire un pose-graph.
R-22	Out	Store optimisé pose.
R-23	In	Data of Signatures in WM pour construire un pose-graph.
R-23	Out	Positions optimisées du magasin extérieur.

Tableau II.6 : Description des dépendances présentes dans le module d'optimisation du graphe de pose.

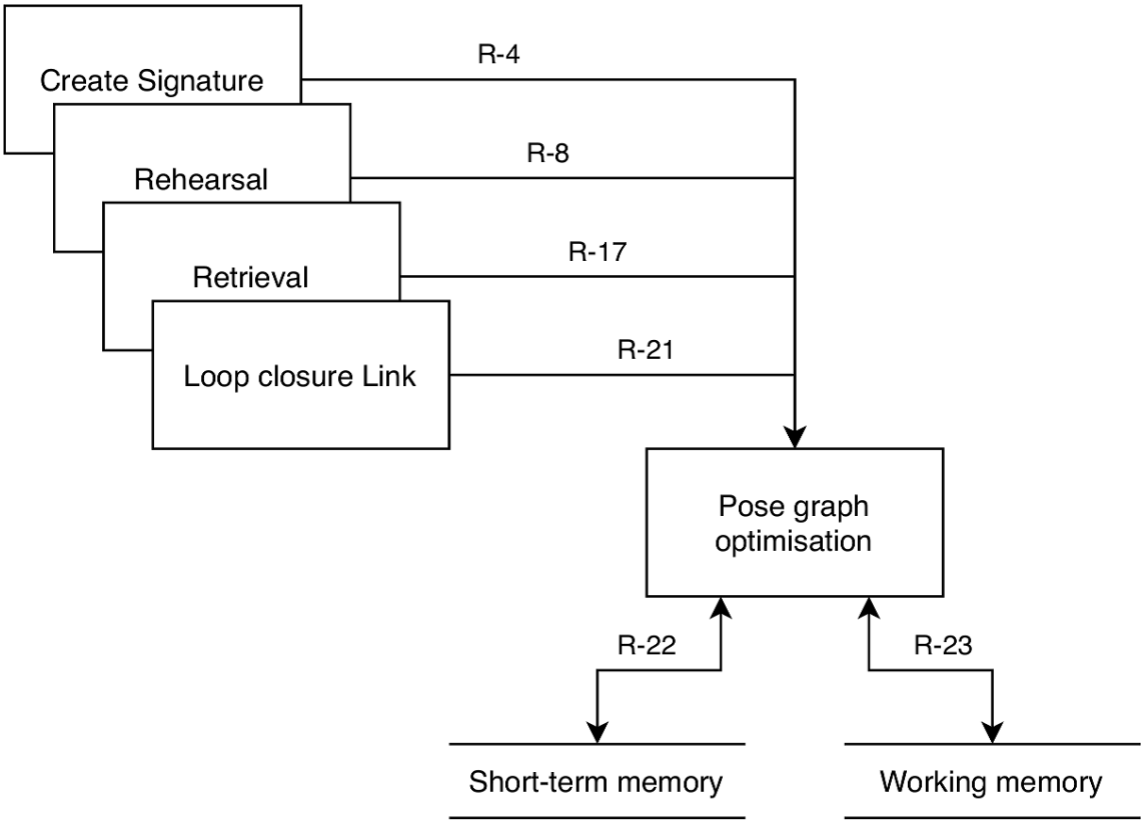


Figure II.12 : Dépendances présentes dans le module d'optimisation du graphe de pose

Enfin, la figure II.13 montre les dépendances du module de transfert, décrites dans le tableau II.7.

Le module Transfert reçoit des informations des modules Génération de signature, Répétition et Récupération, pour décider quelles Signatures ne peuvent pas être transférées. L'information du temps écoulé d'une itération est utilisé pour décider si une Signature doit être transférée. Si le temps écoulé est supérieur à un certain seuil, le module reçoit les données des Signatures dans le STM et le WM pour sélectionner la signature qui doit être transférée et supprimer la signature sélectionnée de la base de données, le WM pour le stocker dans le LTM.

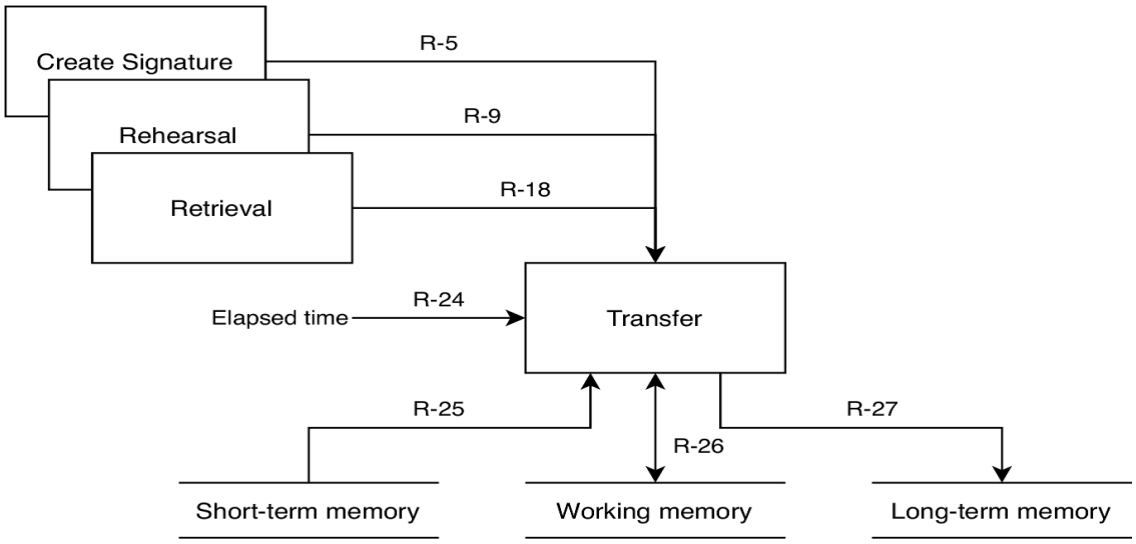


Figure II.13 : Dépendances présentes dans le module Transfert

Dépendance	**Direction**	**Description**
R-5	In	Nombre de la signature nouvellement ajoutée.
R-9	In	Numéro de la signature supprimée.
R-18	In	Montant des Signatures retrouvées. Liste des Signatures qui ne peut pas être transféré à LTM.
R-24	In	Temps écoulé depuis le début de l'itération en cours.
R-25	In	Les données des signatures en STM pour sélectionner un éventuel transfert.
R-26	In	Les données de signatures en WM pour sélectionner un transfert possible.
R-26	Out	Supprimé Signatures transféré à LTM.
R-27	Out	Stocker les signatures transférées dans le LTM.

Tableau II.7 : Description des dépendances présentes dans le module Transfert.

3. Problèmes de modularité RTAB-MAP

RTAB-MAP est une solution SLAM développée depuis plus de 10 ans maintenant et jouit d'une grande réputation auprès de la communauté scientifique dans le secteur. Il a commencé comme un algorithme SLAM basé sur un graphique visuel avec une mémoire basée sur le poids système de gestion, et supporte aujourd'hui un grand nombre de frontaux et d'optimiseurs différents. La plupart de la structure fonctionnelle de RTAB-MAP est disponible dans ses documents de recherche et une brève.

La base de code de RTAB-MAP manque de documentation officielle et une grande partie de ses fonctionnalités est principalement implémenté dans deux grandes classes, ce qui rend la réutilisation du code très complexe.

L'un des principaux objectifs est d'améliorer modularité dans l'un des principaux Framework SLAM open-source, après avoir sélectionné RTAB-MAP pour cette tâche en raison de sa popularité et des travaux antérieurs disponibles avec le Framework

• La différenciation STM et WM : La différenciation entre le STM et le WM augmente fortement la complexité du code, générant de nombreuses dépendances entre modules (R-3, R-4, R-6, R-7, R-10, R-14, R-19, R-22 et R-25).

La fonctionnalité ajoutée par ce la séparation pourrait être mise en œuvre d'une manière différente fusionnant le STM et le WM, réduisant significativement le montant des dépendances. Par exemple, la méthode de répétition n'a besoin d'accéder qu'aux deux dernières signatures ajoutées à la mémoire (R-6), qui pourraient également être fait sans l'existence de la STM. L'optimisation dugraphe de pose ne distingue pas entre les mémoires et optimise toutes les signatures dans leur ensemble, il n'est donc pas nécessaire pour dupliquer l'accès aux données (actuellement effectuépar R-22 et R-23) et la dépendance Le R-22 pourrait être inclus dans le R-23. En revanche, d'autres dépendances comme R-10 et R-25 bénéficier de l'existence de la STM (en excluant les signatures les plus récentes pour détection de fermeture de boucle et pour le transfert), mais laresponsabilité de différencier Les signatures récemment générées et plus anciennes pourraient être intégrées dans la boucle de fermeture détection et les modules de transfert eux-mêmes. Cette modification doit être correctement conçu pour ne pas modifier la base de code actuelle affectant d'autres fonctionnalités, mais il serait simplifié considérablement l'architecture du Framework [21].

Le système de fermeture de boucle : Une autre fonctionnalité de RTAB-MAP qui nuit à sa modularité est la détection de fermeture de boucle et la génération de contrainte de fermeture

de boucle. A ses origines, RTAB-MAP était un système basé sur l'apparence et pensait même qu'il prend désormais en charge des capteurs comme les Lidar pour le raffinement de la position et l'odométrie, la technique de détection de fermeture de boucle reste basée sur l'apparence depuis le début [15].

Ce fait rend RTAB-MAP dépendant d'entrée de données visuelles (R-1), rendant obligatoire le flux de données visuelles. De plus, il y n'y a pas d'option pour remplacer le système de fermeture de boucle et le cadre ne fournit pas avec une interface pour la saisie de contraintes supplémentaires de fermeture de boucle. Ce problème affecte la réutilisabilité du cadre pour les plates-formes qui ne montent pas de caméra, ou pour les robots qui travailler dans des environnements dans lesquels la caméra n'est pas une source fiable de données pour la fermeture de la boucle détections.

Dépendances dans le système de gestion de la mémoire : Le système de gestion de la mémoire (les modules de répétition, de transfert et de récupération) utilise des signaux qui portent des drapeaux ou des valeurs de comptage (dépendances R-4, R-5, R-8, R-9, R-17, R-18 et R-21) à échanger des informations sur la création ou la modification de Signatures. Par exemple, il informe Lors de la génération d'une nouvelle Signature, le nombre de Signatures transférées d'une mémoire à une autre, ou la nécessité de déclencher une optimisation de graphe. Ces interfaces augmentent le nombre de dépendances entre les modules et rend la réutilisation du code plus complexe. Pour simplifier l'architecture actuelle, les modules utilisant ces signaux pourraient extraire les informations par elles-mêmes en accédant à la mémoire. Encore une fois, cette modification doit être correctement conçu pour fonctionner avec la version actuelle du Framework.

Un autre aspect du système de gestion de la mémoire qui affecte la modularité est la dépendance du module Rehearsal aux données d'apparence. Cette dépendance apparaît avec l'utilisation du système de pondération visuelle, qui ne fonctionne qu'avec une entrée visuelle en comparant le nombre de mots visuels appariés. L'utilisation de ces poids crée une dépendance indirecte dans le d'autres modules de gestion de la mémoire, Transfer and Retrieval, qui utilisent ce système de pondération pour décider quelles signatures doivent rester dans Wou lesquelles doivent être stockées dans LTM. Cette dépendance aux données visuelles affecte la réutilisabilité du Framework dans SLAM, car même s'il prend en charge d'autres entrées de capteur comme Lidar, le cadre récupérera le saisie de données visuelles en levant une erreur.

De plus, lors de l'étude de RTAB-MAP un autre problème de modularité concernant le pos graphe l'interface de génération a été trouvée. La création de signature est le seul module

documenté qui accepte les entrées pour la génération du graphe. Il n'a que deux entrées possibles : odométrie et visuelle ou des données Lidar. L'entrée odométrie est fine, car elle permet la génération externe de l'odométrie, contribuant à la modularité. Mais le problème est qu'il n'y a pas d'entrée pour la fermeture de boucle ou des contraintes de proximité, ce qui néglige la possibilité d'avoir une boucle externe de bouclage ou de proximité systèmes de détection, et rend obligatoire l'utilisation de ceux intégrés dans RTAB-MAP.

Les paradigmes SLAM les plus courants étaient la conception proposée par Minnema. Cependant, cette conception n'était pas compatible avec certaines fonctionnalités de RTAB-MAP. Pour cette raison, l'étude de EKF et PF SLAM ont été laissés pour d'autres travaux [22].

- organigramme de notre méthode basée sur le RTAB

Dans cette section, la conception du cadre modulaire de la carte RTAB sera construite sur la base du motif décrit ci-dessus

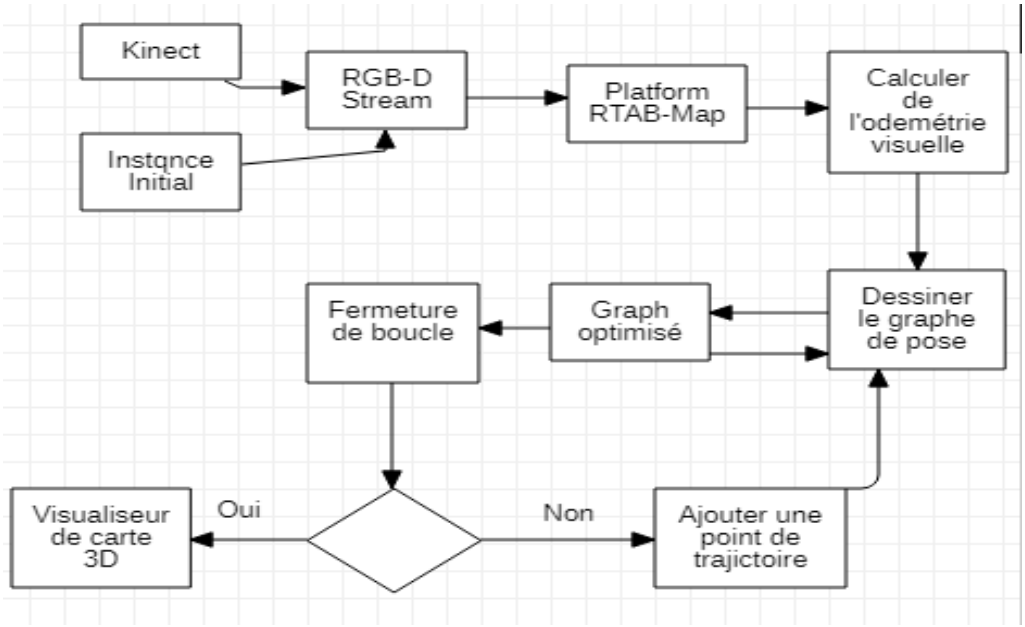


Figure II.14 : organigramme de notre méthode d'acquisition automatique de la carte 3D.

Dans cet organigramme le Kinect scanner l'image qui trouve dans l'instance initial et entrer dans RGB-D Stream et entrer dans la plateforme du RTAB-MAP, puis calculer l'odométrie visuelle et dessiner le graphe de la pose par rapport la position du Kinect et on a extraire un graphe optimisé comme résultat, à la fin de travail et par rapport au point de trajectoire initial la fermeture de boucle faire un choix. Si la photo est la même, la carte 3D apparaitre si non on ajoute une autre point de trajictoire.

Cet organigramme est basé sur les hypothèses suivantes :

• Il existe un seul graphe de pose avec une origine globale unique (les cadres de coordonnées relatives être exprimée à l'aide de cette référence) formée par les entités et les facteurs définis.

• Le pose-graphe forme un système surdéterminé, ce qui signifie qu'il y aura plus de contraintes que les nœuds.

Le développeur est responsable de fournir au Framework les bonnes informations. Si les deux conditions sont satisfaites, le cadre générera le pose-graphe, l'optimisera et représentera la trajectoire. Le Framework est également capable de fournir au Front-End l'optimisation poses pour des estimations plus précises.

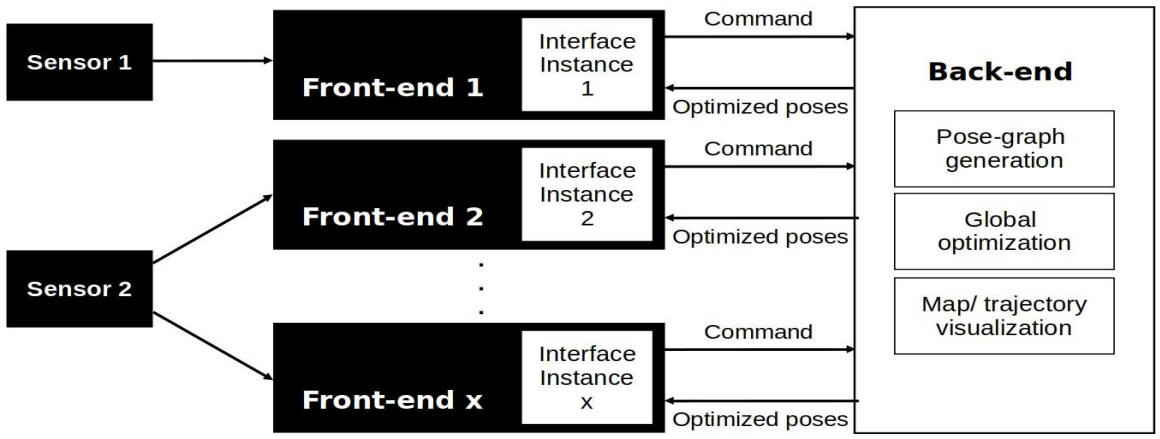


Figure II.15 : Aperçu général de l'architecture du Framework [15]

3.1. Interface Front-End et communication avec le Back-End

L'interface est un bloc conçu pour être intégré dans les Front-Ends et est celui en charge d'établir le canal de communication entre le Front-End et le Back-End pour permettre interaction. L'objet d'interface établira le canal de communication avec le Back-End dès que le processus est lancé. Une fois le canal de communication établi, l'interface recevra les requêtes du Front-End et enverra les commandes au Back-End.

Chaque frontal aura sa propre interface. L'interface envoie non seulement les commandes au Back-End, mais stocke également les méta-informations des Front-Ends. Par exemple, l'interface stocke les informations de la numérotation de chaque élément du graphique. Chaque frontal a sa propre numérotation système, mais ils doivent être unifiés au système de numérotation du Back-End, donc cette tâche est réalisée par l'interface. Un exemple de ceci peut être vu dans la figure II.16, où il y a deux frontaux connectés au Framework envoyant des informations de points de repère détectés dans l'environnement. Dans cet exemple, les deux frontaux utilisent les mêmes informations d'odométrie (nœuds dans blancs reliés par des flèches), mais ils détectent différents types de points de repère (représentés en rouge étoiles

pour le Front-End 1 et vertes pour le Front-End 2). L'interface connectée à chaque frontend reçoit les données de point de repère (flèche noire épaisse) et met à jour les identifiants des points de repère dans un mode FIFO (First In First Out), en maintenant une cohérence globale pour la numérotation.

Les points de repère avec des identifiants mis à jour sont ensuite envoyés au Back-End pour les ajouter au pose-graph.

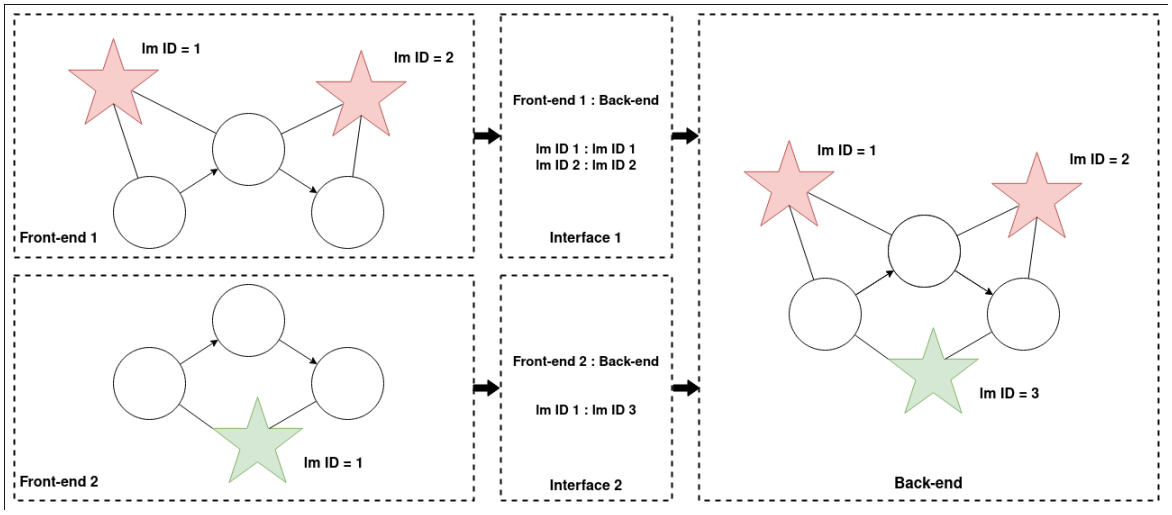


Figure II.16 : Exemple de gestion de numérotation d'interface avec repères.

3.2. Génération de graphes de pose

Le Back-End est conçu pour prendre en charge les fonctionnalités les plus essentielles de SLAM, étant l'une des plus importants la génération et le stockage du pose-graph. Le graphe de pose Le module de génération est celui qui effectue cette tâche en recevant les commandes envoyées par l'interface.

Le graphe généré est composé des trois éléments : nœuds, bords et points de repère. Dans les lignes suivantes, il est expliqué comment ces éléments sont utilisés pour former le graphe de pose utilisé par notre Back-End.

• Nœuds : les nœuds sont des éléments du graphique qui représentent des informations spatiales, comme des poses de le robot à différents instants. Ces éléments sont identifiés par un identifiant unique. Les données spatiales stockées par un nœud sont en coordonnées globales, par rapport à un référentiel global (généralement la pose initiale {0,0,0}), et ces données sont utilisées comme estimation initiale lors de l'optimisation du graphe de pose.

• Repères : Les repères sont des éléments qui représentent des caractéristiques uniques dans l'environnement. Ils sont liés au nœud dans lequel ils ont été détectés, en gardant une trace de tous les nœuds qui a observé le même point de repère.

Ces éléments utilisent également un ID. Repères sont également utilisés pour déclencher des optimisations de graphe de pose lorsqu'un point de repère connu est revisité.

• Arêtes : Ces éléments représentent des contraintes spatiales entre deux nœuds ou entre un nœud et un point de repère. L'information spatiale stockée par les arêtes est relative aux deux éléments connectés du graphe. Avec les informations spatiales, les bords stockent l'incertitude de l'observation qui l'a généré.

Le graphe de pose pourrait utiliser plus d'informations, comme des données cinématiques pour ajouter de la redondance et faire le système plus robuste. D'autresauteurs ont étudié l'ajout de différents types de facteurs dans le problème SLAM, mais afin de tester le noyau fonctionnalité de RTAB-MAP, nous n'aurons qu'à considérer les éléments les plus essentiels dans graphbased CLAQUER [23].

3.3. Optimisation globale

Le Back-End contient également un module d'optimisation de graphe de pose, qui est une partie essentielle d'un Cadre SLAM. L'optimiseur est capable d'accéder au graphe de pose en mémoire pour utiliser le poser des données lors de l'optimisation. L'optimisation est déclenchée dès qu'une contrainte de fermeture de boucle est ajouté au pose-graph, ou en d'autres termes, lorsqu'une nouvelle contrainte est ajoutée entre deux nœuds non consécutifs.

Pendant l'optimisation, l'erreur accumulée est réduite en utilisant la minimisation des moindres carrés. Pour résoudre le problème de minimisation, il existe plusieurs algorithmes disponibles dans les bibliothèques de pointe. Au cours de cette thèse, une seule bibliothèque sera implémentée, mais le module d'optimisation est détaché du reste pour implémenter facilement d'autres solutions.

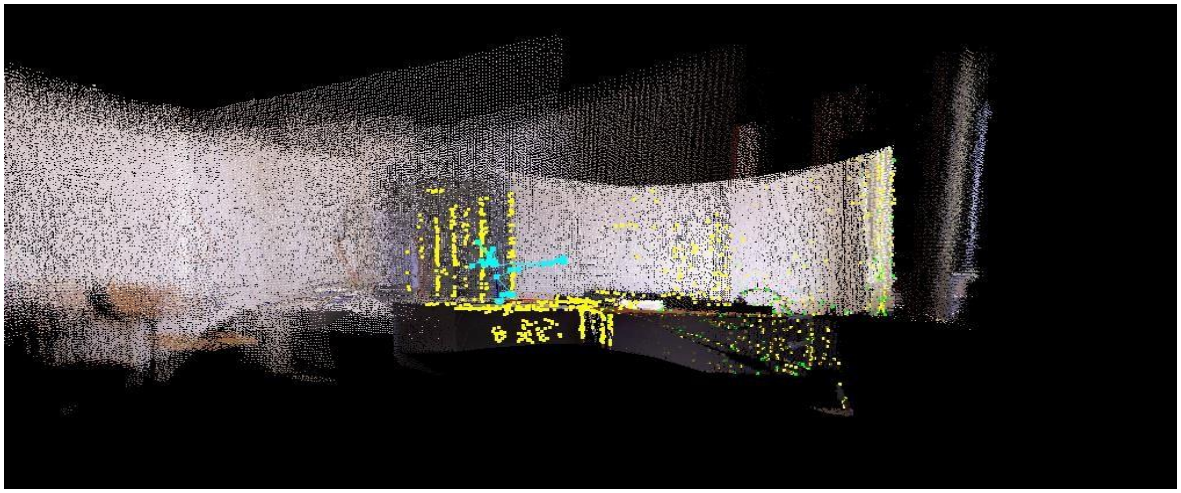


Figure II.17 : un module d'optimisation de graphe de pose.

3.4. Visualisation de la trajectoire

Le Framework comprend également un outil de visualisation. Ce visualiseur sert au débogage, car il trace la trajectoire en temps réel, mais il sert aussi pour les tests. Le Framework peut enregistrer les données générées pose-graph dans un fichier pour son utilisation dans l'étape d'évaluation.

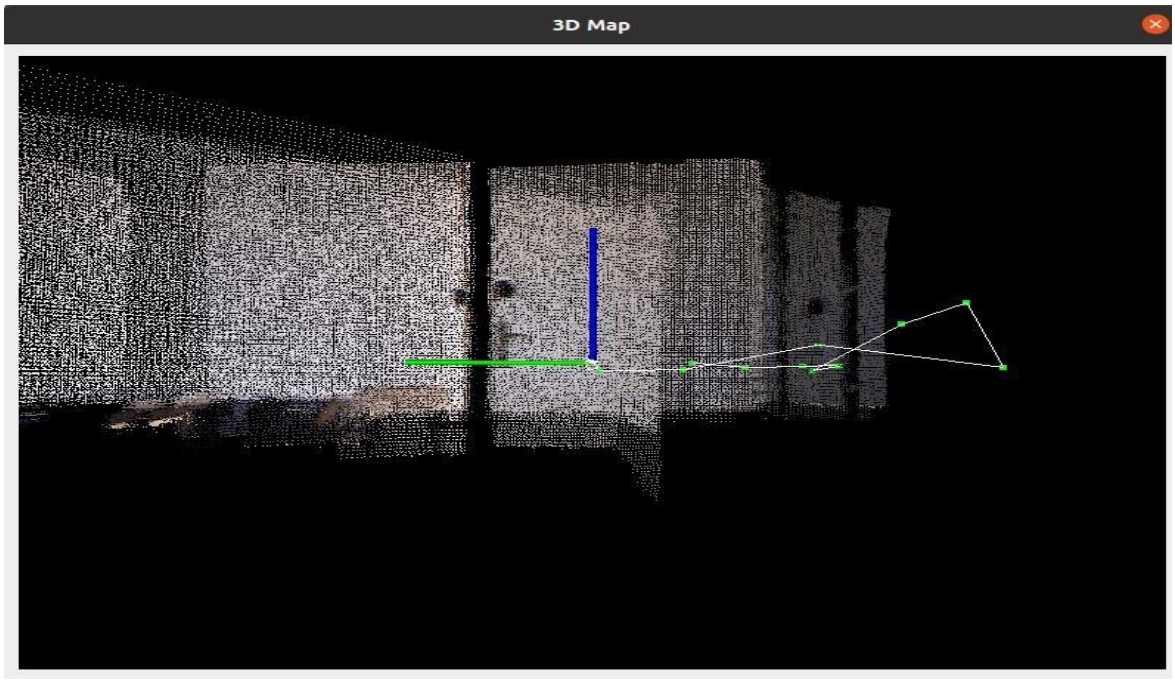


Figure II.18 : Le Framework comprend également un outil de visualisation.

4. Conclusion

Dans ce chapitre, nous avons cerné les objectifs de notre système. Cette phase nous a permis de bien élaborer les patterns de conception utilisés pour concevoir notre système. Par la suite, nous avons passé à la conception détaillée pour expliquer le déroulement de notre système. Ensuite, nous avons conçu quelques problèmes de modularité RTAB-MAP.

Dans le prochain chapitre, nous passerons à une description de l'état de l'implémentation du projet.

Chapitre 03

Implémentation

1. Introduction

Ce chapitre constitue la dernière étape conduisant à la réalisation de notre logiciel. L'étape d'implémentation consiste à traduire le résultat obtenu lors de l'étape de conception en un programme ou logiciel informatique exécuté sur une machine en utilisant les outils de programmation adaptés au problème à traiter.

Dans ce qui suit, nous allons décrire les composants de l'environnement de travail, ainsi que les outils d'implémentation et on donnera quelques exemples d'interfaces qui ont été réalisés dans notre logiciel.

2. Environnement de développement

La réalisation de ce projet a nécessité l'utilisation d'un nombre d'outils et de technologies cités ci-dessous.

2.1. Environnement matériel

La puissance de traitement utilisée est constituée d'un MSI G60 ordinateur portable avec un processeur Intel Core i7-4720HQ à 2,6 GHz avec 8 cœurs, 16 Go de mémoire de travail et un GPU Nvidia GTX 960M.

L'ordinateur portable fonctionne sur Ubuntu 20.04 (Focal Fossa) et utilise ROS Noétique. Le simulateur retenu est le simulateur Gazebo basé sur ROS dans sa 11ème version (gaz (2020)). Gazebo est capable de simuler des environnements 3D et dispose de nombreux plugins disponibles, par exemple, le plugin d'entraînement différentiel est capable de recevoir des commandes de vitesse et des informations d'odométrie en sortie, ce qui est très utile pour simuler un robot téléopéré.

A) Le Kinect

L'ensemble est installé sur un socle motorisé permettant de faire pivoter les capteurs optiques et le projecteur infrarouge afin d'adapter le champ de vision du périphérique en fonction du positionnement des utilisateurs. Début 2012, Microsoft commercialise une version Kinect pour Windows, le capteur de profondeur ayant été modifié pour pouvoir fonctionner à une distance plus proche. De plus, Microsoft a publié un SDK facilitant la programmation du capteur sous Windows 7 et 8, ceci permettant d'ouvrir ses applications pour un coût modique. Des pilotes libres (non Microsoft) existent aussi, ce qui permet d'utiliser la Kinect pour d'autres applications que celles prévues par Microsoft (traitement d'image, le suivi d'objets, etc.)[18].

La Kinect présente une évolution ingénieuse dans l'univers des caméras 3D. Ce périphérique de Microsoft se forme principalement d'une barre horizontale constituant

l'élément principale de la technologie Kinect et contenant une camera RGB fournissant des images couleur et un capteur de profondeur pour détecter la profondeur de chaque pixel de l'image. A partir de ces deux images, la reconstruction 3D de la scène est rendu possible.

La Kinect contient aussi un petit moteur pour déplacer l'appareil, un accéléromètre 3 axes afin d'améliorer la précision du moteur et un petit ventilateur.

B) Système audio

1. Chat vocal Xbox Live et chat vocal dans les jeux vidéo.
2. Suppression de l'écho.
3. Reconnaissance vocale multilingue.

16 bits à 16 KHz grâce aux 4 microphones (figure III.1).

C) Capteur couleur RGB

Le capteur de couleur RGB consiste en 3 photodiodes silicium sensibles chacune à une couleur (rouge, verte ou bleue) (figure III.2). Il est utilisé pour les balances de couleurs, pour les corrections de couleurs des afficheurs, le contrôle d'impression, etc.

Longueur d'onde :

- Rouge : 630 nm
- Vert : 560 nm
- Bleu : 470 nm.
- Rouge : 0,35 µA
- Vert : 0,05 µA
- Bleu : 0,30 µA

Vf : 0,5 V

640 x 480 en couleur 32 bits à 30 images par seconde pour le capteur couleur

D) Capteur Infrarouge associé à un émetteur détectant la profondeur

Le capteur infrarouge est constitué d'un récepteur qui détecte l'intensité lumineuse dans la gamme des lumières infrarouge et d'un émetteur de lumière infrarouge (figure III.2).

Ce capteur peut être utilisé comme capteur de contact. On fait une mesure avec la LED infrarouge éteinte et une avec la LED infrarouge allumée. S'il n'y a aucun obstacle proche, la valeur lue est la même. Sinon, l'obstacle aura réfléchi la lumière infrarouge et la deuxième mesurée donnera un résultat plus élevé. 320 x 240 en couleur 16 bits à 30 images par seconde pour le capteur infrarouge [17].

E) LED

Un voyant est placé entre la caméra et le projecteur infrarouge. Il est utilisé pour indiquer l'état de l'appareil Kinect. La couleur verte de la LED indique que les pilotes de

périphériques Kinect sont chargés correctement. Si vous branchez la Kinect dans un ordinateur, la LED va commencer avec une lumière verte une fois que votre système détecte le périphérique.

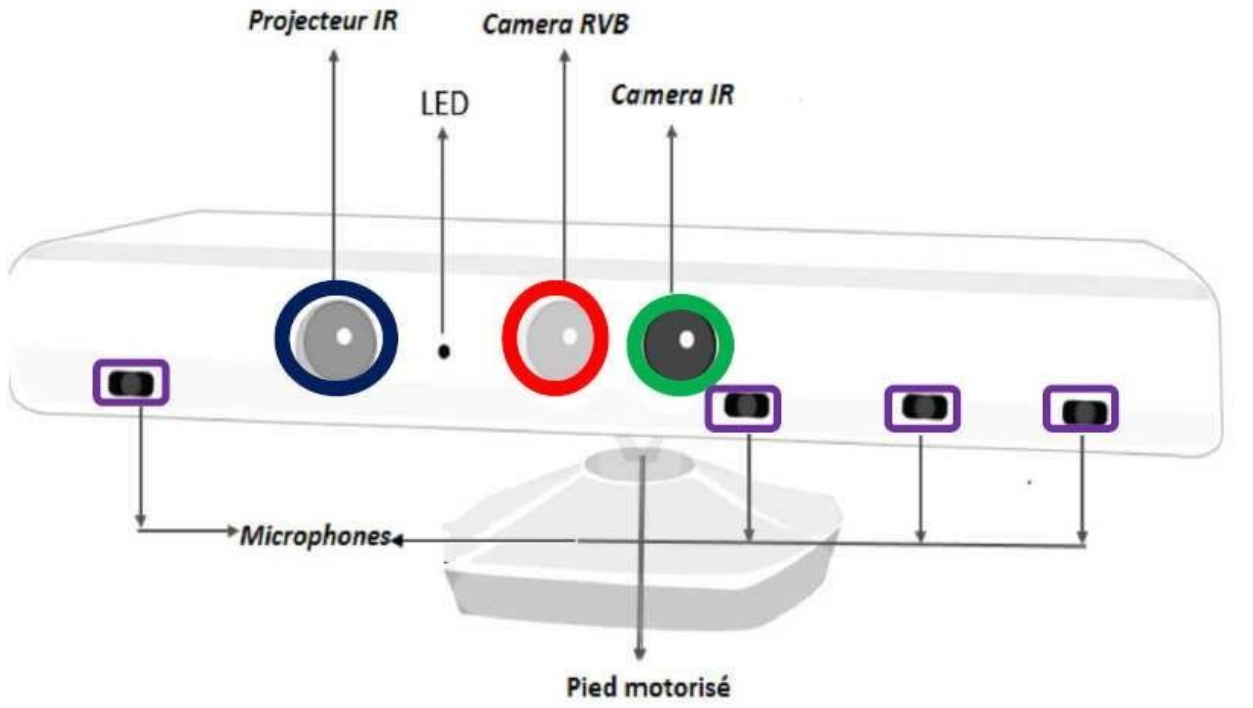


Figure III.1 : Composante de la Kinect.

E) Champ de vision

1. Champ de vision horizontal : 57 degrés.
2. Champ de vision vertical : 43 degrés.
3. Marge de déplacement du capteur (motorisation): +/- 27 degrés grâce au socle (figure III.2).
4. Portée du capteur : 1,2 m – 3,5 m (à partir de 50 cm pour la version Kinect for Windows) [16].

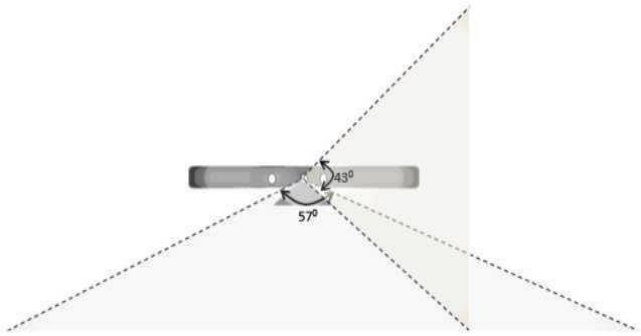


Figure III.2 : La plage de visualisation de la Kinect

F) Caméra de profondeur

La Kinect est une caméra 3D active de type stéréoscopique, elle comprend deux parties :

– une source de lumière infrarouge « structurée ».

– une caméra infrarouge.

La méthode se base sur les mêmes principes géométriques que ceux utilisés pour la stéréoscopie. En stéréoscopie, on exploite le fait que plus un objet central est proche des deux

caméras, plus l'objet est décalé vers la droite dans l'image de gauche et vers la gauche dans l'image de droite. De même, pour un objet distant, le décalage est plus faible.

Cependant, grâce à sa propre source lumineuse, la Kinect lève les indéterminations inhérentes à la stéréoscopie dans les zones dépourvues de texture (la figure III.3).

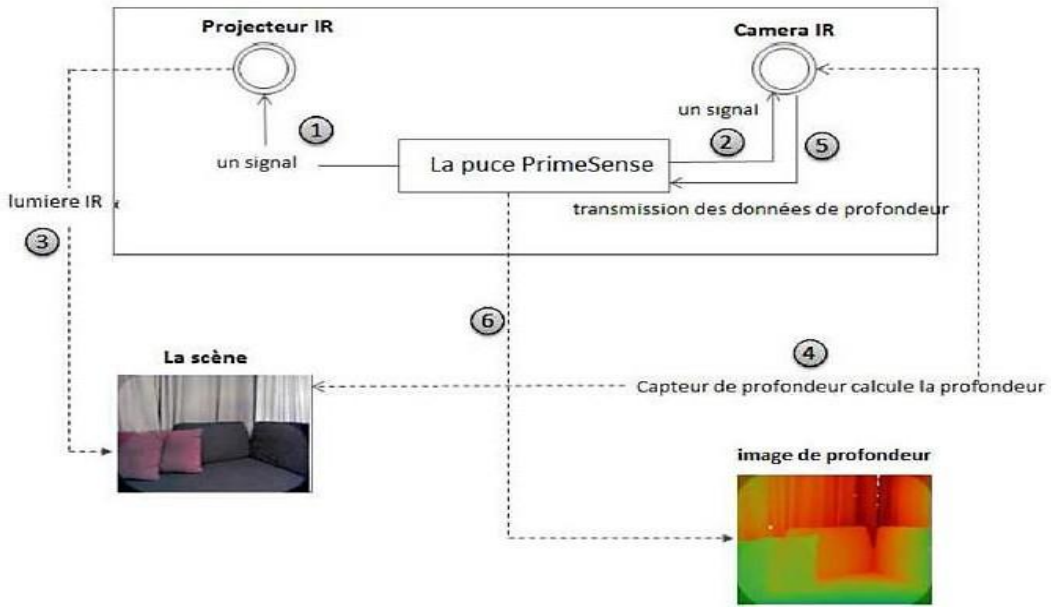


Figure III.3 : Principe de fonctionnement de la Kinect.

G) La mesure de profondeur (Depth)

La mesure de la profondeur se fait par un processus de triangulation. L'émetteur laser émet un faisceau unique est subdivisé en faisceaux multiples par un réseau de diffraction afin de créer un modèle constant de mouchetures projetées sur la scène (figure III.3). Cette tendance est capturée par la caméra infrarouge et est comparée avec un modèle de référence.

Le motif de référence est obtenu en capturant un plan à une distance connue du capteur, et est stocké dans la mémoire de la Kinect. Cette configuration de la Kinect est réalisée en usine avant la commercialisation. Quand la nuée de points est projetée sur un objet dont la distance au capteur est plus petite ou plus grande que celui du plan de référence alors la nuée de points dans l'image infrarouge est décalée dans la direction de la ligne de base entre le laser projecteur et le centre optique (centre focal, point f) de la caméra infrarouge. Ces changements sont mesurés pour toutes les taches par une simple procédure de corrélation, ce qui donne une image de disparité [17].

Pour chaque pixel la distance de la sonde peut alors être extraite de la disparité correspondante.

Pour mesurer la profondeur la Kinect utilise la lumière structurée, le principe de lumière structurée permet d'obtenir l'information de profondeur d'objets présents dans une

scène ne projetant un motif lumineux sur les objets et en analysant la déformation de ce motif au contact des objets.

Cette lumière structurée se présente sous forme d'un ensemble de rayons calibrés, Ces rayons calibrés évitent le besoin d'une seconde caméra tels qu'illustré dans la (Figure III.4).

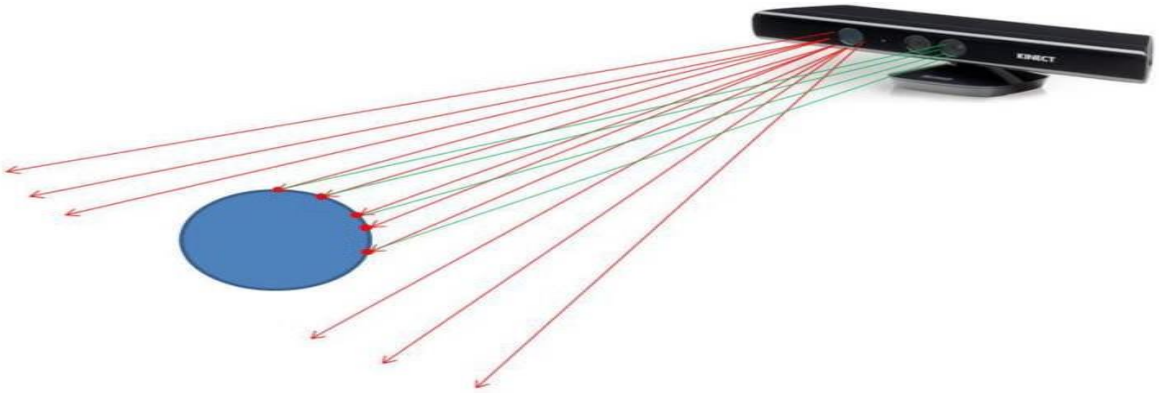


Figure III.4 : Principe de fonctionnement de la Kinect.

Kinect utilise 11 bits pour représente les valeurs de profondeur, ce qui limite la profondeur.

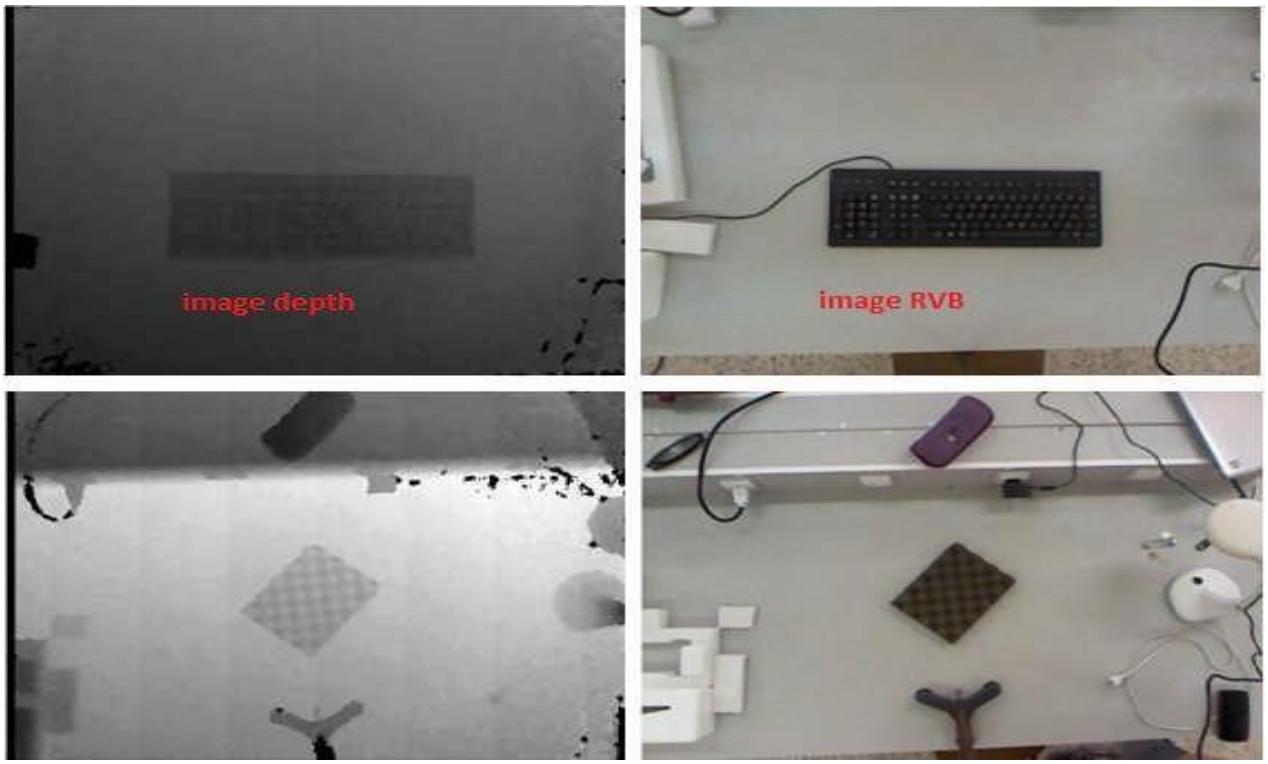


Figure III.5 : Image profondeur et image couleur fournie par la Kinect.

2.2. Coté Logiciel

2.2.1. Les pilotes de la caméra Kinect

A) CLNUI

CLNUI est une plate-forme fournie par Code Laboratory. Les utilisations possibles sont nombreuses, comme par exemple la robotique, la surveillance, la capture de mouvement,

le suivi de personne ou d'objet ou encore la numérisation 3D. CLNUI propose aussi une application pour la caméra KINECT qui affiche l'image couleur et l'image de profondeur, en offrant la possibilité d'incliner la caméra et de faire clignoter sa LED [17].

B) Open Ni

Le Cadre Open NI est une couche abstraite qui fournit l'interface pour des périphériques physiques et composants middleware. L'API (*Application Programming Interface*) permet à de multiples composants d'être enregistrés dans le cadre Open NI. Ces composants sont appelés modules, et sont utilisés pour produire et traiter les données sensorielles.

Les modules compatibles avec Open Ni sont les suivants :

1. Capteurs 3D
2. Caméra RGB
3. Caméra infrarouge
4. Matériel audio

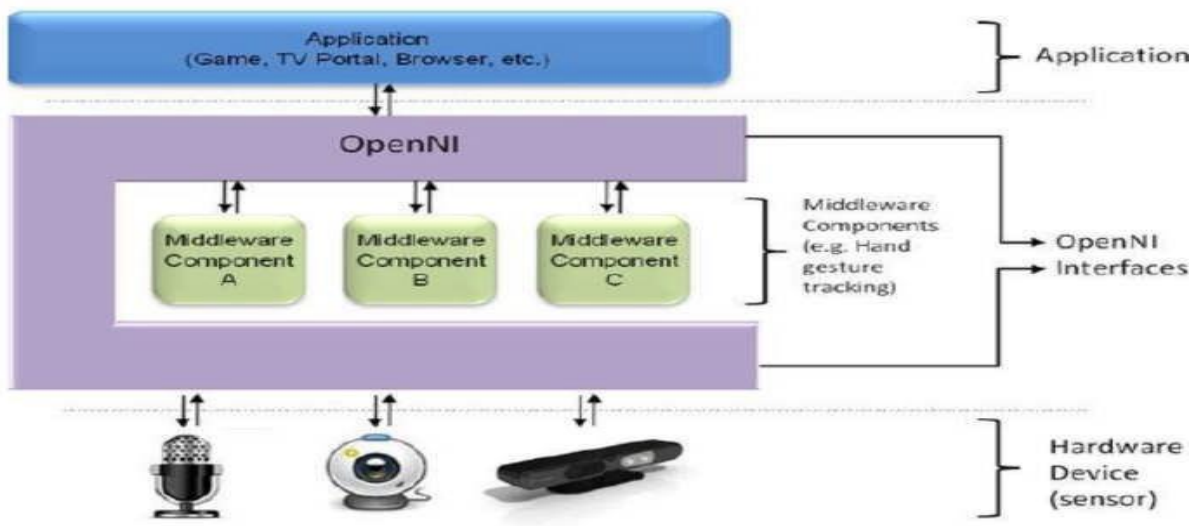


Figure III.6 : Modules Open NI

C) Applications de la Kinect en vision par ordinateur

Avec l'invention et le faible coût de la caméra Kinect de Microsoft, qui offre une carte de profondeur à haute résolution et une image couleur (RGB), son utilisation est devenue généralisée et cela ouvre un grand nombre de nouveaux travaux de recherche et de développement.[18]

La nature complémentaire de la profondeur et de l'information visuelle fournie par la Kinect ouvre de nouvelles opportunités pour résoudre les problèmes fondamentaux dans le domaine de la vision par ordinateur.

De nombreuses applications utilisant la Kinect ont été développés dans le domaine de vision par ordinateur, tels que le prétraitement, le suivi et la reconnaissance d'objets, l'analyse de l'activité humaine, l'analyse et la reconnaissance des gestes de la main [16] (figureIII.5), et la cartographie intérieure 3-D [17] (figure III.6) …etc.

Dans la littérature, on trouve un état de l'art complet sur cette problématique, à savoir l'utilisation de la kinect en vision par ordinateur [18].

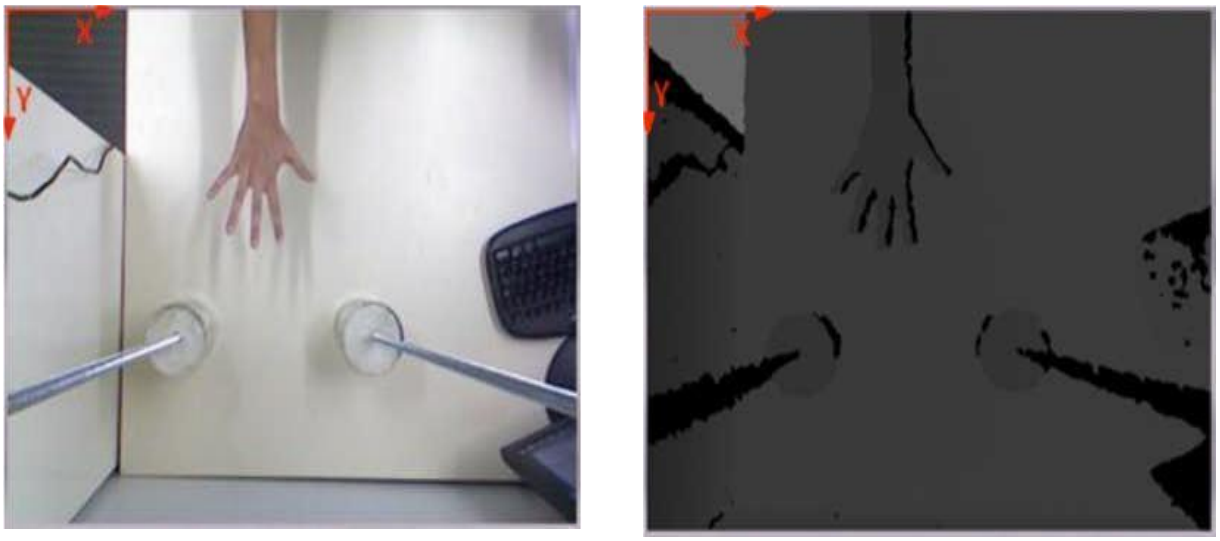


Figure III.7 : Image de profondeur et couleur.

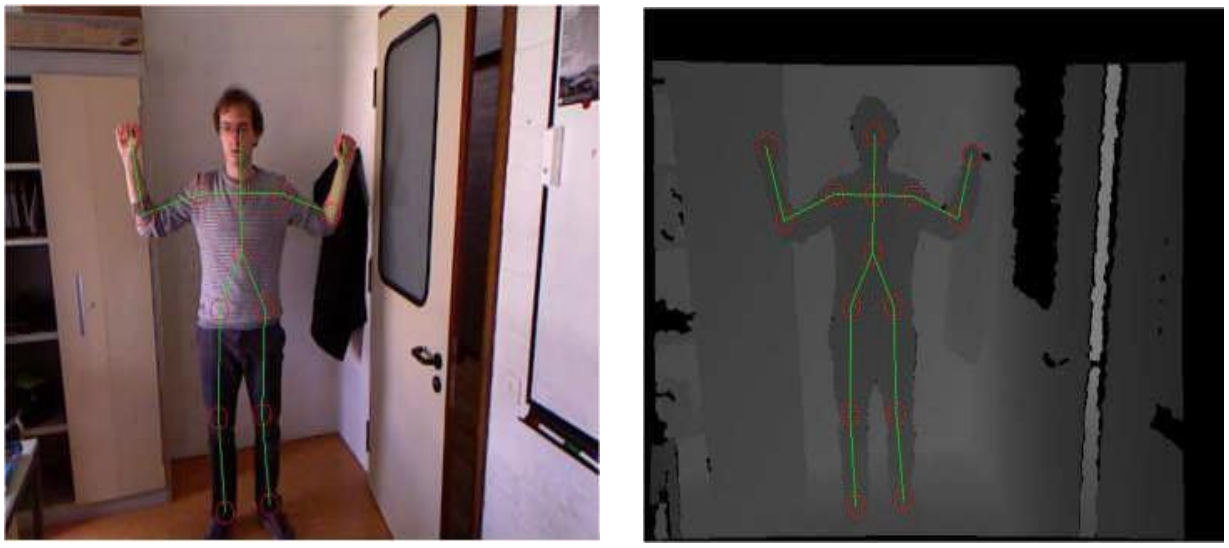


Figure III.8 : Articulations détectées par NITE. Affichées sur la carte de profondeur et sur couleurs.

2.2.2. Installation de Ubuntu

2.2.2.1. Présentation d'Ubuntu 20.04 LTS

Ubuntu 20.04 : est la meilleure porte d'entrée pour découvrir, adopter puis utiliser Linux au quotidien. Développé par Canonical, Ubuntu est une distribution Linux « user-

friendly » accessible et simple d'utilisation. Jouissant d'une forte communauté, Ubuntu est parfait pour une utilisation quotidienne que ce soit sur PC de bureau ou PC portable.1]

A part sa simplicité, Ubuntu est aujourd'hui considéré comme l'**alternative n°1 à Windows** même si d'autres distributions comme Linux Mint valent également le coup d'œil, notamment pour les débutants.

Aujourd'hui, nous allons voir comment installer Ubuntu 20.04 LTS sur un PC vierge ou bien sur un PC lequel Windows est déjà installé.

L'objectif n'est pas de faire un dual-boot Ubuntu/Windows mais de **remplacer Windows par Ubuntu 20.04** afin de faire d'Ubuntu son système d'exploitation principal. Ubuntu 20.04 « Focal Fossa » est une version **LTS** (Long Term Support) dont le support technique sera assuré pendant cinq ans (jusqu'en 2025) [25].

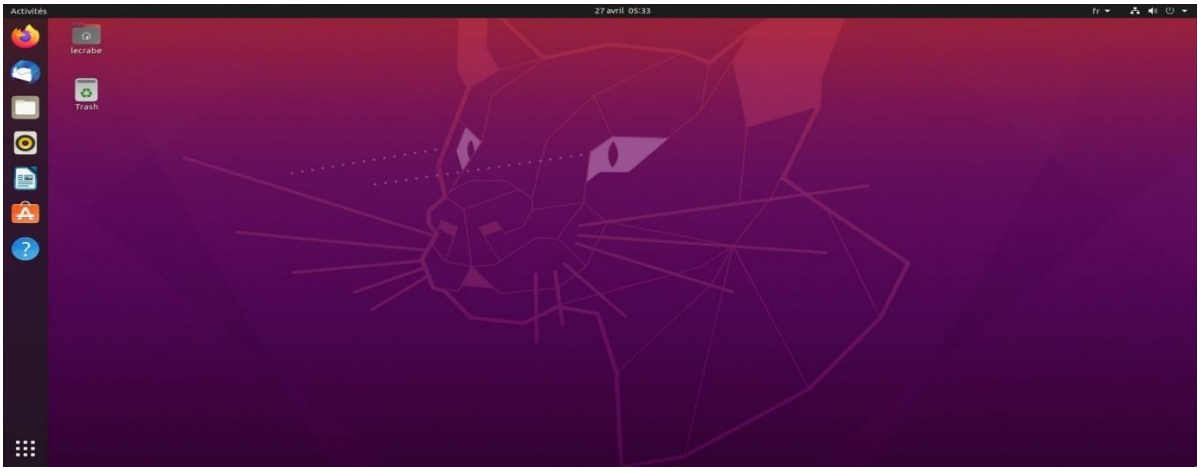


Figure III.9 : Présentation d'Ubuntu 20.04 LTS.

2.2.2.2. Installer Ubuntu 20.04 LTS

Étape 1 : créer une clé USB d'installation d'Ubuntu

Téléchargez l'ISO d'Ubuntu 20.04.

1. Vérifiez l'intégrité du fichier ISO avec l'empreinte SHA-256 du fichier original.

2. Créez une clé USB d'installation d'Ubuntu 20.04 à partir de l'ISO télécharger [25] :
   - depuis Windows : créer une clé USB bootable de Linux
   - depuis Linux : créer une clé USB bootable de Linux depuis Linux

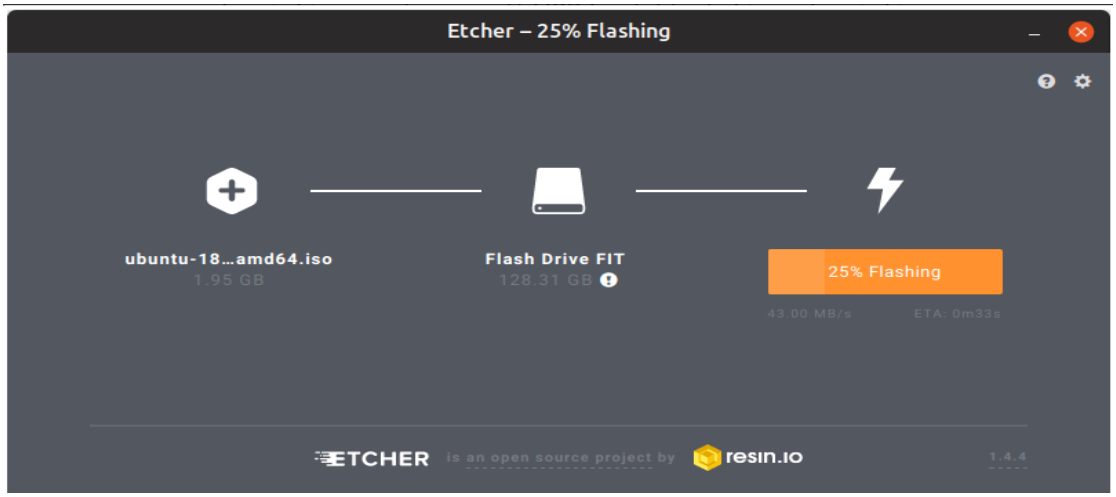


Figure III.10 : l'installation d'Ubuntu 20.04 LTS.

3. L'installeur d'Ubuntu, alias Ubiquity, se lance. Sélectionnez la langue **Français** puis cliquez sur le bouton **Installer maintenant**.

Vous avez également la possibilité de tester Ubuntu sans l'installer en sélectionnant **Essayer Ubuntu** (les données nécessaires au fonctionnement de l'OS seront copiées dans la mémoire vive) puis de lancer l'installation depuis l'icône sur le Bureau.

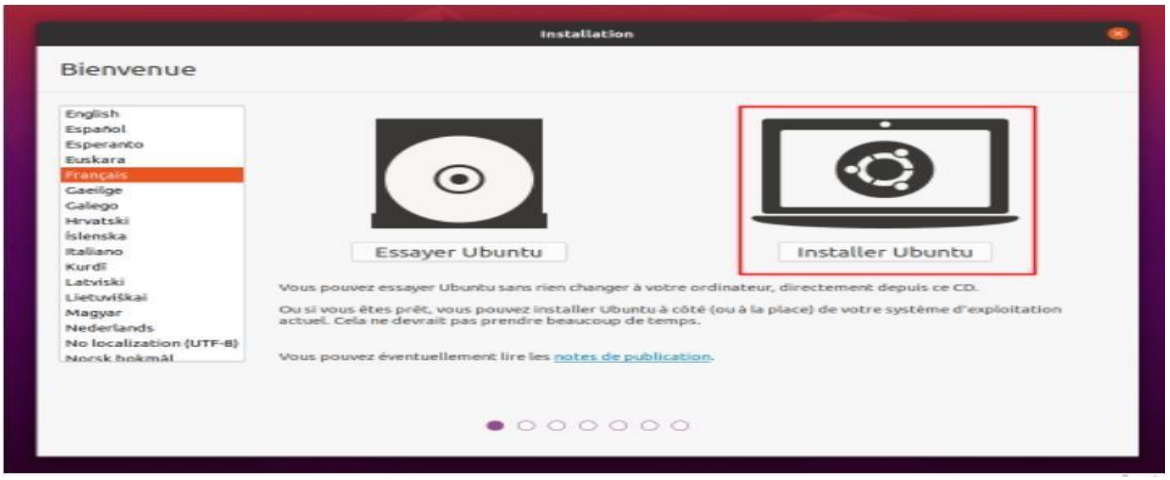


Figure III.11 : Choisissez la langue et installez.

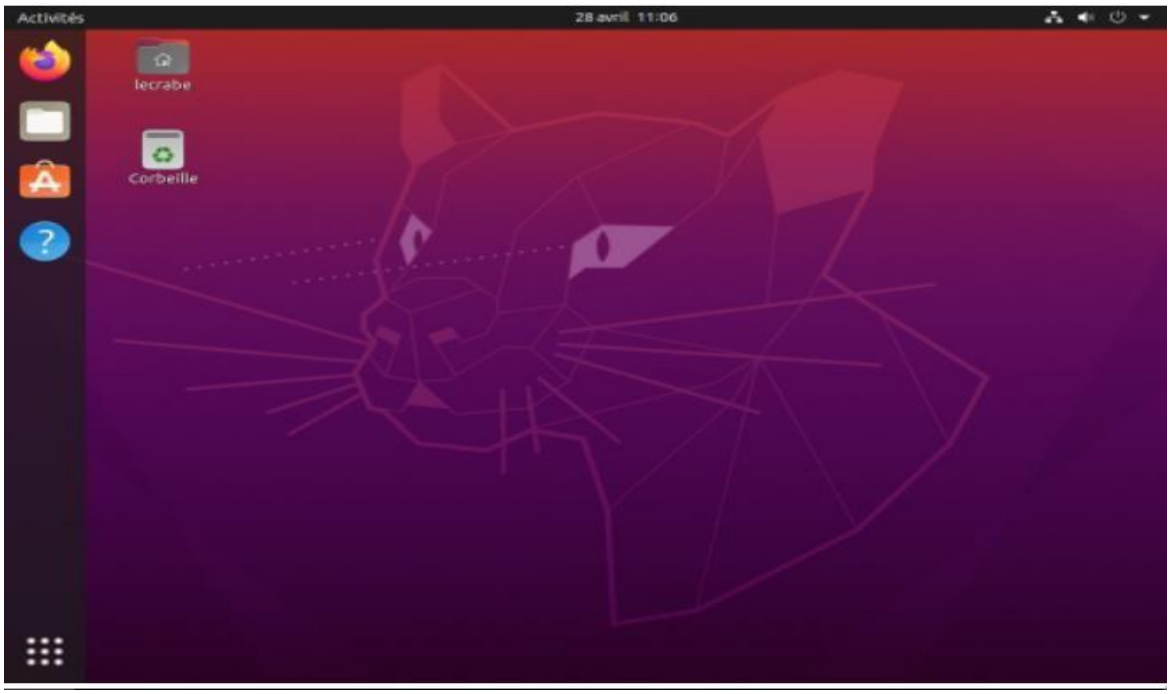


Figure III.12 : vue du bureau pour UBUNTU 20.04 LTS.

2.2.2.2. Installation De Ros

ROS (Robot Operating System) est un projet open source qui fournit un cadre et des outils pour les applications robotiques. Cela aide à concevoir des logiciels complexes sans savoir comment fonctionnent certains matériels.

Noetic est une version LTS de ROS et adaptée à Ubuntu 20.04. Le support ROS Noetic est jusqu'en 2025 (5 ans). Les architectures prises en charge sont amd64, armhf et arm64.

2.2.2.3. Comment installer ROS Noetic sur Ubuntu 20.04 LTS

a-Ajouter le dépôt Noetic officiel à Ubuntu

La première étape de l'installation de ROS Noetic consiste à ajouter le référentiel officiel ROS Noetic au fichier de liste des sources d'Ubuntu 20.04[24].

```
$ echo "deb http://packages.ros.org/ros/ubuntu focal main" | sudo tee /etc/apt/sources.list.d/ros-focal.list
```

b- Ajouter un porte-clés ROS officiel

Ensuite, ajoutez le trousseau de clés ROS officiel à votre système Ubuntu 20.04. Il y a deux façons d'aborder cela.

```
$ curl -sSL 'http://keyserver.ubuntu.com/pks/lookup?op=get&search=
0xC1CF6E31E6BADE8868B172B4F42ED6FBAB17C654'
 | sudo apt-key add -
```

c-Mettre à jour l'index du package ROS

Ensuite, nous mettrons à jour notre système Ubuntu afin d'obtenir les informations du package ROS Noetic à partir du référentiel [24].

```
$ sudo apt update
```

2.2.2.4. Installez ROS Noetic sur Ubuntu 20.04

ROS propose plusieurs méta paquets que vous pouvez choisir d'installer en fonction de vos besoins spécifiques.

Voici les méta paquets ROS Noetic officiels :

ros-noetic-desktop-complet

ros-noetic-desktop

ros-noetic-ros-base

Ros-noetic-ros-core

Choisissez votre métapaquet préféré et installez-le avec l'une des commandes ci-dessous.

ros-noetic-desktop-complet	`$ sudo apt install ros-noetic-desktop-full`
ros-noetic-desktop	`$ sudo apt install ros-noetic-desktop`
ros-noetic-ros-base	`$ sudo apt install ros-noetic-base`
Ros-noetic-ros-core	`$ sudo apt install ros-noetic-core`

2.2.2.5. Configurer l'environnement ROS Noetic

L'étape suivante consiste à configurer l'environnement ROS Noetic. Sourcez d'abord le script setup.bash dans chaque terminal bash qui utilise ROS, tapez :

`$ source /opt/ros/noetic/setup.bash`
`$ echo "source /opt/ros/noetic/setup.bash">>`

```
~/.bashrc
$ tail ~/.bashrc
$  source ~/.bashrc
```

2.2.2.6. Verification Noetic installation

Après avoir installé avec succès ROS Noetic sur Ubuntu 20.04, exécutez simplement la commande roscd.

Vous remarquerez que le répertoire actuel de votre invite change pour /opt/ros/noetic, où nous avons installé Noetic.

Nous pouvons également vérifier l'installation en exécutant la commande roscore dans le répertoire noetic. La sortie affiche la distribution ros et la version ros dans le résumé [24].

```
$ roscd
 $ roscore
```

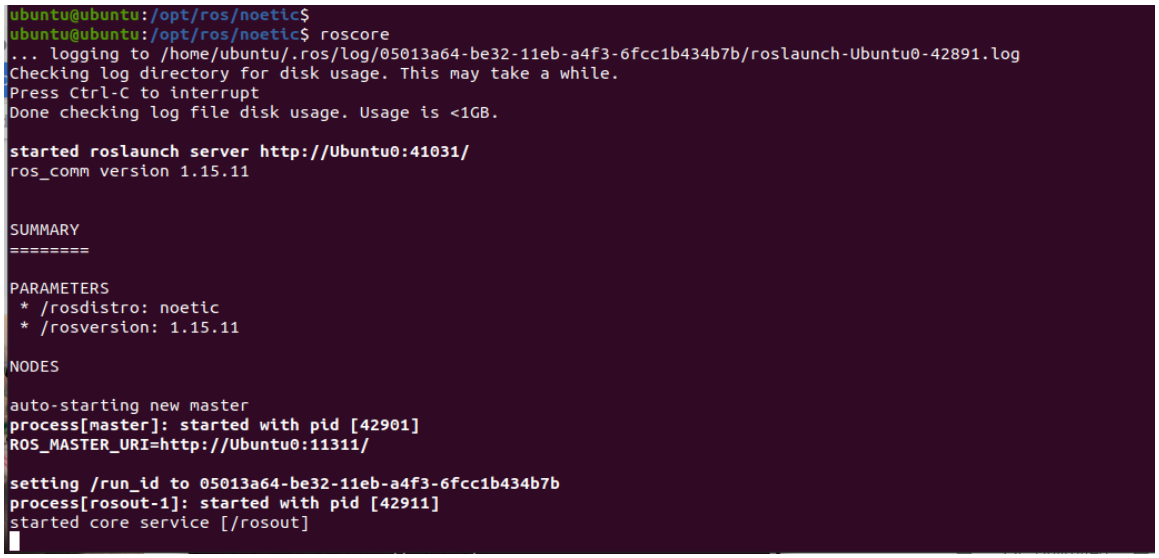


Figure III.13 : vérification de l'installations de ros noetic.

2.3. Programmation Avec Ros

Avant de programmer dans ROS, un espace de travail doit être créé. Le dossier appelé ROS workspace ou catkin workspace est l'endroit où les nouveaux paquets ROS sont créés, les paquets existants sont installés, et les nouveaux exécutables sont construits et créés [28]. Catkin est un ensemble d'outils que ROS utilise pour générer des programmes, des bibliothèques, des scripts et des interfaces que d'autres codes peuvent utiliser [30].

Pour créer un espace de travail catkin,

Un paquet ROS est une combinaison de code, de données et de documentation [29]. Au lieu d'avoir à changer le répertoire des sources vers l'espace de travail catkin créé ci-dessus, chaque fois qu'un nouveau paquet ROS est créé ou qu'un paquet ROS existant est modifié,

Un paquet ROS utile pour la programmation des robots est le paquet tf, où tf signifie « transform ». Un système robotique possède généralement de nombreux cadres de coordonnées 3D, tels que le cadre du monde, le cadre de base, le cadre de tête, etc. qui changent au fil du temps, comme le montre la figure 1. Les cylindres RVB de la figure représentent les axes X, Y et Z des cadres de coordonnées. Tf garde la trace de tous ces cadres et maintient la relation entre les cadres de coordonnées dans une structure arborescente mise en mémoire tampon dans le temps.

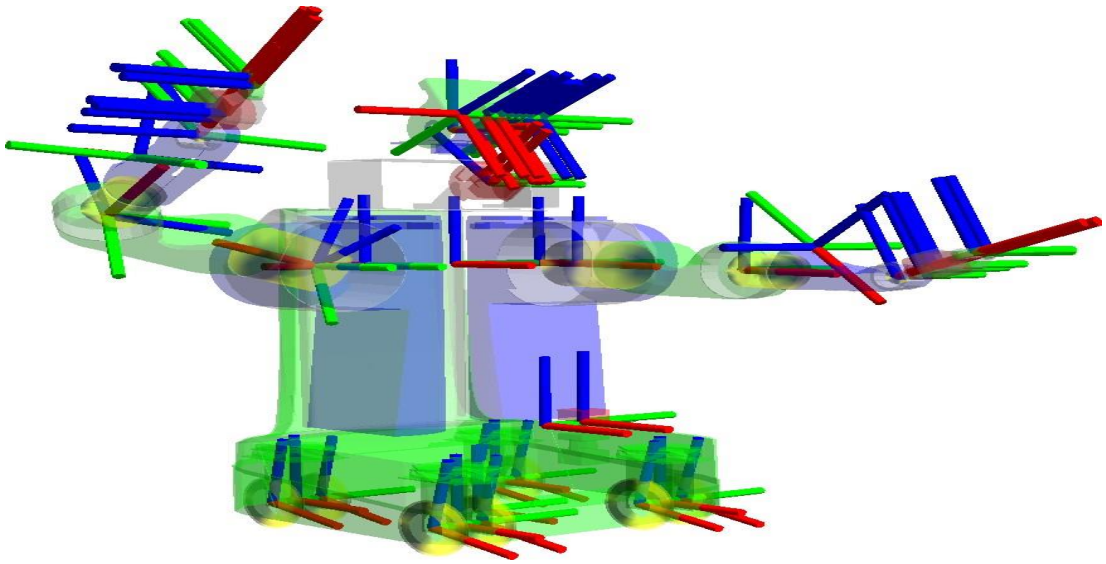


Figure III.14 : Les différents cadres de coordonnées 3D du robot PR2 [31].

La bibliothèque tf a été conçue pour fournir un moyen standard de conserver la trace des cadres de coordonnées et de transformer les données d'un système à un autre, de telle sorte que les utilisateurs de composants individuels puissent être sûrs que les données sont dans le cadre de coordonnées qu'ils souhaitent, sans avoir besoin de connaître tous les cadres de coordonnées du système [31].

2.3.1. L'installation de ROS-noetic à la simulation SLAM de turtlebot3

A. Créer un espace de travail

Si vous installez avec le shell ci-dessus, un espace de travail appelé catkin_ws sera créé, alors ignorez-le. Quiconque l'installe dans l'autre sens créera un espace de travail, quel que soit son nom (catkin_ws ici) [26].

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
$ cd ~/catkin_ws && catkin_make
```

B. Création d'un environnement de simulation pour turtlebot3

Installer le package en spécifiant branch comme noetic-dev

```
$ git clone https://github.com/ROBOTIS-GIT/hls_lfcd_lds_driver.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
```

Je ne veux pas frapper export TURTLEBOT3_MODEL = burger à chaque fois, donc je l'écris dans bashrc. (L'éditeur est familier)

```
$ gedit ~/.bashrc
```

Sur la dernière ligne

Export TURTLEBOT3_MODEL=burger

Écrivez, enregistrez et fermez.

```
$ source ~/.bashrc
```

Cela sera reflété dans. (Si TURTLEBOT3_MODEL donne une erreur inconnue à l'avenir, veuillez recommencer)

C. Installation du simulateur turtlebot3

```
$ cd ~/catkin_ws/src
$ git clone -b noetic-dev https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
$ cd ~/catkin_ws && catkin_make
```

2.3.2. Gazebo

Gazebo est un outil qui permet de simuler des environnements 3D avec différents types de robots avec un haut degré de précision [32]. Le paquet gazebo_ros fournit une interface entre Gazebo et ROS. Il intègre de nombreux capteurs et robots simulés, prêts à fonctionner, et permet d'utiliser les plugins ROS standard existants pour Gazebo. La possibilité de contrôler le monde et le robot à l'aide d'une interface en ligne de commande en fait le choix idéal pour cette recherche. La version 11.0 de Gazebo est actuellement disponible, mais ROS kinetic prend en charge Gazebo 7 et est préinstallé avec Gazebo 7.0. Pour un fonctionnement sans faille, Gazebo7 doit être mis à jour vers la dernière version, qui est actuellement la version 7.16, en utilisant les commandes du tableau 2.

sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable `lsb_release -cs` main"> /etc/apt/sources.list.d/gazebo-stable.list'
wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -
sudo apt-get update && sudo apt-get install gazebo7 -y

2.3.3. Exécutez le simulateur [RViz]

RViz est l'outil de visualisation 3D de ROS. C'est là que les mouvements du robot et le processus de construction de la carte peuvent être observés en temps réel. Pour transformer les données du référentiel local du robot en référence globale, rviz utilise le paquet tf. La commande [33] utilisée pour installer rviz est la suivante :

$ sudo apt-get install ros-noetic-rviz.

Il permet également aux utilisateurs de définir de manière interactive des objectifs intermédiaires et des limites d'exploration pour le robot à l'aide de la définition d'objectifs 2D.

$ roslaunch turtlebot3_fake turtlebot3_fake.launch

Cela lancera turtlebot3.

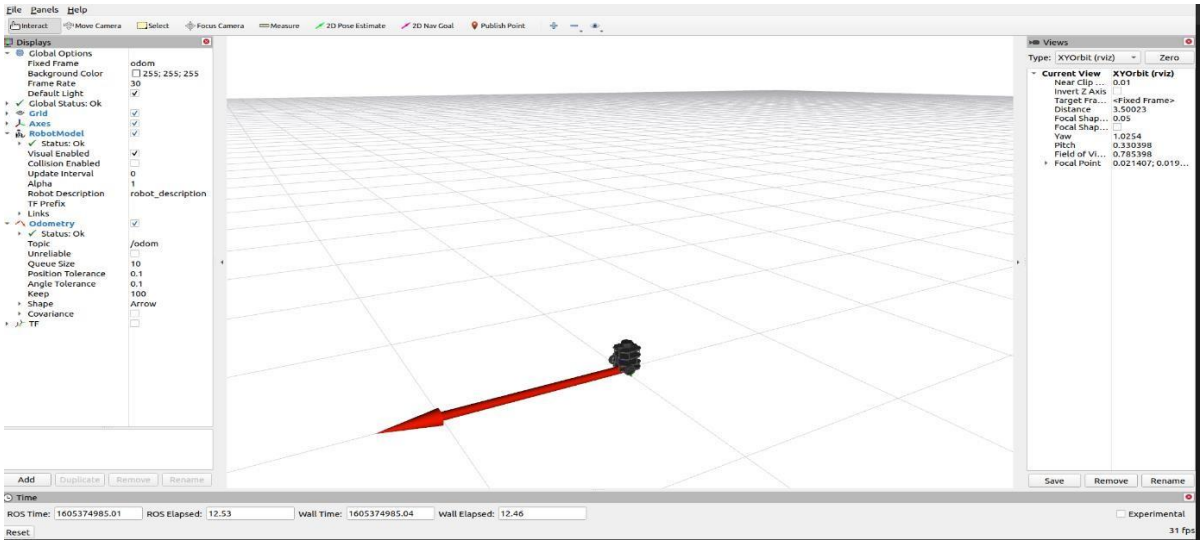


Figure III.15 : Le simulateur [Rviz] [33].

Vous pourrez utiliser le clavier avec la commande suivante, veuillez exécuter "dansun nouveau terminal".

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

A. Exécution du simulateur [Gazebo]

Fermez tous les terminaux et démarrez-en un nouveau.

Ensuite, exécutez la simulation sur Gazebo. RViz est un outil de débogage, mais Gazebo peut le simuler dans n'importe quel environnement [33].

$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
$ roslaunch turtlebot3_gazebo turtlebot3_simulation.launch
$ roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch

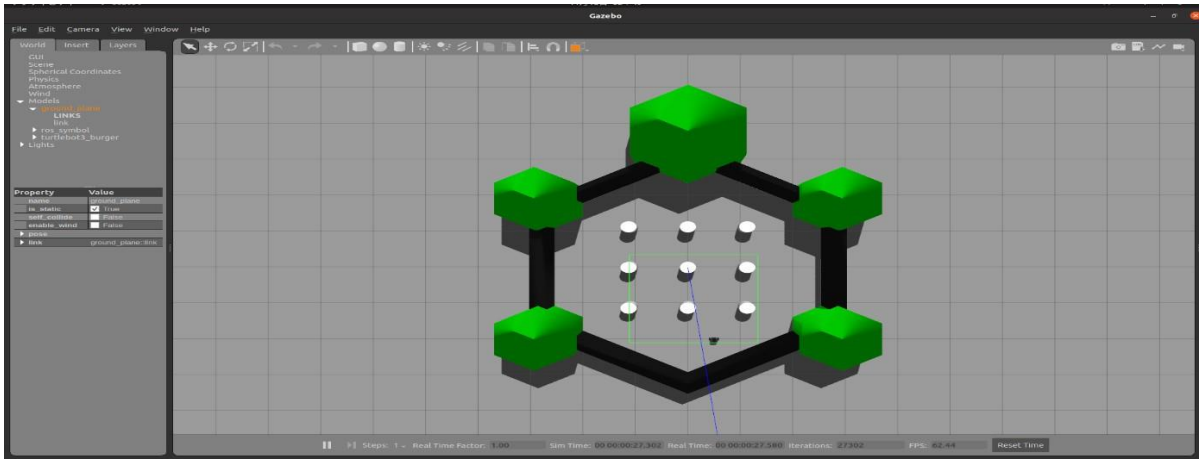


Figure III.16 : Enfin la simulation SLAM.

Installez le module SLAM.

$ sudo apt-get install ros-noetic-slam-gmapping
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch

Une fois Gazebo démarré, exécutez SLAM immédiatement.

$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
$ roslaunch turtlebot3_gazebo turtlebot3_simulation.launch

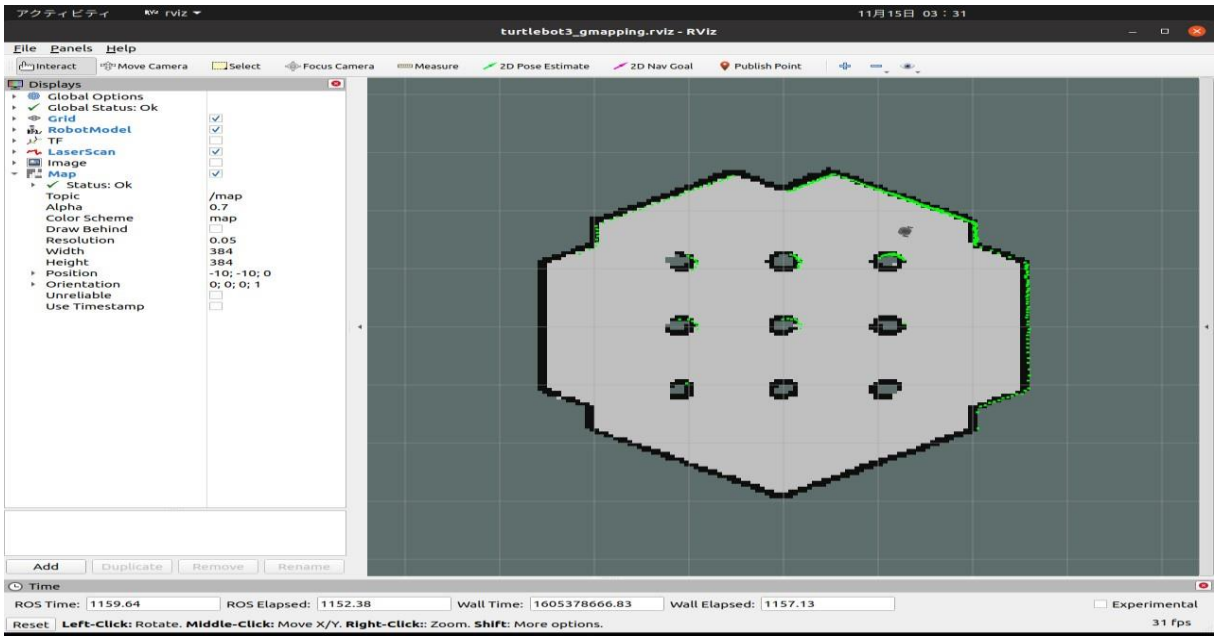


Figure III.17 : turtlebot3_gmapping.rviz.

2.4. Execution RTAB-MAP sur ROS Noetic

2.4.1. Installation de RTAB-MAP (depuis apt)

J'ai exécuté rtabmap_ros avec ROS Noetic sur ubuntu20.04LTS, j'ai donc noté comment le faire. Nous avons confirmé l'opération dans l'environnement suivant [34].

Item	Value
CPU	Ryzen 5
OS	Ubuntu20.04LTS
ROS	Noetic Ninjemys

Installez avec la commande suivante.

sudo apt install ros-noetic-rtabmap-ros ros-noetic-rtabmap

2.4.2. Installation de RTAB-MAP (construit à partir des sources)

Pour une raison quelconque, lorsque je l'ai installé à partir d'apt dans mon environnement, la démo décrite plus tard n'a pas fonctionné, je l'ai donc construit à partir des sources [27]. Voir Construire à partir de la source ci-dessous

https://github.com/introlab/rtabmap_ros

Installez et construisez RTAB-MAP lui-même ci-dessous

$ cd ~
$ git clone https://github.com/introlab/rtabmap.git rtabmap
$ cd rtabmap/build
$ cmake ..
$ make
$ sudo make install

2.4.3. Installer et exécuter rtabmap_ros

$ cd ~/catkin_ws
$ git clone https://github.com/introlab/rtabmap_ros.git src/rtabmap_ros
$ catkin_make -j1

2.4.4. Exécuter la démo

roslaunch rtabmap_ros demo_robot_mapping.launch
rosbag play --clock demo_mapping.bag

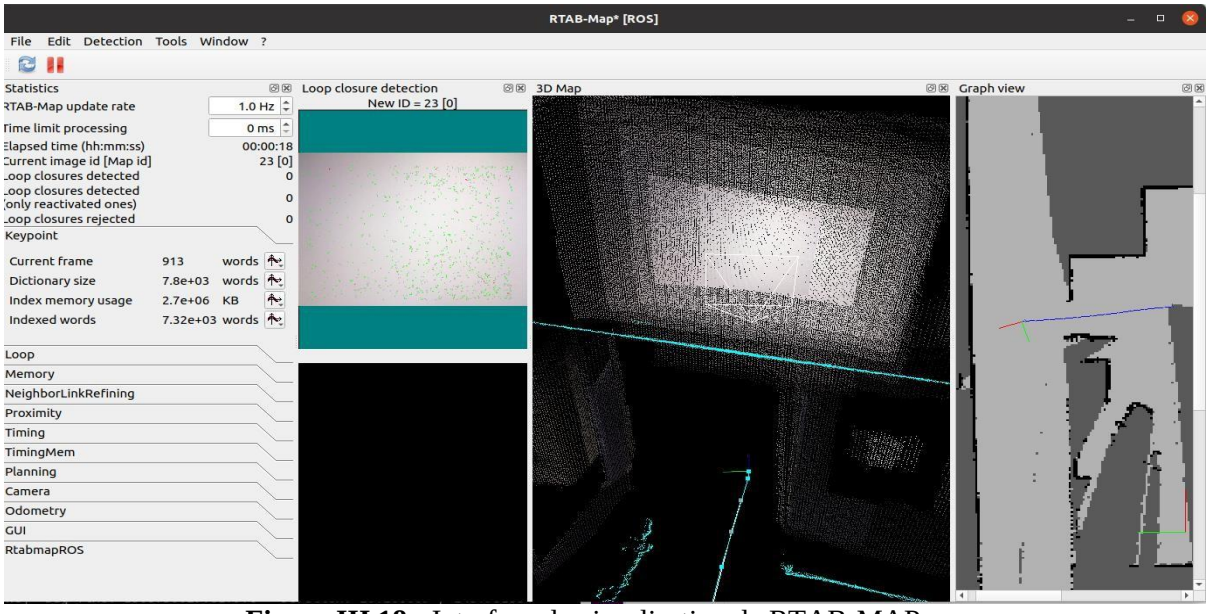


Figure III.18 : Interface de visualisation de RTAB-MAP

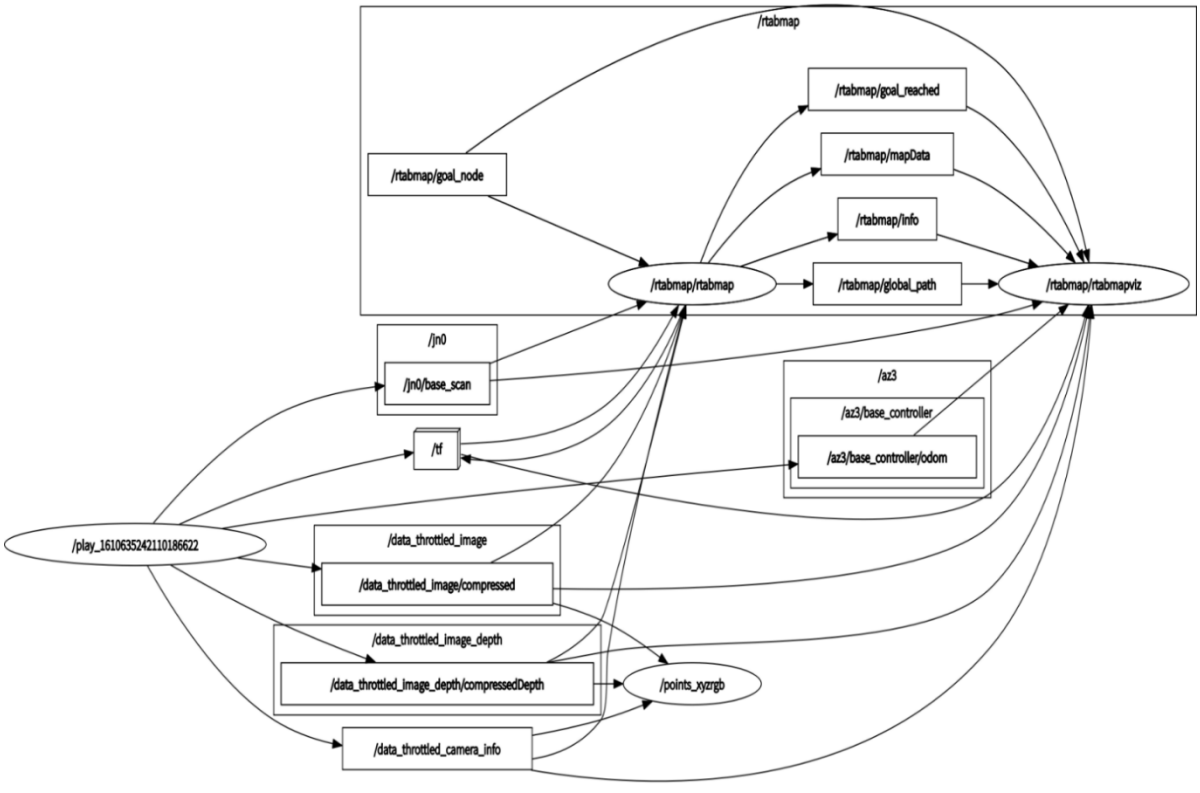


Figure III.19 : RTAB-MAP*(ROS).

3. Exécution

Dans ce chapitre, il est expliqué la mise en œuvre du cadre modulaire conçu et ses intégrations avec d'autres solutions open source. Il existe plusieurs langues possibles dans lesquelles le Framework pourrait être implémenté, mais pour ce projet C++ a été choisi,

pour sa polyvalence et parce que la plupart des solutions open-source utilisent ce langage dans leur base de code (y compris RTAB-MAP), évitant d'éventuelles incompatibilités. Le cadre est conçu comme un outil autonome bibliothèque qui peut être installée à l'aide de CMake pour faciliter l'importation dans différents projets.

3.1. Implémentation avec des frontaux supplémentaires utilisant ROS

La mise en œuvre avec RTAB-MAP sert de preuve de concept qu'il est possible de résoudre le problème SLAM sans la plupart des dépendances qui ont créé les problèmes de modularité dans la version originale. De plus, il montre que le cadre modulaire conçu est capable de travaillant avec la grande variété de modules de traitement de capteurs présents dans RTAB-MAP.

Cependant, d'un point de vue fonctionnel, le cadre conçu n'apporte aucun élément supplémentaire fonctionnalités à RTAB-MAP en plus de réduire potentiellement le nombre de dépendances. Pour ça raison, pour montrer pleinement le potentiel du cadre conçu, nous mettrons en œuvre des frontends pour montrer qu'il est possible d'ajouter facilement de nouvelles fonctionnalités en ayant plusieurs front ends travailler avec notre cadre.

Tout d'abord, nous allons ajouter une interface pour ROS Noetic (ROS (2021)). La raison en est qu’au cours de ce projet, il n'est pas possible de rétro concevoir l'architecture d'un autre gros Framework comme cela a été fait avec RTAB-MAP. Au lieu de cela, nous utiliserons les nœuds ROS disponibles pour implémenter un autre frontal. De plus, ROS a une énorme communauté de développeurs et de passionnés de robotique, il prend en charge une grande variété de capteurs et son architecture est modulaire et a bien interfaces définies.

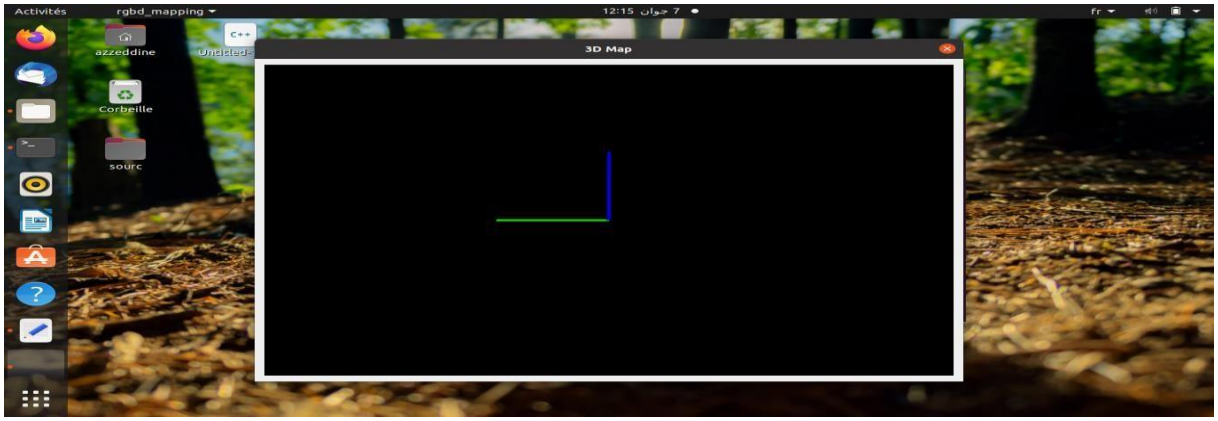


Figure III.20 : Implémentation de interfaces de RTAB MAP.

3.2. Abonnez-vous à RTAB-MAP par défaut à CameraEvent

Le RTAB-MAP est abonné par défaut à CameraEvent, mais nous voulons RTAB-MAP pour traiter OdometryEvent à la place, en ignorant CameraEvent.

Nous pouvons le faire en créant un "tuyau" entre la caméra et l'odométrie, puis seul l'odométrie recevra CameraEvent de cette caméra. RTAB-MAP est également abonné à OdometryEvent par défaut, donc pas besoin de créer de pipe entre, odométrie et RTAB-MAP.

```
218     UEventsManager::createPipe(&cameraThread, &odomThread, "CameraEvent");
219     rtabmapThread.start();
220     odomThread.start();
221     cameraThread.start();
222     mapBuilder.show();
223     app.exec();
224     mapBuilder.unregisterFromEventsManager();
225     rtabmapThread.unregisterFromEventsManager();
226     odomThread.unregisterFromEventsManager();
227     cameraThread.kill();
228     odomThread.join(true);
229     rtabmapThread.join(true);
230
```

Figure III.21 : RTAB-MAP pour traiter OdometryEvent à la place.

3.2.1 Transformer le nuage de points en sa pose (point cloud) et décimation de l'imageavant de créer les nuages (clouds)

```
236     rtabmap->getGraph(optimizedPoses, links, true, true, &nodes, true, true, true, true);
237     pcl::PointCloud<pcl::PointXYZRGB>::Ptr cloud(new pcl::PointCloud<pcl::PointXYZRGB>);
238     for(std::map<int, Transform>::iterator iter=optimizedPoses.begin(); iter!=optimizedPoses.end(); ++iter)
239     {
240         Signature node = nodes.find(iter->first)->second;
241
242
243         node.sensorData().uncompressData();
244
245         pcl::PointCloud<pcl::PointXYZRGB>::Ptr tmp = util3d::cloudRGBFromSensorData(
246                 node.sensorData(),
247                 4,
248                 4.0f,
249                 0.0f);
250         pcl::PointCloud<pcl::PointXYZRGB>::Ptr tmpNoNaN(new pcl::PointCloud<pcl::PointXYZRGB>);
251         std::vector<int> index;
252         pcl::removeNaNFromPointCloud(*tmp, *tmpNoNaN, index);
253         if(!tmpNoNaN->empty())
254         {
255             *cloud += *util3d::transformPointCloud(tmpNoNaN, iter->second);
256         }
257     }
258     if(cloud->size())
259     {
260         printf("Voxel grid filtering of the assembled cloud (voxel=%f, %d points)\n", 0.01f, (int)cloud->size());
261         cloud = util3d::voxelize(cloud, 0.01f);
262
263         printf("Saving rtabmap_cloud.pcd... done! (%d points)\n", (int)cloud->size());
264         pcl::io::savePCDFile("rtabmap_cloud.pcd", *cloud);
265         pcl::io::savePLYFile("rtabmap_cloud.ply", *cloud); // to
```

Figure III.22 : transformer le nuage de points en sa pose (point cloud).

3.2.2. Ajouter des nuages RVB-D (clouds) et Mettre à jour uniquement si la pose a changé

```
const std::map<int, Transform> & poses = stats.poses();
QMap<std::string, Transform> clouds = cloudViewer_->getAddedClouds();
for(std::map<int, Transform>::const_iterator iter = poses.lower_bound(1); iter!=poses.end(); ++iter)
{
    if(!iter->second.isNull())
    {
        std::string cloudName = uFormat("cloud%d", iter->first);

        // 3d point cloud
        if(clouds.contains(cloudName))
        {
            // Update only if the pose has changed
            Transform tCloud;
            cloudViewer_->getPose(cloudName, tCloud);
            if(tCloud.isNull() || iter->second != tCloud)
            {
                if(!cloudViewer_->updateCloudPose(cloudName, iter->second))
                {
                    UERROR("Updating pose cloud %d failed!", iter->first);
                }
            }
            cloudViewer_->setCloudVisibility(cloudName, true);
        }
        else if(iter->first == stats.getLastSignatureData().id())
        {
            Signature s = stats.getLastSignatureData();
            s.sensorData().uncompressData(); // make sure data is uncompressed
            // Add the new cloud
            pcl::PointCloud<pcl::PointXYZRGB>::Ptr cloud = util3d::cloudRGBFromSensorData(
                    s.sensorData(),
                    4,      // decimation
                    4.0f); // max depth
            if(cloud->size())
            {
                if(!cloudViewer_->addCloud(cloudName, cloud, iter->second))
                {
                    UERROR("Adding cloud %d to viewer failed!", iter->first);
```

Figure III.23 : Ajouter des nuages RVB-D (clouds) et Mettre à jour uniquement si la pose a changé.

3.2.3. Ajouter un graphique 3D (afficher toutes les poses)

```
cloudViewer_->removeAllGraphs();
cloudViewer_->removeCloud("graph_nodes");
if(poses.size())
{
    // Set graph
    pcl::PointCloud<pcl::PointXYZ>::Ptr graph(new pcl::PointCloud<pcl::PointXYZ>);
    pcl::PointCloud<pcl::PointXYZ>::Ptr graphNodes(new pcl::PointCloud<pcl::PointXYZ>);
    for(std::map<int, Transform>::const_iterator iter=poses.lower_bound(1); iter!=poses.end(); ++iter)
    {
        graph->push_back(pcl::PointXYZ(iter->second.x(), iter->second.y(), iter->second.z()));
    }
    *graphNodes = *graph;


    // add graph
    cloudViewer_->addOrUpdateGraph("graph", graph, Qt::gray);
    cloudViewer_->addCloud("graph_nodes", graphNodes, Transform::getIdentity(), Qt::green);
    cloudViewer_->setCloudPointSize("graph_nodes", 5);
}
```

Figure III.24 : Ajouter un graphique 3D (afficher toutes les poses).

3.3. Mettre à jour/ajouter la grille d'occupation

```
if(grid_.addedNodes().find(stats.getLastSignatureData().id()) == grid_.addedNodes().end())
{
    if(stats.getLastSignatureData().sensorData().gridCellSize() > 0.0f)
    {
        cv::Mat groundCells, obstacleCells, emptyCells;
        stats.getLastSignatureData().sensorData().uncompressDataConst(0, 0, 0, 0, &groundCells, &obstacleCells, &emptyCells);
        grid_.addToCache(stats.getLastSignatureData().id(), groundCells, obstacleCells, emptyCells);
    }
}

if(grid_.addedNodes().size() || grid_.cacheSize())
{
    grid_.update(stats.poses());
}
if(grid_.addedNodes().size())
{
    float xMin, yMin;
    cv::Mat map8S = grid_.getMap(xMin, yMin);
    if(!map8S.empty())
    {
        //convert to gray scaled map
        cv::Mat map8U = util3d::convertMap2Image8U(map8S);
        cloudViewer_->addOccupancyGridMap(map8U, grid_.getCellSize(), xMin, yMin, 0.75);
    }
}

odometryCorrection_ = stats.mapCorrection();

cloudViewer_->update();

processingStatistics_ = false;
}
```

Figure III.25 : Mettre à jour/ajouter la grille d'occupation (lorsque RGBD/CreateOccupancyGrid est vrai).

4. Mise en œuvre du RTAB-MAP pour l’acquisition automatique de la carte du bureau A19

Le mapping basé sur l'apparence en temps réel (RTAB-MAP) est une solution plus classique en RGB-D CLAQUER. Il implémente tout, depuis FVO, la détection de boucle basée sur les sacs, la carte de pose back-end optimisation et génération de cartes de nuages de points et de mailles triangulaires. Par conséquent,

RTAB-MAP donne un schéma SLAM RVB-D complet.

Nous avons exploité la plate-forme expérimentale pour effectuer un test RTAB-MAP SLAM dans un laboratoire. L'effet de fonctionnement de RTAB-MAP est illustré à la Fig.

L'interface système est divisé en quatre parties : l'interface orange dans le coin supérieur gauche est la détection de bouclage traiter.

Le système détecte les boucles et optimise les poses par reconnaissance de contraste de l'original image. L'interface verte au milieu à gauche est la fenêtre d'avertissement de détection d'erreur.

Le système contrôle les résultats de la cartographie en détectant la "dérive" de la VO frontale. Le bleu

L'interface dans le coin inférieur gauche est le module de fonctionnalités dans le VO frontal. Il continue détecte les points caractéristiques des images adjacentes et effectue la correspondance des caractéristiques pour calculer la pose de mouvement de caméra. L'interface rouge sur la droite affiche les trois intérieurs construits carte de nuage de points dimensionnelle en temps réel, et la ligne blanche est la pose de mouvement de la caméra estimée par le système.

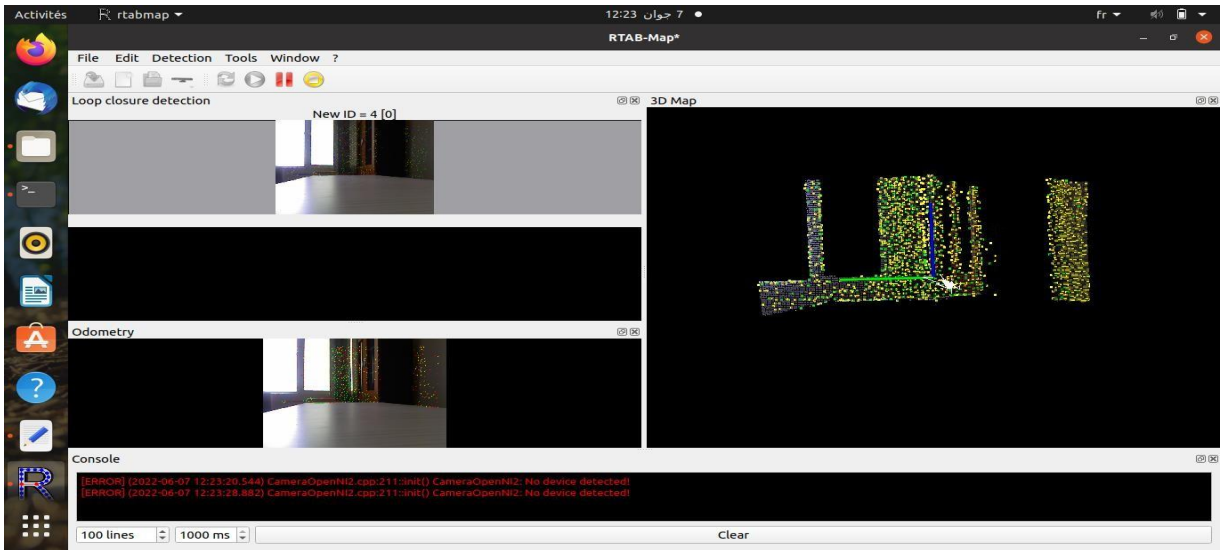


Figure III.26 : Effet de course RTAB-MAP.

4.1. Détection et correspondance des caractéristiques de l'image

Dans cette section, nous comparons et analysons trois détections de points de caractéristiques d'image couramment utilisées algorithmes, à savoir SIFT, SURF et ORB, et algorithmes de correspondance des caractéristiques d'image dans le VO frontal du système RTAB-MAP en termes de performances en temps réel et d'efficacité temporelle.

Tout d'abord, les détections de points caractéristiques SIFT, SURF et ORB sont effectuées sur l'ensemble collecté d'images via la bibliothèque de vision Open CV. L’effet expérimental de l'algorithme de détection de caractéristiques est intuitivement ressenti à travers l'espace répartition des points caractéristiques détectés. Ensuite, 50 images sur près de 2000 images d'images sont sélectionnés au hasard à partir des données expérimentales pour les tests, et le nombre moyen de caractéristiques, les points détectés par différents algorithmes de détection de caractéristiques et le temps consommé sont enregistrés par fixer des seuils raisonnables.

Dans l'expérience de construction de cartes intérieures utilisant RGB-D SLAM, l'efficacité de détection des points de fonctionnalité Front-End et le nombre de correspondances ultérieures sont très importants pour l’efficacité d'exécution globale de l'algorithme. On voit que le temps de détection moyen de chaque fonction ORB est de 0,0632 ms, ce qui est le plus court parmi les trois algorithmes de détection comparés.

Cela montre que ORB a le meilleur temps réel performance. La détection des fonctionnalités SIFT prend le plus de temps. Le nombre de fonctionnalités détectées points est le plus grand et la distribution est relativement uniforme. SURF peut extraire relativement de nombreux points caractéristiques, mais cela prend beaucoup plus de temps que ORB. Par rapport aux fonctions SIFT, la distribution des éléments SURF est principalement concentrée dans la zone périphérique. La fonction ORB la distribution est relativement concentrée et la vitesse d'extraction est la plus élevée. C'est plus adapté pour le système RGB-D SLAM appliqué à la construction de cartes intérieures en temps réel.

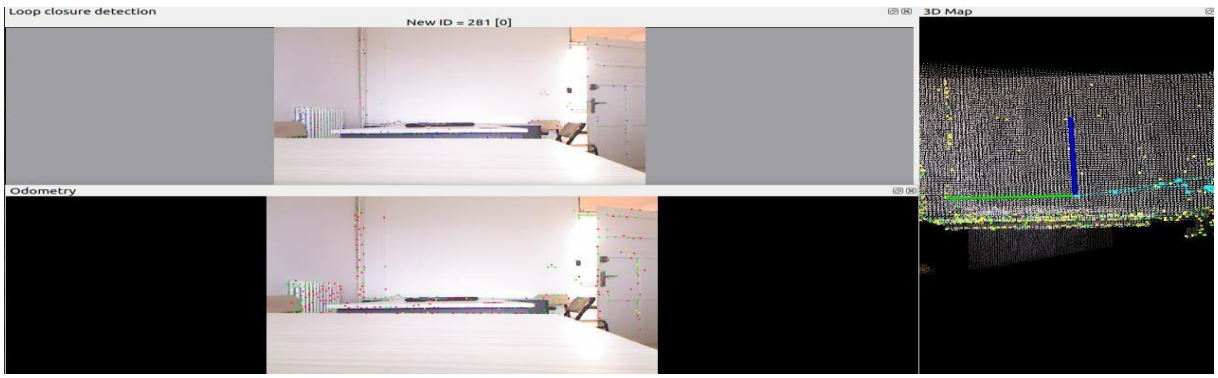


Figure III.27 : Résultats de la détection des points caractéristiques : ORB, SURF et SIFT.

4.2. Étiquetage de scène 2D

Bien que la plupart des algorithmes soient conçus pour traiter uniquement les données RVB, les informations provenant des capteurs peuvent être rassemblés afin que les pixels 2D puissent être convertis en points 3D en analysant le pixel à point transformation et sa relation entre le nuage de points organisé et l'image RVB. Au cours des dernières années, l'émergence des réseaux entièrement convolutifs (FCN) a considérablement contribué à faire progresser l'état de l'art dans le domaine de l'étiquetage de scènes 2D. Deux structures d'architecture doivent être Souligné :

- Architecture Encodeur-Décodeur.
- Architecture de représentation multi-échelle.

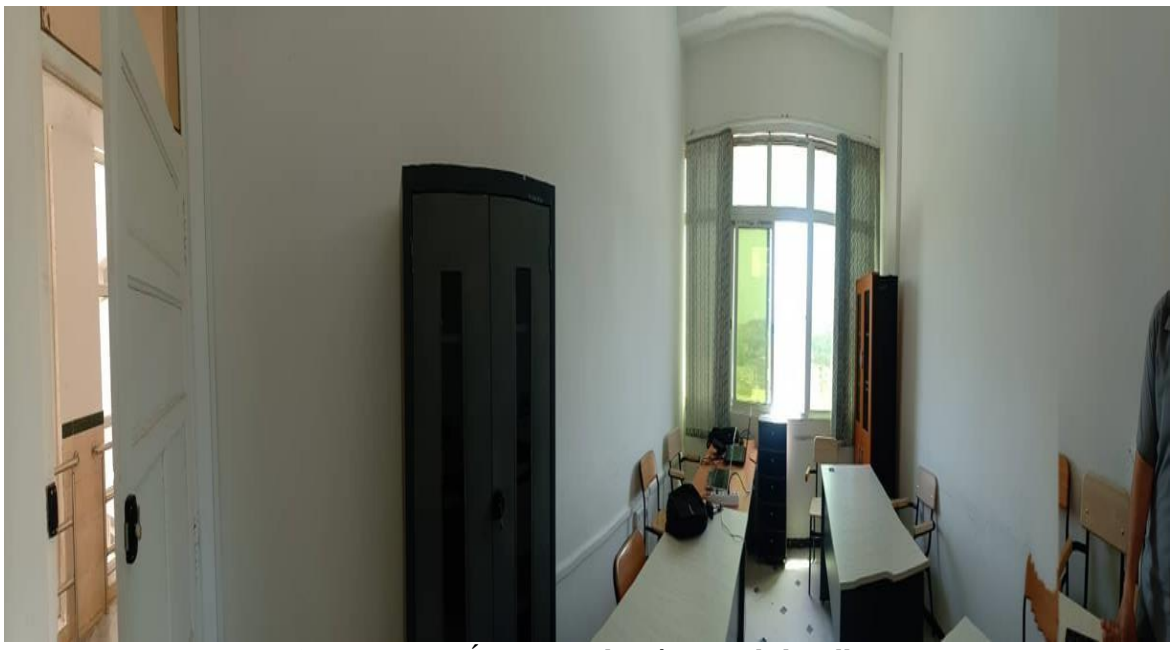


Figure III.28 : Étiquetage de scène 2D de la salle 19

4.3. Étiquetage de scène 3D

Traditionnellement, l'utilisation des informations des capteurs 3D dans les techniques de Deep Learning est moins exploitée que son homologue 2D en raison de sa simplicité. Néanmoins, l'émergence des véhicules autonomes dans les dernières années et, partant, les défis posés par la conduite autonome, a dynamisé l'approche 3D et la génération et l'accès à de grands ensembles de données. Bien qu’ils puissent être adapté pour être réutilisé, ce n'est pas une action simple en raison des différents environnements contextuels.

Comme il existe de nombreuses représentations 3D différentes, l'analyse a été limitée aux approches qui utilisez des nuages de points comme entrée car ils sont également alignés avec les caméras sélectionnées, qui fournissent des points nuages en sortie. Dans ce contexte, les plus intéressants sont :

• PointNet : Étant l'approche la plus populaire au sein des réseaux neuronaux qui consomment nuages de points directement, il s'est avéré être une approche polyvalente pour effectuer tâches de classification, de segmentation des parties et de compréhension sémantique.

• PointNet++ : Une évolution de l'approche PointNet qui surmonte ses principales limitations, comme la reconnaissance de motifs petits et détaillés car il ne capture pas les points locaux structures. De plus, cette version capture les caractéristiques intrinsèques pour la classification des matériaux non rigides objets façonnés.

• Apprendre à segmenter les nuages de points 3D dans l'espace image 2D : Projection de données 3D sur Espace d'image 2D pour tirer parti des réseaux de neurones convolutifs (CNN) bien optimisés Framework comme U-Net pour l'étiquetage de scènes 2D.

Concernant la mise en œuvre de ces techniques dans les Framework de Deep Learning existants, il est convié de mentionner que MMDetection - un Framework open-source sur PyTorch, et l'un dès le plus avancé dans son domaine – a récemment décidé d'intégrer une bibliothèque spécifique pour la mise en œuvre de l'étiquetage de scène 3D. En particulier, la bibliothèque s'appelle mmdetection3d.

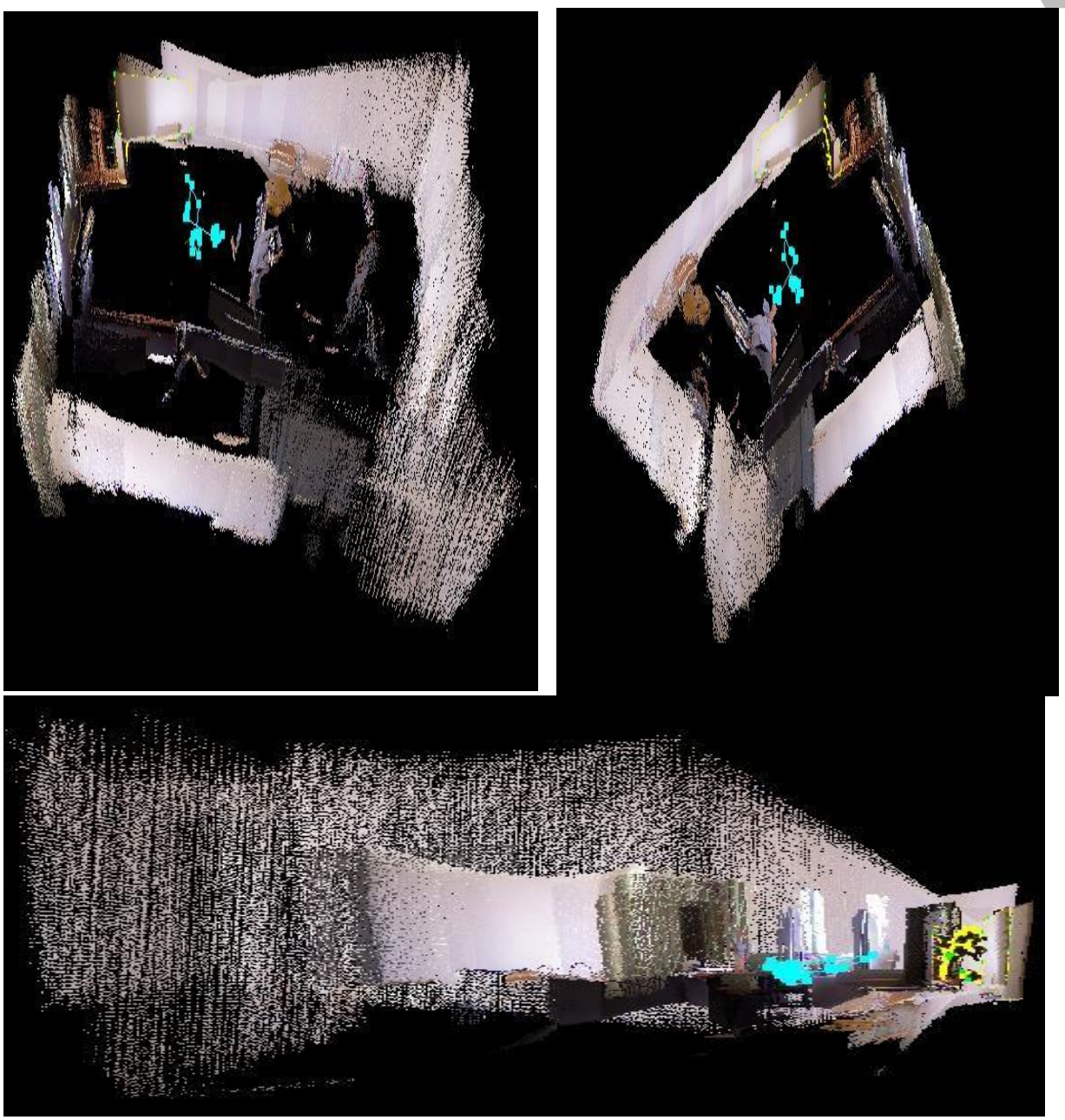


Figure III.29 : Étiquetage de scène 3D différentes.

5. Conclusion

Dans ce dernier chapitre, nous avons essayé de présenter les notions essentielles relatives à l’implémentation de notre système, qui se base sur la Réalisation D'une Outil De Cartographie Automatique De L’environnement.

Pour faciliter la compréhension de notre système, nous avons présenté également les outils, la plateforme et le langage utilisé pour le développement de ce logiciel, ainsi que les matériels utilisés, comme nous avons donné des exemples d’interfaces du notre système et nous avons aussi montré des extraits du code source.

Conclusion générale

Conclusion générale

Tout au long de la préparation de notre projet de fin d'études, nous avons essayé de mettre en pratique les connaissances acquises durant nos études universitaires et cela dans le but de réaliser un outil de cartographie automatique de l'environnement.

Au cours de cette mémoire, nous avons étudié et implémenté et présenter la version étendue RTAB-MAP, qui fournit une intégration complète avec ROS, pour la synchronisation RVB-D, la stéréo, le balayage laser et les thèmes de nuage de points, la capacité de générer Réseaux d'occupationpour tous les capteurs.

Avec la diversité des activités menées, ce projet nous a permis de consolider nos connaissances, essentiellement dans la programmation.

En conséquence, RTAB-MAP est désormais une approche SLAM polyvalente basée sur des graphes qui peut être utilisé prêt à l'emploi par les utilisateurs novices de SLAM et pour le prototypage sur des plates-formes Android avec divers Capteurs et capacités de traitement. Il peut être utilisé pour comparer les performances entre les ensembles de données et mener des évaluations en ligne. Les capteurs nécessaires au SLAM, qu'ils soient bon marché ou chers, sont tout cela il a des limites pour augmenter la précision des traductions, la qualité des cartes et les ressources informatiques.

La flexibilité est démontrée dans cet article en faisant des comparaisons significatives entre le visuel et le lidar. RTAB-MAP est distribué en tant que bibliothèque open source et est déjà disponible pour la communauté.

RTAB-MAP est actuellement l'un des meilleurs packages ROS en utilisation active (plus de 1 600 questions sur ses 18 forums, dépôts GitHub 19 et ROS 20) par la communauté, pour un SLAM à faible coût avec RGB-D et stéréo appareils photo. Notre objectif avec RTAB- MAP est de continuer à intégrer de nouvelles approches de mesure de distance qui manquent de ROS approprié Intégration, pour faciliter la comparaison des garanties SLAM de navigation autonome du robot mobile plates-formes.

Bibliographies & Webographies

Bibliographies & Webographies

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

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

[3] : histoire-des-robots, 2015, Available https://www.timetoast.com/timelines/histoire-des-robots, Accessed 25 May 2022.

[4] : Reuben Hoggett. Sadly he, History Makers, 2016, Available http://davidbuckley.net/DB/HistoryMakers.htm, Accessed 25 May 2022.

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

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

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

[8] : https://actualites-decalees.autojournal.fr/de-droles-de-voitures-fp-223504.html#item=1

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

[10] : Onera, Available, https://www.onera.fr/fr/pepites/modelisation-3d-temps-reel-de-lenvironnement-pour-la-navigation-autonome-des-drones Accessed 25 May 2022

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

[12] : Le Blog Génération Robots, Available, https://blog.generationrobots.com/fr/qu-est-ce-que-la-technologie-lidar/ 24 Juin 2022

[13] Eric Beaudry, Planification de tâches pour robotique mobile, thèse de doctorat, université de Sherbrooke, Canada, Mai 2006.

[14] COMPONENT-BASED SLAM IN RTAB-MAP A. (Ángel) Lorente Rogel

[15] Labbé, M. and F.Michaud (2019), RTAB-Map en tant que lidar open source et bibliothèque visuelle simultanée de localisation et de cartographie pour une opération en ligne à grande échelle et à long terme, dans Journal of Field Robotics, volume 36, Wiley Online Library, pp. 416–446.

[16] Li Y(2012), "Hand gesture recognition using kinect". In IEEE 3rd international conference on software ingineering and service science (ICSESS), June 2012, pp196-199.

[17] Liu X, Fujimura K (2004), "Hand gesture recognition using depth data". In proceeding sixth IEEE international conference on automatic face and gesture recognition, May 2004,pp 529-534.

[18] M.R. Andersen et al. "Kinect Depth Sensor Evaluation for Computer Vision Applications" Electrical and Computer Engineering, Technical report ECE-TR-6, 2012.

[19] Labbé, M. and F.Michaud (2018), Long-termonlinemulti-session graph-based SPLAMwith
memory management, in *Autonomous Robots*, volume 42, Springer, pp. 1133–1150.

[20] caramuzza, D. and F. Fraundorfer (2011), Visual odometry [tutorial], in *IEEE robotics & automationmagazine*, volume 18, IEEE, pp. 80–92.

[21] Mark te Brake (2021), "Modular SLAM improvements for the Real-Time Appearance Based Mapping (RTAB-Map) algorithm",master's thesis, University of Twente.

[22] Labbe,M. and F.Michaud (2013), Appearance-based loop closure detection for online large-scale and long-termoperation, in *IEEE Transactions on Robotics*, volume 29, IEEE, pp. 734–745.

[23] Blanco-Claraco, J.-L. (2019), A Modular Optimization Framework for Localization and Mapping.,
in *Robotics: Science and Systems*.

[24] Winnie Ondara, How to Install ROS Noetic on Ubuntu 20.04 LTS, https://linoxide.com/how-to-install-ros-noetic-on-ubuntu-20-04/?fbclid=IwAR32TCnsAj0QdhGCD7Gs9B1KUqMafQ977eiQVq4nQ89GO4wAwMOt3N0Lt2g, 21 Juin 2022,

[25] Le Crabe Info, Installer Ubuntu 20.04 LTS : le guide complet, Available https://lecrabeinfo.net/installer-ubuntu-20-04-lts-le-guide-complet.html#installer-ubuntu-2004-lts, Accessed 20 April 2022.

[26] Ubuntu20.04 De l'installation ROS-noetic à la simulation SLAM de turtlebot3, https://linuxtut.com/fr/fee91e8e883b4d0eeb74/?fbclid=IwAR2GT9Tun1TV2sAAPCJpIjNdLYwG4x8ZoiR0XE2f_LIWQVYwVZLIQmfWOSs

[27] ubuntu20.04 Run RTAB-MAP with ROS Noetichttps, //linuxtut.com/en/ad730b7f6ac87d9a0304/?fbclid=IwAR0w87dEvLsS0qV6J9_5gLHyrQ2w-I0lDKlR25MYEHponjL4dmr7Q6pmjRo

[28] L. Joseph, *Robot operating system for absolute beginners: Robotics programming made easy*. Berkeley, CA: Apress, 2018.

[29] M. Quigley, B. Gerkey and W. D. Smart, *Programming robots with ROS*, 1st ed. Sebastopol: O'Reilly & Associates Inc., 2015.

[30] R. Siegwart, I.R. Nourbakhsh and D. Scaramuzza, Ed., *Introduction to Autonomous Mobile Robots.* 2nd ed. Cambridge, Mass.: MIT Press, 2011.

[31] T. Foote, "Tf: The transform library," in *TePRA '13 Proceedings of the IEEE International Conference on Technologies for Practical Robot Applications*, Woburn, MA, USA, IEEE, Apr. 22 – 23, 2013. [Online]. doi: 10.1109/TePRA.2013.6556373 [Accessed: Jan. 27, 2019].

[32] Open Source Robotics Foundation, *Gazebo Tutorials*, Gazebo, 2014. [Online]. Available: http://gazebosim.org/tutorials [Accessed: Jan 3, 2019].

[33] Dave Hershberger, David Gossow and Josh Faust, *rviz*, Ros.org, 2016. [Online]. Available: http://wiki.ros.org/rviz [Accessed: Aug 1, 2019].

[34] Mathieu Labbe, *rtabmap_ros*, Ros.org, 2019. [Online]. Available: http://wiki.ros.org/rtabmap_ros [Accessed: Aug 1, 2019].