MEMOIRE DE MASTER

Informatique Automatique et Système Embarqué

Master en Science de l'ingénieur

Réalisé par : Mustapha HAMMOUNY

Navigation bio-inspirée des robots mobiles et odométrie visuelle

Soutenu le 26 Décembre 2016 devant le Jury :

Président :
Mr. A. ETTOUHAMI — **PES, Faculté des Sciences de Rabat.**

Examinateurs :
Mr. N. ZAHID — **PES, Faculté des Sciences de Rabat.**
Mr. M. JEDRA — **PES, Faculté des Sciences de Rabat.**
Mr. Y.RAOUI — **PA, Faculté des Sciences de Rabat. (Encadrant)**

Année universitaire : 2015/2016

Avant-propos

Ce travail a été effectué au sein du Laboratoire Conception et Systèmes, Université Mohamed V, Faculté des Sciences, Rabat, Maroc.

Avant de feuilleter ce modeste rapport que j'espère sera à la hauteur de vos attentes. Je tiens tout d'abord à remercier dans un premier temps, toute l'équipe pédagogique de la faculté des sciences de Rabat, et tous les intervenants professionnels pour avoir assuré notre formation.

Je voudrais exprimer ma profonde reconnaissance à Monsieur Younes RAOUI, professeur à la Faculté des Sciences Rabat, mon encadrant de projet de fin d'étude pour m'avoir initié sur ce sujet, pour ses conseils et ses commentaires précieux qui m'ont permis de surmonter les difficultés, et qui par son expérience et son enthousiasme, m'a aussi donné beaucoup de propositions tout au long de ce mémoire.

Mes sincères remerciements s'adressent également aux membres de jury Noureddine ZAHID, Mohamed JEDRA et Aziz ETTOUHAMI, qui m'ont fait l'honneur de bien vouloir juger ce travail et l'enrichir par leurs différentes remarques, observations, critiques et suggestions.

Un grand merci à mes parents qui sans eux rien ne serait possible. Leurs sages conseils ont été extrêmement fructueux pour mon éducation, ma formation et surtout ma conduite sociale dont je suis particulièrement fière.

A mes familles,

A mes professeurs,

A mes amis.

"I have not failed 700 times. I have not failed once. I have succeeded in proving that those 700 ways will not work. When I have eliminated the ways that will not work, I will find the way that will work."

"Je n'ai pas échoué 700 fois. Je n'ai pas échoué une seule fois. J'ai réussi à prouver que ces 700 façons ne marcheront pas. Lorsque j'aurai éliminé les façons qui ne fonctionnent pas, j'aurai trouvé la façon qui fonctionnera"

Thomas Edison, about creating the light bulb.

Résumé

La cartographie, la localisation et la navigation sont des sujets et des défis majeurs pour la robotique mobile. Pour qu'un robot navigue intelligemment dans un environnement à grande échelle, le robot doit être capable de se localiser dans l'espace et de construire en même temps une carte de l'environnement (localisation et cartographie simultanément -SLAM). La plupart des solutions au problème SLAM ont impliqué des algorithmes probabilistes.

Récemment, une nouvelle approche de problème de SLAM était introduite comme un modèle alternative bio-inspirée au SLAM bayésien, connu sous le nom de RatSLAM. RatSLAM est un algorithme de localisation et de cartographie biologiquement inspiré basé sur des modèles computationnels de l'hippocampe chez les rongeurs. Ce modèle permet à un robot d'exécuter le SLAM en temps réel et de produire des représentations constantes et stables dans une gamme d'environnements intérieurs et extérieurs complexes.

Dans ce présent travail, nous avons étudié l'algorithme de RatSLAM et comparé avec les autre algorithmes de SLAM probabiliste, puis nous avons proposé une méthode d'odométrie visuelle basée sur l'extraction des caractéristiques dans les images pour construire la trajectoire du robot puis la comparer avec la trajecoitre fournie par le RatSLAM.

Mots clés : SLAM, RatSLAM, Bio-inspiré, Odometrie visuelle, Hippocampe, Robot mobile, Navigation.

Summary

Mapping, localization and navigation are major topics and challenges for mobile robotics. In order to navigate intelligently within a large scale environment, the robots must be able to localize themselves in space and to build at the same time a map of the environment (Simultaneous Localization And Mapping -SLAM). The most solutions of SLAM problem have involved probabilistic algorithms.

Recently, a novel motivated approach to SLAM problem was introduced as an alternative bio-inspired approach to Bayesian SLAM, known as RatSLAM. RatSLAM is a biologically inspired localization and mapping algorithm based on computational models of the rodent hippocampus. This model enables a robot to perform SLAM in real time and to produce consistent and stable representations in a range of indoor and outdoor complex environments.

In this present work, we have studied the RatSLAM algorithm and compared it with the other algorithms of probabilistic SLAM, and then we have proposed a visual odometry method based on the extraction of the characteristics in the images to construct the trajectory of the robot and then compare it with the trajectory provided by the RatSLAM.

Keywords : SLAM, RatSLAM, Bio-inspired, Visual odometry, Hippocampus, Mobile Robot, Navigation.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

La thématique de la navigation autonome constitue l'un des principaux axes de recherche dans le domaine des véhicules intelligents et des robots mobiles. Dans ce contexte, on cherche à doter le robot d'algorithmes et de méthodes lui permettant d'évoluer dans un environnement complexe et dynamique, en toute sécurité et en parfaite autonomie.

Les algorithmes assurant les tâches de localisation et de cartographie occupent une place importante parmi les différents algorithmes utilisés dans le domaine de la navigation autonome. En effet, sans information suffisante sur la position du robot (localisation) et sur la nature de son environnement (cartographie), les autres algorithmes (génération de trajectoire, évitement d'obstacles ...) ne peuvent pas fonctionner correctement.

Par ailleurs, un robot a besoin de connaître sa position pour pouvoir cartographier un environnement. D'un autre côté, il doit absolument disposer d'une carte préétablie de son environnement pour pouvoir s'y localiser. Ces deux tâches sont liées. Lorsqu'on ne dispose pas d'une carte de l'environnement, ni de données de localisation, le robot doit trouver ces informations simultanément. Cette opération est effectuée grâce aux d'algorithmes de SLAM : Simultaneous Localization and Mapping.

Ce sujet s'intéresse à la localisation et la cartographie simultanées (SLAM) permettant au robot de calculer sa position et sa carte simultanément en utilisant l'odométrie et les perceptions des repères visuels à l'aide d'une caméra, la méthode étudiée RatSLAM est bio-inspirée effectuant le SLAM avec une carte cognitive comme le rat. En 2004, la méthode RatSLAM était introduite comme alternative bio-inspirée au SLAM bayésien. Le rat a la capacité à naviguer dans un environnement de plusieurs centaines de mètres (ou même des kilomètres) et revenir à sa maison. S'il trouve dans un champ où il se déplace un mûr, des grains ou des fruits, il peut y

revenir un jour sans se perdre grâce à la carte qu'il a construit de son environnement. Le rongeur peut calculer sa position avec les cellules de place et de grille qui se situent dans l'hippocampe. En outre, à l'aide de l'apprentissage nommé Hebbien, le rat réalise une connexion entre les stimuli visuels et les actions générées par des connexions dans le réseau de cellule de place et cellule de vue à l'aide d'attracteurs.
Dans ce projet, la méthode RatSLAM sera étudiée et simulée en utilisant le package RatSLAM disponible sous Robotics Operating System (ROS).

Notre document est organisé de la façon suivante :

Dans le chapitre 1, nous allons voir une introduction générale à la robotique mobile, les moyennes de perceptions et ainsi quelque domaine d'application.

Le chapitre 2 est consacré à l'´étude de la navigation, la localisation et la cartographie. Nous présentons aussi les principales méthodes probabilistes utilisées dans la littérature afin de résoudre le problème du SLAM.

Le chapitre 3 résume les notions de base nécessaires à la compréhension de l'odométrie visuelle. Nous présentons aussi dans ce chapitre des différents algorithmes de vision par ordinateur utilisé pour extraire les caractéristiques dans les images.

Dans le chapitre 4, nous passerons en revue un état de l'art des fondements neuronaux de la navigation chez les rongeurs. Ensuit, nous verrons les différentes unités de traitement de l'algorithme RatSLAM.

Le chapitre 5 se focalisera à l'étude des deux algorithmes de l'odométrie visuelle, le premier qui est utilisé dans le RatSLAM et le deuxième que nous proposons et qui est basé sur l'extraction des points d'intérêt visuel.

Finalement, les expériences et les résultats sont effectués dans le chapitre 6.

Nous terminons le mémoire par une conclusion générale sur l'ensemble de ces études et nous proposons des perspectives à notre travail.

Chapitre 1

Généralités sur la robotique mobile

Ce chapitre nous a permis de présenter un exposé sur le domaine de la robotique mobile, en définissant les robots mobiles, les moyennes de perception de ces machines et également quelque domaine d'application.

Introduction

Un robot est une machine programmable capable d'effectuer une série complexe d'actions d'une façon automatique. Pour certaines tâches, les robots peuvent être supérieurs aux humains en termes de qualité du travail qui sont produit car, contrairement aux humais, ils sont jamais fatigués, ils peuvent travailler dans les conditions physiques qui sont inconfortables ou même dangereux, ils peuvent fonctionner dans des conditions sans air et ils ne s'ennuient pas par la répétition.

1.1 Les différents types des robots

En robotique, on distingue les robots en deux principaux types : les robots manipulateurs et les robots mobiles.

1.1.1 Les robots manipulateurs

Les robots manipulateurs ou stationnaires sont des robots qui travaillent sans changement de leurs positions comme les bras robotique industriel. Se référant du robot comme « stationnaire » ne signifie pas que le robot ne se déplace pas en réalité, le mot « stationnaire » signifie que la base du robot ne se déplace pas pendant le fonctionnement.

FIGURE 1- 1 : Bras robotique

1.1.2 Les robots mobiles

Les robots mobiles sont des robots qui peuvent changer leurs positions et se déplacer dans les déferrent environnement contrairement aux robots stationnaire qui ont une base fixe. Les robots mobiles les plus connue sont :

Les robots mobiles à roue

Des robots mobiles à roues sont les robots qui changent leurs positions avec l'aide de leurs roues. Ce sont en effet les systèmes les plus étudiés, parce qu'ils sont plus simples à réaliser que les autres types de robots mobiles, ce qui permet d'en venir plus rapidement à l'étude de leur navigation. Ce type de robots est notamment très souvent utilisé pour l'étude des systèmes autonomes.

FIGURE 1- 2 : Robot à roue

Les robots mobiles à pattes

Les robots à pattes sont des robots mobiles, semblables aux robots à roues. Comme leur nom indique, ce type de robot utilise les pattes pour contrôler ces mouvements comme par exemple le robot humanoïde, mais également les robots avec un nombre de pattes plus élevés qui offrent de bonnes propriétés pour la locomotion en milieu difficile (milieux forestiers et agricoles). La stabilité des mouvements de ce type de robots est en particulier un thème de recherche important.

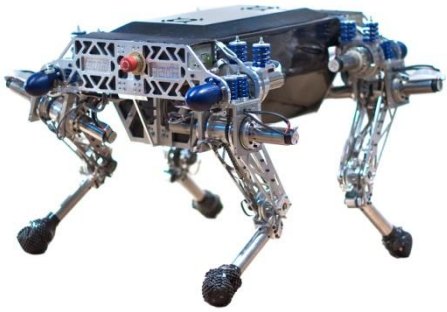

FIGURE 1- 3 : Robot à pattes

Les robots mobiles à chenilles

L'utilisation des chenilles présente l'avantage d'une bonne adhérence au sol et d'une faculté de franchissement d'obstacles .L'utilisation de ce type des robots est orientée vers l'emploi sur sol accidente ou de mauvaise qualité au niveau de l'adhérence.

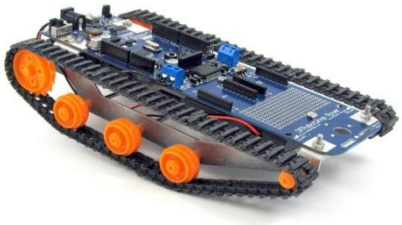

FIGURE 1- 4 : Robot à chenilles

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 robots se fait dans des thématiques spécifiques avec des problèmes particuliers à l'application visée.

1.2 Robot mobile autonome

Il existe 2 principaux modes de fonctionnement pour un robot mobile : télé-opéré et autonome.

En mode télé-opéré [1], une personne pilote le robot à distance. Elle donne ses ordres via une interface de commande (joystick, clavier/souris...), et ceux-ci sont envoyés au robot via un lien de communication (internet, satellite ...). D'ailleurs, suivant le niveau de télé-opération, le terme « robotique » est plus ou moins justifié. Le robot doit donc obéir aux ordres de l'opérateur qui perçoit l'environnement autour du robot, par différents moyens (retour d'image, retour haptique…), de manière à donner des ordres adaptés au robot. Dans ce domaine, les efforts de recherche sont beaucoup portés sur les problèmes liés au réseau de

télécommunication (retards dans le réseau de communication, problèmes de commande, pertes de données).

A l'inverse, en mode autonome [2] le robot doit prendre ses propres décisions. Cela signifie qu'il doit être capable à la fois de percevoir correctement son environnement, mais également de savoir comment réagir en conséquence, suivant le niveau d'autonomie. C'est à lui de planifier son parcours et de déterminer avec quels mouvements il va atteindre son objectif.

Les recherches dans ce domaine portent principalement d'une part sur la localisation du véhicule autonome et la cartographie de son environnement, d'autre part sur le contrôle de tels véhicules (structure de contrôle, stratégies de commande, planification).

1.3 Classification

Une classification est proposée dans la littérature qui définit le degré d'autonomie du robot mobile.

- 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. Ce type de véhicule pose des problèmes d'un niveau de complexité élevé de représentation des connaissances et de capacité décisionnelle.

1.4 Les moyens de perception en robotique mobile

La notion de perception en robotique mobile est relative à la capacité du système à recueillir, traiter et mettre en forme des informations utiles au robot pour agir et réagir dans le monde qui l'entoure. 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'environnement dans lequel il évolue. Le choix des capteurs dépend bien évidemment de l'application envisagée.

L'objectif de capteur est de traduire en une information exploitable des données représentant des caractéristiques de l'environnement. Ce dispositif est soumis à l'action d'une mesurant non électrique, et fournit un signal électrique à sa sortie.

Les capteurs ont une place prépondérante dans le système de traitement d'un robot. Ils peuvent à la fois informer le robot sur le milieu extérieur et l'informer sur ses propres actions en vérifiant l'état de ses actionneurs. Ils sont donc l'élément indispensable à un robot autonome pour savoir ce qu'il fait, ce qui se passe et prendre les bonnes décisions en conséquence.

Les capteurs utilisés pour la perception de l'environnement sont nombreux. Nous citons:

1.4.1 Les capteurs proprioceptifs

Les capteurs proprioceptifs permettent une mesure du déplacement du robot. Ce sont les capteurs que l'on peut utiliser le plus directement pour la localisation, mais ils souffrent d'une dérive au cours du temps qui ne permet pas en général de les utiliser seuls [3].

- Odométrie.
- Les systèmes radar doppler.
- Les systèmes inertiels

1.4.2 Les télémètres

Il existe différents types de télémètres, qui permettent de mesurer la distance aux éléments de l'environnement, utilisant divers principes physiques. Nous citons :

- Télémètres à ultrason.
- Télémètres à infrarouge.
- Télémètres laser.

1.4.3 Les caméras

L'utilisation d'une caméra pour percevoir l'environnement est une méthode attractive car elle semble proche des méthodes utilisées par les humains et fournit une grande quantité d'information sur l'environnement.

Le traitement des données volumineuses et complexes fournies par ces capteurs est cependant souvent difficile, mais c'est une voie de recherche très explorée pour la robotique.

- Caméras simples.
- Caméras stéréoscopiques.
- Caméras panoramiques.

1.5 Applications

Aujourd'hui, le marché commercial de la robotique mobile est toujours relativement restreint en dehors des robots aspirateurs vendus à plusieurs millions d'exemplaires. Cependant, 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. Le tableau ci-après résume de manière non exhaustive les diverses applications des robots mobiles [3].

Domaines	Applications
Industrie nucléaire	- surveillance de sites - manipulation de matériaux radioactifs
Sécurité civile	- neutralisation d'activité terroriste - déminage - pose d'explosif - surveillance de munitions
Militaire	- surveillance, patrouille - pose d'explosifs
Chimique	- surveillance de site - manipulation de matériaux toxiques
Médecine	- assistance d'urgence - aide aux handicapés physiques, aux aveugles
Lutte contre l'incendie	- localisation d'une source d'incendie - détection de fumée - suppression de flammes
Sous-marine	- recherche de navires immergés - inspection des fonds marins
Agricole	- cueillette de fruits - traite, moisson, traitement des vignes...
Nettoyage	- coque de navire - nettoyage industriel
Espace	- exploration
Industriel	- convoyage - surveillance

Table 1- 1 : Domaines d'applications des robots mobiles

Conclusion

Un robot mobile est un système mécanique, électronique et informatique agissant physiquement sur son environnement en vue d'atteindre un objectif qui lui a

été assigné. Cette machine est polyvalente et capable de s'adapter à certaines variations de ses conditions de fonctionnement. Elle est dotée de fonctions de perception, de décision et d'action.

Ainsi, le robot devrait être capable d'effectuer des tâches diverses, de plusieurs manières, et accomplir correctement sa tâche, même s'il rencontre de nouvelles situations inattendues.

Ce premier chapitre fournit une présentation générale sur les robots mobiles, en définissant c'est quoi un robot, et les principaux types en robotique. II donne également une explication de l'architecture des robots mobiles et les moyenne de perception de ces machines et finalement quelque domaine d'application.

Chapitre 2

Navigation

Ce chapitre est consacré à l´étude de la navigation, la localisation et la cartographie, Nous décrivons également les principales méthodes probabilistes utilisées dans la littérature afin de résoudre le problème du SLAM à savoir, le filtrage de kalman étendu et le filtrage particulaire.

Introduction

Il existe de nombreuses applications pour lesquelles le développement de robots autonomes, capables d'évoluer en terrains naturels, présente un intérêt. La plupart du temps, celles-ci prennent place dans des contextes où l'homme ne peut intervenir directement et où apparaissent parfois de fortes contraintes sur les délais et les débits des communications : exploration planétaire, déminage, reconnaissance militaire ou, plus généralement, intervention en milieu hostile et distant.

Le point commun entre ces applications est qu'elles requièrent l'exécution de missions de haut niveau comme « rallier tel objectif », « explorer telle région » ou « surveiller telle zone ». Ces missions se déclinent en différentes tâches qui, en particulier, forment la capacité de naviguer.

La navigation autonome est définie comme la capacité d'enchaîner des actions de perception et de déplacement dans un environnement initialement peu ou pas connu. Il s'agit, d'une composante fondamentale pour les applications visées par la robotique mobile [4].

La réalisation de telles tâches de navigation autonome requiert la mise en œuvre, l'intégration et la coopération d'un très large spectre de fonctionnalités : capacités d'exécution d'actions sensori-motrices « bas-niveau », construction de représentations de l'environnement, planification et de contrôle de l'exécution de trajectoires, élaboration de stratégies de déplacements et de perception, etc. Pour ces raisons, la navigation autonome est une illustration exemplaire de l'intelligence des machines. Elle reste à ce jour l'ambition d'un grand nombre de recherches en robotique mobile.

2.1 Définition de navigation

La navigation est la capacité du robot mobile autonome de planifier son mouvement en temps réel et de naviguer en toute sécurité d'une place à une autre [5]. Les éléments nécessaires à un robot pour faire la navigation sont :

- Un système mécanique qui est capable de faire le déplacement d'un point A à un point B dans un environnement.
- Capteurs pour faire des perceptions de la situation actuelle.
- La cognition pour percevoir l'information détectée et de décider comment agir pour atteindre un tel objectif.
- Un système de contrôle pour mettre en œuvre le comportement désiré.

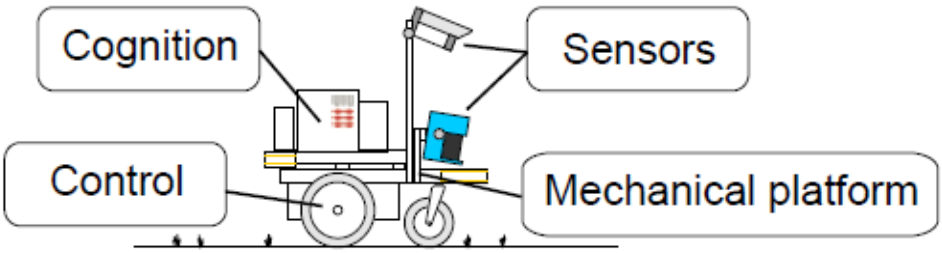

FIGURE 2- 1 : Les principaux éléments pour la navigation

Le processus robuste de navigation exige quatre aspects, à savoir :

- Perception : le robot doit interpréter les informations fournies par ses capteurs pour extraire des données significatives.
- Localisation : le robot doit déterminer sa position par rapport à des objets spécifiques dans l'environnement.
- Planification de chemin : est le processus de trouver un chemin optimal entre un point de départ vers l'emplacement cible sans aucune collision
- Contrôle du mouvement : est la capacité du robot de transférer les informations sensorielles en un mouvement physique dans un monde réel, le robot doit moduler ses sorties de moteur pour atteindre la trajectoire souhaitée.

2.2 Stratégies de navigation

Le robot recherche à naviguer dans un environnement qui a généralement un point de départ, un objet de cible et un certain nombre d'obstacles de tailles et formes aléatoires. Le robot se déplace du point de départ avec l'objectif de trouver la cible. Le robot doit trouver un chemin libre et sans obstacle qui couvre tout l'environnement. Il doit également se localiser dans l'environnement et être conscient lorsque le processus de recherche est accompli [6].

Les stratégies de navigation diffèrent selon le type d'environnement statique (obstacles statiques) ou dynamique (obstacles statiques et dynamiques). Les deux catégories peuvent être subdivisées en environnements inconnus et connus. Dans les différents environnements, il existe de nombreux algorithmes qui traitent du problème de la navigation du robot.

La plupart des algorithmes de planification de navigation supposent que le robot mobile possède une connaissance détaillée des emplacements de départ et de destination, et donc de la direction entre eux, afin de trouver un chemin optimal entre ces deux emplacements et d'éviter les obstacles. Certains algorithmes nécessitent des informations environnementales supplémentaires ou même une carte complète. Selon Zhu, et al. [7], les algorithmes de navigation sont classés dans la planification globale et locale.

2.2.1 Planification globale de la navigation

Les algorithmes de navigation globale planifient le chemin du robot du début au but en recherchant un graphe qui représente une carte de l'environnement global. Le graphe d'environnement est construit soit hors ligne, soit en ligne. Dans le premier cas, la carte complète est initialement chargée dans le robot, puis l'algorithme de navigation détermine le chemin optimal avant que le robot ne commence son mouvement. La méthode hors ligne suppose que le robot se déplace dans un environnement statique et il a un système de mouvement précis pour satisfaire les conditions de navigation. Ces déclarations sont peu réalistes pour un robot réel et donc, cette méthode est rarement utilisée dans la navigation de robot.

Au contraire, dans la technique en ligne, bien que la carte d'environnement soit chargée dans le robot, l'algorithme de navigation continue de la mettre à jour en utilisant les capteurs du robot. Cette méthode permet au robot de naviguer dans des environnements dynamiques et de corriger de façon continue son emplacement dans la carte. Cependant, les méthodes de planification globale ont trois inconvénients intrinsèques : leur coût de calcul est très élevé, complexes à construire et il est difficile d'obtenir un modèle graphique précis.

2.2.2 Planification locale de la navigation

Les algorithmes de navigation locale n'ont pas besoin d'informations a priori de l'environnement car ils utilisent directement les informations actuelles et locales des capteurs, sans construire une carte globale de l'environnement. Par conséquent, ces algorithmes sont utilisés pour donner une capacité à un robot mobile de naviguer depuis le point de départ vers le point cible dans des environnements inconnus ou dynamiques.

Pendant que le robot navigue, il évite les obstacles qui se trouvent sur son chemin et continue à mettre à jour des informations significatives, telles que la distance entre son emplacement actuel et la position cible. Typiquement, les algorithmes de navigation locaux sont faciles à construire et optimale pour les

applications temps réel. La solution optimale pour la trajectoire prescrite n'est pas garantie si en utilisant seulement les informations locales. Si le robot ne détecte aucun chemin qui existe, c'est-à-dire un minimum local, les algorithmes terminent son mouvement et indiquent que la cible est inaccessible.

Ces deux approches sont souvent utilisées ensemble, à savoir la méthode globale est utilisée pour réaliser une trajectoire possible et la méthode locale est utilisée pour l'optimisation locale de la trajectoire et pour éviter des obstacles inattendus.

2.3 Navigation utilisant une carte

Le processus complet qui permet à un robot de mémoriser son environnement, puis de s'y déplacer pour rejoindre un but, peut être découpé en trois phases [3] : la cartographie, la localisation et la planification. Ces trois phases permettent de répondre aux trois questions fondamentales pour la tâche de navigation par carte : Où suis-je ? Où sont les autres lieux par rapport à moi ? Et Comment puis-je atteindre mon but ?

- La cartographie est la phase qui permet la construction d'une carte reflétant la structure spatiale de l'environnement à partir des différentes informations recueillies par le robot.
- Une telle carte étant disponible, la localisation permet alors de déterminer la position du robot dans la carte qui correspond à sa position dans son environnement réel.
- La planification, enfin, est la phase qui permet, connaissant la carte de l'environnement et la position actuelle du robot, de décider des mouvements à effectuer afin de rejoindre un but fixé dans l'environnement.

Ces trois phases sont évidemment fortement interdépendantes. L'ordre dans lequel elles sont citées fait directement apparaître le fait que la seconde phase dépend de la première.

En effet, estimer sa position au sein d'une carte de l'environnement suppose implicitement que cette carte existe et qu'elle contient la position courante du robot. De même, la troisième phase dépend des deux premières, car la planification suppose que l'on connaisse sa position et que la carte de l'environnement représente une portion de l'environnement contenant au moins un chemin reliant cette position au but qui doit être atteint. Mais la relation entre les deux premières phases est plus subtile qu'une simple relation d'antériorité : c'est le même problème que pour l'œuf et la poule. Chacun des deux éléments peut, en effet, être considéré comme préalable à l'autre mais dépend aussi de l'autre pour sa réalisation.

Dans le cas de la cartographie et de la localisation, nous avons déjà vu que la localisation repose sur une phase préalable de cartographie. Mais pour construire une carte, il est nécessaire de savoir où ajouter, dans la carte partielle déjà existante, toute nouvelle information recueillie par le robot. Cela requiert donc une phase

préalable de localisation au sein de la carte partielle déjà existante. Pour un robot complètement autonome, il est donc impossible de ne pas traiter ces deux problèmes simultanément. Dans la littérature scientifique, on parle ainsi de problème de "Simultaneous Localization and Mapping" (SLAM).

Dans le cas où l'on autorise un opérateur humain à intervenir dans le processus, il est évidemment possible de découpler ces deux phases. Dans les applications réelles, il est fréquent que l'on fournisse au robot une carte construite au préalable et qu'on ne s'intéresse qu'à l'estimation de la position au sein de cette carte pour qu'il puisse accomplir sa tâche.

La carte peut alors être obtenue de différentes manières. Il est par exemple possible d'utiliser un plan d'architecte d'un bâtiment pour le transformer en une carte utilisable par le robot. Il est également possible d'utiliser le robot dans une phase supervisée de cartographie. Au cours de cette phase, la position du robot peut être calculée de manière précise par un dispositif externe au système de navigation, et ne nécessite donc pas que le système estime de lui-même la position.

Connaissant la position précise du robot, il est alors relativement simple de construire une carte de l'environnement. Il est également possible d'utiliser un algorithme de SLAM comme sur un robot autonome, mais de corriger la carte avant de l'utiliser réellement.

2.4 Localisation et cartographie

2.4.1 Localisation

Il existe trois types de capacités regroupées sous le terme «localisation», de complexités différentes.

Le suivi de position

C'est la capacité de mettre à jour une estimation existante de la position au vu de données proprioceptives ou de perceptions nouvellement acquises. Dans le cas des données proprioceptives, cette mise à jour concerne un déplacement du robot et va en général diminuer la précision de l'estimation courante de la position, à cause de l'erreur sur la mesure.

Dans le cas de perceptions, au contraire, cette mise à jour va en général permettre d'améliorer cette estimation grâce au lien avec l'environnement fourni par ces données. L'utilisation de cet ancrage dans l'environnement est fondamentale pour assurer que l'estimation de la position reflète correctement la position du robot dans l'environnement réel. Cette mise à jour intégrant les deux types de données permet de combiner les avantages inhérents aux deux types d'information afin d'estimer au mieux la position du robot.

En pratique, toutefois, le suivi de position est problématique car il repose sur une estimation initiale de la position qui doit souvent être fournie par une source extérieure. De plus, si la position estimée s'écarte trop de la position réelle du robot, il peut très bien être impossible de parvenir à corriger l'erreur et de retrouver la position réelle, ce qui conduit à une dérive de l'algorithme.

La localisation globale

La localisation globale est plus générale et permet de retrouver la position du robot sans qu'aucune estimation initiale ne soit fournie. Cette capacité est très importante du point de vue de l'autonomie, car elle permet au robot de trouver sa position initiale, dans toutes les conditions, sans intervention extérieure. Elle permet, par exemple, de couper l'alimentation d'un robot à des fins de maintenance, puis de remettre ce robot dans une position quelconque de l'environnement sans se soucier d'initialiser correctement son estimation de la position.

2.4.2 Cartographie :

Comme nous l'avons expliqué dans les parties précédentes la cartographie est indissociable de la localisation, ce qui implique de disposer d'une méthode de localisation robuste pour espérer avoir une carte correcte. Ceci est le principal problème lors de la phase de cartographie, car une localisation correcte repose en général sur une bonne carte de l'environnement.

La tâche de cartographie est donc intrinsèquement plus complexe que celle de localisation. En effet, la localisation revient à rechercher, parmi les positions possibles représentées dans la carte, celle qui correspond au mieux à la position courante du robot. Cette recherche se déroule donc dans un espace fermé, car on postule que la position recherchée se trouve parmi les positions enregistrées dans la carte.

Dans le cas de la cartographie, une difficulté importante provient du fait que l'estimation de la position du robot se déroule dans un espace ouvert, puisque le robot peut découvrir des zones encore inconnues. De plus cet espace est de faible dimension pour la localisation, tandis que la construction de la carte se déroule dans un espace de beaucoup plus grande dimension. Par exemple, pour une carte de N amers 2D, il y aura 2N paramètres. Or que ce soit pour la cartographie ou la localisation, les données disponibles (odométrie et perceptions) restent les mêmes, et il faudra donc mieux les exploiter pour construire une carte.

L'incomplétude de la carte rend de plus la plupart des méthodes de localisation globale précédemment évoquées difficiles à utiliser car elles supposent une comparaison des probabilités des différentes hypothèses de position. Or avec une carte en cours de construction, le robot peut se situer dans une zone qui n'est pas encore cartographiée et il sera donc impossible d'évaluer la probabilité de cette position.

La plupart des systèmes reposent donc sur une méthode de localisation qui réalise un suivi de position. En effet, si le robot atteint un lieu qui n'est pas représenté dans la carte, il est possible, grâce à ces méthodes locales, de définir sa position par rapport à une position précédente connue au sein de la carte.

2.5 Localisation et cartographie simultanées : SLAM

2.5.1 Définition

Dans le domaine des véhicules intelligents et des robots mobiles l'un des éléments les plus stratégiques est la question de la mobilité ou navigation, qui représente la capacité des robots de s'orienter, de se déplacer et de reconnaître en temps réel leur environnement. Pour que ces robots puissent accomplir leurs tâches efficacement, ils ont besoin d'un minimum d'intelligence artificielle. Cette intelligence leur offre généralement des capacités de cognition, de prise de décision et de manipulation de différents objets.

L'un des challenges majeurs reste la conception d'un système intelligent capable de se déplacer de manière autonome et sûre dans un environnement inconnu, dynamique et large, en se basant seulement sur les données de ses capteurs.
Les deux tâches, la localisation et la cartographie, sont bien connues et il existe des solutions à chacune des deux. Mais lorsque le robot ne dispose ni d'une carte de l'environnement, ni de sa position, il doit effectuer ces deux tâches simultanément. Ce problème est appelé SLAM.

Le SLAM est l'acronyme de Simultaneous Localization and Mapping, ce qui veut dire: Localisation et Cartographie Simultanées. Ce nom désigne l'ensemble des algorithmes offrant à un véhicule mobile robotisé la possibilité de cartographier un terrain inconnu tout en assurant sa propre localisation simultanément.
L'idée principale du SLAM consiste à estimer l'état général du système robotisé. Cet état inclut la position du véhicule ainsi que celles des différents points de l'environnement.

Bien que plusieurs algorithmes aient été développés dans le but de résoudre le problème du SLAM, ils nécessitent généralement de grandes ressources système pour atteindre ce but correctement et efficacement. Pourtant, un système embarqué dispose habituellement de ressources limitées, en matière d'espace de stockage, d'énergie et de puissance de calcul. Ceci impose de fortes contraintes aux algorithmes [8].

2.5.2 Formulation du problème de SLAM :

Le SLAM est composé d'un ensemble de méthodes permettant à un robot de construire une carte d'un environnement et en même temps de se localiser en utilisant cette carte. La trajectoire du véhicule et la position des amers dans la carte sont estimées au fur et à mesure, sans avoir besoin de connaissances a priori. On

cherche à retrouver à chaque instant l'état du robot, noté x, ainsi que la position d'un certain nombre d'amers dans l'environnement. Le robot ne pourra mesurer que des informations relatives par rapport aux amers.

Considérons un robot se déplaçant dans un environnement inconnu, en observant un certain nombre d'amers grâce à un capteur embarqué sur le robot. La figure 2-2 montre une illustration du problème.

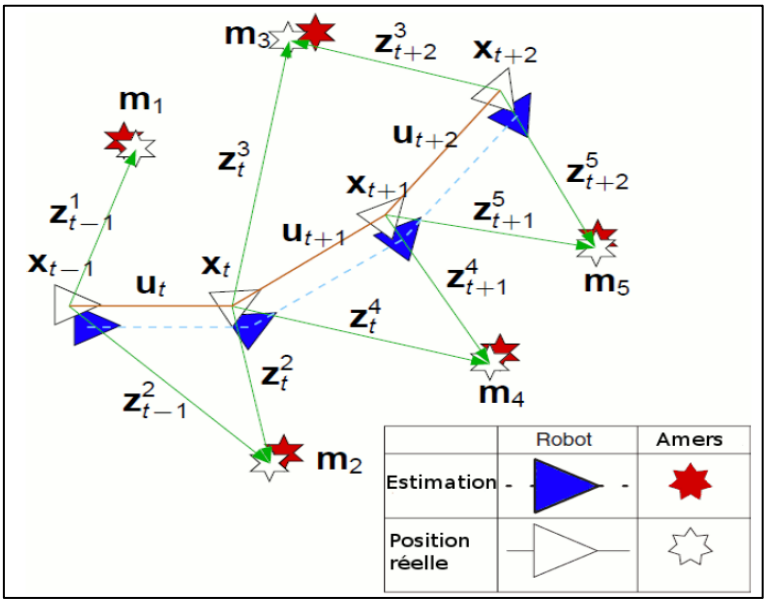

FIGURE 2- 2 : L'idée de base du SLAM

A l'instant $\boldsymbol{t}$ on définit les quantités suivantes :

- $\boldsymbol{x_t}$: le vecteur d'état. Il s'agit en général de la position et de l'orientation, mais cet état peut contenir aussi les vitesses par exemple.
- $\boldsymbol{u_t}$: le vecteur de contrôle. L'application de $\boldsymbol{u_t}$ à l'instant $\boldsymbol{t-1}$ mène le robot de l'état $\boldsymbol{x_{t-1}}$ à l'état $\boldsymbol{x_t}$.
- $\boldsymbol{m_i}$: vecteur contenant la position de l'amer $\boldsymbol{i}$.
- $\boldsymbol{z_{t,(i)}}$: le vecteur d'observation du $i^{ème}$ amer obtenu depuis la position $\boldsymbol{x_t}$ à l'instant $\boldsymbol{t}$.
- $\boldsymbol{z_t}$: l'observation à l'instant $\boldsymbol{t}$.

On définit aussi les ensembles suivants :

- $\boldsymbol{X_{0:t} = \{x_0, x_1, \dots, x_t\} = \{X_{0:t-1}, x_t\}}$: l'ensemble des vecteurs d'état jusqu'à l'instant $\boldsymbol{t}$
- $\boldsymbol{U_{0:t} = \{u_0, u_1, \dots, u_t\} = \{U_{0:t-1}, u_t\}}$: l'ensemble des vecteurs de commande jusqu'à l'instant $\boldsymbol{t}$
- $\boldsymbol{Z_{0:t} = \{z_0, z_1, \dots, z_t\} = \{Z_{0:t-1}, z_t\}}$: l'ensemble des observations jusqu'à l'instant $\boldsymbol{t}$
- $\boldsymbol{m = \{m_0, m_1, \dots, m_n\}}$: la carte de l'environnement contenant une liste d'objets statiques.

Enfin, en effectuant les hypothèses suivantes :

- aucune information a priori n'est disponible quant à la position des amers.
- l'état initial $\boldsymbol{x_0}$ est connu.
- la séquence de contrôles $\boldsymbol{U_{0:t}}$ est connue.
- les mesures $\boldsymbol{Z_{0:t}}$ sont aussi connues.

La localisation en SLAM

Le problème de localisation du robot consiste à estimer sa position dans un environnement donné, en utilisant l'historique de ses observations, l'historique des commandes et la connaissance de l'environnement. La figure 2-3 schématise ce principe.

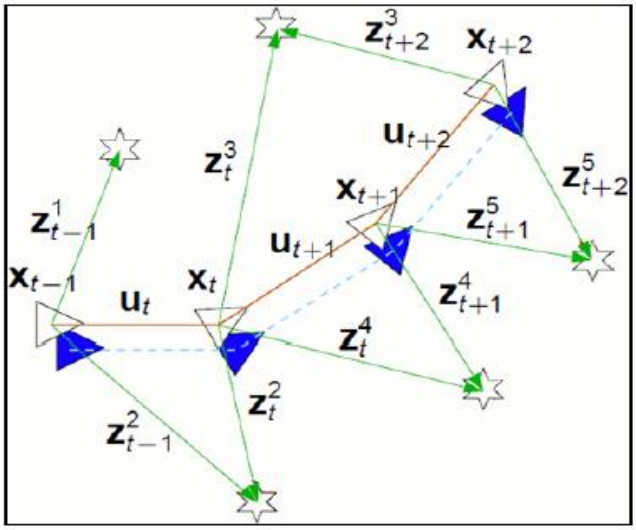

FIGURE 2- 3 : La localisation

On peut analytiquement représenter cette opération par l'estimation de la probabilité de distribution :

$$\boldsymbol{P(x_t|Z_{0:t}, U_{0:t}, m)}$$

L'estimation d'une telle quantité définit la localisation globale, dans la mesure où on utilise toutes les données de l'historique des observations et des commandes pour estimer la position. On obtient ainsi une estimation robuste de la position a postériori, mais on augmente largement la complexité des calculs.

Afin de simplifier l'algorithme, on peut définir une localisation locale, où on utilise uniquement les données de l'instant $\boldsymbol{(t-1)}$ pour estimer la position à l'instant $\boldsymbol{t}$. On représente analytiquement cette opération par l'estimation de la distribution de probabilité :

$$\boldsymbol{P(x_t|z_{t-1}, u_{t-1}, x_{t-1}, m)}$$

En utilisant cette méthode, on simplifie largement la complexité de l'algorithme, mais on risque de dévier de la position correcte du robot, sans pouvoir corriger cela.

La cartographie en SLAM

Le problème de cartographie consiste à déterminer la carte d'un environnement, en utilisant les données des capteurs et l'historique des positions réelles du robot. Sur le schéma de la figure 2-4, le système connait sa position exacte et estime la carte de l'environnement en utilisant les données de ses capteurs.
On peut décrire ce problème mathématiquement par :

$$\boldsymbol{P(m_t|Z_{0:t}, X_{0:t})}$$

Les positions réelles du robot peuvent être obtenues en utilisant par exemple un récepteur GPS. Ces positions doivent être précises et correctes afin d'obtenir une bonne cartographie.

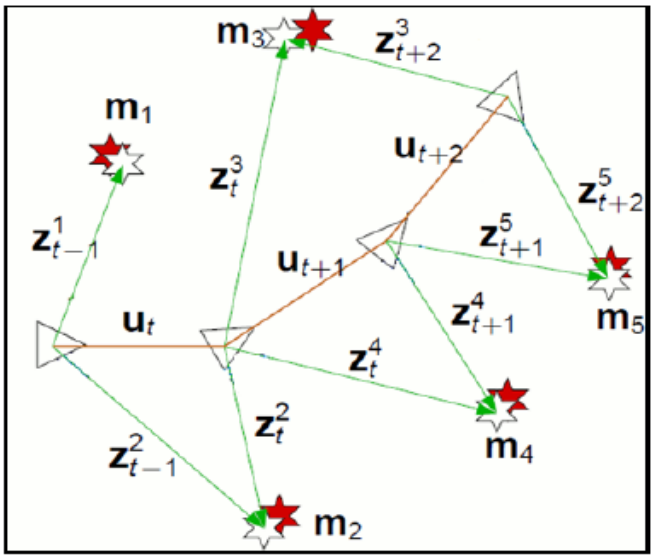

FIGURE 2- 4 : La cartographie

Description mathématique du SLAM

La formulation probabiliste du problème de SLAM nécessite le calcul, à chaque instant *t*, la densité de probabilité de la position et de la carte sachant l'ensemble des mesures, des contrôles et l'état initial. Il s'agit donc de trouver pour tout $\boldsymbol{t}$:

$$\boldsymbol{P(x_t, mt|Z_{0:t}, U_{0:t}, x_0)}$$

L'hypothèse Markovienne nous permet d'obtenir la densité a priori de l'état du robot et de la carte à l'instant $\boldsymbol{t}$ (c.-à-d. la densité de $\boldsymbol{x_t}$ et $\boldsymbol{mt}$ sachant les mesures jusqu'à l'instant $\boldsymbol{t-1}$ l'historique des commandes et l'état initial) :

$$\boldsymbol{P(x_t, m|Z_{0:t-1}, U_{0:t}, x_0) = \int [P(x_t, x_{t-1}, m|Z_{0:t-1}, U_{0:t}, x_0)] dx_{t-1}}$$

$$\boldsymbol{= \int [P(x_t|x_{t-1}, m, Z_{0:t-1}, U_{0:t}, x_0) * P(x_{t-1}, m|Z_{0:t-1}, U_{0:t}, x_0)] dx_{t-1}}$$

$$\boldsymbol{= \int [P(x_t|x_{t-1}, u_t) * P(x_{t-1}, m|Z_{0:t-1}, U_{0:t-1}, x_0)] dx_{t-1}} \quad \boldsymbol{(2.1)}$$

En appliquant **la règle de Bayes** de deux façons différentes, on obtient :

$$\boldsymbol{P(x_t, m, z_t|Z_{0:t-1}, U_{0:t}, x_0) = P(x_t, m|Z_{0:t}, U_{0:t}, x_0) * P(z_t|Z_{0:t-1}, U_{0:t}, x_0)}$$

$$\boldsymbol{P(x_t, m, z_t|Z_{0:t-1}, U_{0:t}, x_0) = \underbrace{P(z_t|x_t, m, Z_{0:t-1}, U_{0:t}, x_0)}_{P(z_t|x_t, m)} * P(x_t, m|Z_{0:t-1}, U_{0:t}, x_0)} \quad \boldsymbol{(2.2)}$$

En combinant les deux équations précédentes, il vient :

$$\boldsymbol{P(x_t, m|Z_{0:t}, U_{0:t}, x_0) = \frac{P(z_t|x_t, m) * P(x_t, m|Z_{0:t-1}, U_{0:t}, x_0)}{P(z_t|Z_{0:t-1}, U_{0:t})}} \quad \boldsymbol{(2.3)}$$

Le dénominateur de la dernière équation ne dépend ni de l'état du robot, ni de la carte. Il peut être vu comme un facteur de normalisation. En combinant les équations (1) et (2), on aboutit finalement à :

$$\boldsymbol{P(x_t, m|Z_{0:t}, U_{0:t}, x_0)}$$

$$\boldsymbol{= \eta * P(z_t|x_t, m) * \int [P(x_t|x_{t-1}, u_t) * P(x_{t-1}, m|Z_{0:t-1}, U_{0:t-1}, x_0)] dx_{t-1}} \quad \boldsymbol{(2.4)}$$

- η : est une constante de normalisation.
- $P(x_t, m|Z_{0:t}, U_{0:t}, x_0)$: Densité de probabilité a posteriori de l'´état du robot et de la carte à l'instant t
- $P(x_{t-1}, m|Z_{0:t-1}, U_{0:t-1}, x_0)$: Densité de probabilité a posteriori de l'´état du robot et de la carte à l'instant $t-1$
- $P(z_t|x_t, m)$: La densité associée aux observations sachant la position du robot et la carte
- $\boldsymbol{P(x_t|x_{t-1}, u_t)}$ **:** La densité associée à la chaîne de Markov décrivant l'évolution de la position du robot en fonction de la commande et de sa dernière position.

Cette structure du SLAM est représentée sur le schéma de la figure 2-5. Sur ce schéma, les cercles gris représentent les données connues, tandis que les cercles blancs désignent les quantités à estimer.

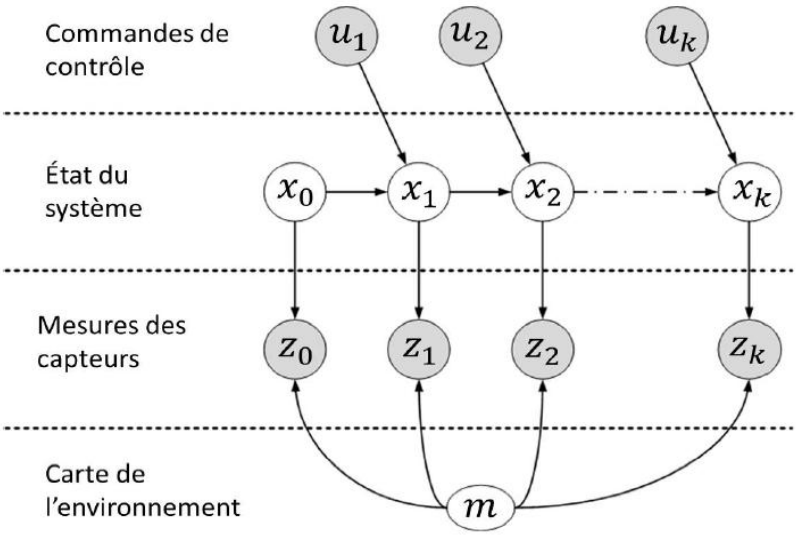

FIGURE 2- 5 : Représentation graphique du problème de SLAM

2.6 Résolution du SLAM

De nombreuses recherches tentent de résoudre le problème de la localisation et la cartographie simultanées, communément appelé SLAM. Il existe trois manières principales de traiter le problème du SLAM dont dérivent beaucoup d'algorithmes.

2.6.1 SLAM basé sur le filtre de Kalman étendu : EKF-SLAM

La première méthode apparue est basée sur le filtrage de Kalman Etendue. Elle utilise un vecteur d'état pour représenter la position du robot et des amers dans la scène.

Le vecteur d'état est défini par le vecteur suivant :

$$\boldsymbol{\mu} = (\boldsymbol{x_R}, \boldsymbol{x_M})^{\boldsymbol{T}} \qquad (\mathbf{2.5})$$

Le vecteur $\boldsymbol{x_R} = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta})^{\boldsymbol{T}}$décrit la position du robot et le vecteur $\boldsymbol{x_M}$ désigne la position des amers dans l'environnement.

La matrice de covariance mesurant l'incertitude du vecteur d'état μ est la suivante :

$$\Sigma = \begin{pmatrix} \sum_{\boldsymbol{R}} & \sum_{\boldsymbol{RM}} \\ \sum_{\boldsymbol{RM}}{}^{\boldsymbol{T}} & \sum_{\boldsymbol{M}} \end{pmatrix} \qquad (\mathbf{2.6})$$

$\sum_{\boldsymbol{R}}$ représente la matrice de covariance associée au vecteur $\boldsymbol{x_R}$ et $\sum_{\boldsymbol{M}}$ la matrice de covariance associée à $\boldsymbol{x_M}$. Les éléments de la matrice $\sum_{\boldsymbol{RM}}$ représentent les corrélations entre les éléments de la carte et les états du robot.

Les transitions d'état sont modélisées par la fonction non-linéaire $\boldsymbol{g}$. Cette équation d'évolution du système SLAM (le robot et la carte) entre deux instants $\boldsymbol{t}$ et $\boldsymbol{t-1}$ s'écrit :

$$\bar{\boldsymbol{\mu}}_{\boldsymbol{t}} = \boldsymbol{g}(\boldsymbol{u}_{\boldsymbol{t}}, \boldsymbol{\mu}_{\boldsymbol{t-1}}) \tag{2.7}$$

Où $\boldsymbol{u}_{\boldsymbol{t}}$ correspond aux commandes appliquées au robot.

La matrice de covariance associée au vecteur d'état $\bar{\boldsymbol{\mu}}_{\boldsymbol{t}}$ s'écrit :

$$\bar{\sum}_{\boldsymbol{t}} = \boldsymbol{G}_{\boldsymbol{t}} \sum_{\boldsymbol{t-1}} \boldsymbol{G}_{\boldsymbol{t}}^{\boldsymbol{T}} + \boldsymbol{R}_{\boldsymbol{t}} \tag{2.8}$$

Où $\boldsymbol{R}_{\boldsymbol{t}}$ est la matrice de covariance associée au bruit du processus et $\boldsymbol{G}_{\boldsymbol{t}}$ est la matrice jacobienne de la fonction g par rapport à l'état $\boldsymbol{\mu}_{\boldsymbol{t-1}}$

$$\boldsymbol{G}_{\boldsymbol{t}} = \begin{pmatrix} \boldsymbol{G}_{\boldsymbol{t}}^{\boldsymbol{x_R}} & \boldsymbol{0}_{\boldsymbol{3\times(n-3)}} \\ \boldsymbol{0}_{\boldsymbol{(n-3)\times 3}} & \boldsymbol{I}_{\boldsymbol{(n\times 3)\times(n-3)}} \end{pmatrix} \tag{2.9}$$

n correspond aux nombres d'états du vecteur x_M et $G_t^{x_R}$ est la matrice jacobienne de la position du robot.

$$\boldsymbol{G}_{\boldsymbol{t}}^{\boldsymbol{x_R}} = \frac{\boldsymbol{\partial g}(\boldsymbol{u}_{\boldsymbol{t}}, \boldsymbol{\mu}_{\boldsymbol{t-1}})}{\boldsymbol{\partial x}_{\boldsymbol{R_{t-1}}}} \tag{2.10}$$

Dans un contexte SLAM, les équations d'observation sont utilisées pour projeter la position estimée des amers dans le repère associé à la position du robot. Le but est de comparer, dans un repère commun, la position des amers de la carte estimée avec les mesures recueillies. La fonction d'observation $\boldsymbol{h}$ s'écrit :

$$\hat{\boldsymbol{z}}_{\boldsymbol{t}} = \boldsymbol{h}(\bar{\boldsymbol{\mu}}_{\boldsymbol{t}}) \tag{2.11}$$

La matrice de covariance associée à cette observation est donnée par :

$$\boldsymbol{S}_{\boldsymbol{t}} = \boldsymbol{H}_{\boldsymbol{t}} \bar{\sum}_{\boldsymbol{t}} \boldsymbol{H}_{\boldsymbol{t}}^{\boldsymbol{T}} + \boldsymbol{Q}_{\boldsymbol{t}} \tag{2.12}$$

Où $\boldsymbol{H}_{\boldsymbol{t}}$ est la matrice jacobienne de la fonction h par rapport à l'état $\bar{\boldsymbol{\mu}}_{\boldsymbol{t}}$ et $\boldsymbol{Q}_{\boldsymbol{t}}$ est la matrice de covariance associée au bruit de mesure.

Le gain du filtre de Kalman K_t se calcule avec l'équation :

$$\boldsymbol{K}_{\boldsymbol{t}} = \bar{\sum}_{\boldsymbol{t}} \boldsymbol{H}_{\boldsymbol{t}}^{\boldsymbol{T}} \boldsymbol{S}_{\boldsymbol{t}}^{\boldsymbol{-1}} \tag{2.13}$$

Finalement, une correction de l'estimation de μ_t et de la covariance $\sum_t$ est réalisée par les équations suivantes:

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{t}} &= \bar{\boldsymbol{\mu}}_{\boldsymbol{t}} + \boldsymbol{K}_{\boldsymbol{t}}(\boldsymbol{z}_{\boldsymbol{t}} - \hat{\boldsymbol{z}}_{\boldsymbol{t}}) \\ \sum_{\boldsymbol{t}} &= (\boldsymbol{I} - \boldsymbol{K}_{\boldsymbol{t}} \boldsymbol{H}_{\boldsymbol{t}}) \bar{\sum}_{\boldsymbol{t}} \end{aligned} \tag{2.14}$$

A chaque nouvel amer observé de nouvelles variables d'état sont ajoutés au vecteur d'état du système ; la taille de la matrice de covariance croit quadratiquement.

Cette approche a été la première développée. Elle est de moins en moins utilisée aujourd'hui notamment du fait des temps de calcul qui la rendent moins intéressante que les autres.

2.6.2 SLAM basé sur le filtre particulaire : FAST-SLAM

Une seconde approche pour traiter le problème du SLAM est basée sur l'utilisation de filtres particulaires. Le principe est de suivre un grand nombre d'hypothèses en parallèle qui sont autant de trajectoires possibles. Ces différentes hypothèses correspondent à un échantillonnage de la distribution de probabilité des trajectoires.

Pour chacune d'elles on construit la carte en fonction des perceptions du robot à l'aide d'un filtre de Kalman. Cependant dans ce cas le traitement est simplifié puisque la trajectoire est connue : les perceptions successives des différents amers ne sont plus corrélées et la matrice de covariance se simplifie puisqu'on ne mémorise plus que les variances individuelles des amers. La complexité des calculs passe ainsi de $0(N^2)$ à $O(N)$.

Le problème principal de cette technique est que la représentation de la carte et de la trajectoire du robot devient vite très lourde. En effet il faut que le nombre de particules soit suffisant pour échantillonner correctement la distribution de probabilité des trajectoires.

Par ailleurs lors de fermeture de boucle, seules les trajectoires correctes sont retenues ce qui entraine un ré-échantillonnage du filtre particulaire, conduisant à une forte perte d'information. Ce problème est d'autant plus important que l'environnement contient plusieurs cycles (plusieurs ré-échantillonnages successifs).

2.6.3 SLAM basé sur les graphes : Graph-SLAM

La troisième méthode est basée sur la théorie des graphes. Elle consiste à considérer les positions successives du robot et des différents amers comme les nœuds d'un graphe. Les arrêtes sont alors constituées des contraintes fournies par l'odométrie ou par l'observation des amers. Pour illustrer ce propos voyons comment procède le robot pour construire la carte.

La méthode Graph-SLAM peut être décomposée en deux étapes :

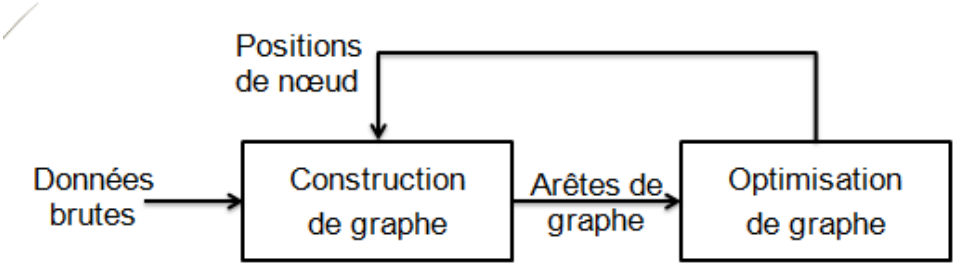

FIGURE 2- 6 : Approche Graph-SLAM

Construction de graphe

A l'instant de départ le robot observe l'amer 1. Le graphe est donc constitué de deux nœuds, la position du robot et celle de l'amer, ainsi que d'une arrête : la contrainte observationnelle entre le robot et l'amer 1.

A l'instant $\boldsymbol{t_2}$ le robot a avancé d'une distance $\boldsymbol{u_2}$ fournie par l'odométrie et observe les amers 1 et 2. Le graphe est maintenant constitué de quatre nœuds: les positions du robot à $\boldsymbol{t_1}$ et $\boldsymbol{t_2}$, reliées par l'estimation de déplacement (odométrie), les positions des amers, reliées aux positions du robot par les contraintes observationnelles.

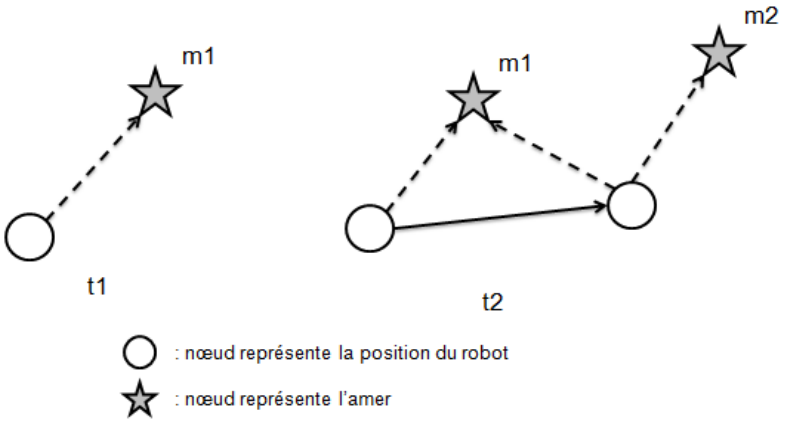

FIGURE 2- 7 : Exemple de création d'un graphe

Optimisation du graphe

Une fois le graphe construit on cherche à l'optimiser en minimisant l'erreur sur les contraintes du graphe. Du fait des incertitudes sur les mesures il existe en effet des erreurs dans l'estimation de la position du robot et des amers. L'entrée pour la procédure d'optimisation est un graphe décrit comme suit :

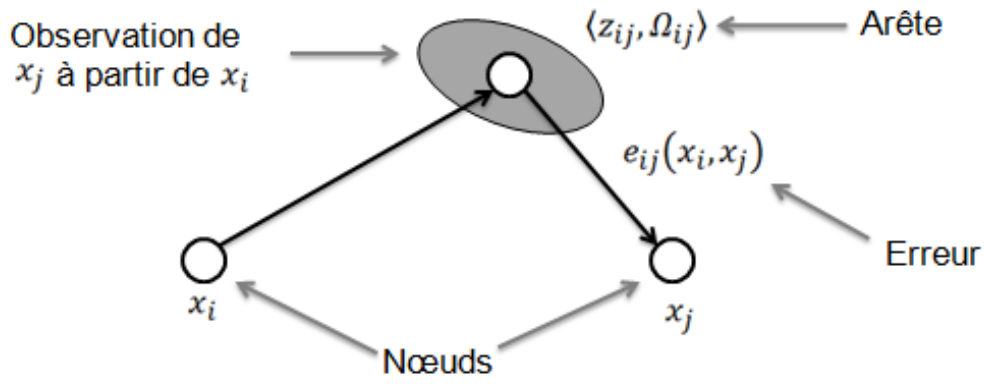

FIGURE 2- 8 : L'erreur sur les contraintes

La méthode Graph-SLAM consiste à trouver la configuration spatiale des nœuds qui minimisent l'erreur introduite par les contraintes :

$$x^* = \underset{x}{\operatorname{argmin}} \sum_{ij} e_{ij}^T \Omega_{ij} e_{ij} \qquad (\mathbf{2.15})$$

Où Ω_{ij} la matrice d'information d'une observation du noeud j vue à partir de nœud i en supposant que le bruit est gaussien et e_{ij} un vecteur d'erreur qui exprime la différence entre une observation et la configuration courante des nœuds.

Le gros avantage du Graph-SLAM est qu'il permet de gérer les cartes composées d'un très grand nombre de nœuds (>10^8 à ce jour) ce qui est impossible avec les autres techniques. Cependant l'optimisation du graphe peut être très lourde.

Conclusion

La robotique est la branche de l'intelligence artificielle concernée par l'étude des systèmes automatiques capables d'interagir directement avec le monde physique. C'est une automatisation de ses machines, o l'objectif est d'augmenter les capacités de localisation et de navigation dans son espace de travail. Ce chapitre nous a permis de présenter un exposé sur le domaine de la robotique mobile, en définissant la navigation autonome des robots mobiles, les différentes méthodes de navigation et les fonctionnalités nécessaires pour exécuter la tâche de navigation telle que la localisation et la planification.

Les problèmes de localisation et de cartographie sont des problèmes majeurs en robotique. On parle de localisation lorsque l'on cherche à retrouver la position du robot au sein d'une carte connue. Lorsqu'on cherche à construire une représentation de l'environnement connaissant la trajectoire du robot, on parle de cartographie. Pris séparément, ces deux problèmes sont aujourd'hui résolus.

Néanmoins, un des objectifs de la robotique mobile est de développer des robots autonomes capables de se déplacer sans connaissance a priori sur l'environnement. Pour cela, les problèmes de localisation et de cartographie doivent être résolus conjointement : on parle alors de SLAM (Simultaneous Localization and Mapping). La résolution complète de ce problème permettra d'assurer une autonomie accrue des robots mobiles.

On a présenté aussi dans ce chapitre le problème de SLAM et les principales méthodes probabilistes utilisées dans la littérature afin de résoudre ce problème à savoir filtrage de Kalman étendu (EKF-SLAM), et filtrage particulaire (FAST-SLAM).

Chapitre 3

Odométrie visuelle

Nous nous intéresserons dans ce chapitre à l'odométrie visuelle qui consiste à estimer le mouvement d'un robot en utilisant seulement l'entrée d'une ou plusieurs caméras. Nous présentons aussi les différentes méthodes de détection des caractéristiques dans l'image.

Introduction

L'odométrie visuelle consiste à estimer le déplacement d'un véhicule équipé d'un capteur de vision, à partir des images successivement recueillies. L'odométrie visuelle permet ainsi d'estimer de façon incrémentale la pose d'un véhicule par l'analyse des images acquises par ses caméras embarquées. Cette dernière décennie, les méthodes d'odométrie visuelle se sont considérablement développées avec un fort intérêt pour les applications temps réel. L'odométrie visuelle est donc un sujet de recherche du moment mais qui prend ses racines dans les années 1980. Actuellement, dans cette branche de la recherche, on trouve tout type de caméra, monoculaire ou stéréo, perspective, omnidirectionnelle, fish eye, etc.

En environnement connu, deux grandes familles de méthodes existent, à savoir la recherche de références géométriques dans les images pour y reconnaître un objet ou une partie d'une scène 3D connue et préalablement modélisée (modèle CAO) ou le travail direct sur l'image. C'est cette dernière approche que nous allons développer car elle ne nécessite pas de connaissances précises de l'environnement dans lequel évolue le robot mobile a priori. Ainsi, l'idée générale va être de détecter les similitudes entre deux images prises successivement par le robot et de déterminer le déplacement de la caméra qui a permis de passer de l'une à l'autre.

Pour estimer le mouvement de la camera entre deux images il faut pouvoir déterminer comment ont bouges les objets observes. Pour cela on ne peut utiliser que des objets visibles dans les deux images et que l'on est capable de retrouver facilement dans chacune d'elles. Pour des raisons de simplicité, on utilise en général des points d'intérêt comme objets de référence.

3.1 L'odométrie

3.1.1 Définition

On appelle « odométrie » l'estimation d'une trajectoire à partir d'un ensemble de relevés de positions relatives. Ainsi, on retrouve, dans le cas de robots mobiles se déplaçant sur des roues, des capteurs sur ces roues, composés d'un capteur infrarouge et d'une roue codeuse, dont les données relevées permettent d'interpoler le déplacement du robot dans un laps de temps très court.

La technique de l'odométrie permet de déterminer la position d'un robot par intégration de ses déplacements élémentaires par rapport à un repère lié à sa configuration initiale. L'algorithme de localisation est basé sur le comptage des impulsions générées par des codeurs durant une période d'échantillonnage connue. Les avantages de l'odométrie résident dans sa simplicité de mise en œuvre et dans son coût faible.

3.1.2 Calcul de déplacement

L'odométrie est certainement la méthode de localisation la plus couramment employée pour les robots disposant d'une structure de locomotion à roues. Son principe consiste à déduire une position, de façon incrémentale, à partir de la vitesse et de la géométrie des roues. La mise en œuvre de cette méthode est des plus simples et ne nécessite qu'une puissance de calcul très limitée [4].

La position odométrique du robot s'obtient par cumul de déplacements élémentaires, calculés périodiquement tous les $\Delta\mathbf{t}$. La fréquence du calcul doit être suffisamment élevée pour que la vitesse des roues puisse être supposée constante entre deux périodes et les déplacements élémentaires peuvent alors être approximés par des arcs de cercles le long desquels le robot parcourt une distance $\boldsymbol{\Delta s_a}$ pour une variation angulaire $\boldsymbol{\Delta\theta}$ comme illustré par la figure 3-1.

$$\begin{aligned} \boldsymbol{\Delta s_a} &= \boldsymbol{\Delta t . R . (\omega_r + \omega_l)/2} \\ \boldsymbol{\Delta\theta} &= \boldsymbol{\Delta t . R . (\omega_r - \omega_l)/e} \end{aligned} \qquad (\mathbf{3.1})$$

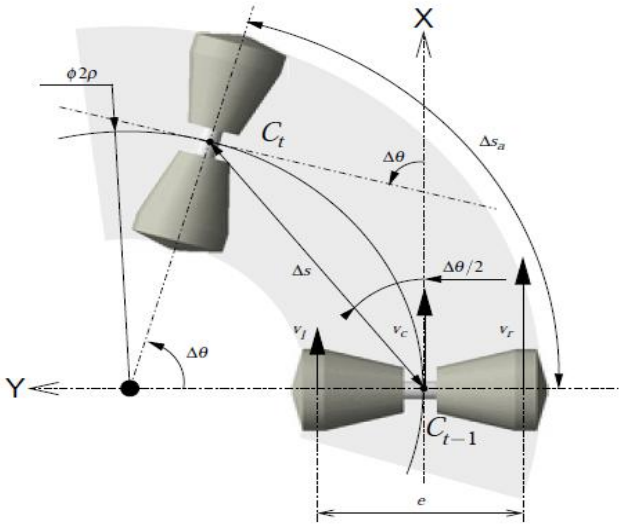

FIGURE 3- 1 : Calcule de la position d'un Robot par l'odométrie

Connaissant les vitesses de chacune des roues, il est possible de calculer périodiquement un déplacement élémentaire (représenté de façon agrandie ici) que l'on peut approximer par un arc de cercle de longueur Δs_a et de variation angulaire $\Delta\theta$. La longueur Δs de la corde de cet arc peut, la plus part du temps, être approximée par la longueur Δs_a de l'arc. Les déplacements élémentaires sont cumulés et la position 2-D peut ainsi être exprimée dans un repère global.

Le mouvement du robot est supposé plan et horizontal et la figure 3-1 permet de remarquer que le déplacement élémentaire $(\Delta x, \Delta y)$ s'exprime en fonction de Δs et $\Delta\theta$ dans le repère lié à la position à l'instant $t-1$:

$$\begin{aligned} \Delta x &= \Delta s.\cos(\Delta\theta/2) \\ \Delta y &= \Delta s.\sin\left(\frac{\Delta\theta}{2}\right) \end{aligned} \qquad (3.2)$$

L'odométrie fournit la mesure de la longueur Δs_a de l'arc de cercle parcouru. Il est donc nécessaire de calculer la translation Δs en fonction de Δs_a:

$$\Delta s = \begin{cases} \Delta s_a & ,\Delta\theta = 0 \\ \frac{2\Delta s_a}{\Delta\theta}\sin\left(\frac{\Delta\theta}{2}\right) & ,\Delta\theta \neq 0 \end{cases} \qquad (3.3)$$

Comme on le voit, ce calcul introduit une singularité pour le cas de la ligne droite ($\Delta\theta = 0$) et il est courant de confondre Δs et Δs_a pour éviter des problèmes d'instabilité numérique lorsque $\Delta\theta$ est petit. Cette approximation est d'autant plus valable que la fréquence d'échantillonnage des vitesses est grande devant la vitesse du robot. La position 2D $(x; y; \theta)$ du robot s'obtient ensuite de façon triviale par cumul du déplacement élémentaire avec la position à l'instant ***t-1***, exprimée dans un repère global :

$$\begin{aligned} x_t &= x_{t-1} + \Delta s.\cos(\theta_{t-1} + \Delta\theta/2) \\ y_t &= y_{t-1} + \Delta s.\sin(\theta_{t-1} + \Delta\theta/2) \\ \theta_t &= \theta_{t-1} + \Delta\theta \end{aligned} \qquad (3.4)$$

3.1.3 Inconvénient

L'odométrie permet d'obtenir une estimation en temps réel de la position du robot. Cependant, dès que la distance parcourue augmente, des erreurs d'origines diverses s'accumulent. Elles résultent de l'imprécision introduite par les codeurs sur la mesure des déplacements angulaires des roues et d'une modélisation approximative de l'évolution du robot mobile.

Cependant, l'odométrie est fragile. Tout d'abord, elle nécessite une connaissance précise de la géométrie du robot : le diamètre des roues doit bien sûr être déterminé, mais d'autres dimensions, comme le point de contact avec le sol, doivent également être prises en compte. Ces paramètres sont généralement difficiles à obtenir de façon précise et, en terrains naturels, dépendent fortement de la nature du sol.

De plus, les erreurs de mesure, qui se produisent lors de déplacements non mesurés par exemple le glissement d'une roue sur le sol, sont dramatiques pour l'estimation de la position. Cette position obtenue est d'autant plus inexacte que ces glissements sont importants. Selon le type de terrain ils peuvent rendre quasiment inutile l'odométrie en tant que méthode de localisation.

3.2 Points d'intérêt visuels (FAST, BRIEF, ORB)

Un point d'intérêt est un point qui caractérise de façon unique une partie de l'image. Plus précisément il s'agit d'un point qui présente de fortes discontinuités d'intensité lumineuse. Pour être utile, ce point doit être très bien identifié à l'aide d'un descripteur unique et il doit être possible de le retrouver facilement. Il existe une multitude de méthodes de détection/description de points d'intérêt parmi lesquelles SIFT, SURF et ORB. Il s'agit de méthodes particulièrement appréciées pour leur robustesse et leur efficacité.

Le principe de détection correspond à rechercher des zones d'intensité lumineuse particulière et offrant une grande stabilité. Pour cela on utilise le plus souvent des méthodes basées sur le calcul du gradient de luminosité. Les extrema correspondent aux points ou s'annule le gradient.

Une fois les points identifiés, il faut décrire l'information locale dans l'image de façon unique et aussi invariante que possible de l'échelle d'observation, des rotations, des variations de luminosité etc. Pour cela on calcule un descripteur pour chaque point. Le descripteur est simplement une suite de valeurs (64 ou 128 le plus souvent) qui décrivent la région autour de chaque point caractéristique détectée.

3.2.1 FAST (Features from Accelerated Segment)

Features from Accelerated Segment Test (FAST), que l'on peut traduire par « caractéristique issues de tests accélérés de segments », est un algorithme de détection

de caractéristique, proposé à l'origine par Rosten et Drummond [9] pour identifier les points d'intérêt dans une image.. Il est utilisé dans le domaine de vision par ordinateur, pour des tâches de détection d'objet ou de reconstruction 3D.

L'algorithme fonctionne en deux étapes : dans la première étape, un test de segment basé sur les luminosités relatives est appliqué à chaque pixel de l'image traité, la deuxième étape permet d'affiner et de limiter les résultats par la méthode suppression non-maximum.

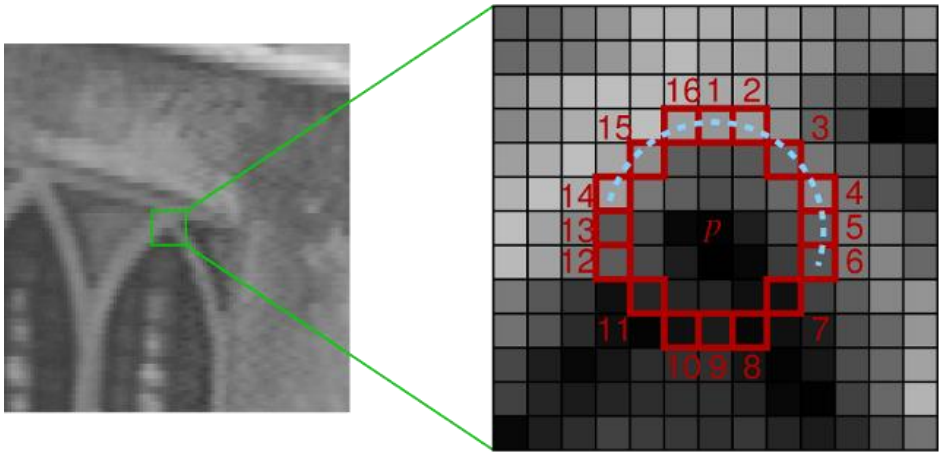

FIGURE 3- 2 : Modèle de détection de point FAST

Les différentes étapes de la détection FAST

1. Sélectionner un pixel p dans l'image qui doit être identifié comme un point ou non d'intérêt. Que son intensité soit $\boldsymbol{I_p}$.
2. Sélectionner une valeur de seuil approprié $\boldsymbol{t}$.
3. considérons un cercle de 16 pixels autour du pixel en cours de test.
4. Maintenant le pixel p est un coin s'il existe un ensemble de n pixels contigus dans le cercle (de 16 pixels) qui sont tous plus lumineux que $\boldsymbol{I_p + t}$, ou tout plus sombre que $\boldsymbol{I_p - t}$.
5. Pour rendre l'algorithme rapide, on compare d'abord l'intensité des pixels 1, 5, 9 et 13 du cercle avec $\boldsymbol{I_p}$. Comme la montre la figure 3-2, au moins trois de ces quatre pixels devraient satisfaire le critère de seuil pour que le pixel soit un point d'intérêt.
6. Si p est un coin, alors au moins trois d'entre eux doivent tous être plus lumineux que $\boldsymbol{I_p + t}$ ou plus foncée que $\boldsymbol{I_p - t}$. Si ce n'est pas le cas alors $\boldsymbol{p}$ ne peut pas être un coin.
7. Répéter la procédure pour tous les pixels de l'image.

Il y a quelques limitations à l'algorithme. Tout d'abord, pour $\boldsymbol{n} < 12$, l'algorithme ne fonctionne pas très bien dans tous les cas parce que lorsque $\boldsymbol{n} < 12$ le nombre de points d'intérêt détectées sont très élevés. En second lieu, l'ordre dans lequel les 16 pixels sont testés détermine la vitesse de l'algorithme.

Une approche d'apprentissage automatique à été ajouté à l'algorithme pour traiter ces problèmes.

La technique de l'apprentissage automatique

1. Sélectionner un ensemble d'images pour l'apprentissage.
2. Pour chaque image, exécuter l'algorithme FAST afin de détecter les points d'intérêt en prenant un pixel à la fois et évaluer tous les 16 pixels dans le cercle.
3. Pour chaque pixel $\boldsymbol{p}$, stocker les 16 pixels qui l'entourent dans un vecteur.

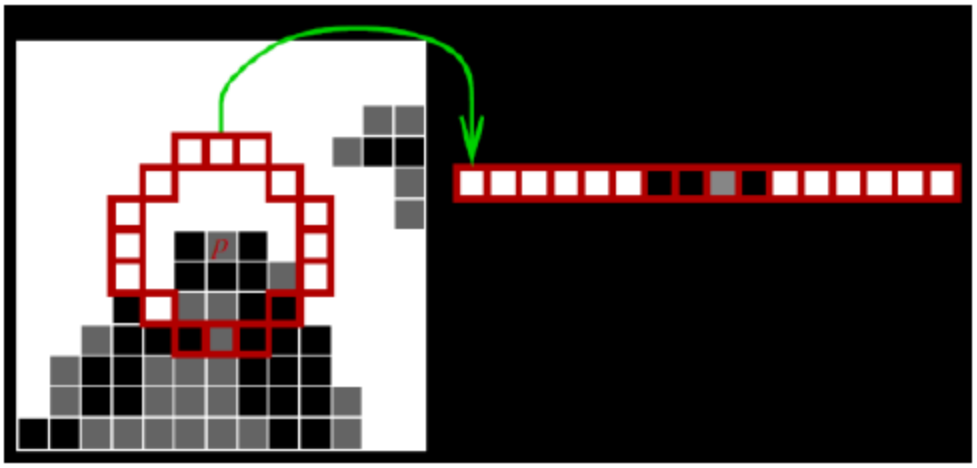

FIGURE 3- 3 : Les 16 valeurs qui entourent le pixel p stockés dans un vecteur

4. Répéter cette opération pour tous les pixels de toutes les images. Ceci est le vecteur $\boldsymbol{P}$ qui contient toutes les données de l'apprentissage.
5. Chaque valeur (un des 16 pixels, disons x) dans le vecteur, peut prendre trois états. Plus sombre que $\boldsymbol{p}$, plus claire que p ou similaire à $\boldsymbol{p}$.
   Mathématiquement :

$$\boldsymbol{S_{p\to x}} = \begin{cases} d, & \boldsymbol{I_{p\to x} \le I_p - t} \quad (\textbf{Sombre}) \\ s, & \boldsymbol{I_p + t < I_{p\to x} < I_p + t} \quad (\textbf{Similaire}) \\ b, & \boldsymbol{I_p + t \le I_{p\to x}} \quad (\textbf{Brillant}) \end{cases} \qquad (\mathbf{3.5})$$

$\boldsymbol{S_{p\to x}}$ est l'état de x, $\boldsymbol{I_{p\to x}}$ est l'intensité du pixel x,t est la seuil.

6. Selon les états, la totalité du vecteur $\boldsymbol{P}$ sera subdivisé en trois sous-ensembles$\boldsymbol{P_d}$, $\boldsymbol{P_s}$ et $\boldsymbol{P_b}$.
7. Définir une variable $\boldsymbol{K_p}$ qui est vrai si $\boldsymbol{p}$ est un point d'intérêt et fausse si $\boldsymbol{p}$ n'est pas un point d'intérêt.
8. Utiliser l'algorithme ID3 (classificateur – arbre de décision) pour interroger chaque sous-ensemble en utilisant la variable $\boldsymbol{K_p}$, pour la connaissance de la classe correcte. Il sélectionne le $\boldsymbol{x}$ qui donne le plus d'informations pour savoir si le pixel candidat est un coin. mesurée par l'entropie de $\boldsymbol{K_p}$.

L'entropie pour l'ensemble $\boldsymbol{P}$ peut être mathématiquement représentée comme :

$$\begin{aligned} & \boldsymbol{H(p) = \ (c + \bar{c}) log_2(c + \bar{c}) - c\, log_2\, c - \bar{c}\, log_2\, \bar{c}} \\ o\grave{u} \quad & \boldsymbol{c = \ \left|\{ p \mid K_p\ is\ true\}\right|} \quad \boldsymbol{(nombre\ de\ coins)} \\ et \quad & \boldsymbol{\bar{c} = \ \left|\{ p \mid K_p\ is\ false\}\right|} \quad \boldsymbol{(nombre\ de\ non\ coins)} \end{aligned} \tag{3.6}$$

9. Ceci est appliqué de façon récursive à tous les sous-ensembles jusqu'à ce que son entropie soit nulle.
10. L'arbre de décision ainsi créée est utilisé pour la détection pour d'autres images.

Suppression non-maximale

La détection de plusieurs points d'intérêt adjacents les uns aux autres est un des autres problèmes de la version initiale de l'algorithme. Cela peut être traité en appliquant la suppression non maximale après détection des points d'intérêt.
L'algorithme est décrit ci-dessous :

1. Calculer une fonction de score,V pour tous les points caractéristiques détectés.V est la somme de différence absolue entre p et les 16 pixels qui l'entourent.
2. Considérons deux points clés adjacentes et calculons leur score V.
3. Exclure celui qui a le score V le plus faible.

L'ensemble de processus peut être résumé mathématiquement comme suit :

$$\boldsymbol{V = max} \begin{cases} \sum (\boldsymbol{valeurs\ de\ pixel - p}) \quad \boldsymbol{si} \quad (\boldsymbol{valeur - p}) > t \\ \sum (\boldsymbol{p - valeurs\ de\ pixel}) \quad \boldsymbol{si} \quad (\boldsymbol{p - valeur}) > t \end{cases} \tag{3.7}$$

$\boldsymbol{p}$ est le pixel central, $\boldsymbol{t}$ est le seuil pour la détection et les valeurs de pixels correspondent aux $\boldsymbol{N}$ pixel contigus dans le cercle.

3.2.2 BRIEF (Binary Robust Independent Elementary Features)

Le descripteur SIFT travaille avec des vecteurs de 128 dimensions. Puisqu'il utilise des nombres flottants, il prend essentiellement 512 octets. De la même façon SURF prend également au minimum 256 octets (pour 64 dimensions).

La création d'un tel vecteur de milliers de points prend beaucoup de mémoire. Cela peut être un inconvénient pour les applications en temps réel telles que SLAM qui gardent la trace de nombreux points, ainsi que pour les algorithmes qui nécessitent le stockage d'un très grand nombre de descripteurs, par exemple pour la reconstruction 3D à grande échelle.

BRIEF est une méthode récente qui utilise un test binaire simple pour extraire le descripteur autour de point caractéristique [10]. Le descripteur obtenu par BRIEF est très simple et son espace de stockage est plus petit par rapport aux SIFT et SURF.

Le descripteur BRIEF est une description d'une chaîne binaire d'un patch de l'image construit à partir d'un ensemble de tests binaires d'intensité.
Considérons un patch de l'image lissée, p. Un τ de test binaire est définie par :

$$\boldsymbol{\tau(p;x,y) \coloneqq \begin{cases} 1 & si\ I(p,x) < I(p,y) \\ 0 & \text{sinon} \end{cases}} \quad (3.8)$$

Où $\boldsymbol{I(p,x)}$ est l'intensité de pixel au point $\boldsymbol{x = (u,v)^T}$ qui est dans le patch de l'image P après le processus de filtrage. Un ensemble de points peut uniquement identifier un descripteur binaire τ. On définit le descripteur BRIEF comme une chaîne binaire de dimension $\boldsymbol{n_d}$.

$$\boldsymbol{f_{n_d}(p) \coloneqq \sum_{1 \le i \le n_d} 2^{i-1}\, \tau(p; x_i, y_i)} \quad (3.9)$$

$\boldsymbol{n_d}$ peut être égale à 128, 256 et ainsi de suite,

Les tests binaire prennent seulement les informations des pixels uniques en considération et sont donc très sensibles au bruit. Par un pré-lissage de patch, cette sensibilité peut être réduite, en augmentant ainsi la stabilité des descripteurs [11].

Un point important est que BRIEF est un descripteur, il ne fournit aucune méthode pour trouver les caractéristiques. L'application de BRIEF nécessite d'utiliser n'importe quels autres détecteurs de caractéristique comme SIFT, SURF etc.

3.2.3 ORB (Oriented FAST and Rotated BRIEF)

L'algorithme ORB, qui est basé sur les algorithmes FAST et BRIEF, est une méthode pour décrire des points caractéristiques à l'aide de la chaîne binaire. Puisque le point caractéristique d'ORB est détecté par la détection de caractéristique améliorée FAST et qui est décrit à l'aide d'un descripteur de caractéristique BRIEF amélioré, donc les plus grandes caractéristiques de cet algorithme est la rapidité et l'invariance de rotation et la réduction de la sensibilité au bruit [12].

L'algorithme commence par la détection des points caractéristiques dans l'image par la méthode FAST qui prend un paramètre, le seuil d'intensité entre le pixel central et ceux dans un anneau circulaire autour du centre. Puis il applique l'algorithme de Harris pour détecter les points intéressants dans les points caractéristiques qui ont été détectés. La méthode de Harris donne les meilleurs N points qui sont les plus convenables.

FAST ne produit pas les caractéristiques multi échelle, pour cette raison, ORB utilise la pyramide d'échelle de l'image pour résoudre ce problème. Pour obtenir les

points caractéristiques, le test FAST doit être effectué à chaque niveau de la pyramide.

Plusieurs détecteurs des points clé incluent un opérateur d'orientation (SIFT et SURF sont deux exemples), mais FAST n'a pas cet opérateur. L'algorithme ORB peut résoudre le problème de l'orientation par la méthode de barycentre d'intensité. Le barycentre d'intensité suppose que l'intensité d'un coin est décalée par rapport à son centre, et ce vecteur peut être utilisé pour attribuer une orientation. Tout d'abord définir les moments du patch (voisinage circulaire) est :

$$\boldsymbol{m_{pq} = \sum_{x,y} x^p y^q I(x,y)} \qquad (\mathbf{3.10})$$

x, y est la position du point caractéristique FAST, dans un voisinage circulaire de rayon r, $x, y \in [-r, r]$, Ensuite, on calcule le centre de gravité du patch :

$$\boldsymbol{C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right)} \qquad (\mathbf{3.11})$$

L'angle formé par le point caractéristique et le centre de gravité est défini comme la direction du point caractéristique FAST :

$$\boldsymbol{\theta = arctan\left(\frac{m_{01}}{m_{10}}\right) = arctan\left(\frac{\sum_{x,y} yI(x,y)}{\sum_{x,y} xI(x,y)}\right)} \qquad (\mathbf{3.12})$$

Finalement, l'algorithme ORB extrait les descripteurs BRIEF selon la direction fournie par cette équation.

3.3 Formulation du problème de l'odométrie visuelle

Un robot se déplace dans un environnement et prend des images avec un système de vision aux instants discrets k. Dans le cas d'un système monoculaire, l'ensemble des images prises à des instants k est notée par : $\boldsymbol{I_{0:n} = \{I_0, \dots\dots, I_n\}}$.

Dans le cas d'un système stéréo, il y a une image gauche et une image droite à chaque instant de temps, noté : $\boldsymbol{I_{l,0:n} = \{I_{l,0}, \dots\dots, I_{l,n}\}}$ et $\boldsymbol{I_{r,0:n} = \{I_{r,0}, \dots\dots, I_{r,n}\}}$. La figure 3-4 montre une illustration de ce paramètre.

Pour plus de simplicité, on suppose que la coordonnée de caméra coïncide à la coordonnée du robot. En cas d'un système stéréo, sans perte de généralité, le système de coordonnée de caméra gauche peut être utilisé comme l'origine [13]. Deux positions de caméra aux instants adjacents $\boldsymbol{k-1}$ et $\boldsymbol{k}$ sont liées par la transformation du corps rigide $\boldsymbol{T_{k,k-1} \in R^{4\times4}}$ de la forme suivant :

$$\boldsymbol{T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}} \qquad (\mathbf{3.13})$$

Où $\boldsymbol{R_{k,k-1}} \in \boldsymbol{SO(3)}$ est la matrice de rotation, et $\boldsymbol{t_{k,k-1}} \in \boldsymbol{R^{3\times1}}$ est la vecteur de translation.

L'ensemble $\boldsymbol{T_{1:n}} = \{\boldsymbol{T_{1,0}}, \dots\dots, \boldsymbol{T_{n,n-1}}\}$ contient tous les mouvements ultérieurs. Pour simplifier la notation, $\boldsymbol{T_k}$ sera utilisé au lieu de $\boldsymbol{T_{k,k-1}}$. Finalement, l'ensemble de positions de caméra $\boldsymbol{C_{0:n}} = \{\boldsymbol{C_0}, \dots\dots, \boldsymbol{C_n}\}$ contient les transformations de la caméra.
La position actuelle $\boldsymbol{C_n}$ peut être calculée par la concaténation de toutes les transformations $\boldsymbol{T_k(k = 1, \dots\dots, n)}$, et par conséquent :

$$\boldsymbol{C_n = C_{n-1}T_n} \qquad (\boldsymbol{3.14})$$

Avec $\boldsymbol{C_0}$ est la position de caméra à l'instant $\boldsymbol{k = 0}$, qui peut être fixé arbitrairement par l'utilisateur.

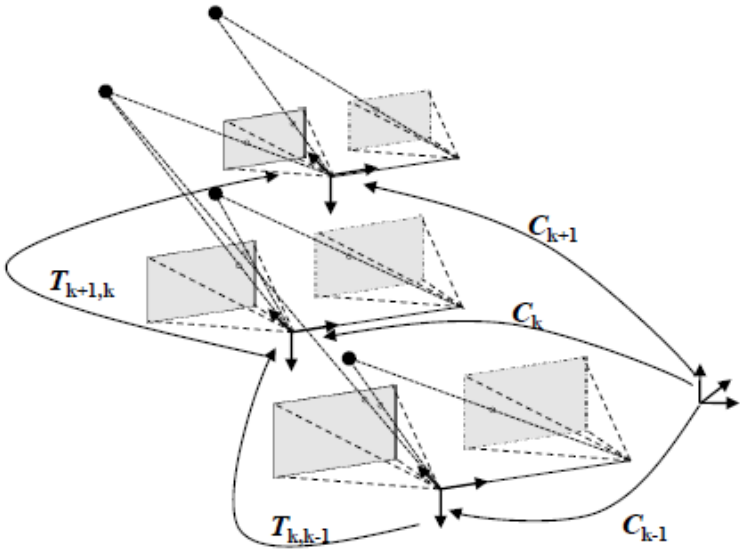

FIGURE 3- 4 : Une illustration du problème d'odométrie visuelle

La tâche principale en odométrie visuelle est de calculer les transformations relatives $\boldsymbol{T_k}$ à partir des images $\boldsymbol{I_k}$ et $\boldsymbol{I_{k-1}}$, puis concaténer les transformations pour récupérer la trajectoire complète $\boldsymbol{C_{0:n}}$ de la caméra. Cela signifie que l'odométrie visuelle récupère le chemin progressivement, position après position.

En odométrie visuelle monoculaire, il existe deux approches principales pour calculer le mouvement relatif $\boldsymbol{T_k}$: les méthodes basées sur l'apparence (ou global), qui utilisent les informations de l'intensité de tous les pixels dans les deux images d'entrée, et des méthodes basée sur les caractéristiques de l'image, qui utilisent seulement des caractéristiques répétitivement extraites à travers les images.

Les méthodes globales sont moins précises que les méthodes basées sur les caractéristiques de l'image et plus coûteuses. La plupart des méthodes basées sur l'apparence ont été appliqués à l'odométrie visuelle monoculaire. Ceci est dû à la facilité de mise en œuvre par rapport à la caméra stéréo. Les méthodes basée sur les caractéristiques de l'image exigent la capacité de suivre les caractéristiques à travers des images d'une manière robustes, mais sont plus rapides et plus précises que les

méthodes globales. Par conséquent, la plupart des implémentations de visuelle odométrie utilisent les méthodes basée sur les caractéristiques de l'image.

Les tâches principales en odométrie visuelle sont résumées dans la figure 3-5. Pour chaque nouvelle image $\boldsymbol{I_k}$ (ou du couple d'images dans le cas d'une caméra stéréo), les deux premières étapes consistent à détecter et trouver la correspondance des caractéristiques 2D avec celles des images précédentes.
La troisième étape consiste à calculer le mouvement relatif $\boldsymbol{T_k}$ entre les instants $\boldsymbol{k-1}$ et $\boldsymbol{k}$.

Selon si les correspondances sont spécifiées dans trois ou deux dimensions, il y a trois approches distinctes pour résoudre ce problème. En fin, La position de caméra $\boldsymbol{C_k}$ est calculée par concaténation $\boldsymbol{T_k}$ des avec la position précédente.

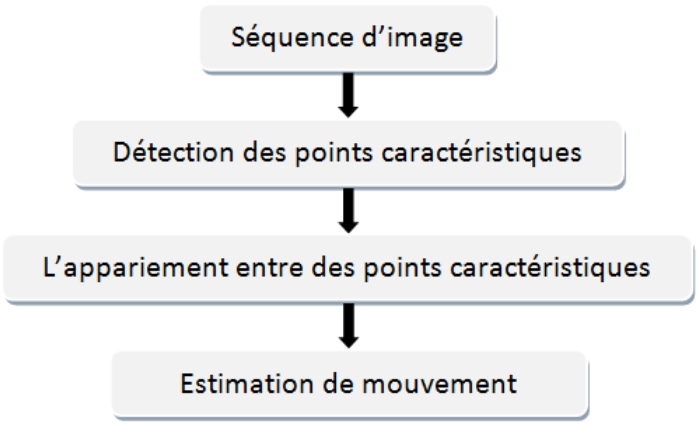

FIGURE 3- 5 : les principaux composants d'un système de visuelle odométrie.

3.3.1 L'appariement entre des points caractéristiques

Il existe deux approches principales pour trouver des points caractéristiques et leurs correspondances. La première est de trouver des points d'intérêt dans une image et de les suivre dans les autres images en utilisant des techniques de recherche locale, telles que la corrélation.

La seconde consiste à détecter d'une manière indépendante des caractéristiques dans toutes les images et les comparer, en se basant sur leurs descripteurs. La manière la plus simple pour trouver les points correspondants entre deux images consiste à comparer tous les descripteurs des points d'intérêt de la première image à tous les descripteurs des points d'intérêt de la deuxième image.

3.3.2 Estimations de mouvement

L'estimation de mouvement est l'étape de calcul principal effectué pour chaque image dans un système VO. Plus précisément, dans cette étape, le mouvement de caméra entre l'image actuelle et l'image précédente est calculé. Par la concaténation de tous ces mouvements simples, la trajectoire complète de la caméra

et de robot (en supposant que la caméra est montée de façon rigide) peut être récupérée.

La transformation entre deux images $\boldsymbol{I_{k-1}}$ et $\boldsymbol{I_k}$ peut être calculée à partir de deux ensembles de correspondances $\boldsymbol{f_{k-1}}$, $\boldsymbol{f_k}$ à des instants $\boldsymbol{k-1}$ et $\boldsymbol{k}$, respectivement. Il existe trois méthodes différentes selon les correspondances sont spécifiées en deux ou trois dimensions. Dans le cas monoculaire, la structure 3D doit être triangulée à partir de deux vues de caméra adjacentes (p. ex. $\boldsymbol{I_{k-2}}$ et $\boldsymbol{I_{k-1}}$) et ensuite mis en correspondance avec les caractéristiques des images 2D en un troisième point de vue (p. ex. $\boldsymbol{I_k}$), dans le schéma monoculaire, la mise en correspondance nécessite au moins trois points de vue différente. Les caractéristiques peuvent être des points, des régions ou des contours.. Les points sont les plus exploités, car les algorithmes géométriques tels que le calcul de position ou la géométrie épipolaire travaillent le plus souvent sur des points.

L'algorithme3D à 3D

Dans cette méthode, les deux correspondances $\boldsymbol{f_{k-1}}$ et $\boldsymbol{f_k}$ sont spécifiés en 3D. Pour appliquer cette méthode, il est nécessaire de trianguler les points 3D à chaque instant, par exemple, en utilisant un système de caméras stéréo.

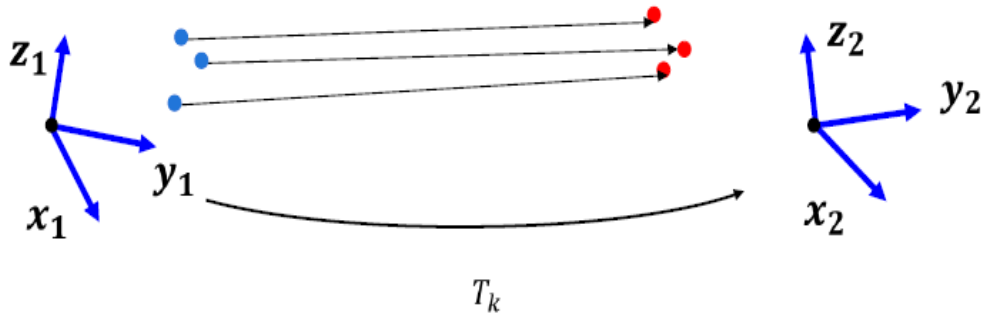

FIGURE 3- 6 : Illustration de la méthode 3D à 3D

L'algorithme 3D à 3D est représenté la table suivant :

Algorithme 3D à 3D
1) Capturer une paire d'images stéréo $I_{l,k}$ et $I_{r,k}$ 2) Détection des point d'intérêt dans les deux images $I_{l,k-1}$ et $I_{l,k}$ 3) Mise en correspondance entre les images $I_{l,k-1}$ et $I_{l,k}$ 4) Triangulation des points correspondants pour chaque paire $I_{l,k-1}$, $I_{r,k-1}$ et $I_{l,k}$, $I_{r,k}$ 5) Calcul de T_k à partir des points 3D : X_{k-1} et X_k 6) La concaténation de la transformation T_k , en calculant $C_k = C_{k-1}\ T_k$ 7) Répéter à partir de 1)

Table 3- 1 : Algorithme 3D à 3D

L'algorithme 3D à 2D

Dans cette méthode, les points caractéristiques $\boldsymbol{f_{k-1}}$ sont spécifiés en 3D et les points $\boldsymbol{f_k}$ sont leurs ré-projections 2D correspondantes dans l'image $\boldsymbol{I_k}$

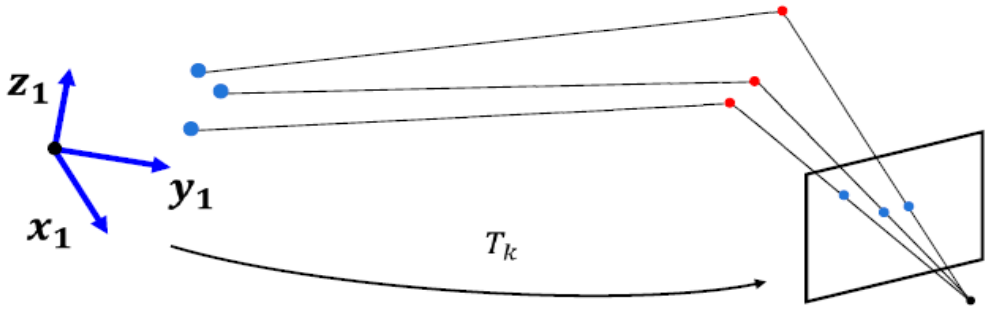

FIGURE 3- 7 : Illustration de la méthode 3D à 2D

L'algorithme 3D à 2D est représenté le tableau suivant :

Algorithme 3D à 2D
1) Faire une seule fois : 1.1) Capturer deux images I_{k-2} et I_{k-1} 1.2) Détection des point d'intérêt dans les deux images I_{k-2} et I_{k-1} 1.3) Mise en correspondance entre I_{k-2} et I_{k-1} 1.4) Triangulation des points correspondants de I_{k-2} , I_{k-1} 2) Faire à chaque itération : 2.1) Capturer une nouvelle image I_k 2.2) Détection des points d'intérêt dans l'image I_k 2.3) Mise en correspondance entre I_{k-1} et I_k 2.4) Estimation de la position de caméra à partir des points correspondants 3D à 2D, en utilisant l'approche PnP (Perspective n Points) 2.5) Triangulation des nouveaux points correspondants de I_{k-1} et I_k 2.6) La concaténation de la transformation T_k par le calcul de $C_k = C_{k-1} T_k$ 2.7) Répéter à partir de 2.1)

Table 3- 2 : Algorithme 3D à 2D

L'algorithme 2D à 2D

Dans cette méthode, les deux paires correspondants $\boldsymbol{f_{k-1}}$ et $\boldsymbol{f_k}$ sont spécifiés en 2D.

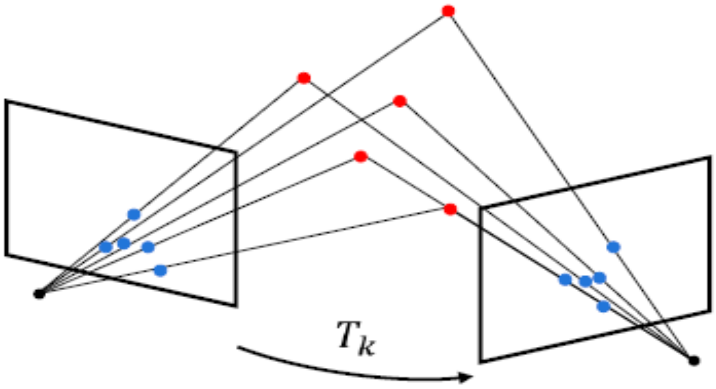

FIGURE 3- 8 : Illustration de la méthode 2D à 2D

L'algorithme 2D à 2D est résumé comme suite :

Algorithme 2D à 2D
1) Capturer une paire d'images stéréo I_k 2) Détection des point d'intérêt dans les deux images I_{k-1} et I_k 3) Mise en correspondance entre les images I_{k-1} et I_k 4) Estimation de la matrice essentielle E 5) extraction de t_k et R_k par la décomposition de la matrice essentielle E 6) Former T_k à partir de t_k et R_k 7) La concaténation de la transformation T_k , en calculant $C_k = C_{k-1} T_k$ 8) Répéter à partir de 1)

Table 3- 3 : Algorithme 2D à 2D

Conclusion

Le principe de la localisation par vision monoculaire d'un véhicule est identique pour la grande majorité des méthodes. La première position de la caméra est inconnue et est donc fixée arbitrairement. Le déplacement en 3 dimensions de la caméra dans l'environnement qui l'entoure va se traduire visuellement dans l'image par un déplacement en 2 dimensions des éléments d'intérêt de l'image. Ces éléments d'intérêt sont généralement des zones ou plus couramment des points de l'image qu'il est possible de suivre au fil du flux vidéo, c'est-à-dire entre les différentes images.

Ce chapitre représente les différentes techniques pour détecter ces points d'intérêt avec leurs descripteurs dans l'image, à savoir, FAST, BRIEF et ORB.
A partir de l'observation du déplacement de ces éléments d'intérêt, il est possible d'inférer le mouvement de la caméra. Ceci peut être réalisé directement à partir de l'information 2D : on parlera alors généralement d'odométrie visuelle. On a vu également dans ce chapitre l'intérêt de l'odométrie visuelle, en expliquant les étapes nécessaires pour estimer le mouvement de la caméra entre deux points de vue.

Chapitre 4

RatSLAM

Ce chapitre fera l'objet d'une étude soigneuse de la navigation chez les rongeurs. Nous décrivons aussi le modèle RatSLAM qui est un modèle informatique bio-inspiré de l'hippocampe du rat et son architecture. Et finalement, nous citons les différentes unités de traitement qu'il se compose.

Introduction

Pour qu'un robot navigue intelligemment dans un environnement à grande échelle, il doit posséder un moyen pour stocker des informations sur l'expérience passée, et d'avoir la capacité d'utiliser ces informations pour prendre des décisions sur le comportement approprié.

Au cours de la dernière décennie, les chercheurs ont trouvé une méthodologie pour résoudre ce problème basée sur la construction d'une carte interne de l'environnement, puis de se déplacer en utilisant cette carte pour l'estimation de la position du robot. Cette méthodologie est connue sous le nom de localisation et de cartographie simultanée SLAM.

Plusieurs approches sont publiées pour résoudre le problème SLAM et chacune de ces approches a des avantages et faiblesses, et la recherche d'une bonne solution fait l'objet de nombreuses recherches actuelles. Les algorithmes SLAM réguliers qui sont décrits dans le chapitre 2 souffrent de plusieurs problèmes. En outre, la restriction de l'exécution sur des environnements statiques, et la complexité des calculs sont des facteurs importants.

Toutefois, dans la vie quotidienne, nous pouvons observer plusieurs systèmes exécutant le SLAM de manière assez précise sans calcul intensif. Ces systèmes sont capables de résoudre les objectifs de navigation dans une approche facile et robuste. Les systèmes apparemment puissants sont les humains, les oiseaux, les poissons, les rongeurs, etc.

Le RatSLAM est un système robotisé de SLAM visuel inspiré des modèles de calcul dans l'hippocampe des rongeurs. Il a été déployé sur une gamme de plates-formes de robot et véhicule dans différents environnements, mais toujours sur des robots terrestres. Toutefois, les recherches sur l'hippocampe restent un sujet qui

intrigue beaucoup de chercheurs. Il est généralement admis que les rongeurs ont des zones responsables de la localisation qui sont activées en fonction de la position du rongeur dans l'espace. Les champs de lieu (place fields) sont modulés par l'activité du rongeur comme quand il se déplace ainsi que par des stimuli visuels. Cette relation entre les stimuli accédant au cortex visuel et les cellules de place et de grille est exploitée pour faire du SLAM. Ce processus est celui de l'estimation de la position du robot et de la carte visuel calculée dans le SLAM bayésien [14].

4.1 Fondements biologiques

Les rongeurs sont parmi les animaux les plus étudiés quand il s'agit des fondements neuronaux de la navigation et la cartographie. Il y a une base bien établie de la recherche sur leurs capacités dans un large champ d'environnements expérimentaux. En outre, l'hippocampe de rongeur est l'une des régions du cerveau les plus étudiées chez les mammifères (Figure 4-1) depuis les travaux d'O'keef [15].Cela a conduit à un certain nombre de modèles neuronaux bien établis de cartographie et de navigation chez les rongeurs

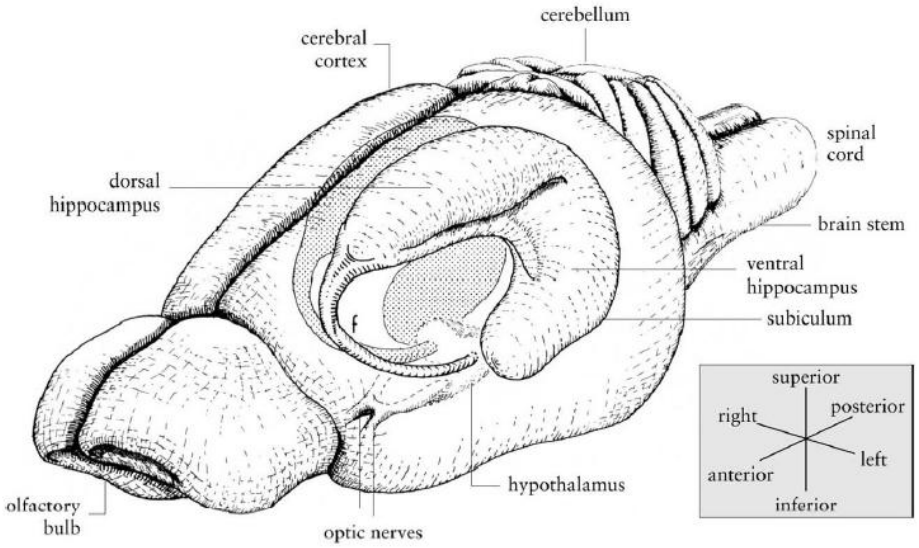

FIGURE 4- 1 : Hippocampe de rongeur

Au cœur de cette recherche il y a la carte cognitive. Le concept de carte cognitive incarne l'hypothèse qu'un animal ou un insecte peut apprendre des informations pas spécifiquement liées ou utiles à la tâche en cours. Le rat a la capacité à naviguer dans un environnement de plusieurs centaines de mètres (ou même des kilomètres) et revenir à sa maison. S'il trouve dans un champ où il se déplace un mûr, des grains ou des fruits, il peut y revenir un jour sans se perdre grâce à la carte qu'il construit de son environnement.

Des travaux expérimentaux ont montré qu'il existe au moins deux populations distinctes de cellules dans l'hippocampe de rongeur appelés cellules de place et cellules de direction de la tête. Les cellules de place sont associées à certains

emplacements physiques du robot et les cellules de direction de tête sont associées à certaines orientations du robot.

Des recherches récentes ont également découvertes les cellules de grille. Ils ont constaté que ces cellules chez le rat se sont activées dans des endroits spécifiques qui formaient un motif (Figure 4-2).

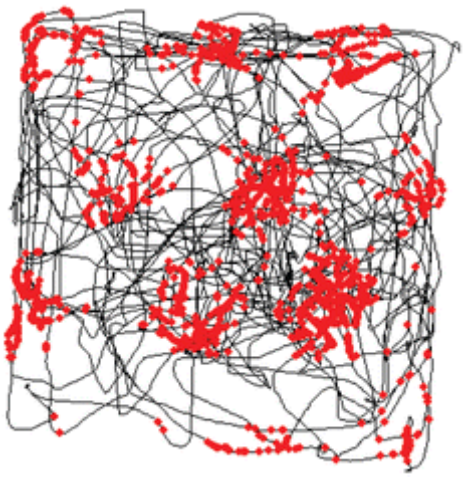

FIGURE 4- 2 : Trajectoire d'un rat dans un environnement carré (représenté en noir). Les points rouges indiquent les emplacements où une cellule de grille s'est activée.

Cependant, la plupart des modèles de l'hippocampe du rongeur se sont jusqu'à présent concentrés sur des cellules de direction de la tête et des cellules de place.

4.1.1 Cellules de direction de la tête

Les cellules de direction de tête sont sensibles à l'orientation de la tête du rongeur. Ces cellules s'activent non pas lorsque l'animal se trouve dans un lieu donné de son environnement, mais lorsque sa tête est orientée d'une façon particulière [16]. Une direction donnée de la tête induit de plus des activations préférentielles.

L'activation de la cellule de direction de la tête est indépendantes de l'emplacement du rat ou du comportement actuel (immobile, marchant, courant, mangeant), bien que quelques cellules de direction de la tête semblent dépendre des facteurs tels que la vitesse de translation et la vitesse angulaire de l'animal. Les travaux récents ont trouvé une relation entre la vitesse du rongeur et l'activation des cellules de direction de la tête pour une majorité des cellules de direction de la tête observées.

La réponse des cellules de direction de tête est triangulaire (ou gaussienne). Il y a un seul maximum, correspondant à la direction dite "préférée" de la cellule, ce qu'on pourrait appeler sa direction nominale. Une cellule "nord" ne s'active que quand la tête du rongeur est orientée au nord, plus ou moins une trentaine de degrés

(bien entendu, "nord" désigne ici un nord arbitraire, mais il s'agit bien d'une direction allo-centrée, définie par rapport à l'environnement global).

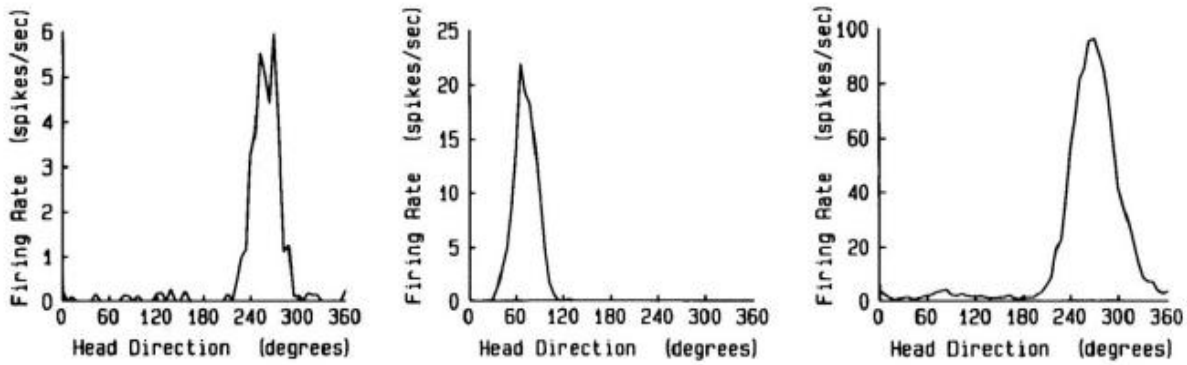

FIGURE 4- 3 : La réponse des cellules de direction de la tête.

4.1.2 Cellules de place

Les cellules de place sont les cousins deux dimensionnels des cellules de la direction de la tête. Ces cellules s'activent quand un animal arrive à un endroit déterminé de son environnement. L'activité de ces cellules décroît quand cet animal s'éloigne de cet endroit.

La zone de l'environnement dans lequel une cellule de place particulière s'active est connue comme le champ de lieux de cette cellule. Cependant, l'activation de cellules de place est également affectée par un certain nombre d'autres facteurs, tels que les repères visuels, obstacles et l'emplacement de récompense alimentaire.

Comme les cellules de la direction de la tête, l'activité des cellules de place peut changer afin de refléter le mouvement du rongeur même en l'absence de repères visuels. Dans les expériences sur des rats qui avaient été aveugles, il a été démontré que les cellules de place présentaient encore les fonctionnalités normales.

Le taux de l'activation de cellule était, en général, plus bas dans les rats aveugles que des rats aperçus, avec une raison possible étant que ces taux dépendent d'une combinaison d'entrées sensorielles, avec la privation de l'entrée visuelle aboutissant à un taux de l'action global inférieur.

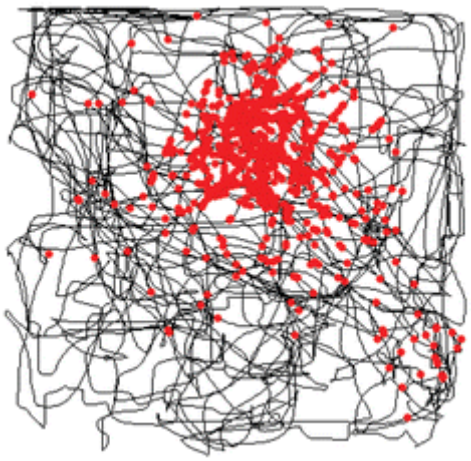

FIGURE 4- 4 : Trajectoire d'un rat dans un environnement carré (représenté en noir). Les points rouges indiquent les emplacements où une cellule de place s'est activée.

4.1.3 Réseaux attracteurs compétitifs

Les réseaux attracteurs compétitifs sont un type de réseau de neurones qui habituellement convergent vers un modèle stable d'activation à travers leurs unités. Les unités de réseau peuvent être arrangée en plusieurs configurations, mais généralement chaque unité excitera des unités près de lui et inhibera ceux-là plus loin, ce qui conduit à un bloc d'activité connue sous le nom d'un paquet d'activité éventuellement dominant.

L'activité injectée dans le réseau près de ce paquet gagnant aura tendance à déplacer ce paquet vers lui. L'activité injectée loin du paquet gagnant créera un autre paquet qui rivalise avec l'original. Si suffisamment d'activité est injectée le nouveau paquet peut 'gagner' et le vieux paquet disparaît. Cette structure de réseau est utilisée pour représenter les cellules de direction de la tête et les cellules de place.

4.2 Architecture du système de RatSLAM

RatSLAM est un modèle basé sur la vision, bio-inspiré, capable d'obtenir des résultats SLAM compétitifs dans les environnements du monde réel avec un capteur caméra et, éventuellement, les capteurs qui rassemblent les données odométrique.

Les capteurs comme le laser, les capteurs ultrasons ou des capteurs de profondeur ne sont pas utilisés. RatSLAM est un modèle informatique bio-inspiré de la structure hippocampique du rat et qu'utilise des techniques de correspondance entre les vues en combinaison avec des informations odométriques pour former un réseau d'attracteurs (continuous attractor network) (CAN).

RatSLAM se compose de plusieurs unités de traitement : cellule de vue local (local view cells), cellule de position (pose cells) et carte d'expérience (experience map).

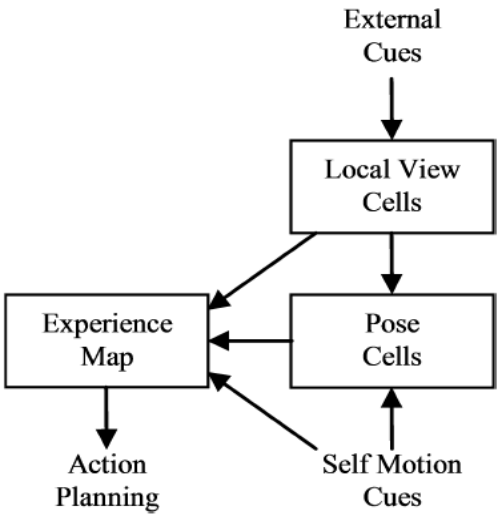

FIGURE 4- 5 : Système RatSLAM

4.2.1 Cellules de position

Les cellules de position forment un réseau d'attracteur continu tridimensionnel (Continuous Attractor Network). CAN a été un choix populaire pour modéliser les cellules trouvées dans le cerveau du rongeur. Le CAN est un réseau de neurones qui se compose d'un ensemble d'unités avec des connexions pondérées fixes qui peuvent être extractrices et inhibitrices [17].

Contrairement à la plupart des réseaux de neurones, la CAN opère principalement en variant l'activité des unités de neurones, plutôt que par la modification de la valeur des connexions pondérées.

RatSLAM utilise un taux de codage CAN, ce qui signifie que chaque groupe neuronal a une valeur d'activation continue entre zéro et un. L'interprétation du niveau d'activation d'unité est généralement comme une mesure du taux d'activation d'une cellule biologique réelle. Ainsi, un niveau d'activation plus élevée, représente le taux d'activation le plus rapide.

Chez les rongeurs, les cellules spatialement réactives telles que les cellules de place s'activent plus rapidement lorsque le rat est situé à un certain emplacement, et réduisent leur taux d'activation lorsque le rat s'éloigne de cet emplacement. Dans RatSLAM, la valeur d'activation d'une unité neuronale augmente lorsque le robot se rapproche d'une position associée à cette unité de neurones.

Au cours du temps, le réseau de cellules de position aura généralement un seul groupe d'unités fortement actives : le paquet d'activité. Le centre du paquet d'activité fournit une estimation de la position du robot qui peut être déterminée par la structure du réseau de cellule de position. Le réseau de cellules de position est organisé en une structure de prisme rectangulaire, comme illustré dans la Figure 4-6.

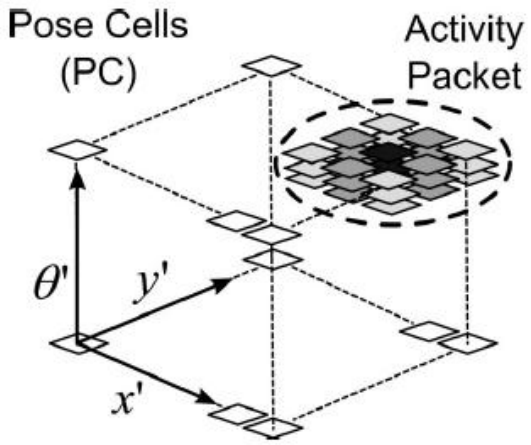

FIGURE 4- 6 : Modèle trois dimensions de cellule de position.

Chacune des trois dimensions du réseau correspond à une des trois dimensions spatiales x′, y′ et θ′. Des coordonnées prêtes sont utilisées pour en différencier l'espace de celui utilisé avec la carte d'expérience. L'emplacement des unités neuronales actives dans la structure de prisme rectangulaire corrèle avec la pose du robot (x′, y′, θ′) dans l'espace.

Attracteur dynamique

L'attracteur dynamique est conçue pour maintenir un paquet d'activité unique dans le réseau attracteur continu. Les connexions excitatrices locales augmentent l'activité des unités qui sont proches dans (x′, y′, θ′) l'espace à une unité active, générant le groupe principal. Les connexions inhibitrices suppriment l'activité des groupes de petites activités.

Pour chaque cellule de position, l'inhibition et l'excitation locale est obtenue grâce à une distribution gaussienne tridimensionnelle des connexions pondérées, comme indiqué par les flèches reliant les cellules de pose dans la figure 4-8. La distribution **ε**, est donnée par :

$$\boldsymbol{\varepsilon_{a,b,c}} = \boldsymbol{e^{-(a^2+b^2)/k_p^{exp}} e^{-c^2/k_d^{exp}} - e^{-(a^2+b^2)/k_p^{inh}} e^{-c^2/k_d^{inh}}} \qquad (\mathbf{4.1})$$

Où k_p et k_d sont les constantes de la variance pour le lieu et la direction respectivement, et a, b, et c représentent les distances entre les unités dans les coordonnées x′, y′, et θ′ respectivement.

Les variances d'inhibition sont plus grandes que pour l'excitation. Les connexions s'enrouler sur les six faces du réseau de cellules de pose, comme indiqué par les flèches plus longues connectant des cellules de pose dans la figure 4-8. Les indices a, b et c sont donnés par :

$$\begin{aligned} \boldsymbol{a} &= (\boldsymbol{x'} - \boldsymbol{i})(\boldsymbol{modn_{x'}}) \\ \boldsymbol{b} &= (\boldsymbol{y'} - \boldsymbol{i})(\boldsymbol{modn_{y'}}) \\ \boldsymbol{a} &= (\boldsymbol{\theta'} - \boldsymbol{i})(\boldsymbol{modn_{\theta'}}) \end{aligned} \qquad (\mathbf{4.2})$$

Le changement dans le niveau d'activité ΔP de la cellule est donnée par :

$$\Delta \boldsymbol{P}_{x',y',\theta'} = \sum_{i=0}^{n_{x'}-1} \sum_{j=0}^{n_{y'}-1} \sum_{k=0}^{n_{\theta'}-1} \boldsymbol{P}_{i,j,k} \boldsymbol{\varepsilon}_{a,b,c} - \boldsymbol{\varphi} \qquad (\mathbf{4.3})$$

Où $n_{x'}$, $n_{y'}$ et $n_{\theta'}$ sont la taille du réseau en nombre de cellules au long de chacune des dimensions **x**, **y** et **θ**. **φ** permet de contrôler le niveau d'inhibition globale. Toutes les valeurs de P sont limitées à des valeurs non négatives et normalisés.

Intégration de chemin

L'intégration de chemin implique le changement du paquet d'activité dans le réseau de cellule de position basé sur des informations de mouvement (comme l'odométrie). Même si un modèle biologique de haute-fidélité changerait l'activité à l'aide d'un ensemble approprié de connexions pondérées, RatSLAM déplace simplement une copie de l'état d'activité actuel par une quantité basée sur des zones spatiales nominales et les bandes d'orientation d'une cellule de position. La reproduction et le changement de l'activité offrent la performance d'intégration de chemin stable sur une gamme plus large de vitesses de mouvement et sous des taux d'itération de système irréguliers [17].

Le processus d'intégration de chemin d'accès peut provoquer un groupe d'activité dans les cellules de position pour déplacer une des faces de la structure de cellule de position et l'enrouler autour de l'autre, comme le montre les deux paquets dans la figure 4-8.

Si le robot effectue une translation, l'activité est décalé dans le plan (x, y), s'il tourne, l'activité est décalé dans la direction θ. La magnitude du mouvement dans le plan (x, y) dépend de la vitesse de translation v, et le mouvement le long de l'axe θ dépend de la vitesse de rotation w. L'équation (4.4) représente l'énergie injectée dans chaque cellule de position provient d'un groupe de cellules de position décalées par les quantités entières δx_0 , δy_0 et $\delta\theta_0$. La quantité d'activité injectée est basée sur le produit de l'activité de l'unité P et un composant de résidu α. Ce résidu est basé sur les composants fractionnaires des décalages δx_f , δy_f et $\delta\theta_f$.

$$\Delta \boldsymbol{P}_{lmn} = \sum_{x=\delta x_0}^{\delta x_0+1} \sum_{y=\delta y_0}^{\delta y_0+1} \sum_{z=\delta\theta_0}^{\delta\theta_0+1} \boldsymbol{\alpha}_{xyz} \boldsymbol{P}_{(l+x)(m+y)(n+z)}$$

$$\boldsymbol{\delta x_0 = \lfloor K_x v \cos\theta \rfloor, \delta y_0 = \lfloor K_y v \sin\theta \rfloor, \delta\theta_0 = \lfloor K_\theta w \rfloor}$$

$$\boldsymbol{\delta x_f = K_x \cos\theta - \delta x_0, \delta y_f = K_y \sin\theta - \delta y_0, \delta\theta_f = K_\theta w - \delta\theta_0} \qquad (\mathbf{4.4})$$

$$\boldsymbol{\alpha_{ijk} = g(\delta x_f, i - \delta x_0) g(\delta y_f, j - \delta y_0) g(\delta\theta_f, k - \delta\theta_0)}$$

$$\boldsymbol{g(a,b) = \begin{cases} 1-a, & b=0 \\ a, & b=1 \end{cases}}$$

Cellule de vue locale

Les cellules de vue locale sont un tableau extensible d'unités, dont chacun représente une scène visuelle distincte dans l'environnement. Quand une nouvelle scène visuelle est perçue, une nouvelle cellule de vue locale est créée et associée avec les données de pixel brutes dans cette scène.

En outre, un lien excitateurs $\boldsymbol{\beta}$ est appris entre cette cellule de vue locale et le barycentre du paquet activité dominante dans les cellules de position à ce moment-là. L'apprentissage suit une version modifiée de la loi Hebbian, où la connexion entre la cellule de vue local V_i et la cellule de la position est $P_{x',y',\theta'}$ donné par :

$$\boldsymbol{\beta}^{t+1}_{i,x',y',\theta'} = \boldsymbol{max}\left(\boldsymbol{\beta}^{t}_{i,x',y',\theta'}, \boldsymbol{\lambda V_i P_{x',y',\theta'}}\right) \quad (\mathbf{4.5})$$

Où λ est le taux d'apprentissage.

Lorsque cette vue est vu à nouveau par le robot, la cellule de vue locale devient active et injecte l'activité dans les cellules de pose via ce lien excitateur :

$$\Delta \boldsymbol{P_{x',y',\theta'}} = \boldsymbol{\delta} \sum_{\boldsymbol{i}} \boldsymbol{\beta_{i,x',y',\theta'} V_i} \quad (\mathbf{4.6})$$

Où la constante δ détermine l'influence des repères visuels sur l'estimation de pose du robot.

Un processus de saturation assure que chaque modèle visuel peut seulement injecter l'activité pendant une courte période de temps, pour éviter des fausses relocalisations quand le robot est stationnaire.

Si une séquence suffisamment longue de scènes visuelles familières est expérimentée dans la séquence correcte, l'injection constante de l'activité dans les cellules de position se traduira par une relocalisation, autrement dit, le paquet d'activité dominante se produisant à la même position que la scène a été vu dans la première fois.

Les unités dans les cellules de vue locales deviennent associées avec des unités dans les cellules de positions grâce à des connexions pondérées apprises entre les deux réseaux.

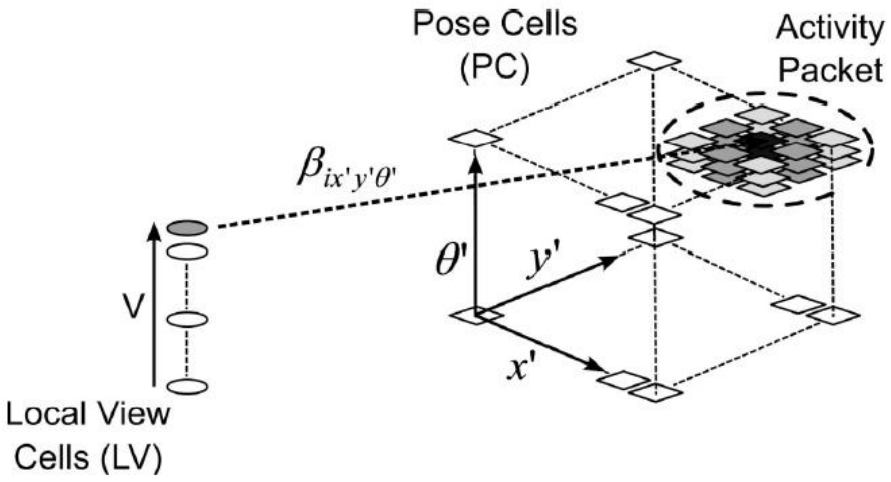

FIGURE 4- 7 : Illustration de réseau de cellules de vue locales et réseau de cellules de positions.

4.2.2 Carte d'expérience

La carte d'expérience est une carte semi métrique-topologique contenant des représentations de places, appelés des expériences **e**, et des liens entre les expériences décrivant les transitions t, entre ces places. Chaque expérience est définie par la conjonction d'un certain état de l'activité P_i dans les cellules de pose et une cellule de vue locale active V_i.

Cependant, chaque expérience est placée à une position p_i dans l'espace de l'expérience, qui est similaire à l'espace cartésien du monde réel, mais avec des contraintes de connectivité. Par conséquent, l'état complet d'une expérience peut être défini comme :

$$\boldsymbol{e_i = \{P^i, V^i, p^i\}} \qquad (\mathbf{4.7})$$

La figure 4.8 représente la région de cellules de position et de cellule de vue locale associée à l'expérience actuellement active A. Les cellules de vue locales représentent des scènes uniques apprises dans l'environnement. Les cellules de position représentent la croyance de la pose actuelle.

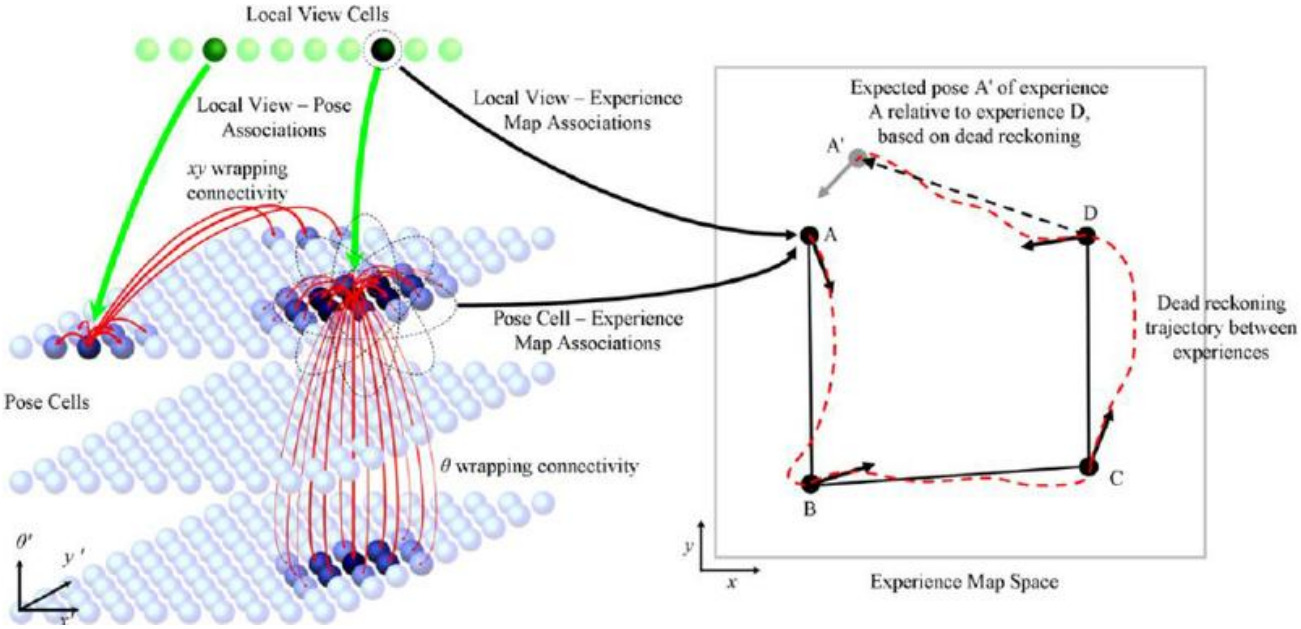

FIGURE 4- 8 : Modules majeurs du système RatSLAM.

Création d'expérience

Lorsque la combinaison de l'état d'activité de cellules de pose et cellules de vue locale n'est pas suffisamment décrite par l'une des expériences existantes, une nouvelle expérience est créée.

Une métrique de score S est utilisée pour comparer comment étroitement la pose actuelle et l'état de vue locale correspondent à ceux associés à chaque expérience, donnée par :

$$\boldsymbol{S^i = \mu_p \left| P^i - P \right| + \mu_v \left| V^i - V \right|} \qquad (\mathbf{4.8})$$

Où μ_p et μ_v sont respectivement des poids de contributions des positions et de vue locale au score de correspondance. Si $min(S) \geq S_{max}$, une nouvelle expérience est créé, défini par l'état d'activité de la pose actuelle et de cellule de vue locale.

Toutefois, s'il existe plusieurs scores d'expérience inférieure au seuil, le score le plus bas est choisi comme l'expérience « actif » et représente la meilleure estimation de l'emplacement du robot à l'intérieur de la carte de l'expérience.

Transitions d'expérience

La création d'une nouvelle expérience crée aussi une transition l_{ij} liant l'expérience précédemment active e_i à la nouvelle expérience e_j. Les liens de transition codent le changement de la position, Δp^{ij}, calculé directement à partir de l'odométrie et le temps écoulé, Δt^{ij}, depuis la dernière expérience était active:

$$\boldsymbol{l_{ij} = \{\Delta p^{ij}, \Delta t^{ij}\}} \qquad (\mathbf{4.9})$$

Où Δp^{ij} est la pose d'odométrie relative entre les deux expériences, et Δt^{ij} est le temps nécessaire pour se déplacer entre les deux expériences.

Les informations odométrie définissent l'emplacement initial dans l'espace d'expérience d'une nouvelle expérience créée :

$$\boldsymbol{e_j = \{P^j, V^j, p^i + \Delta p^{ij}\}} \qquad (\mathbf{4.10})$$

Le robot utilise cette information temporelle pour planifier les chemins à partir de son emplacement actuel vers un emplacement désiré.

Relaxation de Carte d'expérience

Dans les premières étapes de la construction de carte, les relations spatiales entre les expériences qui sont liée entre eux dans l'espace d'expérience, définis dans l'équation précédente, exactement correspondent à l'information odométrie qui existe dans les transitions entre ses expériences.

En cas de fermeture de boucle, il est probable que la position relative des deux expériences liées dans la carte ne correspond pas aux informations de transition de l'odométrie entre les deux.

La méthode de relaxation de carte expérience vise à minimiser l'écart entre l'information de transition de l'odométrie et la position absolue dans l'espace de l'expérience, en appliquant un changement dans l'emplacement de l'expérience Δp^i:

$$\boldsymbol{\Delta p^i = \alpha \left[\sum_{j=1}^{N_f} (p^j - p^i - \Delta p^{ij}) + \sum_{k=1}^{N_t} (p^k - p^i - \Delta p^{ki}) \right]} \qquad (\mathbf{4.11})$$

Où α est le taux de correction, N_f est le nombre de liens à partir de l'expérience e_i vers une autre expérience, N_t est le nombre de liens provenant d'autres expériences vers l'expérience e_i.

Cette équation est appliquée itérativement à tout moment pendant le fonctionnement du robot. il n'y a aucune détection de fermeture de boucle explicite qui déclenche la correction de carte.

4.3 SLAM bayésien vs SLAM bio-inspire

4.3.1 Les méthodes de validation

Les méthodes de validation les plus utilisées pour comparer les algorithmes SLAM sont les suivantes [18] :

Temps de calcul

Il représente le temps nécessaire à l'algorithme pour atteindre l'objectif du problème SLAM. Il peut être mesuré pour une étape particulière de l'algorithme ou pour L'ensemble de l'algorithme de la cartographie et de la localisation.

Coût de calcul

Il mesure la complexité de l'algorithme de SLAM en termes de nombre d'opérations. Ce coût a un impact direct sur la vitesse (le temps de calcul) de l'algorithme de SLAM. Dans le cas où le coût est très élevé, il peut dépasser la puissance du matériel utilisé pour le calcul SLAM.

La fermeture de la boucle

La fermeture de boucle est la tâche de décider si vraiment le robot doit retourner à une zone précédemment visitée ou non. La fermeture de boucle fiable est à la fois essentielle et difficile dans le SLAM. Par conséquent, la distance atteinte avant une fermeture correcte de la boucle est un indice important de la robustesse et de la précision de l'algorithme.

Adaptabilité

La localisation et la cartographie dans des environnements inconnus devient plus difficile dans les environnements à grande échelle. En fait, lorsque la taille de l'environnement augmente le nombre des amers visuels et donc le coût de calcul de l'algorithme de SLAM augmente.

Par conséquent l'adaptabilité ou la capacité du robot pour fonctionner correctement, même si la taille de l'environnement cartographiée ou le nombre des amers visuels augmente, est un critère important pour estimer la performance de l'algorithme de SLAM.

Dynamisme de l'environnement du robot

Ce critère mesure la capacité du robot à travailler en cas de changement d'emplacement d'autres agents dans l'environnement. Cela crée un grand défi pour les algorithmes SLAM, car il introduit des mesures de capteur incohérentes.

Problème non linéaire et bruit non gaussien

Puisque la plupart des algorithmes de SLAM sont basés sur l'hypothèse que les variables sont linéaires et le bruit est gaussienne, ce critère estime la capacité de l'algorithme à travailler dans un environnement non linéaire et avec un bruit non gaussien.

Robustesse contre bruit et incertitudes

Un algorithme SLAM robuste devrait pouvoir performer bien même en cas de données bruyantes ou d'observations ambiguës et de fortes incertitudes. La robustesse est un critère important dans le SLAM car elle est nécessaire pour obtenir de bons résultats même avec de mauvaises conditions.

4.3.2 Comparaison entre SLAM probabiliste et SLAM bio-inspiré

Premièrement, en ce qui concerne le dynamisme de l'environnement du robot, il n'y a presque aucun algorithme de SLAM bayésien capable de traiter la cartographie dans des environnements dynamiques. La plupart des solutions probabilistes de problème de SLAM consiste à construire la carte des environnements qui sont supposées statiques.

Il existe un algorithme probabiliste SLAM appelé vSLAM qui est capable de travailler dans un environnement intérieur dynamique. Cependant, lorsqu'il est testé dans un environnement extérieur, il ne peut atteindre qu'une distance de 200 m de cartographie qui n'est pas comparable aux distances obtenues par des algorithmes de SLAM bio-inspirés. L'expérience de Milford et Wyeth en 2010 [17] a montré que le système RatSLAM est capable de se déplacer d'une façon autonome dans un environnement intérieur fortement dynamique. Il est été capable de réaliser 1143 tâches de livraison dans un environnement intérieur (bureau de travail), et de parcourir une distance totale plus de 40 km pendant 37 heures avec une précision supérieure à 98%. Le SLAM bayésien ne peut pas atteindre cette performance en raison de l'augmentation de la complexité avec le nombre des amers visuels qui apparaissent dans un environnement dynamique. Cette complexité croissante pour des approches probabilistes avec le nombre des amers visuels est la raison principale pour laquelle les algorithmes SLAM bio-inspirés ont une meilleure adaptabilité et un temps de calcul plus petit que les algorithmes réguliers.

Par exemple, l'une des meilleures cartographies obtenues par les algorithmes de SLAM bayésien était dans Victoria Park avec l'algorithme FastSLAM. FastSLAM a été capable de cartographier un environnement extérieur ayant une longueur de 3,5 km. De l'autre côté, le RatSLAM a été testé avec succès dans un environnement extérieur de longueur 66 km avec une voiture à travers un réseau routier complexe. Utilisant seulement une caméra Web fonctionnant à 10 Hz [19], RatSLAM a généré une carte cohérente de l'environnement entier à vitesse réelle. Il semble que presque tous les algorithmes de SLAM bayésien ne peuvent pas construire des cartes cohérentes pour les zones très grandes, principalement en raison de la forte augmentation du coût de calcul et des incertitudes qui deviennent plus grandes lorsque l'environnement devient plus large.

Concernant le problème de fermeture de boucle, qui est l'un des plus grands défis pour les algorithmes SLAM aujourd'hui, il existe différentes approches pour le résoudre. Certaines des approches probabilistes courantes utilisent les estimations, produites par l'algorithme SLAM lui-même, de la position du robot et des

emplacements des amers visuels. Généralement, cette approche ne donne pas de bons résultats pour estimer la fermeture de boucle. La fermeture de boucle s'agit d'un problème très critique car si une boucle n'est pas détectée, par exemple quand il y a des erreurs significatives dans les estimations de la position, les zones précédemment visitées seront re-mappées par le robot, mais dans des mauvais emplacements global, ce qui entraîne une accumulation importante d'erreurs.

En revanche, les algorithmes de SLAM bio-inspirés sont également très performants dans la détection de la fermeture de boucle. Par exemple, le RatSLAM a été capable de cartographier en temps réel une boucle d'une longueur maximale de 5km en utilisant uniquement des informations sensorielles visuelles. Ce système a montré sa haute performance dans la fermeture de grandes boucles même en cas des erreurs significatives d'intégration de chemin. Il détecte également des boucles internes à l'intérieur d'une boucle externe.

Lorsqu'il s'agit du temps de calcul, les deux classes de SLAM ont montré leurs performances en temps réel. Cependant, le problème avec les approches probabilistes est qu'ils ne peuvent pas s'adapter à un grand environnement même s'ils peuvent effectuer une cartographie et localisation rapide pour les petits environnements. Comme mentionné précédemment, la raison de cette mauvaise adaptabilité est la croissance de la complexité.

La force clé des techniques probabilistes est peut-être leur robustesse et leur faible sensibilité au bruit. Par conséquent, dans le cas de mauvais capteurs, c'est-à-dire avec beaucoup de bruit et d'incertitude, les algorithmes probabilistes donnent généralement une carte plus robuste que les algorithmes bio-inspirés.

Enfin, il existe un problème qui est intrinsèquement lié aux approches probabilistes. Beaucoup d'entre eux sont basés sur l'hypothèse de linéarité et de bruit gaussien, qui n'est pas toujours le cas dans l'environnement. Cela limite significativement la performance de ces algorithmes et nécessite quelques extensions et de nouvelles solutions afin de pouvoir traiter tous les types possibles d'environnement.

Conclusion

La plupart des travaux actuelles sur le SLAM utilisent des techniques probabilistes. Les techniques bio-inspirés récents ont prouvé leurs capacités à résoudre le problème de SLAM, en utilisant des méthodes qui sont apparemment robustes, flexibles et bien intégrées dans le robot.

Les algorithmes bio-inspirés, comme leur nom indique, sont inspirés par les systèmes biologiques, principalement le cerveau d'humain et de rongeurs. Les rats peuvent naviguer et mettre à jour leur représentation de la position même sans des indices externes parce qu'ils ils utilisent seulement des estimations de l'auto-

mouvement ou de ce que l'on appelle l'intégration du chemin. Ils ont également une capacité remarquable pour stocker et se rappeler des indices visuels.

Ce chapitre nous a permis de représenter les différentes cellules, cellule de direction de la tête, cellule de place et cellule de grille, qui sont responsables de la navigation chez le rat. Nous avons vu aussi le système RatSLAM qui est l'un des algorithmes de SLAM bio-inspirés les plus connus et qui offrent une solution au problème de la cartographie et de la localisation à grande échelle. Il utilise un modèle de calcul simplifié de l'hippocampe de rongeur pour construire une carte en ligne et en temps réel.

Nous avons établi une comparaison entre les deux classes de SLAM à savoir le SLAM probabiliste et le SLAM bio-inspiré basée sur quelques méthodes de validation utilisées pour évaluer la performance d'un algorithme SLAM. Nous avons vu que des algorithmes probabilistes peuvent nous fournir des algorithmes très précis et fidèles dans l'environnement réel, mais que cette haute précision a un coût de calcul élevé qui les rend plus lents et plus complexes que les algorithmes bio inspirés spécialement pour les environnements à grande échelle.

Nous avons également vu que les approches probabilistes sont plus adaptées aux situations où il y a une quantité importante de bruit et d'incertitude, mais que les restrictions sur le modèle gaussien de bruit et les hypothèses de linéarité peuvent limiter de façon significative l'utilisation de méthodes probabilistes. Finalement, nous avons montré que pour les environnements dynamiques, les systèmes de SLAM bio-inspirés sont beaucoup plus adaptés et peuvent obtenir de très bons résultats.

Chapitre 5

RatSLAM basé sur les points d'intérêt visuels

Nous présentons dans ce chapitre une démarche théorique des deux méthodes concernant l'odométrie visuelle : première méthode qui est utilisée dans l'algorithme RatSLAM et la deuxième méthode basée sur les points d'intérêt visuels que nous proposons..

Introduction

Contrairement au système de cartographie et de la localisation, le système de vision utilisé dans le RatSLAM n'a aucune inspiration biologique. Ce système est basé sur l'utilisation des techniques de l'odométrie visuelle et de mise en correspondance d'image.

L'intérêt de l'odométrie visuelle est de déterminer le mouvement de la caméra, en comparant des images successives. Il existe plusieurs approches pour réaliser une odométrie visuelle plus fiable. L'une des approches d'odométrie visuelle les plus connues qui est basée sur l'utilisation des points caractéristiques pour faire la mise en correspondance entre les images. Selon les points de correspondance spécifiés, il existe trois méthodes différentes 2D-à-2D, 3D-à-2D et 3D-à-3D et chacune a des avantages et des inconvénients. L'utilisation de la méthode 3D-à-3D nécessite une caméra stéréo. Dans le cas d'une caméra monoculaire, la méthode de 2D-à-2D est préférable comparée au cas de 3-D-à-2-D puisqu'elle évite la triangulation des points.

L'objectif de ce chapitre est d'appliquer l'une de ces méthodes de l'odométrie visuelle pour déterminer le déplacement du robot mobile et la comparer avec celle utilisée dans le RatSLAM. Nous nous intéressons par la méthode 2D-à-2D puisqu'on utilise une caméra monoculaire.

5.1 Description du module odométrie visuel de RatSLAM

L'algorithme de vision utilisé dans le RatSLAM fournit une estimation odométrique basée sur les changements dans la scène visuelle, en utilisant les profils d'intensité. Le profil d'intensité est un vecteur unidimensionnel formé en additionnant les valeurs d'intensité dans chaque colonne de pixels et puis normalisant le vecteur. Ce profil est utilisé pour estimer la rotation et la vitesse

de translation entre les images et de comparer l'image actuelle avec l'image vue précédemment pour effectuer le calibrage de vue local [19].

5.1.1 Estimation de rotation

Les informations de rotation sont estimées en comparant les tableaux d'images consécutifs. La figure 5-1(a) montre les profils d'intensité de deux images consécutives. La comparaison entre les profils est réalisée en calculant la différence d'intensité absolue moyenne entre les deux profils d'intensité $f(s)$ tels qu'ils sont décalés l'un par rapport à l'autre.

$$\boldsymbol{f(s, I^j, I^k) = \frac{1}{w - |s|}\left(\sum_{n=1}^{w-|s|} \left| I^j_{n+max(s,0)} - I^k_{n-min(s,0)} \right|\right)} \qquad (\mathbf{5.1})$$

Où I^j et I^k sont les profils d'intensité à comparer, s est le décalage en pixel, et w est la largeur de l'image.

Le décalage s_m entre les deux images consécutives I^j et I^k est la valeur de s qui minimise la fonction f pour ces deux profils.

$$\boldsymbol{s_m = \underset{s \in |\rho - w, w - \rho|}{argmin}\, f(s, I^j, I^k)} \qquad (\mathbf{5.2})$$

Où le décalage ρ assure qu'il existe un chevauchement suffisant entre les profils.

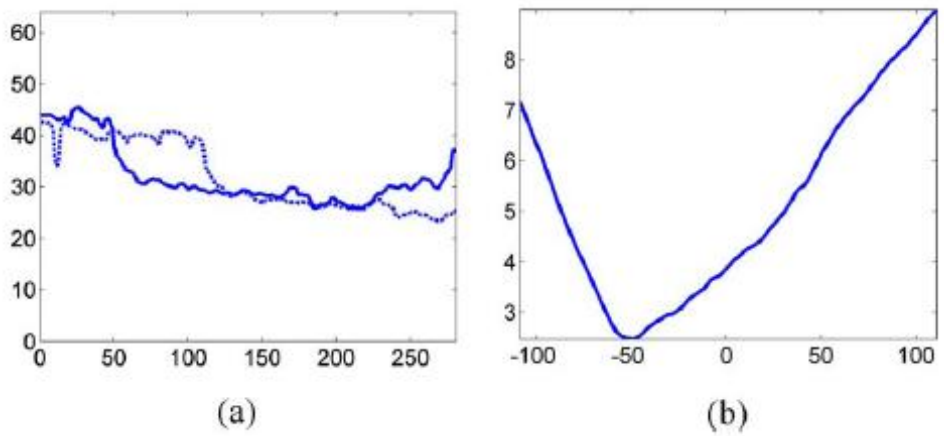

FIGURE 5- 1 : (a) Profils de deux images consécutives. (b) Décalage entre les profils de (a).

Le décalage en pixel est multiplié par un gain constant σ, pour le convertir en un décalage angulaire $\Delta\theta$:

$$\boldsymbol{\Delta\theta = \sigma s_m} \qquad (\mathbf{5.3})$$

5.1.2 Estimation de la vitesse

La vitesse est estimée selon le taux de changement de l'image. Cette vitesse de mouvement est représentée dans l'espace perceptuel plutôt que l'espace physique.
La mesure de la vitesse v est obtenue à partir de la différence d'intensité absolue moyenne filtrée entre deux profils d'intensité.

$$\boldsymbol{v = min[v_{cal} f(s_m, I^j, I^{j-1}), v_{max}]} \quad (\mathbf{5.4})$$

Où v_{cal} est une constante empiriquement déterminée qui convertit la vitesse perceptuelle en une vitesse physique, et v_{max} est un seuil de la vitesse maximum.

5.1.3 Calcul de cellule de vue locale

Une cellule de vue locale est active si la vue actuelle de l'environnement est suffisamment similaire à la vue précédemment visitée de l'environnement associé à cette cellule. La vue actuelle est comparée aux vues précédentes, en utilisant le même profil d'intensité.

Les profils d'intensité vus précédemment sont stockés comme des modèles de vues avec chaque modèle associé à une cellule de vue locale. Le profil d'intensité actuelle est comparé aux modèles stockés, et si une correspondance est trouvée comme la figure 5-2(a) montre, la cellule de vue locale associée au modèle stocké est devient active. S'il n'y a aucun correspondant à un modèle déjà connue comme indiqué dans la figure 5-2(b), un nouveau modèle est stocké et une nouvelle cellule de vue locale est créée et associée au nouveau modèle.

La comparaison entre le profil actuel et les modèles est exécutée utilisant la fonction de différence d'intensité absolue moyenne qu'on a déjà vue. La comparaison est effectuée sur une petite plage de décalages de pixels ψ pour fournir une certaine généralisation en rotation aux correspondances. Le meilleur appariement est trouvé comme :

$$\boldsymbol{k_m = \underset{k}{argminf}(s, I^j, I^k), s \in [-\psi, \psi]} \quad (\mathbf{5.5})$$

Où I^j est le profil actuel et I^k sont les profils stockés.

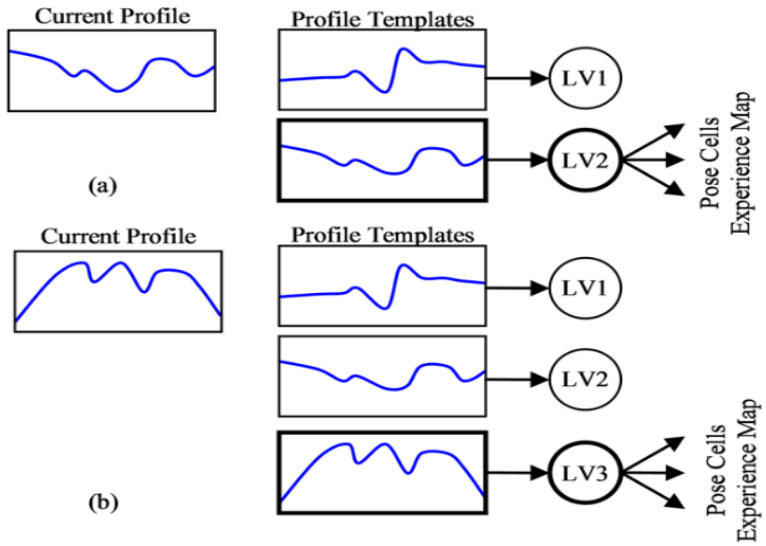

FIGURE 5- 2 : modèle de correspondance pour le calcul de vue local.

La qualité de la correspondance d est calculée par :

$$\boldsymbol{d = \min_{s\in[-\psi,\psi]} f\left(s, I^{j}, I^{k_m}\right)} \qquad (\mathbf{5.6})$$

La valeur d est comparée avec un seuil d_m pour déterminer si le profil est suffisamment similaire au modèle ou si un nouveau modèle doit être créé.
Le vecteur de vue local est ensuite défini par :

$$\boldsymbol{V_i = \begin{cases} d_m - d_i & d_i \le d_m \\ 0 & d_i > d_m \end{cases} \forall_i} \qquad (\mathbf{5.7})$$

5.2 Module de visuelle odométrie basé sur les points d'intérêt

Dans le RatSLAM, les images acquises ne sont pas bien interprétées. Pour cette raison, une autre méthode d'odométrie visuelle est utilisée en se basant sur les points d'intérêt. L'algorithme utilisé est déjà décrit dans le chapitre 3.

5.2.1 Détection des points d'intérêt

La première étape consiste à détecter une région où se trouve le point clé dans les deux images prise successivement par le détecteur ORB, ensuite chaque région détectée autour du point d'intérêt est convertie en un descripteur compact qui peut être mise en correspondance avec d'autres descripteurs, en utilisant le descripteur BRIEF qui décrit dans le chapitre 3.

5.2.2 La mise en correspondance

Ensuite, l'étape de la mise en correspondance est effectuée. La façon la plus simple pour la mise en correspondance entre deux images est de comparer tous les descripteurs trouvés dans la première image à tous autres descripteurs dans la deuxième image. Les descripteurs sont comparés en utilisant une mesure de similarité.

5.2.3 La méthode RANSAC (RANdom SAmple Consensus)

Les points appariés contiennent des valeurs aberrantes (outliers), qui sont, des mauvaises associations de données. Les causes possibles de ces points aberrants sont le bruit dans l'image, les changements de points de vue et l'éclairage que le descripteur ne prend pas en considération.

Pour estimer le mouvement de la caméra avec précision, il est important que les points aberrants soient éliminés. La suppression de ces points est une tâche plus délicate en visuelle odométrie. La solution pour éliminer les points aberrants consiste à prendre les avantages des contraintes géométriques introduites par le modèle de mouvement. L'idée derrière RANSAC est de calculer les hypothèses du modèle à partir d'ensembles des points de données choisis au hasard, et puis vérifier ces hypothèses sur les autres points de données [20].

L'hypothèse qui montre le consensus le plus élevé avec les autres données est sélectionnée comme une solution. Pour l'estimation de mouvement entre deux vues avec l'odométrie visuelle, le modèle estimé est le mouvement relatif (R, t) entre les deux positions de la caméra, et les points de données sont les candidats des points correspondances. L'algorithme RANSAC est illustré dans la table 5-1.

Le nombre d'itérations N qui est nécessaire pour garantir qu'une solution correcte est trouvée peut-être calculée par :

$$\boldsymbol{N = \frac{log(1-p)}{log(1-(1-\varepsilon)^s)'}} \qquad (\mathbf{5.8})$$

Où s est le nombre de points de données dont le modèle peut être instancié, ε est le pourcentage des points aberrants dans les points de données et p est la probabilité de succès demandée. Comme observé, RANSAC est une méthode probabiliste et non déterministe pour cela il expose une solution différente sur les différentes pistes ; cependant, la solution tend à être stable lorsque le nombre d'itérations augmente.

Algorithme RANSAC
Entrées : • k : le nombre maximal d'itérations de l'algorithme • d : un seuil pour déterminer si un point correspond à un modèle • A : un ensemble de N points correspondants

Sorties :

- m : les inliers
- M : le meilleur modèle trouvé

Répéter tant que le nombre d'itération < k :
Sélection d'un échantillon au hasard de s points de puis A
Estimation d'un modèle à partir de ces points
Calcul de la distance de tous les autres points de A, en utilisant le modèle estimé
Construire l'ensemble d'inliers (en comptant le nombre de points dont la distance par rapport au modèle < d)
Si le nombre d'inliers trouvé > m
m <= nombre d'inliers

Calcul du modèle M en utilisant tous les inliers m

Table 5- 1 : Algorithme RANSAC

5.2.4 Calcule de la matrice essentielle

Les relations géométriques entre les deux images $\boldsymbol{I_k}$ et $\boldsymbol{I_{k-1}}$ d'une caméra calibrée sont décrites par la matrice dite essentielle $\boldsymbol{E}$. Cette matrice contient les informations de translations et de rotations de la première caméra vers la deuxième à un facteur d'échelle près, exprimée par la relation suivant :

$$\boldsymbol{E_k = \hat{t}_k R_k} \qquad (\mathbf{5.9})$$

Où $\boldsymbol{t_k = [t_x, t_y, t_z]}$ et

$$\hat{\boldsymbol{t}}_{\boldsymbol{k}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{-t_z} & \boldsymbol{t_y} \\ \boldsymbol{t_z} & \boldsymbol{0} & \boldsymbol{-t_x} \\ \boldsymbol{-t_y} & \boldsymbol{t_x} & \boldsymbol{0} \end{bmatrix}$$

La matrice essentielle peut être calculée à partir de correspondances 2D à 2D, et la rotation et la translation peuvent être directement extraites de $\boldsymbol{E}$. La propriété principale d'estimation de mouvement 2D à 2D basée est la contrainte épipolaire, qui détermine la ligne sur laquelle se situe l'observation correspondante p' de p dans l'autre image (Figure 5-3).

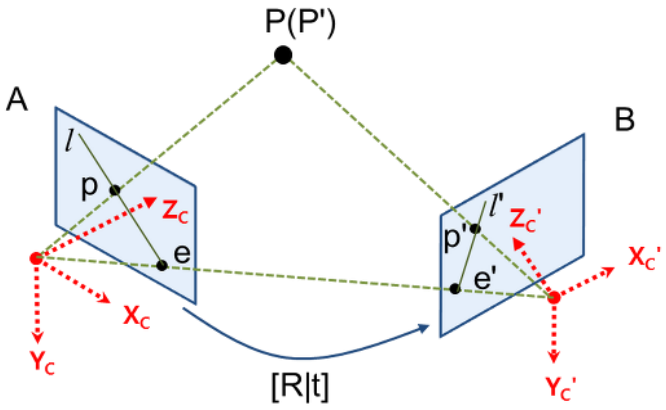

FIGURE 5- 3 : Contrainte épipolaire

Cette contrainte peut être formulée par :

$$\tilde{\boldsymbol{p}}'^{T}\boldsymbol{E}\,\tilde{\boldsymbol{p}} = \boldsymbol{0} \qquad (\boldsymbol{5.10})$$

Où $\tilde{\boldsymbol{p}}' = [\tilde{\boldsymbol{u}}', \tilde{\boldsymbol{v}}', \boldsymbol{1}]^{\boldsymbol{T}}$ est l'emplacement du point d'intérêt dans l'image $\boldsymbol{I_{k'}}$ et $\tilde{\boldsymbol{p}} = [\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \boldsymbol{1}]^{\boldsymbol{T}}$ est l'emplacement de son homologue dans l'autre image $\boldsymbol{I_{k-1}}$. Les points $\tilde{\boldsymbol{p}}'$ et $\tilde{\boldsymbol{p}}$ sont représentés en coordonnées d'image normalisées.

$$\tilde{\boldsymbol{p}} = [\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}, \boldsymbol{1}]^{\boldsymbol{T}} = \boldsymbol{K^{-1}}[\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{1}]^{\boldsymbol{T}} \qquad (\boldsymbol{5.11})$$

Où $\boldsymbol{K}$ est la matrice des paramètres intrinsèques de la caméra.

Dans le cas où les paramètres intrinsèques des caméras ne sont pas connus, une matrice analogue à la matrice essentielle existe : la matrice fondamentale (notée $\boldsymbol{F}$). La relation qui lie la matrice essentielle à la matrice fondamentale est la suivante :

$$\boldsymbol{F} = \boldsymbol{K^{-T}EK^{-1}} \qquad (\boldsymbol{5.12})$$

Plusieurs méthodes existent pour calculer la matrice essentielle. On peut citer notamment la méthode de 8 points introduite par [21]. Une autre méthode a été proposée par [22] nécessite seulement 5 points pour déterminer la matrice essentielle.

Pour n'importe quels deux homologues, on peut écrire :

$$[\tilde{u}\tilde{u}' \;\; \tilde{u}'\tilde{v} \;\; \tilde{u}' \;\; \tilde{u}\tilde{v}' \;\; \tilde{v}\tilde{v}' \;\; \tilde{v}' \;\; \tilde{u} \;\; \tilde{v} \;\; 1]\boldsymbol{e} = \boldsymbol{0} \qquad (\boldsymbol{5.13})$$

Où $\boldsymbol{e} = [\boldsymbol{e_{11}}\ \boldsymbol{e_{12}}\ \boldsymbol{e_{13}}\ \boldsymbol{e_{21}}\ \boldsymbol{e_{22}}\ \boldsymbol{e_{23}}\ \boldsymbol{e_{31}}\ \boldsymbol{e_{32}}\ \boldsymbol{e_{33}}]^{\boldsymbol{T}}$ et $\boldsymbol{e_{ij}}$ désigne l'élément de $\boldsymbol{E}$ dans la $i^{ème}$ ligne et la $j^{ème}$ colonne.

Les vecteurs de coefficients obtenus à partir d'un ensemble de $\boldsymbol{n}$ paires $\tilde{\boldsymbol{p}} \leftrightarrow \tilde{\boldsymbol{p}}'$ peuvent être empilés pour former un système d'équations linéaires.

$$\boldsymbol{Ae} = \begin{bmatrix} \tilde{u}_1\tilde{u}'_1 & \tilde{u}'_1\tilde{v}_1 & \tilde{u}'_1 & \tilde{u}_1\tilde{v}'_1 & \tilde{v}_1\tilde{v}'_1 & \tilde{v}'_1 & \tilde{u}_1 & \tilde{v}_1 & 1 \\ & & & \vdots & & & & & \\ \tilde{u}_n\tilde{u}'_n & \tilde{u}'_n\tilde{v}_n & \tilde{u}'_n & \tilde{u}_n\tilde{v}'_n & \tilde{v}_n\tilde{v}'_n & \tilde{v}'_n & \tilde{u}_n & \tilde{v}_n & 1 \end{bmatrix} \boldsymbol{e} = \boldsymbol{0} \qquad (\boldsymbol{5.14})$$

Ce système d'équations linéaires peut facilement être résolu à l'aide de l'algorithme de 8 points. Cet algorithme utilise les correspondances de 8 points pour trouver la matrice essentielle en se basant sur la décomposition en valeur singulière SVD [21]. La décomposition de $\boldsymbol{A}$ est de la forme $\boldsymbol{A = USV^T}$ et la solution de moindres carrés qui minimise $\|\boldsymbol{Ae}\|$ avec $\|\boldsymbol{e}\| = \mathbf{1}$ peut être trouvée comme la dernière colonne de $\boldsymbol{V}$.

Cependant, cette estimation linéaire de $\boldsymbol{e}$ ne respecte pas les contraintes d'une matrice essentielle, car le vecteur résultant $\boldsymbol{e}$ qui peut être utilisé pour former la matrice $\boldsymbol{E}$ ne contient pas nécessairement les bonnes valeurs singulières $\{\mathbf{1}, \mathbf{1}, \mathbf{0}\}$.
Pour cette raison, une décomposition SVD de $\boldsymbol{E = USV^T}$ est effectuée et ensuite le calcul de la matrice essentielle est fait en utilisant : $\boldsymbol{E = U\, diag}([\mathbf{1}, \mathbf{1}, \mathbf{0}])\boldsymbol{V^T}$.

5.2.5 Extraction de R et t à partir de E

A partir de l'estimation de $\boldsymbol{E}$, les éléments de rotation et de translation peuvent être extraites. Les solutions de $\boldsymbol{R}, \boldsymbol{t}$ sont :

$$\boldsymbol{t} = [\boldsymbol{u_{13}}\ \boldsymbol{u_{23}}\ \boldsymbol{u_{33}}]^T$$
$$\boldsymbol{R_a = UWV^T} \quad \boldsymbol{ou} \quad \boldsymbol{R_b = UW^TV^T} \qquad (\mathbf{5.15})$$

Où $\boldsymbol{W} = \begin{bmatrix} \mathbf{0} & \mathbf{-1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$

Toutes les combinaisons de $\boldsymbol{R}$ et $\boldsymbol{t}$ satisfont la contrainte épipolaire (5.10). Pour résoudre les ambiguïtés inhérentes, on suppose que la première matrice de la caméra est $[\boldsymbol{I}|\mathbf{0}]$ et que $\boldsymbol{t}$ est de longueur unitaire. Il y a donc quatre solutions possibles pour la deuxième matrice de la caméra :

$$\boldsymbol{P_A} = [\boldsymbol{R_a}|\,\boldsymbol{t}] \qquad \boldsymbol{P_B} = [\boldsymbol{R_a}|-\boldsymbol{t}] \qquad \boldsymbol{P_C} = [\boldsymbol{R_b}|\,\boldsymbol{t}] \qquad \boldsymbol{P_D} = [\boldsymbol{R_b}|-\boldsymbol{t}] \qquad (\mathbf{5.16})$$

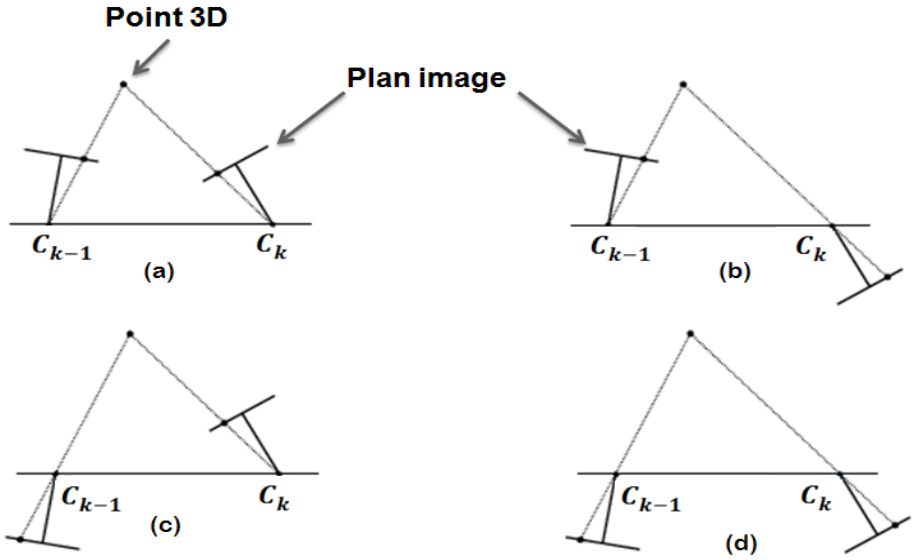

FIGURE 5- 4 : Les quatre solutions possibles de la deuxième matrice de la caméra.

Lorsque le calcul de ces quatre solutions est effectué, la solution correcte peut être déterminée par triangulation d'un point pour tous le choix de $\boldsymbol{P}$ et la solution est choisie où le point est devant les deux caméras.

5.2.6 Estimation de la position du robot

La transformation de l'image n-1 à l'image n est formé par :

$$\boldsymbol{T_n} = \begin{bmatrix} \boldsymbol{R_{n,n-1}} & \boldsymbol{t_{n,n-1}} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \quad (\mathbf{5.17})$$

Où $\boldsymbol{R_{n,n-1}}$ est la matrice de rotation et $\boldsymbol{t_{n,n-1}}$ est le vecteur de translation

La trajectoire complète est récupérée par la concaténation de toutes les transformations.

$$\boldsymbol{C_n} = \boldsymbol{C_{n-1}T_n} \quad (\mathbf{5.18})$$

Où $\boldsymbol{C_n}$ désigne la position de la caméra l'instant n.

Conclusion

Ce chapitre représente deux approches pour le calcul de l'odométrie, la méthode utilisée en RatSLAM qui est basé sur la comparaison des profils d'intensité qui donne les vitesses de rotation et de translation.

Par contre notre approche est complètement différente car elle utilise les caractéristiques de l'image pour calculer le déplacement effectué par le robot. Les techniques qui ont été utilisées pour cette approche ne sont pas les plus sophistiquées disponibles, mais elles sont simples et rapides. L'étape la plus coûteuse de l'algorithme est l'estimation d'une solution pour la matrice essentielle, où la décomposition en valeur singulière est utilisée. La complexité computationnelle du calcul de la SVD d'un $m \times n$ matrice est $O(mn^2)$. Le nombre de colonnes est fixé dans le contexte donné, de sorte que la complexité de l'algorithme décrit ici est $O(n^2)$ pour chaque étape, avec n points correspondants.

Un inconvénient de l'approche actuelle est que seulement deux images sont utilisées à la fois. Il existe des méthodes pour estimer le mouvement des caméras et la structure de scène à partir de plus de deux images, comme l'ajustement du faisceau « bundle adjustment » qui peut réduire la dérive par rapport à l'odométrie visuelle à deux vues car il utilise des mesures de caractéristiques sur plus de deux images.

Chapitre 6

Résultats expérimentaux

Le dernier chapitre est destiné à présenter l'es outils utilisés pour faire ce travail à savoir le méta-système d'exploitation ROS, la bibliothèque OpenCV. Ensuite nous allons illustrer les différents résultats obtenus et puis les comparer afin de faire évoluer ce travail.

Introduction

Nous avons déjà vu que l'algorithme de RatSLAM se compose de plusieurs nœuds ou bien des unités. Dans ce chapitre, nous décrivons l'organisation de ces nœuds en ROS et ainsi de l'approche de l'odométrie visuelle que nous proposons et qui est déjà expliqué dans le chapitre 5. En suite, le résultat de chaque nœud sera détaillé, et finalement nous allons comparer les résultats trouvées.

6.1 Présentation de ROS

La communauté robotique a fait des progrès impressionnants ces dernières années. Le matériel de robots fiable et coûteux - des robots mobiles terrestres, aux hélicoptères, aux humanoïdes - est plus largement disponible que jamais auparavant. La communauté a également développé des algorithmes qui aident ces robots à fonctionner avec des niveaux croissants d'autonomie. Parmi ces travaux l'un des plus abouti est sans doute ROS.

ROS (Robot Operating System) est un méta-système d'exploitation open-source qui fournit l'ensemble des services comme le contrôle des périphériques de bas niveau, la gestion des packages installés et la communication interprocessus. Il fournit également des outils et des bibliothèques pour l'obtention, la construction, l'écriture et l'exécution de code sur plusieurs ordinateurs.

6.1.1 Nœuds

Les nœuds sont des processus qui effectuent des calculs. Un système de commande de robot comprend généralement plusieurs nœuds. Par exemple, un nœud contrôle un télémètre laser, un nœud contrôle les moteurs de roues, un nœud effectue

la localisation, un nœud réalise la planification de chemin, un nœud fournit une représentation graphique du système, et ainsi de suite.

6.1.2 Les messages

Les nœuds communiquent entre eux en faisant passer des messages. Un message est simplement une structure de données. Les messages peuvent inclure de types standards (entier, flottant, booléen, etc.), des tableaux, et également d'autres messages. Un nœud qui s'intéresse à un certain type de données souscrira au topique approprié.

6.1.3 Topique

Le principal mécanisme que les nœuds utilisent pour communiquer est d'envoyer des messages. Les messages sont routés via un système de transport publier/souscrire. Un nœud envoie un message en le publiant à un topique donné. Le topique est un nom qui est utilisé pour identifier le contenu du message. Les nœuds qui sont intéressés à un certain type de données vont alors souscrire à ce topique pour recevoir ces messages.

Il peut y avoir plusieurs éditeurs et abonnés simultanés pour un seul topique, et un seul nœud peut publier et/ou souscrire à plusieurs topiques tel que les éditeurs et les abonnés ne sont pas conscients de l'existence de chacun. Logiquement, on peut penser à un topique comme un bus de message fortement typé. Chaque bus a un nom, et tout le monde peut se connecter au bus pour envoyer ou recevoir des messages tant qu'ils sont du bon type.

6.1.4 Les services

Le modèle de publisher/subscriber est un paradigme de communication très flexible. Le transport à sens unique ne convient pas pour des interactions de demande/réponse, qui sont souvent nécessaires dans un système distribué. Demande/réponse est effectué via des services, qui sont définis par une paire de structures de message : une pour la demande et l'autre pour la réponse.

Un nœud fournissant offre un service sous un nom et un client utilise le service en envoyant le message de demande et attendant la réponse. Les bibliothèques clientes de ROS présentent généralement cette interaction pour le programmeur comme si elle était un appel de procédure à distance.

6.1.5 Le fichier bag

Bag est un format de fichiers pour stocker, lire et traiter des données de message ROS telles que les données des capteurs, qui peuvent être difficiles à recueillir, et ils sont nécessaires pour développer et tester des algorithmes dans ROS.

6.2 Présentation d'OpenCV

OpenCV (Open source Computer Vision Library), est une bibliothèque open source de vision par ordinateur et Machine Learning. Elle est écrite en C et C++ et fonctionne sous Linux, Windows et Mac OS. Il est actif sur le développement des interfaces pour Python, Ruby, Matlab et autres langues.

OpenCV a été conçu pour l'efficacité de calcul et avec une forte concentration sur les applications en temps réel. Cette bibliothèque a plus de 2500 algorithmes optimisés pouvant être utilisés pour détecter et reconnaître des visages, identifier des objets, classifier des actions humaines dans des vidéos, suivre le mouvement de caméra, suivre les objets en mouvement, extraire les modèles 3D des objets, produire des nuages de point 3D à partir de caméras stéréo, assembler des images pour produire une image haute résolution d'une scène entière, trouver des images semblables dans une base de données, suivre des mouvements d'œil, etc.

6.3 Présentation OpenRatSLAM

OpenRatSLAM est une version open-source de RatSLAM [23], Développée à l'Université de Technologie du Queensland (QUT) à Brisbane, en Australie. Il est écrit en C++ et intégré avec ROS (Robot Operating System) pour tirer parti des avantages tels que l'abstraction de robot et de capteur, la gestion de réseau, la lecture de données et la visualisation.

L'implémentation OpenRatSLAM de RatSLAM se compose de quatre nœuds. Ces quatre nœuds et les connexions entre eux sont représentés sur la Fig. 6.1.

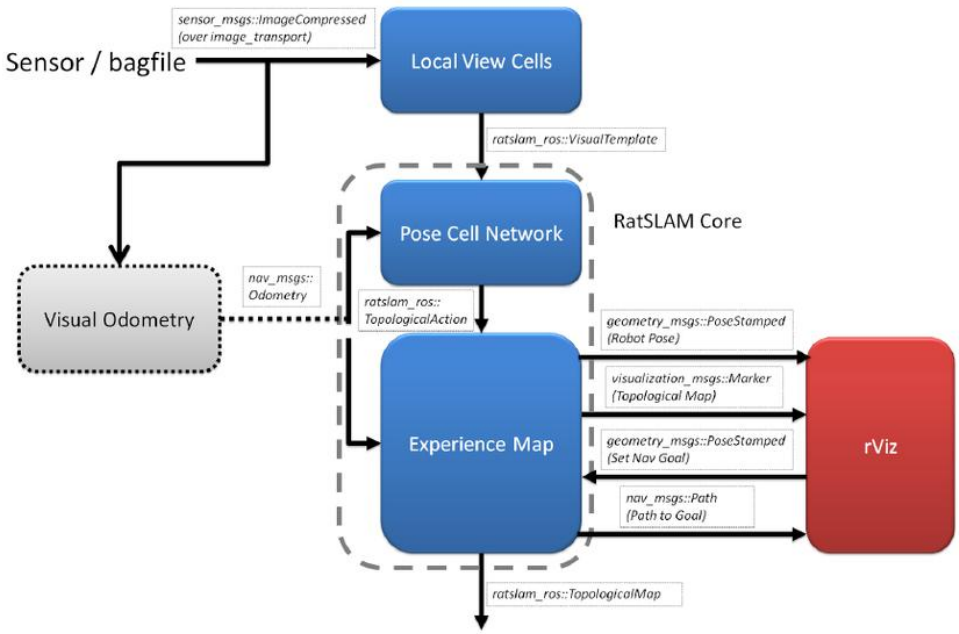

FIGURE 6- 1 : La structure de nœuds et de messages pour OpenRatSLAM

6.3.1 Nœud de l'odométrie visuelle (Visual Odometry)

Ce nœud fournit une estimation odométrique basée sur des changements dans la scène visuelle. Il détermine le mouvement de la caméra en comparant des images successives. Le nœud odométrie visuelle souscrit à un topique qui publie les message « sensor_msgs::CompressedImage » (à l'aide de la bibliothèque image_transport de ROS) pour recevoir les images fournit par le fichier bag, et publie des messages « nav_msgs::Odometry » dans le topique de l'odométrie.

6.3.2 Nœud de cellules de Vue Locales (Local View Cells)

Ce nœud détermine si une scène donnée par la vue actuelle est nouvelle ou familière en utilisant des techniques de comparaison des images. Le nœud de cellule de vue locale souscrit à un topique qui publie les messages « sensor_msgs::CompressedImage » (à l'aide de la bibliothèque image_transport de ROS) et publie un message personnalisé « ratslam_ros::ViewTemplate » qui inclut un identifiant de modèle se référant à la cellule de vue locale actuellement active.

6.3.3 Nœud de réseau de cellule de pose (Pose Cell Network)

Ce nœud gère le paquet d'énergie qui représente la position en réponse aux connexions de l'odométrie et de vue locale. Il prend également des décisions concernant le nœud de carte d'expérience et la création de liaison. Le nœud de cellules de position répond à deux types d'entrée. Ces entrées sont reçues sous forme de messages ROS. Ce nœud souscrit en deux topiques : le premier topique publie les messages « nav_msgs::Odometry » pour l'entrée odométrie et le deuxième topique publie les messages « ratslam_ros::ViewTemplate » pour l'entrée de cellule de vue locale.

Après avoir terminé, le nœud doit déterminer une action pour le graphe topologique de la carte d'expérience. Les actions possibles sont créés dans un nouveau nœud (qui implique implicitement la création d'un lien à partir du nœud précédent), créer une liaison entre deux nœuds existants ou définir l'emplacement d'un nœud existant. Cette action est envoyée dans un nouveau message personnalisé, « ratslam_ros::TopologicalAction ».

6.3.4 Nœud de la carte d'expérience (Experience Map)

Ce nœud gère la construction des graphes, la relaxation des graphes et la planification de chemin. Le nœud de carte d'expérience souscrit à un topique qui publie les messages de type « ratslam_ros::TopologicalAction » et utilise les actions reçues pour créer des nœuds et des liens ou pour définir le nœud courant.

Le nœud de carte d'expérience publie trois messages pour exposer l'état de la carte d'expérience. La première est une représentation complète de la carte d'expérience comme une carte topologique dans un message personnalisé « ratslam_ros::TopologicalMap » qui consiste en une liste de nœuds et de liens. Le

deuxième est la pose du robot dans la carte de l'expérience dans un message « geometry_msgs::PoseStamped » . Le troisième message est un message de type « visualization_mags::Marker » pour rendre la carte dans rviz.

6.3.5 Rviz

Il existe plusieurs options disponibles pour visualiser l'état du système OpenRatSLAM. Comme indiqué dans la Fig. 6.1, l'outil rviz de ROS peut être utilisé pour visualiser la carte d'expérience, la pose du robot et le chemin du robot vers un but.

6.4 Expériences et résultats

OpenRatSLAM prend en entée des séquences d'images monoculaire et l'odométrie sous forme d'un fichier bag, qui est la méthode normalisée pour stocker les données de message ROS. Il existe une variété d'outils pour enregistrer, lire, analyser et visualiser les données du message des fichiers bag de ROS.

Les trois fichiers bag disponibles sur Internet sont : St lucia 2007, Oxford New College 2008 et iRat 2011 Australia. Le fichier bag de St Lucia 2007 contient des images uniquement. Le nœud de l'odométrie visuelle est nécessaire pour calculer l'odométrie comme indiqué dans la Fig. 6.1. Lorsque l'odométrie est déjà fournie par le fichier bag ou bien par les capteurs de robot, comme les fichiers bag de New College 2008 et iRat 2011, le nœud visuel odométrie ne sera pas nécessaire.

6.4.1 Résultat de RatSLAM de StLucia

L'ensemble des données de St Lucia se compose de séquences de caméras Web provenant d'un parcours de 66 km en voiture dans une banlieue australienne, à travers une large gamme de types d'environnement.

Les figures suivantes montrent l'évolution du système RatSLAM pour l'ensemble de données St Lucia 2007.

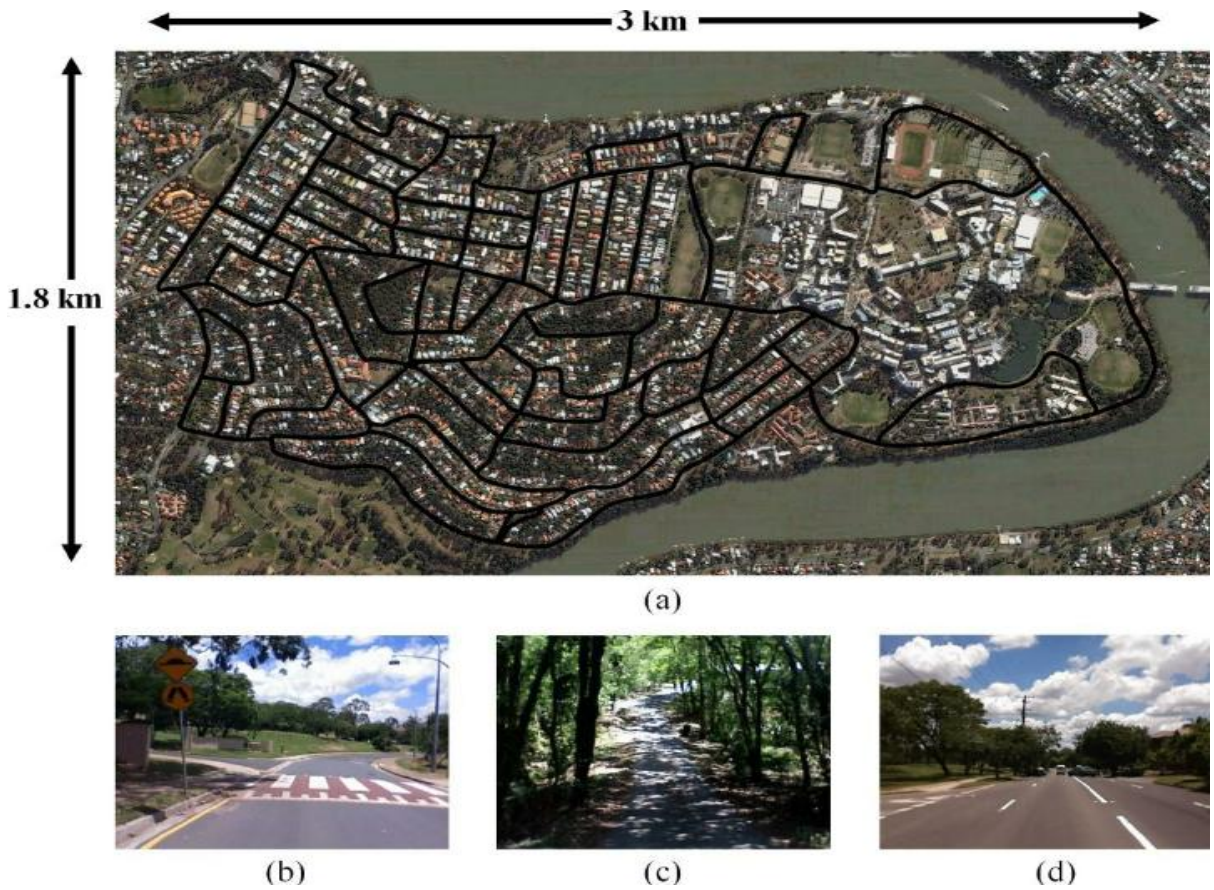

FIGURE 6- 2 : L'ensemble de données St. Lucia 2007 avec les différents d'endroits rencontrés

Réseau de cellules de position

La fig. 6.3 montre une séquence d'activités de cellules de position lors d'un événement de relocalisation typique. À 1138 s, une scène visuelle familière active des cellules de vue locales, qui, à son tour, activent des cellules de pose par des liens entre les positions et la vue locale, ce qui fait apparaître un paquet d'activités concurrentes au-dessus du paquet d'activité existant. Des scènes visuelles plus familières injectent une autre activité, ce qui fait que le nouveau paquet devient légèrement dominant en 1140 s. Le paquet original ne reçoit aucun support visuel supplémentaire, s'éteignant en 1142 s.

Sur une période de 6 s, le système de vision reconnaît des scènes visuelles familières et injecte l'activité dans les cellules de position, créant finalement un nouveau paquet d'activité dominant.

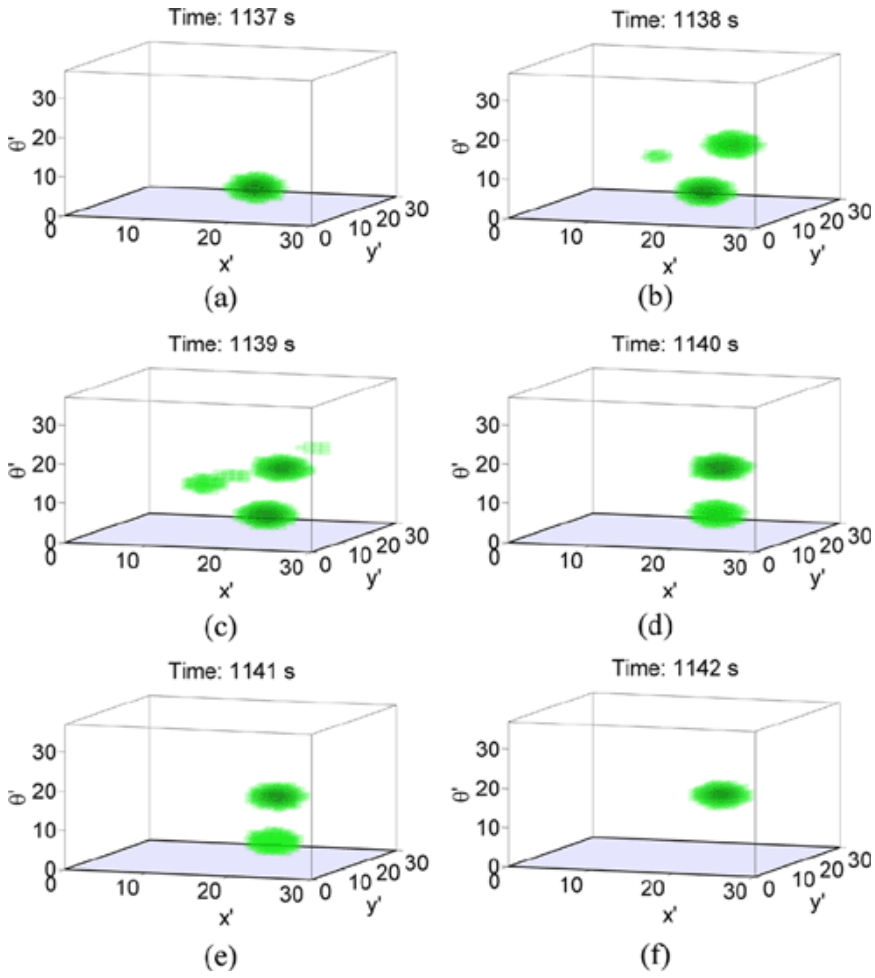

FIGURE 6- 3 : Séquence d'activités de la cellule de position lors d'un événement de relocalisation

Cellules de vue locale

Une cellule de vue locale est active si la vue courante de l'environnement est suffisamment similaire à la vue précédemment vue de l'environnement associé à cette cellule. Dans l'exemple de la fig. 6.4. La cellule de vue (VT2) est activée.

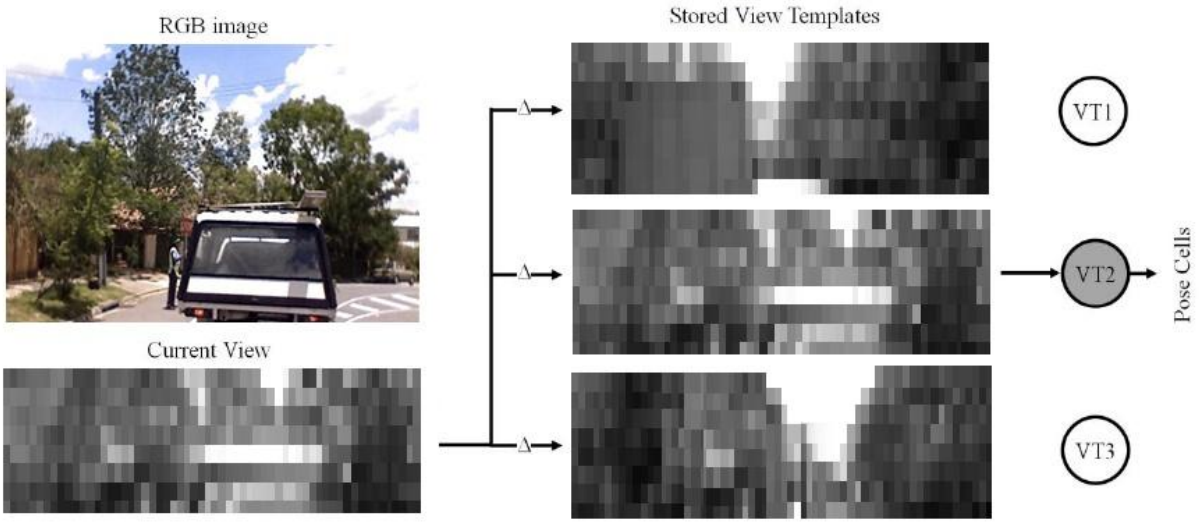

FIGURE 6- 4 : Reconnaissance de place à l'aide de modèle de vue pour StLucia

Carte d'expérience

L'algorithme de correction de carte d'expérience s'exécute seulement après que tous les autres modules de système ont terminé le calcul. La fig. 6.4 montre l'évolution de la carte d'expérience, pour les intervalles d'un quart de l'ensemble de données. La carte finale est représentée dans la fig. 6.4 (d).

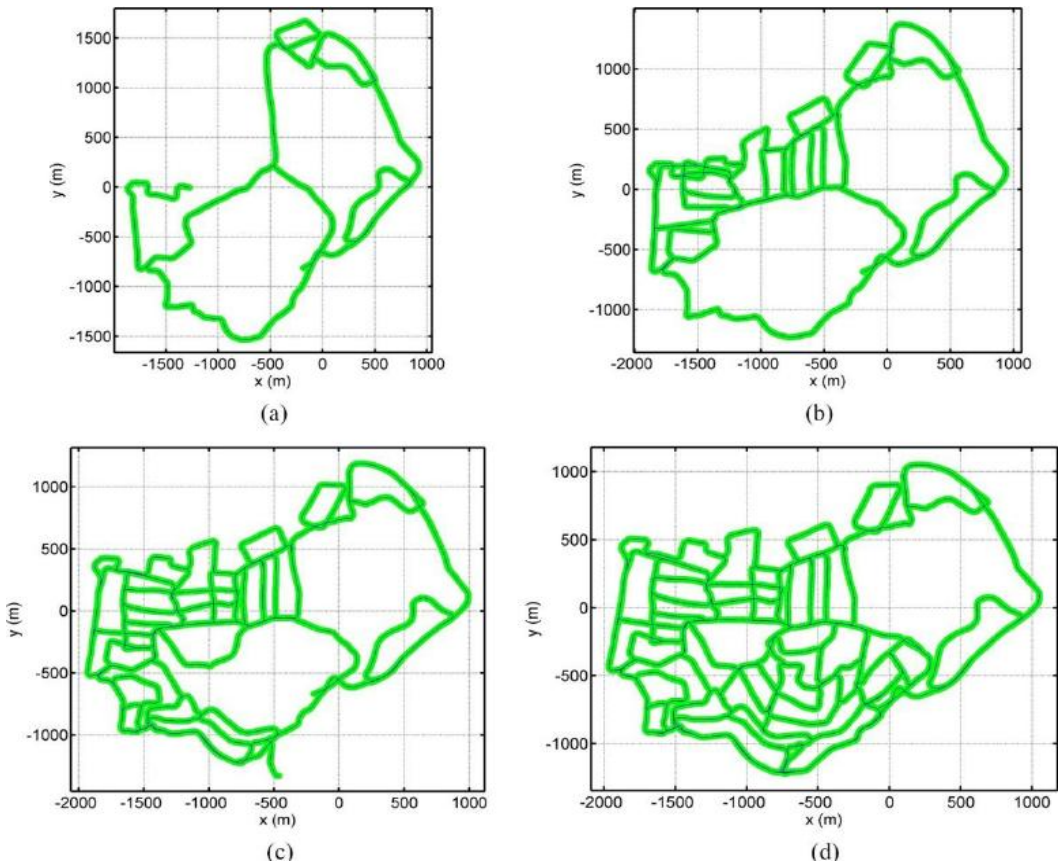

FIGURE 6- 5 : L'évolution de la carte d'expérience de StLucia

6.4.2 Résultat de RatSLAM de iRat

La technologie iRat (intelligent Rat animat technology) est un petit robot mobile de taille et de forme similaires à celle d'un grand rongeur (Fig 6.2 c). Le robot a deux roues motrices différentielles, des haut-parleurs et un microphone, capteurs de proximité infrarouges et encodeurs de roues. Il dispose également d'un ordinateur embarqué 1GHz ×86 256Mo RAM exécutant Ubuntu (Fig 6.2 a-b).

L'iRat a des algorithmes de centre et de mur qui lui permettent d'explorer de façon autonome cet environnement. Pour cet ensemble de données, l'exploration de l'iRat a été guidée par un humain qui a donné des directives sur de quel côté se tourne à chaque intersection.

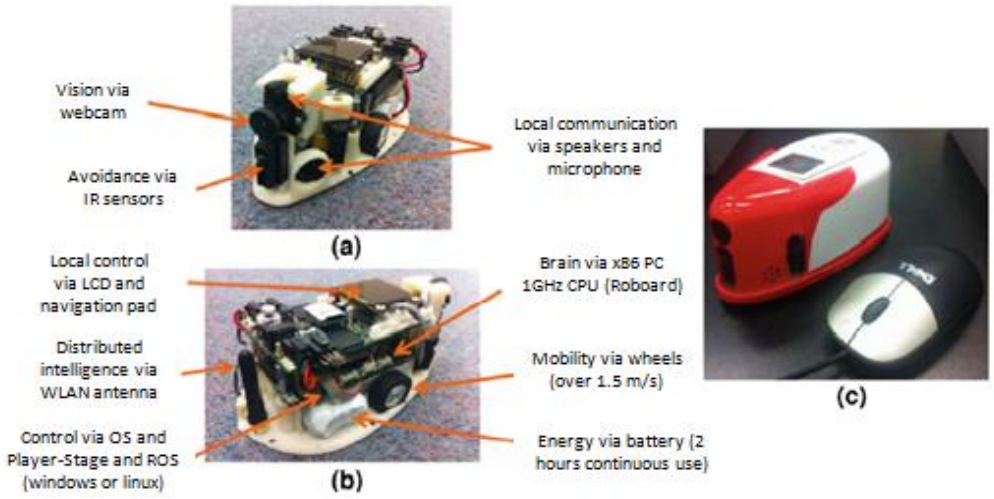

FIGURE 6- 6 : Les différents composants de robot iRat

L'ensemble de données a été obtenu tandis que l'iRat a exploré un film de route de ~16 min basé sur la géographie australienne, contenant des repères australiens proéminents tels que l'Opéra de Sydney et Uluru.

Les figures suivantes montrent l'évolution de RatSLAM pour les données d'iRat. La figure 6-7 (a) représente les cellules de vue locale affichant l'image fournie par la camera de robot et ainsi le modèle de la scène actuelle et leur correspondant et (b) montre la carte d'expérience. Le cube de la figure (c) désigne le réseau de cellules de position et le suivi du mouvement de centre du paquet d'activité dans le plan (x,y) représente par le Vert. La figure (d) représente le mouvement du robot dans l'environnement entier et la dernière figure (e) c'est la carte topologique et la position fournie par l'outil Rvis de ROS.

La flèche rouge montre la position du robot qui correspond à l'emplacement du robot dans l'image (d) et dans la carte d'expérience (b).

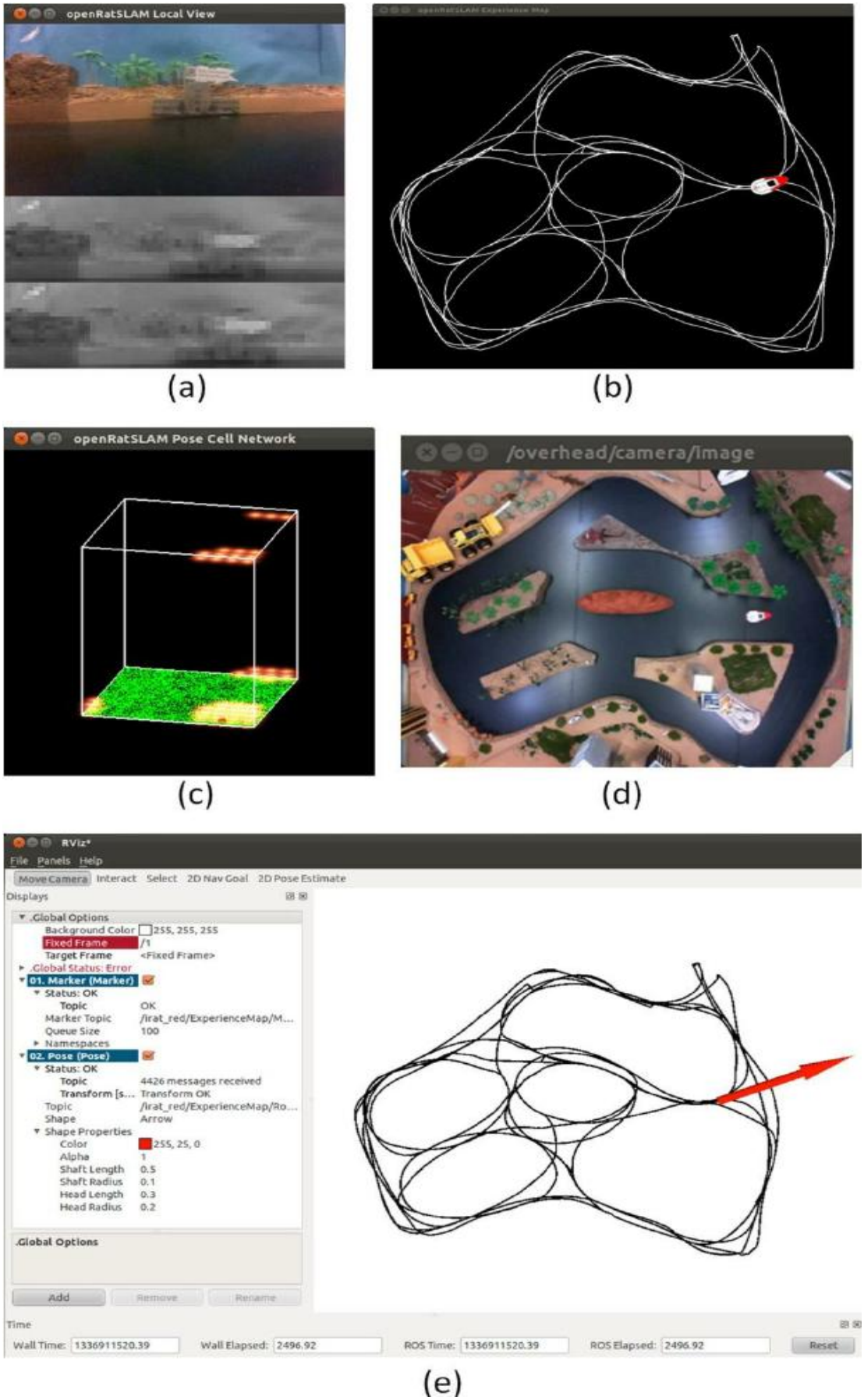

FIGURE 6- 7 : Captures d'écran d'OpenRatSLAM en action

6.4.3 Résultat de notre approche

Pour notre algorithme, nous utiliserons le langage de programmation C++ en se basant sur ROS et la bibliothèque OpenCV et également le logiciel Matlab pour visualiser la trajectoire.

Tout d'abord, nous créons un nœud « /MonoVisoOdo » qui va souscrire au topique « /stlucia/camera/image/compressed » pour recevoir les messages « sensor_msgs::CompressedImage » de fichier bag StLucia, puis nous allons appliquer sur chaque image reçue l'algorithme de l'odométrie visuelle qu'on a déjà expliqué dans le chapitre 5. La figure suivante illustre l'organisation des nœuds.

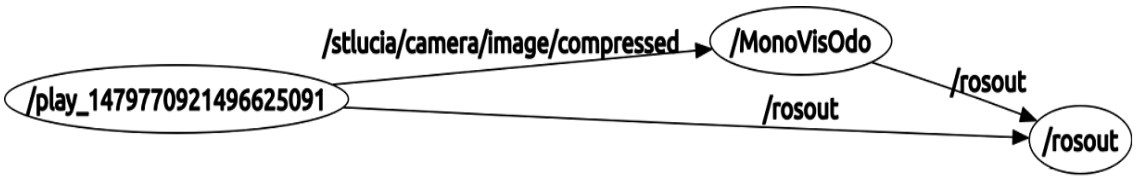

FIGURE 6- 8 : Organisation des nœuds et des topiques de l'application

Détection des points d'intérêt

La figure 6-7 montre la première étape qui consiste à détecter les points d'intérêt dans l'image. Ces points sont représentés en bleu dans la figure.

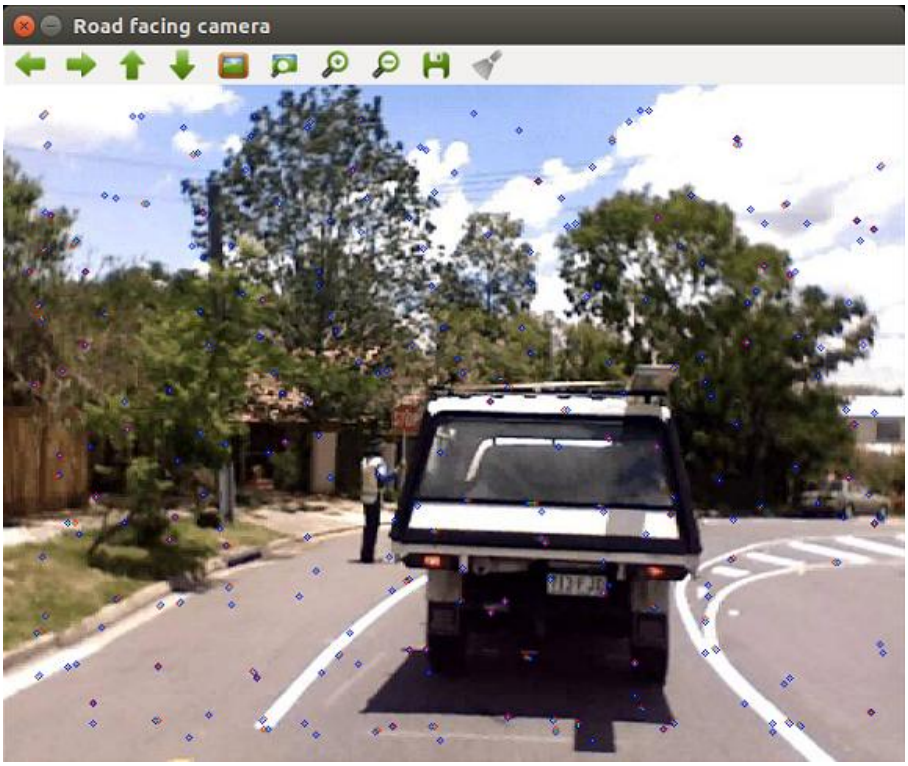

FIGURE 6- 9 : Détection des points d'intérêt dans l'image

La mise en correspondance entre les points d'intérêt

La figure suivante représente les points d'intérêt avec leurs correspondants. Les points en bleu sont les points d'intérêt de l'image acquise à l'instant t-1 et les points rouges sont les points d'intérêt de l'image acquise à l'instant t. Chaque couple de point est relié avec un segment vert comme illustre la figure suivant.

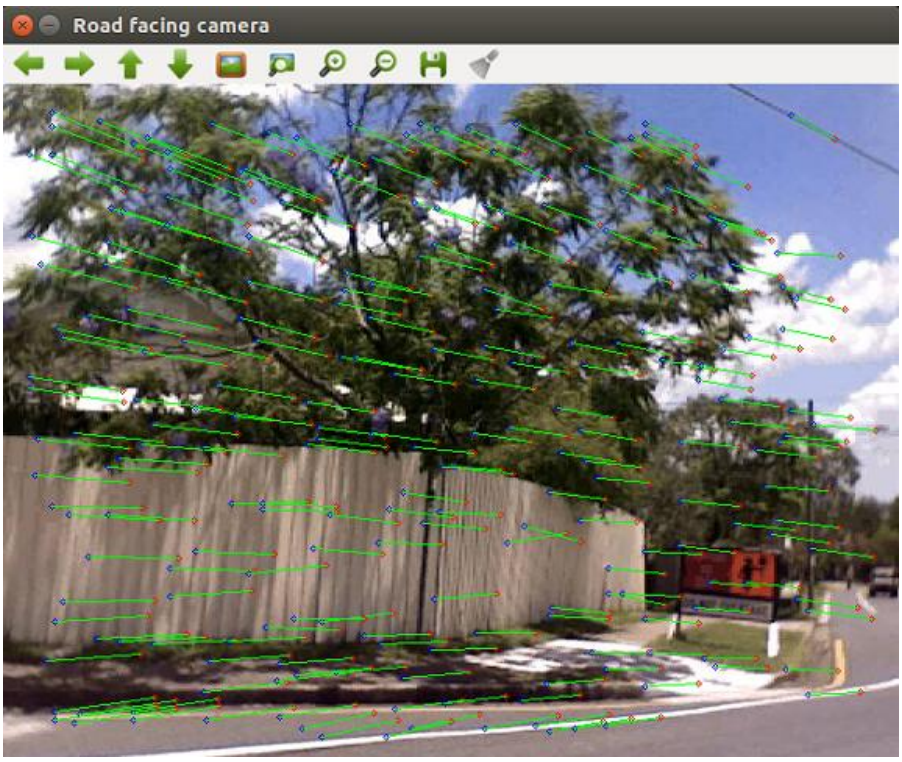

FIGURE 6- 10 : La mise en correspondance entre deux images

Estimation de la trajectoire

Après avoir détecté les points d'intérêt avec leurs correspondants, la matrice essentielle est déterminée à l'aide de ces points en se basant sur les contraintes épipolaires. Ensuite, la transformation entre les images est calculée par la décomposition de la matrice essentielle. Nous effectuons la concaténation de toutes les transformations obtenues pour tracer la trajectoire du robot (cette méthode est déjà détaillée dans le chapitre 5).

6.4.4 Analyse des expériences

Pour faire la comparaison entre le RatSLAM et notre approche basée sur les points d'intérêt visuel, nous prenons 1800 images de fichier bag de stlucia.

Les positions trouvées par notre approche ne sont disponibles qu'à une échelle arbitraire. Notre résultat est transformé par une transformation globale T pour le comparer avec le résultat de RatSLAM.Les trajectoires fournies par les deux algorithmes sont présentées dans la figure suivant :

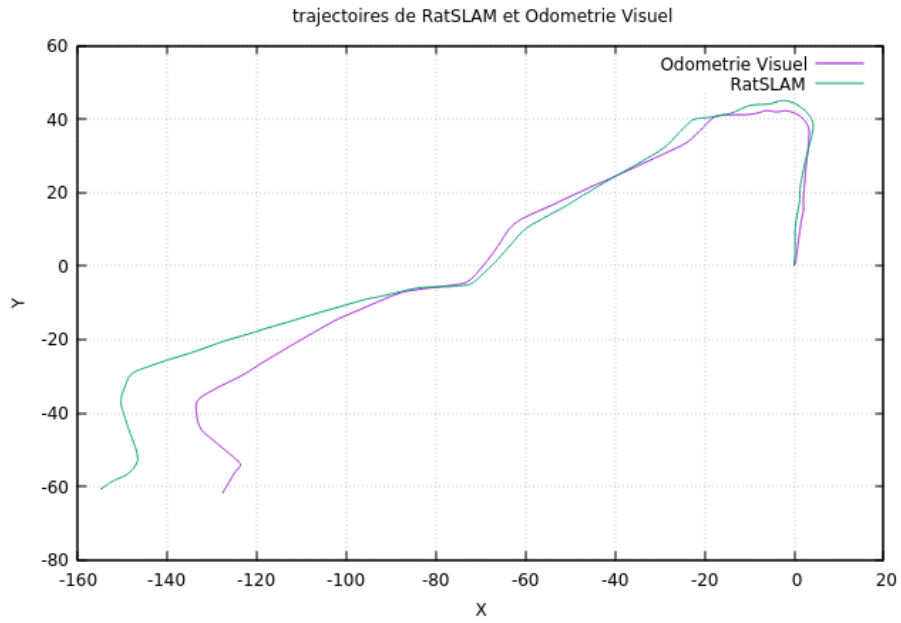

FIGURE 6- 11 : Trajectoires de RatSLAM et de notre approche

La comparaison entre les rotations obtenues pour les deux algorithmes est illustrée dans la figure suivant :

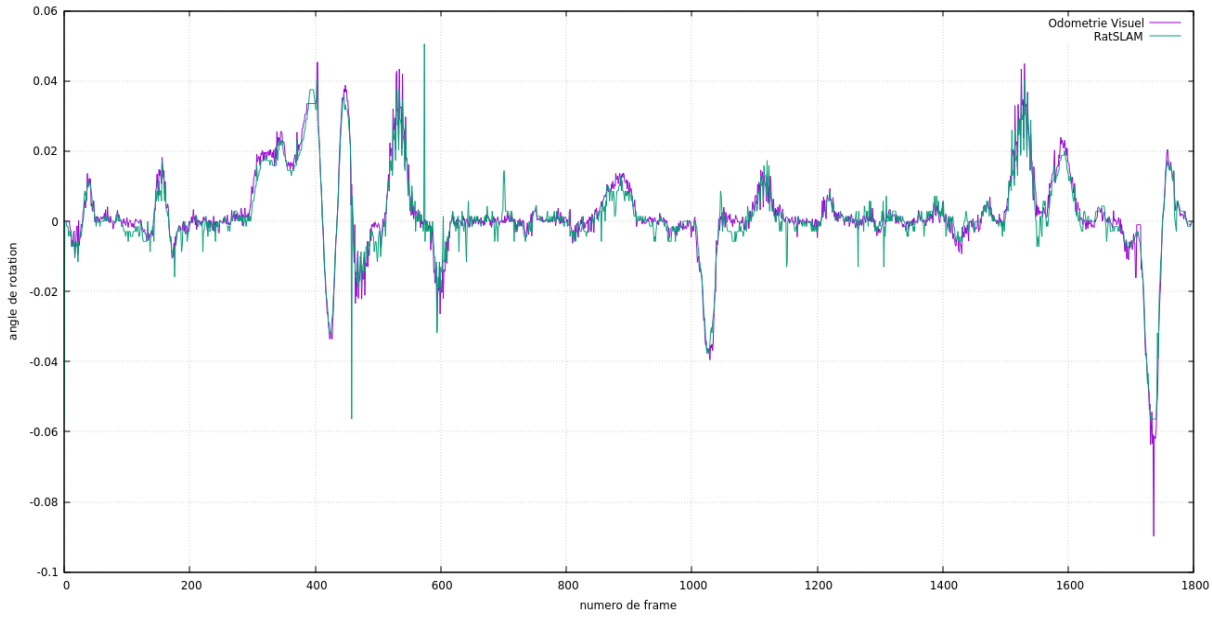

FIGURE 6- 12 : L'orientation en fonction de numéro **de frame**

6.5 Discussions

Pour l'estimation de la rotation entre deux images successive, les deux algorithmes donnent presque le même résultat comme la figure 6-12 montre. Les images extraites ne sont pas bien interprétées par le nœud de l'odométrie visuelle

utilisée dans le RatSLAM. Par contre notre approche est basée sur la détection des caractéristiques visuelles dans les images qui sont les points d'intérêt dans notre cas.

Nous nous sommes basés sur les points d'intérêt car ils sont répétables (c.-à-d, Un grand nombre de points peuvent être redétectées dans les images suivantes), efficaces dans le calcul, robustes au bruit, distinctifs (les points détectés peuvent les suivre avec précision à travers différentes images), et invariants (à la fois aux changements photométriques par exemple l'éclairage et les changements géométriques par exemple la rotation).

Contrairement à l'approche d'odométrie visuelle utilisée dans le RatSLAM, nous avons trouvé des résultats plus comparables aux résultats de l'algorithme RatSLAM comme la figure 6-11 montre.

Conclusion

L'objectif de ce chapitre est de comparer la méthode de l'odométrie visuelle que nous avons proposée avec celle utilisée dans le RatSLAM. Notre programme ne fait pas la mise à jour des positions de robot pour corriger les erreurs ou bien pour fermer des boucles. Ces erreurs influence sur la qualité de la trajectoire trouvée.

L'odométrie visuelle est un bon choix pour estimer la position du robot car il se base seulement sur les informations fournies par le capteur caméra. Mais l'utilisation seulement de l'algorithme de l'odométrie visuelle monoculaire pour récupérer la trajectoire réel de robot n'est pas suffisamment. Pour cette raison il est important d'ajouté des mécanismes capable de corriger ces erreurs et de mettre à jour les positions qui ne sont pas correctes.

Conclusion générale

La navigation autonome d'un robot mobile est actuellement un champ actif de recherche en robotique. De nombreuses approches de navigation ont été développées par des chercheurs dans le but de contrôler les robots pour naviguer avec succès dans un environnement inconnu et non structuré.

Dans ce projet, nous nous sommes concentrés sur l'étude du problème de la localisation et de la cartographie simultanément « SLAM ». Plus précisément la localisation de la position du robot en se basant seulement sur l'utilisation des capteurs de vision.

D'abord, Nous avons cité les bases de la robotique mobile, en commençant par la définition des robots en général, puis en décrivant les deux types des robots qui sont les robots manipulateurs et les robots mobiles. Nous avons cité également les moyens de perception de ces robots tels que les télémètres, les capteurs proprioceptifs et les caméras et finalement les diverses applications de la robotique.

Puis, Nous avons passé en revue l'état de l'art des modèles permettant la navigation autonome d'un robot mobile. Ensuite, nous avons abordé la formulation du problème de SLAM « Simultaneous Localisation And Mapping » qui se base principalement sur l'idée de construire une carte interne détaillée de l'environnement et de l'utiliser pour qu'un robot navigue de manière précise dans cet environnement. Enfin, Nous avons présenté trois algorithmes pour la résolution du SLAM à savoir EKF-SLAM, Fast-SLAM et Graph-SLAM.

Par ailleurs, nous avons exposé les étapes nécessaires pour accomplir la tâche de l'odométrie visuelle qui est le processus de l'estimation de position du robot, en utilisant une ou plusieurs caméras. Nous avons conclu que

L'avantage de l'odométrie visuelle par rapport à l'odométrie de roues est que l'odométrie visuelle n'est pas affectée par le glissement de roue dans un terrain accidenté ou autres conditions défavorables. Nous avons cité également les trois algorithmes d'estimation de mouvement 2D-à-2D, 3D-à-2D et 3D-à-3D.

Enfin, nous avons conclu qu'il existe une autre catégorie des algorithmes inspirés par les systèmes biologiques principalement le cerveau de l'homme et des rongeurs. Ces algorithmes sont capables de résoudre le problème de SLAM ainsi utilisant des méthodes qui sont apparemment robustes, flexibles et bien intégrés dans le robot. Dans le cadre du modèle bio-inspiré, nous avons étudié les types des cellules existantes dans l'hippocampe qui sont responsables sur la navigation chez les rats. Ensuite, nous avons détaillé l'algorithme RatSLAM qui se compose de quatre éléments importants : odométrie visuelle, cellule de vue locale, réseaux de cellule de position et carte d'expérience. Les algorithmes probabilistes montrent leurs limites face à des environnements dynamiques et/ou de grandes tailles par contre les algorithmes bio-inspirés sont adéquats dans les environnements dynamiques.

Après ces études théoriques, nous avons abordé notre approche de l'odométrie visuelle appliquée au RatSLAM, qui récolte tous nos efforts tout au long de ce mémoire. Notre approche est basée sur la détection des points d'intérêt visuel, puis leur comparaison avec l'image précédente. Après la mise en correspondance des deux images, la matrice essentielle est calculée en utilisant la contrainte épipolaire. L'estimation de la translation et de la rotation est faite par la décomposition de la matrice essentielle. Finalement, la trajectoire est obtenue en concaténant toutes les transformations effectuées par le robot. Nous avons également décrit la méthode utilisée dans le RatSLAM pour calculer l'odométrie qui est basée sur la différence entre les profils d'intensité.

Pour conclure, les résultats obtenus par notre approche basée sur les points d'intérêt visuel ont montré de bonne performance comparant à la méthode de l'odométrie visuelle utilisée dans le système RatSLAM.

Nous pouvons envisager des travaux futurs concernant l'amélioration d'algorithme de l'odométrie visuelle, afin de réaliser une meilleure estimation de la trajectoire du robot et ainsi la réalisation d'un robot autonome capable de connaitre son position à chaque instant et de construire la trajectoire dans les différents types d'environnements dynamique, inconnue et large.

Annexe

Installation d'OpenCV

La bibliothèque OpenCV « Open source Computer Vision Library » est une bibliothèque open source de vision par ordinateur et Machine Learning. Elle est disponible pour les différents systèmes d'exploitation : Windows, Linux, Mac OS, iOS et Android.

Les paquets nécessaires

Plusieurs paquets nécessaires avant de commencer l'installation d'OpenCV comme par exemple les paquets qui donnent à l'OpenCV la capacité de travailler sur les différents formats de fichiers d'image, et également les paquets pour accéder à la caméra et lire des vidéos, etc. Ces paquets peuvent être installés à l'aide d'un terminal et les commandes suivantes :

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential cmake pkg-config
$ sudo apt-get install libjpeg8-dev libtiff5-dev libjasper-dev libpng12-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
$ sudo apt-get install libgtk-3-dev
$ sudo apt-get install libatlas-base-dev gfortran
$ sudo apt-get install python2.7-dev python3.5-dev
```

Téléchargement d'OpenCV

Il existe deux façon pour télécharger la bibliotéque OpenCV .La première est de télécharger OpenCV depuis http://opencv.org/downloads.html et décompresser. La deuxième est d'utiliser les commandes suivantes :

```
$ cd ~
$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip
$ unzip opencv.zip
```

Le nom de fichier 3.1.0.zip désigne la version d'OpenCV (version 3.1.0 dans notre cas).

Nous avons aussi besoin de télécharger le package opencv_contrib pour accéder aux fonctionnalités telles que SIFT, SURF et ORB …

```
$ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip
$ unzip opencv_contrib.zip
```

Configuration d'OpenCV

Nous allons créer un répertoire temporaire, que nous appelons « build » pour placer les Makefiles générés, les fichiers de projet ainsi que les fichiers d'objets et les binaires de sortie. Nous utilisons les commandes suivantes :

```
$ cd ~/opencv-3.1.0/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
        -D CMAKE_INSTALL_PREFIX=/usr/local \
        -D INSTALL_PYTHON_EXAMPLES=ON \
        -D INSTALL_C_EXAMPLES=ON \
        -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.1.0/modules \
        -D WITH_OPENGL=ON \
        -D BUILD_EXAMPLES=ON ..
```

Après la terminaison de CMake sans aucune erreur, nous pouvons maintenant compiler OpenCV, nous allons entrer dans le répertoire créé (build dans notre cas) et puis nous exécutons les commandes suivants :

```
$ make
$ sudo make install
```

Installation d'OpenRatSLAM

OpenRatSLAM est une implémentation open source de l'algorithme RatSLAM. Nous allons décrire dans cette partie les étapes de l'installation d'OpenRatSLAM.

Installation des dépendances

OpenRatSLAM dépend de ROS : opencv2 et topological_nav_msgs et aussi la bibliothèque graphique 3D Irrlicht. Irrlicht peut être installé sous Ubuntu par la commande suivant :

```
$ sudo apt-get install libirrlicht-dev
```

Construction d'OpenRatSLAM

Le code pour télécharger OpenRatSLAM est disponible depuis :

https://storage.googleapis.com/google-code-archive-source/v2/code.google.com/ratslam/source-archive.zip

Pour configurer les variables d'environnement ROS, nous tapons :

```
$ . /opt/ros/fuerte/setup.sh
```

Le répertoire OpenRatSLAM doit être ajouté à la variable d'environnement ROS_PACKAGE_PATH.

```
$ export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:/path/to/OpenRatSLAM
```

Ensuite, nous construisons OpenRatSLAM par la commande :

```
$ rosmake
```

Exécution d'OpenRatSLAM

Il existe trois fichiers bag pour tester l'algorithme OpenRatSLAM:

- iRat 2011 in Australia
- Car in St Lucia 2007
- Oxford New College 2008 dataset

Ces trois fichiers sont disponibles en ce site :

https://wiki.qut.edu.au/display/cyphy/OpenRatSLAM+datasets.

Nous plaçons le fichier dans le répertoire OpenRatSLAM. Puis nous exécutons le RatSLAM en tapant dans deux terminaux différents soit :

```
$ roslaunch irataus.launch

$ rosbag play irat_aus_28112011.bag
```

Ou :

```
$ roslaunch stlucia.launch

$ rosbag play stlucia_2007.bag
```

Ou:

```
$ roslaunch oxford_newcollege.launch

$ rosbag play oxford_newcollege.bag
```

La carte créée par OpenRatSLAM sera périodiquement publiée sur rviz. Pour exécuter rviz :

```
$ rosrun rviz rviz
```

Bibliographies

[1] Houcine Zerfa. Conception, Réalisation et Commande Floue d'un Robot Mobile. Mémoire de magister de l'université des sciences et de la technologie d'Oran. Algérie 2013.

[2] Bilgic. T and I. Turksen. "Model based localization for an autonomous mobile robot". Proceedings of the IEEE Conference on Systems, Man and Cybernetics, pp. 6, University of Toronto, Canada, 1995.

[3] David Filliat,. "Robotique Mobile. Engineering school. Robotique Mobile , ENSTA ParisTech, pp.175, 2011.

[4] Anthony Mallet. Localisation d'un robot mobile autonome en environnements naturels. Automatique / Robotique. Institut National Polytechnique de Toulouse - INPT, 2001.

[5] Andersen, J. C., Ravn, O., & Andersen, N. A. "Mobile Robot Navigation". 2007.

[6] Khusheef, A. S. "Investigation on the mobile robot navigation in an unknown environment", 2013. http://ro.ecu.edu.au/theses/537

[7] Y. Zhu, T. Zhang, J. Song, and X. Li, "A New Bug-type Navigation Algorithm Considering Practical Implementation Issues for Mobile Robots," in proceedings of the 2010 IEEE International Conference on Robotics and Biomimetics, pp. 531-536, Tianjin, China, 14-18 December, 2010.

[8] Oussama El Hamzaoui. Localisation et cartographie simultanées pour un robot mobile équipé d'un laser à balayage : CoreSLAM. Autre [cs.OH]. Ecole Nationale Supérieure des Mines de Paris, 2012.

[9] E. Rosten and T. Drummond, “Machine learning for high speed corner detection”, in 9th Euproean Conference on Computer Vision, vol. 1, pp. 430–443. Department of Engineering, Cambridge University, UK, 2006

[10] Christoph Strecha, Vincent Lepetit, Tomasz Trzcinski, Michael Calonder, Mustafa Özuysal, Pascal Fua, "BRIEF: Computing a Local Binary Descriptor Very Fast", IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 34, no. , pp. 1281-1298, July 2012.

[11] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: binary robust independent elementary features,” in Computer Vision—ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5–11, 2010, Proceedings, Part IV, vol. 6314 of Lecture Notes in Computer Science, pp. 778–792, Springer, Berlin, Germany, 2010.

[12] Yu Lei, Zhixin Yu, Yan Gong, “An Improved ORB Algorithm of Extracting and Matching Features”, International Journal of Signal Processing, Image Processing and Pattern Recognition, Vol. 8, No. 5, pp. 117-126, 2015.

[13] Scaramuzza, D., Fraundorfer, F., Visual Odometry: Part I - The First 30 Years and Fundamentals, IEEE Robotics and Automation Magazine, Volume 18, issue 4, 2011.

[14] Milford MJ, Wyeth G, Prasser D. “RatSLAM: a hippocampal model for simultaneous localization and mapping”. In IEEE International. Conference. Robotics and Automation, New Orleans, USA. IEEE 2004.

[15] O’Keefe, J. and Nadel, L. The hippocampus as a cognitive map / John O’Keefe and Lynn Nadel. Clarendon Press ; Oxford University Press Oxford : New York, 1978.

[16] Milford, M. J. Robot navigation from nature: Simultaneous localisation, mapping, and path planning based on hippocampal models (Vol. 41). Berlin: Springer, 2008.

[17] Milford, M., & Wyeth, G. Persistent navigation and mapping using a biologically inspired SLAM system. International Journal of Robotics Research, 29, 1131–1153, 2010.

[18] Nafouki, C., & Conradt, J. Spatial Navigation Algorithms for Autonomous Robotics.

[19] Milford, M., &Wyeth, G. Mapping a suburb with a single camera using a biologically inspired SLAM system. IEEE Transactions on Robotics, 24, 1038–1053, 2008.

[20] Fraundorfer, F., Scaramuzza, D., Visual Odometry: Part II - Matching, Robustness, and Applications, IEEE Robotics and Automation Magazine, Volume 19, issue 1, 2012.

[21] H. Longuet-Higgins, “A computer algorithm for reconstructing a scene from two projections,” Nature, vol. 293, no. 10, pp. 133–135, 1981.

[22] D. Nister, “An efficient solution to the five-point relative pose problem”, Proc. CVPR03, pp. II: 195–202, 2003.

[23] D. Ball, S. Heath, J. Wiles, G. Wyeth, P. Corke, and M. Milford, "OpenRatSLAM: an open source brain-based SLAM system", Autonomous Robots, 2013.