Titre: Title:	Navigation globale d'un fauteuil roulant motorisé dans de grands espaces intérieurs
Auteur: Author:	Anas El Fathi
Date:	2012
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	El Fathi, A. (2012). Navigation globale d'un fauteuil roulant motorisé dans de grands espaces intérieurs [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/999/

UNIVERSITÉ DE MONTRÉAL

NAVIGATION GLOBALE D'UN FAUTEUIL ROULANT MOTORISÉ DANS DE GRANDS ESPACES INTÉRIEURS

ANAS EL FATHI
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
DÉCEMBRE 2012

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

NAVIGATION GLOBALE D'UN FAUTEUIL ROULANT MOTORISÉ DANS DE GRANDS ESPACES INTÉRIEURS

présenté par : EL FATHI, Anas
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury d'examen constitué de :

M. SAUSSIÉ David, Ph.D., président
M. GOURDEAU Richard, Ph.D., membre et directeur de recherche
Mme. PINEAU Joelle, Ph.D., membre

À ma mère Saida, mon père Mohamed,
mes sœurs Nada et Heda
et tous mes chers amis.

REMERCIEMENTS

Je commence ce mémoire par remercier grandement tous ceux qui ont partagé avec moi leur temps et leur savoir, m'ont soutenu au long de mon projet de recherche, et ont gravé dans mon esprit tant de moments profitables et agréables que ce soit humainement ou professionnellement.

À mon professeur et directeur de recherche, Richard Gourdeau, je souhaite témoigner de ma gratitude pour m'avoir accueilli dans son équipe et m'avoir proposé ce projet fort intéressant pour ses répercussions scientifiques et très enthousiasmant pour ses applications sociales. Richard, merci pour votre soutien et votre présence, dès le début de ma maîtrise jusqu'au dernier jour.

À mes collègues de laboratoire Hai Nguyen, Xavier Savaris, Beomjoon Kim et Andrew Sutcliffe, j'exprime ma profonde reconnaissance pour leur assistance et leur aide précieuse. Notamment, Hai, merci énormément pour les heures passées à m'assister pendant nos expérimentations et pour tous ces soins donnés au fauteuil roulant en cas de problèmes.

À mes amis d'études à l'École Polytechnique, merci d'être toujours à mes côtés. Vous allez me manquer. Bonne chance à vous tous, en espérant vous revoir souvent.

Je remercie ma responsable de recherche à Supaero, Caroline Bérard, la responsable des échanges internationaux de l'école, Françoise Loytier, ainsi que tous mes collègues et amis à Toulouse, qui me manquent énormément.

À tout le personnel du département de génie électrique, section Automation et Systèmes, merci pour votre présence et votre assistance en cas de besoin.

Je remercie tous les organismes qui ont rendu ce projet possible, notamment le Conseil de Recherches en Sciences Naturelles et en Génies (CRSNG) et le regroupement INTER.

À tous mes chers amis qui m'ont vraiment soutenu au long de ces deux années, je vous remercie énormément.

À mes sœurs Nada et Heda, et mes parents Mohamed et Saida, merci pour vos encouragements constants et pour votre présence bienveillante.

RÉSUMÉ

Dans ce mémoire nous présentons un nouveau module de navigation conçu pour un fauteuil roulant motorisé. Ce module a pour objectif de fournir des fonctionnalités essentielles à l'ensemble des personnes à mobilité réduite utilisant ces fauteuils.

Différents services ont été prévus lors de la conception de ce module, tels que l'assistance à l'évitement d'obstacle, l'exécution de manœuvres automatiques (mouvements rectilignes, suivi de personnes, traversée de passages étroits et stationnement) et la génération de carte de l'environnement, le tout en respectant les modalités de sécurité et convivialité. De plus, il a été conçu et développé sous une plateforme collaborative de développement se distinguant par sa facilité d'intégration et sa modularité.

Même si nous nous limitons sur un fauteuil roulant, ce travail contribue au domaine de la robotique mobile avec l'architecture de contrôle proposée, cette architecture englobant : un superviseur, un mode manuel, un mode semi-manuel et un mode automne. Ainsi, il propose des solutions au problème de navigation globale en fournissant une méthode de construction de carte basée sur différents capteurs, et une méthode de localisation globale permettant de se retrouver sur une carte.

Le prototype développé se caractérise par la multitude de capteurs utilisés : télémètre laser, télémètre ultrason et caméra Kinect. Nous allons présenter un travail de synthèse permettant d'identifier parmi ces capteurs, ceux nécessaires pour assurer les performances recherchées dans un tel module de navigation.

Enfin, l'ensemble des expérimentations que nous avons effectuées, incluant des cartographies de grands espaces, démontre les performances de notre module de navigation, ainsi que ses limites.

ABSTRACT

In this thesis we present a new navigation system designed for powered wheelchairs. This system aims at providing essential functionalities to disabled people using wheelchairs.

Different services were planned during the design of the navigation module, such as collision avoidance, execution of automatic maneuvers (straight movements, following a person, passing through narrow passage and parking) but also generating maps of the environment. All these actions must comply with the terms of safety and security. In addition, it has been designed and developed in a collaborative operating system characterized by its ease of integration and modularity.

Even if we do limit our investigation to motorized wheelchairs, our new control architecture is a contribution to the field of mobile robotics with the proposed control architecture. This architecture includes: a supervisor, a manual mode, a semi-manual mode and an automatic mode. Thus, it offers solutions to the global navigation problem by providing a mapping method based on various sensors, and a global localisation method to localise on a map.

The prototype developed is characterized by a multitude of sensors: laser rangefinder, ultrasound rangefinder and Kinect camera. We will perform a synthesis to identify among these sensors, those necessary to ensure the desired performance in such a navigation module.

Finally, the set of experiments we carried out, including mapping of large interior spaces, demonstrates the performance of our navigation module and its limits.

TABLE DES MATIÈRES

DÉDICACE iii

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT vi

TABLE DES MATIÈRES vii

LISTE DES TABLEAUX x

LISTE DES FIGURES xi

LISTE DES ANNEXES xiii

LISTE DES SIGLES ET ABRÉVIATIONS xiv

CHAPITRE 1 INTRODUCTION 1
- 1.1 Définitions et concepts de base 1
- 1.2 Éléments de la problématique 2
- 1.3 Objectifs de recherche 3
- 1.4 Plan du mémoire 4

CHAPITRE 2 REVUE DE LITTÉRATURE 5
- 2.1 Introduction 5
- 2.2 Navigation locale 6
  - 2.2.1 Méthode basée sur les champs potentiels 6
  - 2.2.2 Méthode basée sur les histogrammes de champ de vecteur 8
  - 2.2.3 Méthode des fenêtres dynamiques 8
- 2.3 Navigation globale 10
- 2.4 Localisation globale dans une carte 11
- 2.5 Cartographie 12

CHAPITRE 3 DESCRIPTION ET PRÉSENTATION DU FRMI 17
- 3.1 ROS : Robot Operating System 17

3.2 Composantes du FRMI, présentation sous ROS 18
3.2.1 La forme brute du FRMI 18
3.2.2 Les rajouts matériels : capteur et interface 20
3.2.3 Le FRMI sous ROS 22
3.3 Calibration des capteurs laser 24
3.4 Intégration des sonars 30
3.5 Intégration de la caméra Kinect 31
3.6 Génération des données laser et des nuages de points 32
3.6.1 Filtrage suivant la trace du FRMI 34
3.6.2 Autres filtres 36
3.7 Conclusion 36

CHAPITRE 4 ARCHITECTURE DE CONTRÔLE DU FRMI 37
4.1 Architecture de contrôle globale 37
4.1.1 Le superviseur 38
4.1.2 Le mode manuel 40
4.1.3 Le mode semi-manuel 44
4.1.4 Le mode automatique 45
4.2 Module d'évitement d'obstacles 45
4.2.1 Méthode de la fenêtre dynamique 48
4.2.2 Navigation assistée 48
4.2.3 Suivi d'une trajectoire 50
4.2.4 Suivi d'une commande haut niveau 50
4.3 Module de cartographie 50
4.3.1 Fusion avec une carte existante 51
4.4 Conclusion 53

CHAPITRE 5 RÉSULTATS ET VALIDATION EXPÉRIMENTALE 56
5.1 Analyse et validation de la cartographie 56
5.1.1 Environnement contrôlé 56
5.1.2 Environnement non contrôlé 66
5.2 Analyse et validation du module de navigation locale 69
5.3 Conclusion 72

CHAPITRE 6 CONCLUSION 73
6.1 Synthèse des travaux 73
6.2 Limitations de la solution proposée 75

6.3 Améliorations futures 75

RÉFÉRENCES 76

ANNEXES 80
A.1 Cartographie 80
A.2 Fusion avec une carte 81
B.1 Localisation globale 84
B.2 Navigation globale 84

LISTE DES TABLEAUX

4.1 Forme du service fourni par le superviseur 39
4.2 Configuration de la navigation assistée 49
5.1 Synthèse des différents tests de cartographie dans un environnement contrôlé 65

LISTE DES FIGURES

2.1 Simulation de la méthode des champs potentiels 7
2.2 Simulation de la méthode des fenêtres dynamiques 9
2.3 Fonction du coût de la méthode des fenêtres dynamiques 9
2.4 Simulation de la planification de trajectoire avec la méthode des fenêtres dynamique 10
2.5 Simulation de l'algorithme MCL 13
2.6 Tracé de la position réelle et la position estimée par itération 14
3.1 Présentation de la communication entre nœuds dans le mode manuel . 19
3.2 Présentation du FRMI, les rajouts matériels en rouge 21
3.3 Arbre de joints présentant l'URDF du FRMI 23
3.4 URDF du FRMI 25
3.5 Calibration de laser pour le FRMI 26
3.6 Résultat de la calibration des trois télémètres lasers, respectivement : à gauche, à l'arrière et à droite 27
3.7 Résultat de la simulation de la calibration des trois télémètres lasers . . 29
3.8 Modèle de propagation d'une onde ultra sonore 30
3.9 Exemple d'une acquisition de données de télémètre ultrason en blanc, superposée sur ceux du laser en bleu 31
3.10 Mesure de nuage de points fourni par la caméra Kinect 32
3.11 Résultats de la fusion des données laser 35
3.12 Trace au sol du FRMI 35
4.1 Présentation de l'architecture de contrôle du module de navigation pour FRMI 38
4.2 Présentation du nœud superviseur 39
4.3 Présentation des roues folles 41
4.4 Schéma du contrôleur PID 42
4.5 Résultat de la simulation de la commande PID de la vitesse 43
4.6 Navigation globale avec MCL 46
4.7 Présentation du module d'évitement d'obstacles 47
4.8 Simulation du générateur des trajectoires sous MATLAB avec une vitesse linéaire de $0.7m/s$ et une vitesse angulaire de $0.5rad/s$ 49
4.9 Exemple d'une carte obtenue par cartographie à gauche, une carte fournie à droite 52

4.10 Résultat de la transformée de Hough pour détection de ligne 53
4.11 Orientation de la carte avec transformée de Hough 54
4.12 Résultat de la fusion des deux cartes 55
5.1 Présentation du circuit pour cartographie : schéma détaillé 57
5.2 Présentation du circuit pour cartographie : vue générale 58
5.3 Résultat de la cartographie avec télémètre laser, de droite à gauche, du haut en bas, la résolution est égale à 1, 5 et 10 degrés 59
5.4 Résultat de la cartographie avec télémètre ultrason seul 60
5.5 Résultat de la cartographie avec la caméra Kinect seule 61
5.6 Résultat de la cartographie avec télémètres ultrason et laser 62
5.7 Résultat de la cartographie avec combinaison de télémètre laser et camera Kinect de droite à gauche, d'en haut en bas, la résolution du laser change de 1, 5 à 10 degrés 63
5.8 Résultat de la cartographie avec une fusion de tous les capteurs 64
5.9 Cartographie du premier étage du bâtiment Lassonde avec une fusion de donnée : télémètre laser (10%), ultrason et caméra Kinect 67
5.10 Cartographie du premier étage du bâtiment Lassonde avec télémètre laser seulement 68
5.11 Trajectoire utilisée pour l'expérimentation du suivi de chemin 70
5.12 Suivi de la trajectoire : instant de traversé de porte 70
5.13 Résultat du suivi de trajectoire 71
A.1 État des 50 particules (en rouge) à différents instants de la cartographie 82
A.2 Carte obtenue par cartographie (à gauche) et une carte fournie (à droite) 82
A.3 Résultat de la fusion des deux cartes 83
B.1 Test de l'algorithme MCL 85
B.2 Navigation globale 1 86
B.3 Navigation globale 2 87
B.4 Navigation globale 3 88

LISTE DES ANNEXES

Annexe A Cartographie et fusion de carte au premier étage de Lassonde 80
Annexe B Résultat d'autolocalisation et navigation point à point 84

LISTE DES SIGLES ET ABRÉVIATIONS

FRMI	Fauteuil Roulant Motorisé Intelligent
FRM	Fauteuil Roulant Motorisé
ROS	Robot Operating System (système d'exploitation pour robot)
SLAM	Simultaneous Localization And Mapping (cartographie et localisation simultanées)
CAO	Conception Assistée par Ordinateur
dt	Èchantillon du temps
x_k	Position du robot la plus récente à l'instant kdt
m_k	Carte la plus récente à l'instant kdt
u_{k-1}	Données de l'odométrie les plus récents à l'instant $(k-1)\,dt$
z_k	Mesures du capteur laser les plus récents à l'instant kdt
URDF	Unified Robot Description Format (format unifié pour description de robot)
ϕ	Orientation des roues folles en avant du FRMI

CHAPITRE 1

INTRODUCTION

Par nature, les êtres humains sont imparfaits ; ils doivent dormir, manger, ils vieillissent et ils tombent malades. Dans leur bataille pour combler leurs imperfections, ils ont dû inventer et créer des méthodes pour ne plus sentir ces faiblesses. Ainsi, depuis quelques décennies, la société commence à être envahie par de petits systèmes électriques, doués d'une intelligence unique, appelés robots.

Dans notre cas particulier, nous nous concentrons sur l'amélioration de la vie des personnes à mobilité réduite utilisant un fauteuil roulant, et ce en rajoutant un aspect autonome et intelligent à ces fauteuils. En effet, cette ancienne invention a connu de nombreuses évolutions depuis la fin de la Deuxième Guerre mondiale. En 1950 le premier fauteuil roulant motorisé a vu le jour avec « Everest & Jennings ». Ensuite, dans les années 1980, avec la maturité de la théorie de l'automatique, plusieurs recherches sur les stratégies de contrôle et l'interaction personne-machine commencèrent à s'élaborer. Actuellement, des études très sérieuses dans des universités renommées sont menées pour assister ces personnes (Simpson, 2005).

1.1 Définitions et concepts de base

Un Fauteuil Roulant Motorisé Intelligent (FRMI) est en réalité un fauteuil roulant électrique, similaire à ceux que l'on retrouve sur le marché actuel, sur lequel est ajouté un module de navigation semi-automne, et un module d'interaction personne-machine. Le but est d'apporter des fonctionnalités automatiques permettant d'effectuer un ensemble de tâches prédéfinies, dont l'ultime objectif d'assister complètement l'usager dans sa vie quotidienne.

À titre d'exemple, le FRMI devrait permettre à l'utilisateur d'éviter de frapper les obstacles qui sont aux alentours ; ceci devrait se faire de façon continue et assez lisse pour que la personne concernée ne sente pas la manœuvre d'évitement. Aussi, l'usager devrait pouvoir exécuter des manœuvres complexes de type passage de porte ou suivi d'une personne. Le FRMI que nous développons se distingue par un module de navigation robuste qui permet de créer et sauvegarder une carte de l'environnement et donc d'atteindre automatiquement des destinations visitées auparavant.

Pour atteindre ce genre de performance, une bonne connaissance de l'environnement et une souplesse de navigation sont cruciales. Ceci est assuré par l'ensemble des capteurs et actionneurs associés au FRMI, mais aussi par les programmes qui s'exécutent en permanence

dans l'ordinateur embarqué. Le prototype actuel dispose de trois télémètres laser, six capteurs ultrasons, et une caméra Kinect pour la perception de l'environnent, d'encodeurs optiques pour le contrôle des roues motrices, d'un contrôleur pour la navigation manuelle assistée, et d'un écran tactile pour l'interaction avec le FRMI. L'ordinateur embarqué fonctionne sous Linux et les programmes sont développés au sein du Robot Operating System (ROS), une surcouche qui assure la communication entre les diverses composantes du système et la gestion du module de navigation.

1.2 Éléments de la problématique

Dans le domaine des robots mobiles, le problème de navigation englobe et couvre un large spectre de systèmes, présente plusieurs contraintes et propose plusieurs solutions basées sur des méthodes plus au moins récentes. En fait, d'une façon simpliste, ce problème se résume aux trois questions suivantes :

- Où sommes-nous ?
- Où allons-nous ?
- Comment nous y rendre ?

Un système permettant de répondre à ces questions, à chaque instant et dans différentes situations, est un système doué d'une intelligence suffisante pour assurer la navigation globale et locale dans de nouveaux environnements. Nous allons donc présenter des éléments de réponses pour ces questions pour un robot mobile en général et pour le FRMI en particulier.

Pour répondre à la première question, il suffit de connaître la position du FRMI dans son milieu. Pour y arriver le FRMI doit être en mesure de percevoir son environnement local, acquérir et filtrer l'information utile, et enfin, estimer sa position en croisant cette information avec une base de données qu'il a en mémoire. En pratique, ceci peut se traduire de deux façons :

- Le FRMI commence d'un point connu puis il garde en mémoire la trace de sa navigation en se basant sur des informations qui lui sont propres.
- Le FRMI dispose d'une carte *a priori* où il peut chercher sa position en se basant sur les informations qui l'entourent.

Ces deux approches sont connues sous le nom de localisation locale et globale. Un robot mobile doit maîtriser et combiner ces deux approches. Par ailleurs, pour pouvoir assurer une telle localisation le FRMI doit disposer de capteurs extéroceptifs, par exemple, sonars, infrarouges, télémètres laser ou caméras, ainsi qu'un ensemble de capteurs proprioceptifs comme des encodeurs optiques sur les roues motrices, ou des accéléromètres et gyromètres.

Pour la deuxième question, c'est l'usager qui détermine la réponse. Cependant, interpréter un ordre de l'usager par une information utile pour le module de navigation, suppose la présence d'un moyen d'interaction avec le FRMI, soit en spécifiant la commande de vitesse angulaire et linéaire directement à l'aide d'un contrôleur [1], ou par des consignes de haut niveau du genre : je veux aller à un local. Ces consignes peuvent être assignées par reconnaissance vocale, au travers de l'écran tactile ou toute autre interface appropriée.

La troisième question soulève les problèmes de la planification et la décision (Neumann et Morgenstern, 1944), le FRMI doit avoir des niveaux d'autonomie lui permettant de prendre des décisions lorsque c'est nécessaire, et laisser l'usager agir lorsqu'il en a besoin. Par contre, si le robot mobile doit naviguer seul sans intervention humaine, il doit être en mesure d'interpréter son environnement pour planifier son chemin.

Maintenant que nous avons un aperçu du problème de navigation, nous pouvons explorer d'autres problèmes liés à notre FRMI. Nous rappelons que le FRMI doit pouvoir naviguer dans des espaces intérieurs assez larges, caractérisés par des obstacles fixes et mobiles, de différentes structures et tailles. Dans ces circonstances assez complexes, quelle stratégie devons-nous adopter pour fournir un module de navigation au FRMI ? En fait, lorsque nous parlons de stratégie nous évoquons essentiellement les deux questions suivantes :

- Quels sont les capteurs matériels nécessaires pour percevoir ce genre d'environnement ?
- Quels sont les algorithmes et les méthodes que nous devons utiliser pour assurer une telle navigation ?

1.3 Objectifs de recherche

L'objectif ultime de notre recherche serait de concevoir et développer une architecture de contrôle fiable et sécuritaire pour le nouveau module de navigation conçu pour FRMI. Ce grand objectif renferme derrière lui une autre finalité de notre recherche, à savoir, identifier l'ensemble de capteurs minimaux permettant d'assurer cette navigation. Tout ceci nous amène à des sous-objectifs adjacents nous guidant vers une réponse à nos problématiques. Nous allons donc :

- mettre en place une base de développement solide, une base caractérisée par son environnement modulaire et compréhensible par la majorité des développeurs ;
- développer un module de navigation manuel, permettant à un usager de contrôler d'une manière lisse et sécuritaire le FRMI directement avec une manette de contrôle ;
- fournir des modules complémentaires pour le traitement de données de base : acquisition de données de capteurs, calibration et fusion de données ;

1. Pour le FRMI, il s'agit d'un « joystick ».

- mettre en place un module de navigation local, se basant sur un algorithme d'évitement d'obstacle et une planification de chemin ;
- mettre en place un module de cartographie, permettant la construction de carte à partir des données de capteurs ;
- effectuer une confirmation et une validation du système de navigation ;
- faire une synthèse sur les capteurs utilisés visant à faire un compromis entre le prix et le rendu nécessaire pour cette étude.

1.4 Plan du mémoire

Dans le présent document, nous allons essayer de répondre à ces objectifs. Pour ce faire nous suivons le plan suivant : dans un premier lieu, citer et rappeler dans une revue de littérature l'ensemble de la théorie de la navigation, d'abord comme notion générale pour robot mobile puis comme application au FRMI. Ceci nous permettra d'encadrer les limites et les difficultés de notre sujet ainsi que de faire apparaître des éléments de solutions au regard de notre problématique.

Dans un second lieu, une description générale du FRMI traitant les deux aspects mécaniques industrielle et électrique, ceci en se concentrant sur les capteurs utilisés, et comment nous avons prévu de les intégrer dans le module de navigation.

Dans un troisième lieu, une présentation du module de navigation conçu pour le FRMI, intégrant les modules de supervision, navigation manuelle, semi-manuelle et automatique, évitement d'obstacle et cartographie.

Dans un quatrième lieu, un chapitre présentant une série d'expériences visant à confirmer et valider l'architecture de contrôle présentée.

Enfin, nous allons finir par une conclusion sommaire de nos travaux de recherche faisant apparaître les limites des solutions proposées et les ébauches d'améliorations possibles pour les travaux futurs.

CHAPITRE 2

REVUE DE LITTÉRATURE

2.1 Introduction

Depuis les années 1980, plusieurs modèles de prototype de fauteuil roulant motorisé autonome ont vu le jour (Simpson, 2005), chacun avec des caractéristiques différentes, plus ou moins intéressantes ayant pour but de satisfaire un ensemble de fonctionnalités désirées (Simpson *et al.*, 2005; Röfer et Lankenau, 1998) [1]. Ces fonctionnalités sont l'essence des besoins et attentes des personnes en bénéficiant. Un FRMI parfait est celui qui répondra à tous ces besoins d'une manière conviviale et efficace. Qui sont alors ces personnes intéressées et quels sont leurs besoins ?

Selon une enquête de Fehr *et al.* (2000) 10 % des patients qui reçoivent une formation sur un FRM trouvent qu'il est extrêmement difficile de l'utiliser pour les activités de la vie quotidienne. De plus, presque la moitié de ces usagers trouvent fatigant de piloter et manœuvrer un FRM à longueur de la journée (ceci est dû au besoin particulier de certains usagers présentant d'autres troubles cognitifs ou visuels). Une autre étude connue de Simpson (2008) confirme que les usagers potentiels du FRMI sont estimés à entre 61% et 91% des usagers du FRM actuel sur le marché. Ce fort taux est justifié par la ressemblance de l'interface d'interaction avec l'usager. Nous soulignons que ces pourcentages sont en croissance continue puisque la population mondiale vieillit (Siegel et , U.S.).

Concernant les besoins à satisfaire, ceci présente le cœur de la problématique de cette recherche. Nous devons concevoir un FRMI assurant une navigation sécuritaire et aisée dans les environnements habituels. Ceci suppose que le FRMI fournit un moyen adapté à ces usagers pour se déplacer naturellement dans ces espaces. En général, un usager cherche à produire le minimum d'effort pour avoir le comportement désiré. Nous devons, alors, assurer une interaction fluide humain-machine fournissant un sentiment de satisfaction et d'appropriation de la machine (Crandall et Goodrich, 2001). Il se trouve que l'utilisation des cartes pour se localiser et naviguer est une action naturelle et conviviale pour les humains (Eden, 1992). Cette solution intuitive pour l'usager est celle que nous allons adopter. Ceci dit, l'interprétation et la création de cette carte demandent un processus cognitif ambiguë dont les FRM actuels ne disposent pas.

1. Un tableau comparant les différentes caractéristique de ces fauteuils, fournie par Simpson (2005), est disponible sur l'adresse `http://www.rehab.research.va.gov/jour/05/42/4/pdf/simpson-appen-table.pdf`

Pour réaliser une navigation basée sur une carte, nous devons disposer et utiliser les trois processus suivants :

Construction de la carte : assimilation et apprentissage de l'environnement en mémorisant les données acquises durant une exploration.

Localisation globale : estimation de la position du robot dans la carte.

Planification de chemin : sélection d'un ensemble d'actions à suivre pour atteindre un objectif, connaissant la position du robot.

Dans la suite de ce chapitre, nous allons passer en revue les méthodes et algorithmes développés pour la navigation locale, globale, ainsi que le problème de cartographie.

2.2 Navigation locale

Lorsque nous parlons de navigation locale, nous évoquons immédiatement le problème d'évitement de collision avec les obstacles aux voisinages du FRMI, tout en assurant un mouvement fluide et réactif. En effet, un FRMI doit nécessairement être très réactif aux changements furtifs qui peuvent se produire dans l'environnement, par exemple, une personne qui traverse devant le fauteuil. Ainsi, plusieurs méthodes robustes tendent vers des algorithmes dynamiques qui planifient en continu le comportement du robot (Kitagawa *et al.*, 2001; Fox *et al.*, 1997). Cette dynamique ne peut être assurée qu'à l'aide des capteurs extéroceptifs du FRMI, fournissant avec une fréquence déterminée des informations sur les obstacles à proximité du robot. Ces informations sont souvent représentées par une distance à l'obstacle et une direction par rapport au robot. À partir de cette information, plusieurs méthodes existent pour dégager les mouvements sécuritaires que le robot peut effectuer.

Il est à noter qu'en général, les méthodes d'évitement d'obstacles pour la navigation locale ne se basent pas sur une carte d'environnement *a priori*, pour la simple raison que, d'une part ces cartes ne sont pas exactes (des obstacles peuvent resurgir dans l'environnement), d'autre part, vu la taille des cartes, ces méthodes coûtent en termes de rapidité de calcul pour planifier les trajectoires admissibles (Latombe, 1990). Par contre, ceci vient avec l'inconvénient que ces méthodes ne sont jamais optimales étant donné qu'elles se limitent à l'environnement immédiat. Elles peuvent donc être facilement coincées dans des minima locaux, comme les culs-de-sac (obstacle en forme de U). Dans les prochaines pages, nous allons détailler les trois algorithmes les plus connus pour la navigation locale.

2.2.1 Méthode basée sur les champs potentiels

Ces méthodes, comme celle introduite par (Khatib, 1986), calcule la direction et la norme de la vitesse, en supposant que les obstacles exercent une force répulsive sur le robot F_{rep}

tandis que le point objectif exerce une force attractive F_{att}. Ces forces dérivent d'un champ potentiel U qui a la forme suivante :

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta \left(\frac{1}{\|o(q)-o(p)\|} - \frac{1}{\rho_0} \right)^2 & \text{si } \|o(q)-o(p)\| \leq \rho_0 \\ 0 & \text{sinon} \end{cases} \tag{2.1}$$

$$U_{att}(q) = \begin{cases} \frac{1}{2}\xi \|o(q)-o(p)\|^2 & \text{si } \|o(q)-o(p)\| \leq d \\ d\xi \|o(q)-o(p)\| & \text{sinon} \end{cases} \tag{2.2}$$

Avec o une mesure de distance, ρ_0 le rayon d'influence de l'obstacle détecté au point p sur le robot au point q, d une distance caractérisant l'attractivité de l'objet. Si d est grand, le robot va se rapprocher plus rapidement de l'objet et vice-versa. Sur la figure 2.1 nous présentons une simulation sur MATLAB de la méthode des champs potentiels. Nous pouvons distinguer : en vert, la position de départ du robot, en rouge les obstacles dans l'environnement et en jaune la position finale. Le robot commence à se diriger vers le puits de potentiel créé par la position finale, puis en se rapprochant des obstacles, il se voit contraint de les contourner vu leur grand potentiel.

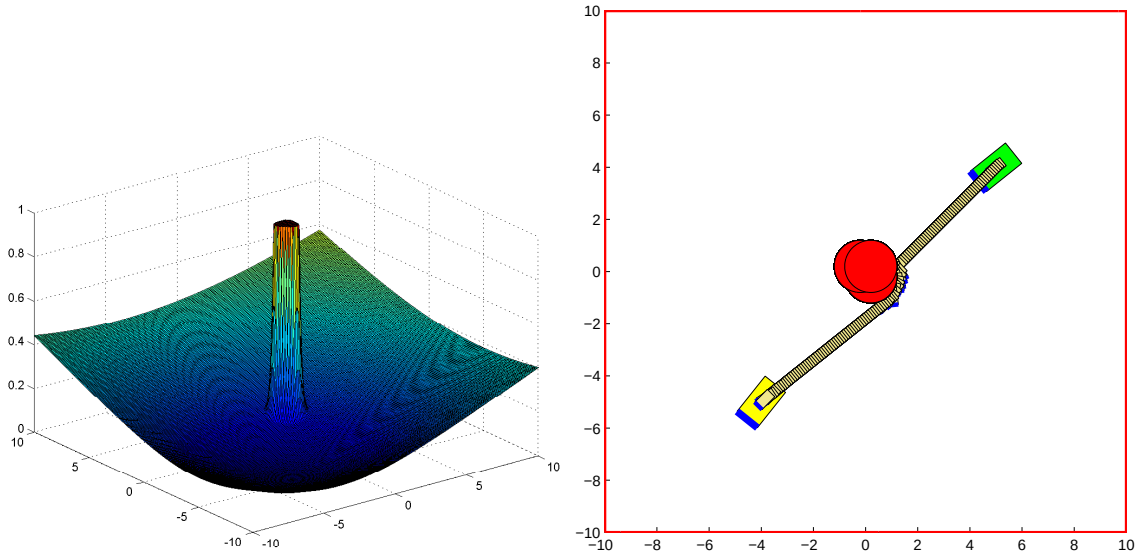


Figure 2.1 Simulation de la méthode des champs potentiels : à gauche la somme des potentiels attractif et répulsif, à droite le chemin planifié.

Bien que ces méthodes soient très rapides et ne considèrent que l'environnement immédiat

du robot, elles échouent facilement à trouver un chemin entre les obstacles très rapprochés (minimum local). Elles produisent aussi des trajectoires oscillatoires dans les corridors (deux forces répulsives parallèles) (Koren et Borenstein, 1991). Depuis, d'autres versions améliorées de cet algorithme ont vu le jour (Yin et Yin, 2008), ceci en modifiant la forme du champ potentiel ou en rajoutant d'autres champs potentiels particuliers, comme, un champ caractérisant la rotation du robot.

2.2.2 Méthode basée sur les histogrammes de champ de vecteur

Borenstein et Koren proposent une méthode basée sur un histogramme de champ de vecteurs ; cette approche utilise une grille d'occupation locale mise à jour à chaque instant. Chaque case contient un coût proportionnel au nombre de fois qu'un obstacle est détecté. Cette carte est ensuite transformée en histogramme qui décrit les espaces libres autour du robot, pour enfin calculer la direction et la norme optimale de la vitesse permettant de traverser l'espace moins encombré (Borenstein et Koren, 1991).

Cet algorithme se caractérise par sa simplicité d'implémentation et rapidité d'exécution ; par contre, la discrétisation de l'espace et la dépendance sur un espace limité peuvent générer des vitesses de rotation brusques pour éviter les nouveaux obstacles apparaissant.

2.2.3 Méthode des fenêtres dynamiques

La méthode des fenêtres dynamiques a été introduite la première fois par Fox *et al.* (1997). Elle vient pour pallier aux lacunes des algorithmes précédents. Elle a été spécialement conçue pour gérer les contraintes imposées sur la vitesse et l'accélération, ainsi qu'éviter les situations liées aux minima locaux.

En bref, la méthode consiste à examiner périodiquement, sur des petits intervalles de temps, les trajectoires circulaires ou courbées envisageables par le robot, ceci, pour choisir la meilleure trajectoire possible. Cette approximation de trajectoire est possible en discrétisant l'espace des vitesses linéaires et angulaires. Enfin de compte, la recherche de la trajectoire optimale devient une recherche dans l'espace des vitesses admissibles satisfaisant un critère du coût fixé *a priori*. En général, la fonction du coût fait intervenir une mesure de la distance et de l'orientation par rapport au point objectif, et une mesure de distance du plus proche obstacle.

La figure 2.2 présente le résultat d'une simulation sur MATLAB de l'algorithme des fenêtres dynamiques. Elle démontre à gauche un ensemble de positions finales admissibles, puis à droite le chemin optimal retenu, nous avons choisi une fonction coût de la forme suivante :

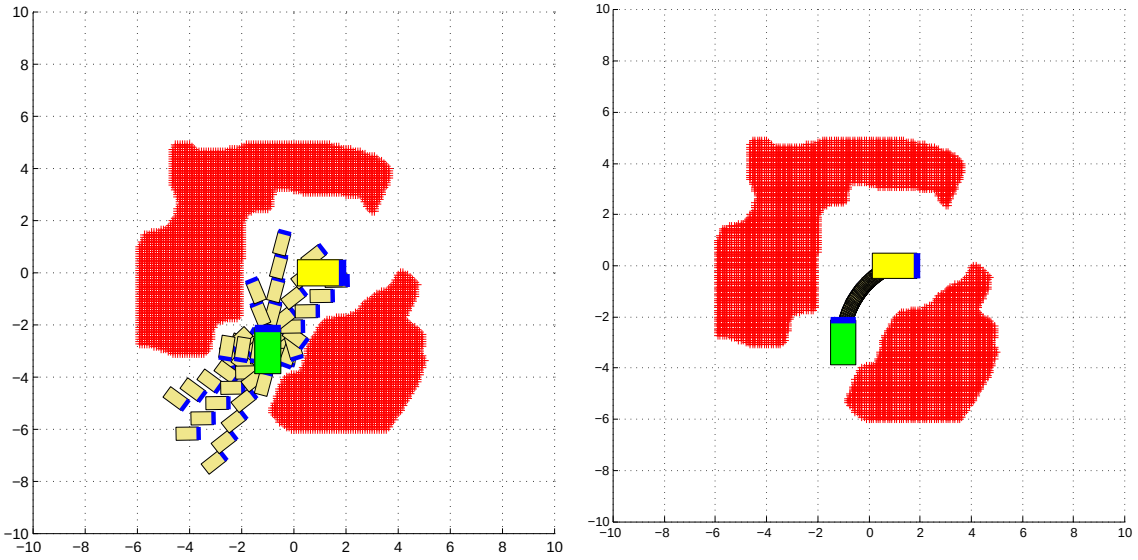


Figure 2.2 Simulation de la méthode des fenêtres dynamiques : à gauche l'ensemble des trajectoires admissibles pour atteindre l'objectif, à droite le chemin optimale choisi selon la fonction coût.

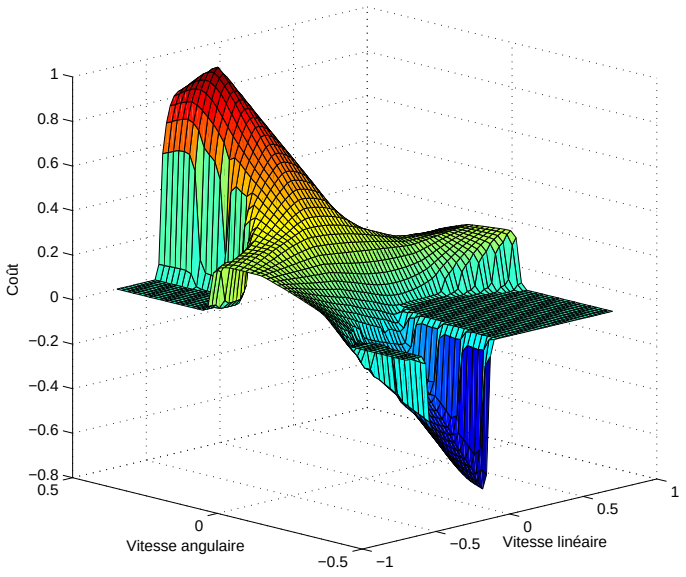


Figure 2.3 Fonction du coût généré par la méthode des fenêtres dynamiques à l'instant initial

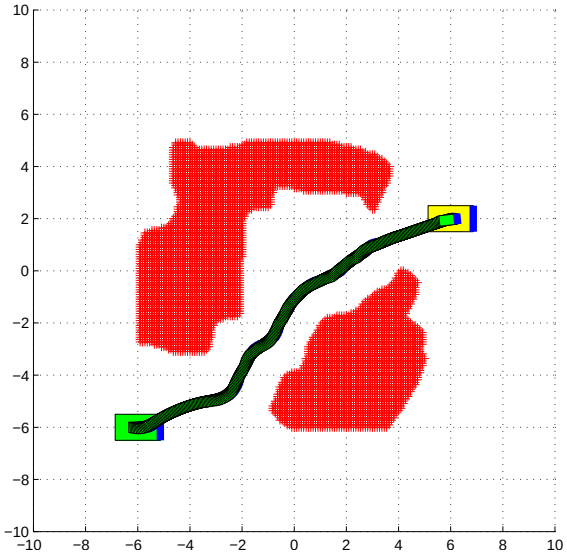


Figure 2.4 Simulation de la planification de trajectoire avec la méthode des fenêtres dynamique

$$G(v,w) = \sigma\left(\alpha \text{ avancement}(v,w) + \beta \text{ encombrement}(v,w) + \gamma \text{ vitesse}(v,w)\right) \tag{2.3}$$

L'avancement est une mesure quantifiant la progression vers le but ; elle est égale à une combinaison entre la distance au but et l'orientation du robot par rapport au but. L'encombrement est égal à la distance au plus proche obstacle rencontré pendant la trajectoire simulée. La vitesse est égale à la valeur absolue de la vitesse du robot durant l'exécution de la trajectoire et la fonction σ est une fonction de lissage pour régulariser la valeur du gain. La figure 2.3 montre la valeur de la fonction du coût sur l'espace de vitesse linéaire et angulaire.

Enfin la figure 2.4 montre une simulation de la planification en temps réel avec cette méthode. Nous pouvons constater que le chemin optimal planifié passe exactement au milieu des obstacles tout en se dirigeant vers le point objectif, ceci confirme la fluidité et la sécurité de cette méthode en simulation.

2.3 Navigation globale

Le problème de navigation globale est un problème de planification globale de chemin pour aller à un point particulier. Pour arriver à ce genre de planification, le robot doit disposer d'une source d'information globale : une carte d'environnement. La carte d'environnement peut être exprimée de différentes façons :

Arbre de point d'intérêt : le principe est de sauvegarder un ensemble de points d'intérêt dans un arbre ; en général ce sont des images des points particuliers dans l'environnement. La localisation devient alors un problème de recherche dans un arbre (Frese, 2006).

Carte du champ magnétique : en exploitant les anomalies du champ magnétique dues aux câbles électriques, l'acier dans les bâtiments, etc., il est possible de construire une carte caractérisant le champ magnétique de l'environnement ; ensuite, à l'aide d'un magnétomètre, et d'algorithmes probabilistes, comme un filtre à particules, nous arrivons à estimer la position globale dans la carte (Haverinen et Kemppainen, 2009).

Carte de grille 2D : à l'aide d'un capteur de proximité et de l'odométrie du robot, nous pouvons identifier les obstacles statiques à chaque instant. Ainsi, nous pouvons assembler tous ces obstacles pour construire une carte caractérisant les obstacles dans l'environnement, surtout les murs. Une fois dans cette carte nous pouvons estimer la position globale du robot (Moravec, 1988).

Dans tous les cas, ces cartes ne représentent qu'une vision globale de l'environnement avec souvent un manque critique de détails. Ainsi, ces cartes globales sont utilisées typiquement pour la planification de route à suivre. Une fois la route établie, le module de navigation locale a pour mission de suivre ce chemin en évitant les obstacles. En général, la navigation globale est soutenue par des modules complémentaires de suivi de mur dans un couloir, de passage d'une porte, ou autres.

2.4 Localisation globale dans une carte

Durant la dernière décennie, le problème de localisation dans une carte en se basant sur des données de télémétrie a occupé largement les travaux concernant les robots mobiles (Burgard *et al.*, 1998; Endres *et al.*, 1998; Thrun *et al.*, 1997). Résoudre ce problème consiste à :

- dans un premier temps, chercher la position du robot dans une carte de l'environnement ; ceci inclut le cas pour lequel nous n'avons aucune information sur la position initiale du robot (Engelson, 1994) ;
- dans un deuxième temps, tracer et estimer sa position au fil du temps. Les récentes approches traitant ce problème démontrent de très bons résultats ; ceci a été obtenu en se basant sur les chaînes de Markov (Nourbakhsh, 1998).

L'idée générale de la localisation de Markov est de représenter la pose du robot par une distribution de probabilité sur les positions possibles, et utiliser la règle de Bayes et la convolution pour mettre à jour cette estimation de la pose à chaque instant. Cette idée n'est

pas nécessairement nouvelle puisque Gelb (1974) l'a déjà exploitée avec un filtre de Kalman ; par contre, nous notons que ce dernier reste limité dans ce genre de problème multimodal et fortement non linéaire.

Les filtres particulaires, aussi connus sous le nom de méthodes de Monte-Carlo séquentielles (MCL), présentent un moyen efficace et rapide pour estimer la probabilité d'une distribution de chaînes de Markov (Fox *et al.*, 1999). L'idée consiste à générer aléatoirement un ensemble de particules p ; chaque particule, étant une position hypothétique du robot, sera associée à un poids. Le poids dépend essentiellement du rapport entre la mesure du robot sur son environnement et ce que la particule aura mesuré connaissant sa position sur la carte. La puissance du MCL est qu'il considère un ensemble limité de particules ; par contre, il gagne en robustesse en associant à chaque particule un bruit de mouvement modélisant l'erreur de l'odométrie, et en échantillonnant la distribution des particules au besoin.

Les figures 2.5 montrent le résultat d'une simulation sur MATLAB de l'algorithme MCL. Le robot est présenté en vert et les particules en jaune. Nous pouvons constater qu'au début, l'algorithme initialise la distribution des particules aléatoirement dans l'environnement, puis à chaque instant, les particules les plus probables (de taille plus grande) survivent au fur et à mesure que le robot découvre son environnement. En fait, nous avons simulé à chaque instant des mesures de balayage laser comme s'ils étaient perçus par le robot dans sa position réelle. Cette mesure est comparée par la suite avec la carte de l'environnement pour fournir une mesure d'incertitude aux particules. La figure 2.6 présente l'erreur entre les positions réelle et estimée par itération.

2.5 Cartographie

Malgré l'abondance des plans des bâtiments qui peuvent être utilisés comme carte de navigation, il est fort probable de tomber sur de nouveaux environnements pour lesquels nous ne disposons pas de carte ou tout simplement que ces cartes soient dépassées, anciennes ou très difficiles à interpréter par un robot. Nous devons, alors, nous procurer un moyen de création ou de correction de ces cartes.

Le problème de cartographie, aussi appelé « SLAM : simultaneous localization and mapping », est l'un des problèmes les plus complexes et fondamentaux dans la robotique mobile. Sa complexité vient du fait que pour construire une bonne carte le robot doit connaître précisément sa position, mais pour connaître sa position il a besoin d'une carte précise (Grisetti *et al.*, 2007). C'est cette corrélation entre la position globale et la carte qui rend le problème difficile et qui exige de chercher une solution dans une dimension encore plus grande.

La seule façon de résoudre ce problème est d'estimer continuellement la position du robot

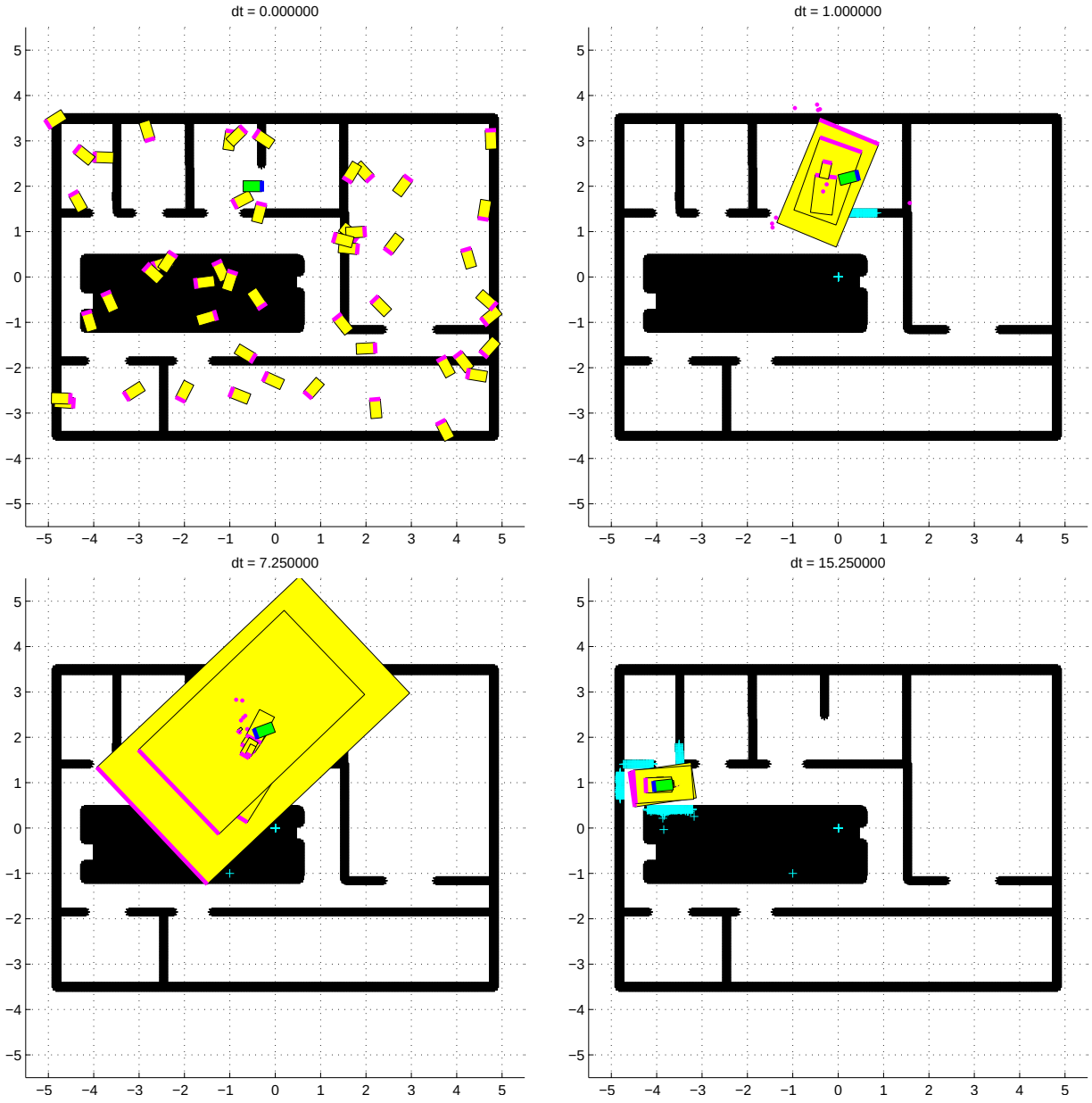


Figure 2.5 Simulation de l'algorithme MCL

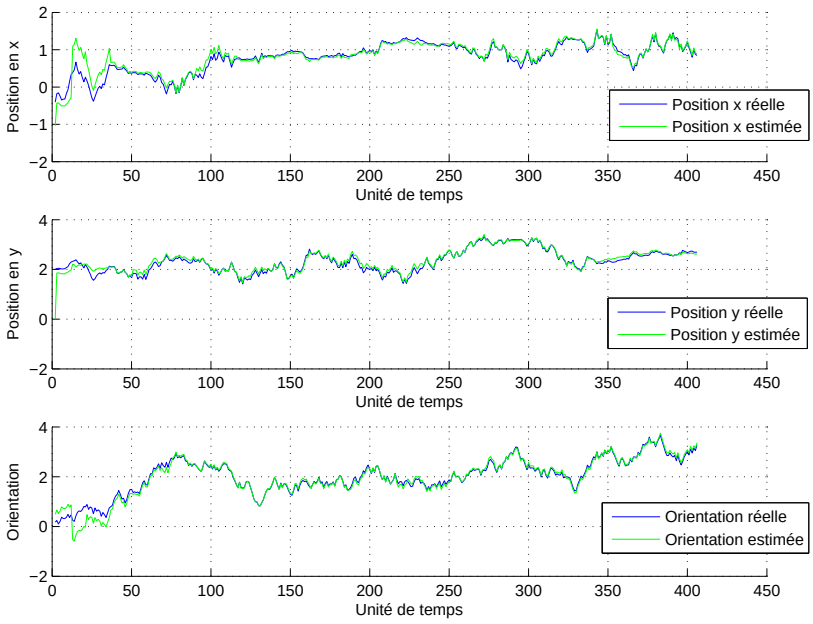


Figure 2.6 Tracé de la position réelle et la position estimée par itération

et la carte simultanément $p(x_t, m_t)$. Mais avant de faire cette estimation, revoyons nos sources d'information. Essentiellement nous disposons de deux types d'information :

Une source idiothétique [2] (Mittelstaedt et Mittelstaedt, 1980) Cette source concerne les informations internes du robot lui-même, comme sa vitesse, son accélération, la rotation de chaque roue. L'intégration continue de ce genre d'information permet de connaître la position relative du robot, par rapport à un point de départ. Bref, c'est ce que nous appelons odométrie.

Une source allothétique [3] (Mittelstaedt et Mittelstaedt, 1980) Cette source représente toutes les informations extérieures au corps du robot, c'est-à-dire, des indices sur son environnement (généralement, ils prennent la forme de données laser, sonar, ou de vision).

Bien que ces sources démontrent individuellement des désavantages critiques, lorsqu'elles sont combinées ensemble, elles s'avèrent très complémentaires (Cox, 1991). En effet, d'une part, l'estimation de la position basée sur les données d'odométrie a recours à des intégrations

2. Terme utilisé pour la navigation chez les animaux : l'orientation d'un animal dans son environnement est dite idiothétique si les indices spatiaux sont tirés de signaux produits par l'organisme lui-même, par exemple, les signaux proprioceptifs dus à l'activité locomotrice, ou les commandes motrices elles-mêmes.

3. Terme utilisé pour la navigation chez les animaux : l'orientation d'un animal dans son environnement est dite allothétique quand les indices spatiaux trouvent leur origine dans la position de l'animal relativement à des facteurs physiques ordonnés spatialement et qui sont indépendants de l'état physiologique et du comportement de l'animal.

continues des valeurs brutes acquises par les capteurs de vitesse : ceci implique que chaque erreur dans ces valeurs sera cumulée, et donc, au long terme l'estimation de la position n'est plus correcte. D'autre part, les sources allothétiques sont statiques dans le temps et ne contiennent pas d'information sur le robot, ce qui rend difficile, voire impossible, de distinguer deux situations qui se ressemblent dans l'environnement (par exemple, un couloir symétrique, des pièces carrées, ...).

La combinaison de ces deux sources pour construire une carte signifie que les sources allothétiques vont permettre de corriger les erreurs accumulées par l'odométrie, tandis que les sources idiothétiques rajouteront une information spatiale, pour donner sens aux mesures externes de l'environnement, qui vont permettre de construire la carte.

L'un des premiers à exploiter cette particularité fut Elfes (1989) en travaillant sur des cartes de grille d'occupations. Avec ces grilles, le problème est d'estimer une valeur associée à chaque case, donnant une estimation sur l'existence d'un objet.

D'un point de vue purement probabiliste (Thrun *et al.*, 2005), le problème est celui d'estimer la carte m_t et la position x_t connaissant les dernières mesures d'odométrie u_{t-1} et de capteurs extéroceptifs z_t, c'est-à-dire : $p(x_t, m_t|z_t, u_{t-1})$. Pour ce faire, nous allons faire une première estimation basée sur le modèle du robot et les informations antérieures dont nous disposons (avec la formule des probabilités totales), puis corriger et améliorer cette estimation avec les mesures actuelles, qui nous donnent une information sur l'état actuel (avec le théorème de Bayes). Ceci s'exprime simplement avec les deux équations :

$$p(x_t, m_t|u_{t-1}) = \sum_{y \in x_{1:t-1}, m_{1:t-1}} p(x_t, m_t|u_{t-1}, y)p(y) \tag{2.4}$$

$$p(x_t, m_t|z_t, u_{t-1}) = \frac{p(z_t, m_t|x_t, u_{t-1})p(x_t, m_t|u_{t-1})}{p(z_t, m_t|u_{t-1})} \tag{2.5}$$

Cependant, la résolution exacte de ces deux équations à chaque instant avec la même fréquence du mouvement du robot est non envisageable, chose qui nous pousse à utiliser des filtres pour l'estimation de ces probabilités.

L'utilisation des filtres de Kalman étendue est la méthode la plus commune (Wolf et Sukhatme, 2005). Cette approche a été expliquée en détail par Smith *et al.* (1990). Dans ce cas, le filtre de Kalman est utilisé pour estimer la position d'un ensemble de points de repère « landmark » retrouvé sur les mesures $z(t)$ au même temps que la position du robot $x(t)$. Par contre, dans les grands environnements le nombre de ces points de repère explose facilement. De nombreuses recherches ont conduit Stentz *et al.* (2003) à proposer une méthode pour réduire la complexité du problème. Cette méthode est basée sur les filtres à particules Rao-Blackwellized (Doucet *et al.*, 2000).

Les filtres de Rao-Blackwellized permettent de justifier l'hypothèse de factoriser l'estimation de la carte et de la position $p(x_t, m_t|z_t, u_{t-1})$ en :

$$p(x_t, m_t|z_t, u_{t-1}) \quad = \quad p(m|x_{1:t}, z_{1:t})p(x_{1:t}|z_{1:t}, u_{1:t-1}) \tag{2.6}$$

Ceci signifie que nous pouvons estimer uniquement la trajectoire du robot et puis calculer la carte en se basant sur cette trajectoire, puisque la carte dépend fortement de l'estimation de la pose du robot.

CHAPITRE 3

DESCRIPTION ET PRÉSENTATION DU FRMI

Avant de s'attaquer aux problèmes de plus haut niveau comme le contrôle, la navigation ou la cartographie, il est nécessaire de bien connaître la structure du robot mobile sur lequel nous travaillons, c'est-à-dire, le fauteuil roulant motorisé intelligent. Dans ce chapitre, nous présentons en détail les composantes du FRMI :

la partie logicielle qui se manifeste sous ROS : un système d'exploitation conçu pour les robots. Nous commençons par une description générale de cet outil permettant de justifier pourquoi nous l'avons adopté, puis une description détaillée de l'esprit de développement sous ROS, montrant comment nous l'avons adapté aux spécifications de notre FRMI.

la partie matérielle qui contient l'ensemble des actionneurs et capteurs du FRMI. Ceci, dans le but de faire l'inventaire de notre matérielle disponible, connaître ces atouts, mais aussi ses limites, et pouvoir conclure sur une méthode efficace de traitement ce flux des données de capteurs.

3.1 ROS : Robot Operating System

ROS est un méta système d'exploitation conçu pour les robots, installé sur un système d'exploitation classique, de préférence Ubuntu Linux. Il est conçu par des développeurs expérimentés dans le domaine robotique [1], avec comme objectif de fournir un environnement simple et pratique pour implémenter des applications robotiques :

- il offre des fonctionnalités de système d'exploitation, comme des outils de navigation entre applications, création et compilation d'application, ou d'édition d'un fichier source d'application ;
- il supporte plusieurs langages de programmation : C++, Python ou JAVA ;
- il inclut une bibliothèque complète de pilotes (drivers) pour différents types de capteurs ;
- il gère la communication interne entre les nœuds [2], mais aussi la communication externe entre machines par réseau, permettant ainsi de distribuer les nœuds sur plusieurs

1. ROS a vu le jour au laboratoire d'intelligence artificielle de l'université de Sandford, puis, a été entretenue par willow garage un laboratoire de recherche robotique connu pour son robot personnel PR2.

2. Une nœud sous ROS est une application/processus conçue pour effectuer une tâche particulière (acquisition d'un capteur, application d'un filtre, implémentation d'un algorithme, ...), un nœud peut s'inscrire ou publier des messages à d'autres nœuds.

ressources de calcul ;
– chaque application développée peut être mise en ligne pour qu'elle soit révisée et corrigée par la communauté de développeurs de ROS.

Toutes ces caractéristiques intéressantes nous ont poussés à migrer de l'ancienne plateforme robotique Acropolis [3] (Zalzal, 2009) à ROS, même si ceci n'a pas été direct à cause de quelques incompatibilités matérielles. Cependant, la similitude de l'architecture entre ces deux systèmes (un arbre de nœud/plugin qui communiquent entre eux) nous a permis de réécrire les applications fonctionnelles qui existaient sous Acropolis sous une forme compatible avec ROS.

Pour clarifier le fonctionnement de ROS, la figure 3.1 présente un exemple de ce qui se passe derrière l'écran dans le simple cas d'une navigation manuelle avec acquisition de données laser. Les formes circulaires représentent des processus qui s'exécutent en continu avec une fréquence déterminée (des nœuds), les formes carrées représentent les messages communiqués entre ces nœuds (des dossiers [4]), et les flèches démontrent le flux d'information. Avec ces notations nous pouvons lire : la manette de contrôle (*/joy*) et les télémètres lasers (*/right_hokuyo_node*, */rear_hokuyo_node* et */left_hokuyo_node*) lisent en continu les données d'entrées. Le superviseur interprète le message de la manette et supervise la navigation (voir chapitre 4), le message de commande est acquis par le robot virtuel (*/virtualFRMI*) qui s'occupe de la communication matérielle, de mettre à jour l'état du FRMI, ainsi que publier d'autres messages pour la visualisation. Les nœuds */merge_scan* et */merge_cloud* permettent de synchroniser et combiner les messages lasers pour afficher une donnée laser exploitable (voir section 3.6).

3.2 Composantes du FRMI, présentation sous ROS

3.2.1 La forme brute du FRMI

Il est crucial de connaître précisément la structure d'un robot avant de commencer à traiter les problèmes de contrôle ou de navigation. Dans notre cas, nous travaillons sur un FRM, en général nous pouvons distinguer les parties suivantes (voir figure 3.2) :

Le siège : c'est l'ensemble de pièces de confort pour l'usager. Il contient une base, un accoudoir et un dossier. La forme de siège change d'un modèle à un autre.

3. Acropolis est l'ancienne plateforme de développement robotique qui était présente sur le FRMI, développée entièrement par des chercheurs de l'École Polytechnique de Montréal. Elle permettait d'implémenter des applications robotiques sous formes de « plugins » (équivalent des nœuds) et de gérer les liens entre eux. Malheureusement, à l'heure actuelle elle est en voie de devenir obsolète à cause du manque de mise à jour.

4. Sur ROS un dossier est référé par *topic*.

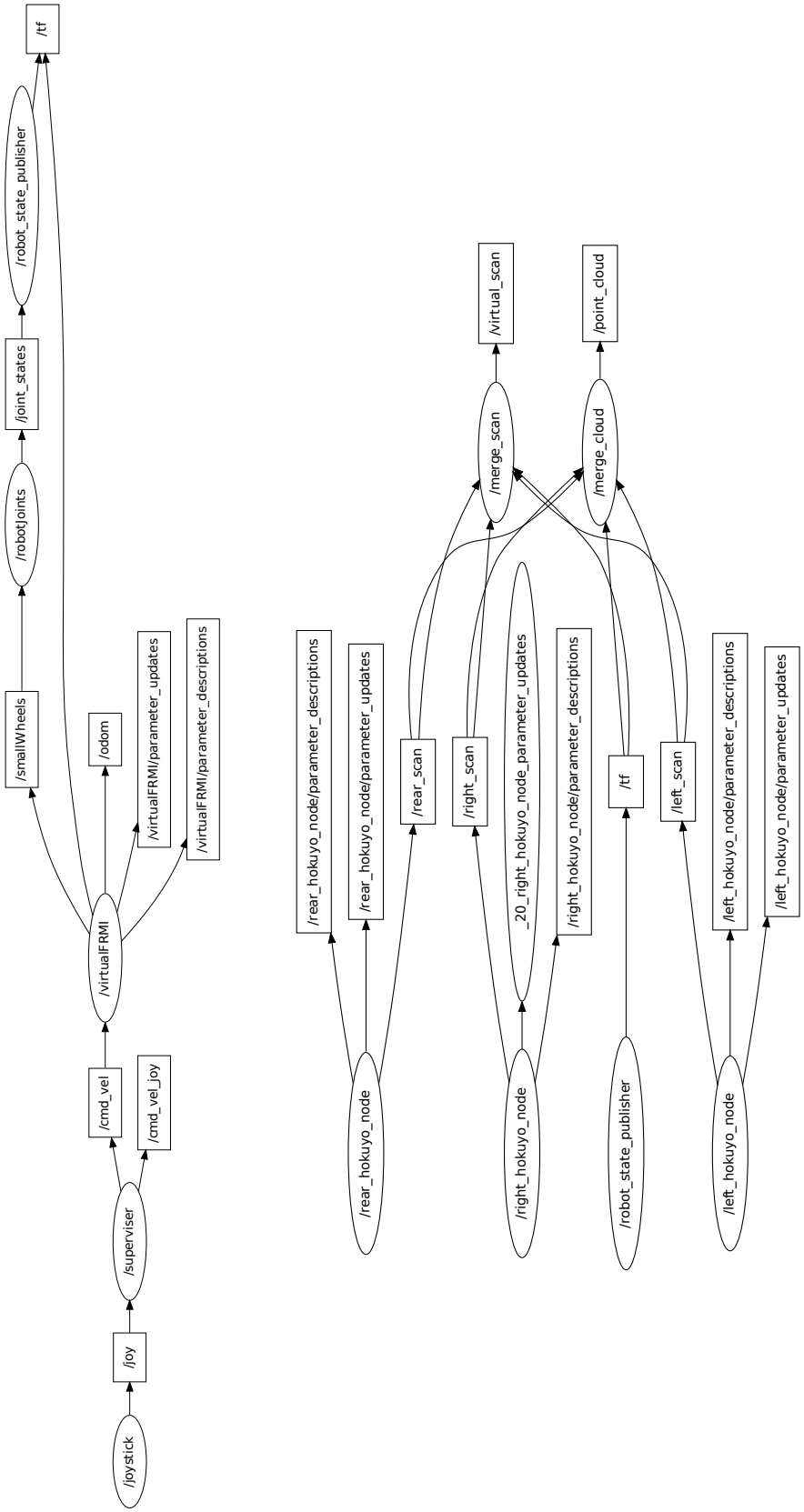


Figure 3.1 Présentation de la communication entre nœuds dans le mode manuel

Les roues motrices : ce sont les deux roues arrières qui reçoivent la puissance du couple des moteurs et chaque roue peut tourner indépendamment l'une de l'autre dans les deux sens.

Les roulettes : deux en avant et deux en arrière, ce sont des roulettes libres « caster wheel », permettant d'assurer un équilibre stable au fauteuil.

La base du FRMI : c'est un coffret rigide sous le siège entre les deux roues motrices, elle contient tout le matériel informatique nécessaire pour le contrôle du FRMI.

Ces pièces représentent le noyau central d'un fauteuil roulant motorisé ; nous les retrouvons dans tous les modèles du marché actuel, et ce malgré que leurs propriétés individuelles (dimensions, inerties et matériels . . .) puissent changer d'un modèle à l'autre. Afin de pouvoir fournir un logiciel modulaire qui s'adapte à tous les cas possibles, nous allons proposer un outil simple de personnalisation du FRMI.

3.2.2 Les rajouts matériels : capteur et interface

Pour assurer le critère « intelligent » du FRMI, nous avons ajouté du matériel supplémentaire pour percevoir et interagir avec l'environnement, mais puisque nous sommes en phase de développement, nous nous permettons d'essayer un large catalogue de ces capteurs et interfaces d'interaction. Nous notons que le positionnement de ces derniers peut changer, soit volontairement pour faire un test particulier, soit accidentellement à cause de mauvaises liaisons (voir figure 3.2).

Interface d'interaction

Une interface d'interaction permet à l'usager d'agir directement sur le FRMI. Nous distinguons deux interfaces : une manette de contrôle « joystick » pour envoyer des commandes de vitesse lineaire et angulaire, ainsi que pour déclencher et arrêter quelques manœuvres . Un écran tactile qui permet d'avoir un visuel du FRMI, de son environnement et des actions qu'il effectue, ainsi que d'envoyer des commandes de navigation haut niveau (passer une porte, suivre un mur, aller à un endroit . . .).

Capteurs

Le prototype FRMI dispose des capteurs suivants :

Codeurs optiques rotatifs : Ce sont les deux capteurs reliés aux roues motrices, permettant de compter le nombre de tours effectuées par les roues, ce qui fournit une information sur l'odométrie : vitesse linéaire v_{lin} et vitesse angulaire v_{ang} (voir l'équation 3.2).

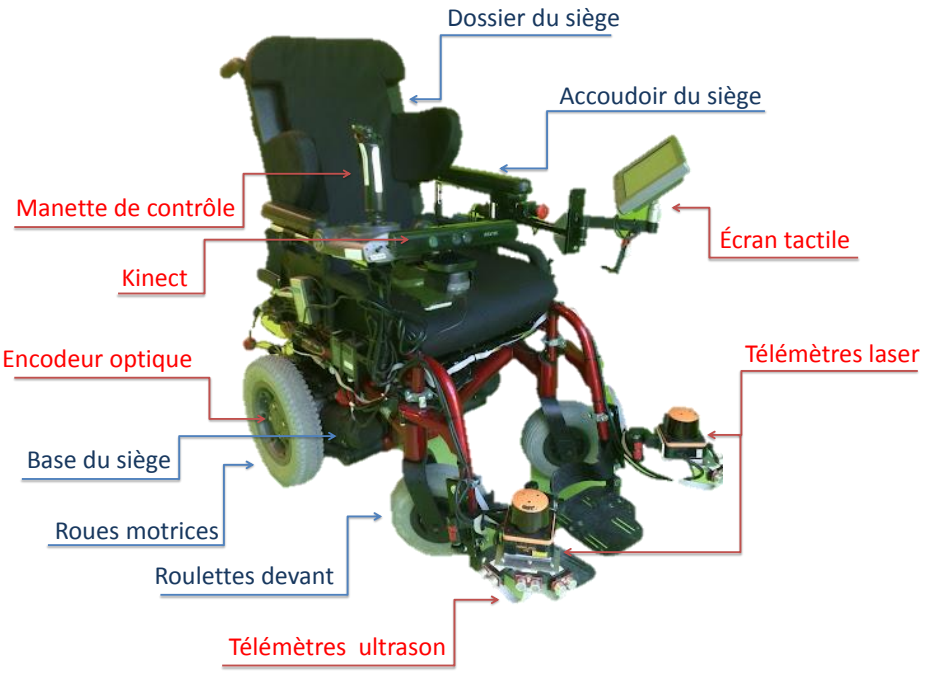


Figure 3.2 Présentation du FRMI, les rajouts matériels en rouge

$$v_{lin} = \frac{\pi R}{n}\left(N_{droite} + N_{gauche}\right) \quad (3.1)$$

$$v_{ang} = \frac{2\pi R}{nL}\left(N_{droite} - N_{gauche}\right) \quad (3.2)$$

n étant la résolution des encodeurs égale à 110 tiques par rotation. N_{droite} et N_{gauche} sont le nombre de tiques compté par les encodeurs [5] pour, respectivement, les roues droite et gauche. L est la distance entre les deux roues.

Télémètre laser : C'est un dispositif utilisant les rayons laser pour la mesure de la distance à un objet. Il se caractérise par sa précision et sa large portée. Les lasers équipant le FRMI sont du type « Hokuyo UHG-08LX » ; ils ont un angle de vue de 270 degrés, une résolution angulaire allant jusqu'à 0.36 degré et une fréquence de balayage de 40Hz.

Télémètre ultrason : Comme un télémètre laser, il permet de mesurer la distance ; par contre, comme son nom l'indique, il utilise des ondes ultrasons au lieu d'ondes optiques.

Caméra Kinect : C'est une caméra stéréoscopique développée par Microsoft pour la reconnaissance des formes et des mouvements. Elle inclut une caméra couleur, un capteur de profondeur 3D qui comprend un projecteur laser infrarouge associé à un capteur CMOS monochrome, et un circuit intégré qui permet de construire un nuage de points de l'environnement à partir de ces deux informations. La Kinect a une portée inférieure à 3.5 mètres, et un champ de vision de 57 degrés sur le plan horizontal et 43 degrés sur le plan vertical.

3.2.3 Le FRMI sous ROS

Malgré l'aspect prototype de notre projet et les différences entre les fauteuils roulants dans le marché, nous voulons fournir un logiciel modulaire qui s'adapte au plus grand nombre des cas possibles ; c'est pourquoi nous proposons un outil simple de personnalisation du FRMI.

ROS propose d'utiliser un format unifié pour la description d'un robot « URDF » (Lu, 2012) que nous allons adopter. L'idée est de stocker dans un fichier XML les pièces nécessaires pour la présentation du robot. À chaque pièce est associée un repère, une forme géométrique, une masse et un type de matériel. Cet ensemble forme un joint et chaque joint peut être lié à d'autres joints par une liaison fixe ou libre. Le tout constitue finalement un arbre de joints (voir figure 3.3).

Dans l'arbre des joints, nous soulignons l'existence de quelques repères clés à retenir :

5. Le nombre de tiques est remis à zéro après chaque lecture.

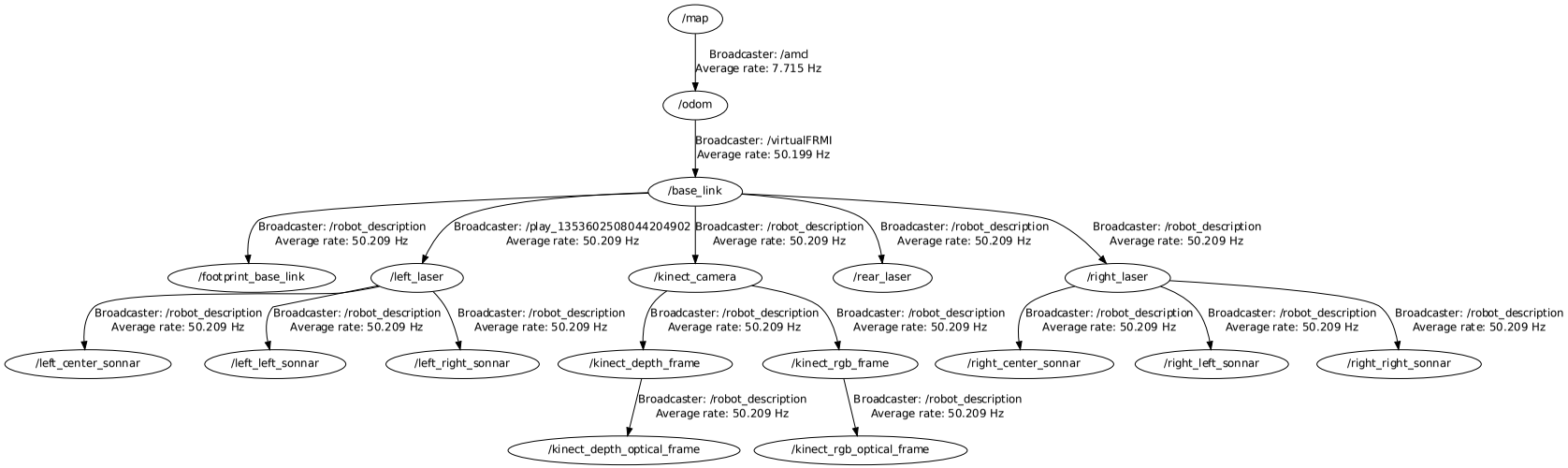


Figure 3.3 Arbre de joints présentant l'URDF du FRMI

- le repère associé à la base du FRMI R_{FRMI} [6] : ce repère est situé au milieu entre les deux roues motrices sur l'axe de rotation du FRMI ;
- le repère associé à la navigation locale R_0 [7] : ce repère est créé au début de la navigation. Il représente le tout premier repère R_{FRMI} initialisé au début de la navigation. Ce repère contient l'information sur l'odométrie (mouvement entre R_{FRMI} et R_0) à chaque instant ;
- le repère associé à la carte R_{monde} [8] : c'est le repère de référence associé à la carte.

L'intérêt du fichier XML est qu'il est facile d'utilisation même en n'ayant que quelques notions en programmation. Ainsi, le changement, le rajout ou la suppression d'une pièce ou d'un capteur se fait avec une simple instruction dans le fichier XML.

Une fois l'URDF réalisé, nous allons en faire une lecture pour récupérer les informations du FRMI, et faire bouger les joints libres. Ainsi, d'une part nous allons pouvoir afficher le modèle du robot (voir figure 3.4), et d'autre part, publier l'information des repères dans ROS [9] .

3.3 Calibration des capteurs laser

Comme indiqué auparavant, nous avons construit un modèle 3D du FRMI interprétable facilement par l'usager d'une part et par les outils de ROS d'autre part, ainsi qu'une façon simple d'éditer ce modèle. Actuellement, les dimensions des pièces du FRMI sont mesurées empiriquement et ne sont pas d'une précision très grande [10]. Cependant, dans le cas des positions des capteurs, comme les télémètres lasers, nous sommes obligés de fournir une information millimétrique sur leurs positions et rotations, surtout pour accomplir des procédures complexes comme les évitements d'obstacles ou la cartographie.

Nous allons présenter un algorithme permettant d'estimer la position du télémètre laser par rapport au centre du FRMI, en faisant une manipulation simple à chaque fois que cette position change, ce que nous allons appeler une calibration des capteurs lasers.

La calibration consiste à mettre le FRMI devant une planche délimitée et distinguable, à une distance connue d_{board} [11], et de façon à ce que le FRMI soit perpendiculaire et au centre de la planche (voir figure 3.5). Il suffit ensuite de lancer le programme. L'algorithme consiste

6. Sur ROS ce repère est noté « */base_link* »
7. Sur ROS ce repère est noté « */odom* »
8. Sur ROS ce repère est noté « */map* »
9. Dans ROS il est important de préciser la position de tous les capteurs par rapport au repère du robot R_{FRMI}, cette information est sauvegardée dans le message contenant toutes les transformations entre joints « */tf* ».
10. Les dimensions du FRMI peuvent être assignées directement par le constructeur dans le futur.
11. d_{board} ne doit être ni très grande ni très petite, si très grande nous risquons de confondre la planche avec d'autres objets, si très petite, nous risquons de ne pas voir la planche au complet

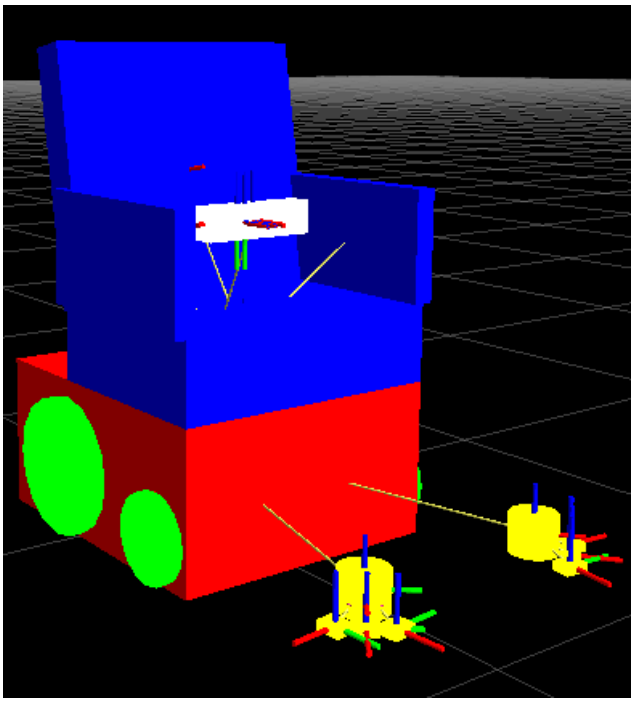


Figure 3.4 URDF du FRMI

à détecter la planche à l'aide des mesures lasers puis déduire la position relative du laser par rapport au FRMI, en ayant comme information la relation entre la position de la planche et celle du FRMI.

Pour distinguer la planche, nous procédons comme suit :

- le fauteuil doit être stable et en face de la planche à la distance d_{board}, et le laser doit pouvoir voir la planche au complet.
- le laser effectue un nombre de balayages fixe : ces mesures seront moyennées pour construire une image de la pièce. Nous effectuons une moyenne de plusieurs balayages pour réduire les bruits de mesures, dus à la réflexion sur les coins [12] et des obstacles mouvants. L'image résultante contient un 1 dans les obstacles et des 0 dans les espaces vides. Les figures 3.6 présentent le résultat du balayage pendant une calibration. Nous pouvons constater la présence d'un ensemble de points séparés présentant les obstacles mobiles ; ces points seront enlevés dans le seuillage qui suit.
- nous effectuons un seuillage de l'image sur une distance supérieure à d_{board}, ensuite, nous effectuons un processus de détection de contour. Puisque nous allons utiliser l'image

12. Ce phénomène apparaît quand le balayage du laser frappe un objet à un angle rasant, ce qui introduit une division de pixel ou une ombre, en pratique ceci se traduit par l'apparition d'un point à mi-chemin entre le coin rasé et un objet à l'arrière-plan.

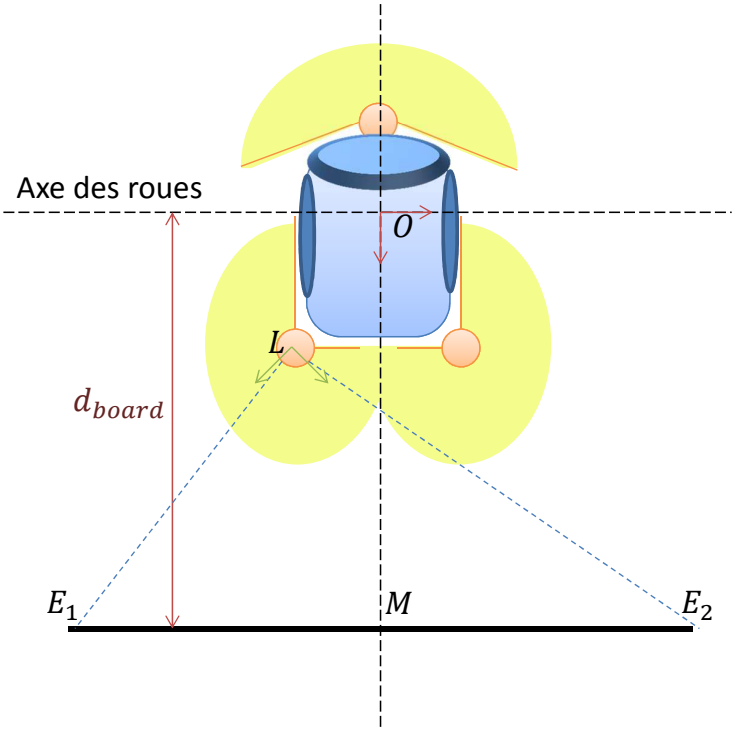


Figure 3.5 Calibration de laser pour le FRMI

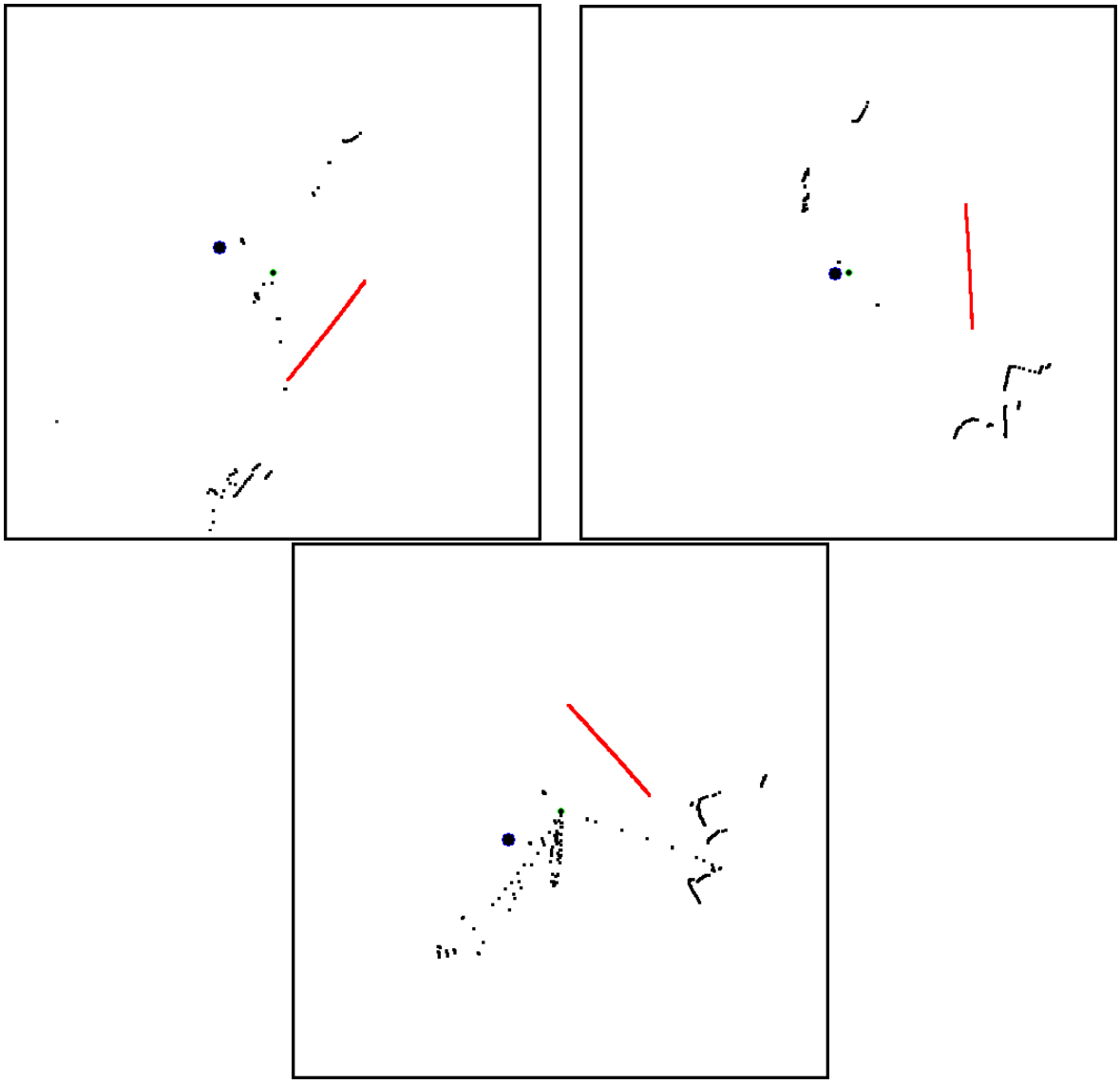


Figure 3.6 Résultat de la calibration des trois télémètres lasers, respectivement : à gauche, à l'arrière et à droite

finale pour faire des calculs de distance nous voulons garder au maximum la taille de la planche (l'objet de calibration). Donc, pour détecter la planche nous allons nous baser sur l'hypothèse qu'elle est le plus grand objet détecté après seuillage, en appliquant le minimum de méthode morphologique (fermeture, ouverture) pour essayer d'enlever le bruit résiduel dans l'image.

– une fois la planche identifiée, nous pouvons extraire les coordonnées de ses deux bords E_1 et E_2 (E_1 étant le premier bord vu dans le sens positif) dans le repère lié au laser R_{laser}. On note M le milieu de la planche $[E_1E_2]$, L le centre du laser (l'origine de R_{laser}) et O le centre du robot (voir figure 3.5). Ainsi, nous avons les deux équations :

$$\overrightarrow{OM} \cdot \overrightarrow{E_1E_2} = 0 \tag{3.3}$$

$$\overrightarrow{OM} \otimes \overrightarrow{E_1E_2} = d_{board} \left\| \overrightarrow{E_1E_2} \right\| \overrightarrow{z} \tag{3.4}$$

– la résolution de ces deux équations dans le repère laser nous donne [13] [14] :

$$O = \begin{pmatrix} \frac{E_{1x}+E_{2x}}{2} - d_{board} \frac{E_{2y}-E_{1y}}{\left\| \overrightarrow{E_1E_2} \right\|} \\ \frac{E_{1y}+E_{2y}}{2} + d_{board} \frac{E_{2x}-E_{1x}}{\left\| \overrightarrow{E_1E_2} \right\|} \\ \arctan\left(\frac{\bar{d}_{board}(E_{2y}-E_{1y})}{-\bar{d}_{board}(E_{2x}-E_{1x})} \right) \end{pmatrix}_{R_{Laser}} \tag{3.5}$$

– une transformation de repère nous donne :

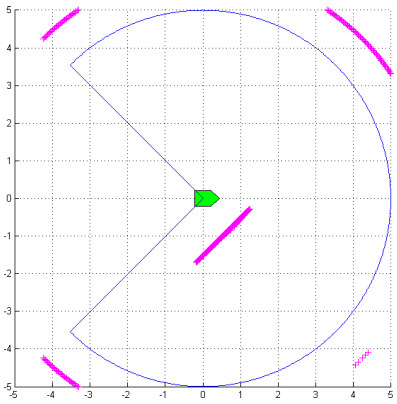
$$L = \begin{pmatrix} -O_x cos(O_\theta) - O_y sin(O_\theta) \\ O_x sin(O_\theta) - O_y cos(O_\theta) \\ -O_\theta \end{pmatrix}_{R_{FRMI}} \tag{3.6}$$

– le programme modifie automatiquement l'URDF du FRMI avec les valeurs calculées, en effectuant une lecture du fichier URDF du FRMI.

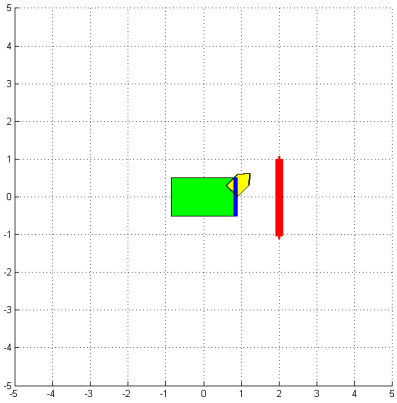
Pour illustrer le calcul précédent, nous avons effectué une simulation sous MATLAB des trois cas possibles : télémètre laser à droite, à gauche et en arrière (voir figure 3.7). Les erreurs remarquées viennent de la résolution de l'image traitée, erreur que nous allons rencontrer dans la calibration réelle, sachant que nous ne pouvons pas choisir une résolution supérieure à celle des télémètres laser. En pratique, nous allons nous contenter d'une résolution de 64 pixels par mètre.

13. Un point M dans le plan est présenté par ces coordonnes M_x,M_y et l'orientation autour de z, M_θ

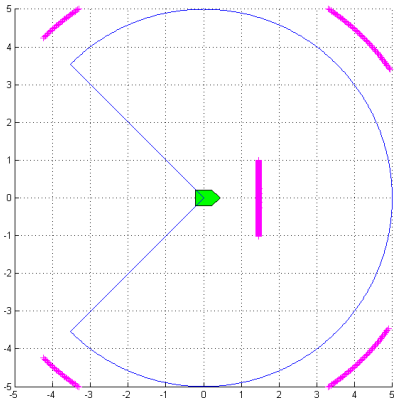
14. $\bar{d}_{board}$ représente la distance algébrique entre la planche et le FRMI, si la planche est devant $\bar{d}_{board} = d_{board}$, si la planche est en arrière $\bar{d}_{board} = -d_{board}$.



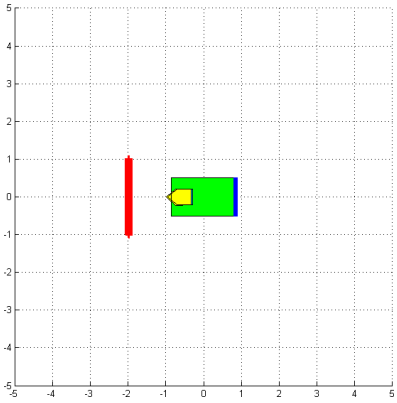
(a) Simulation du balayage effectué par le laser à gauche



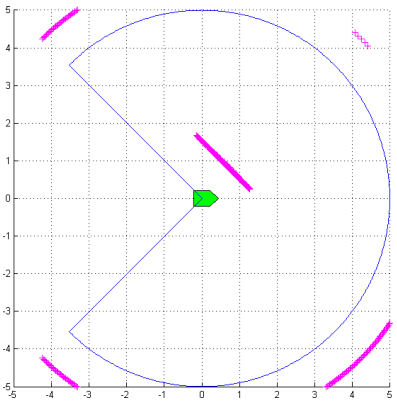
(b) Calibration : résultat du calcul de la position du laser à gauche



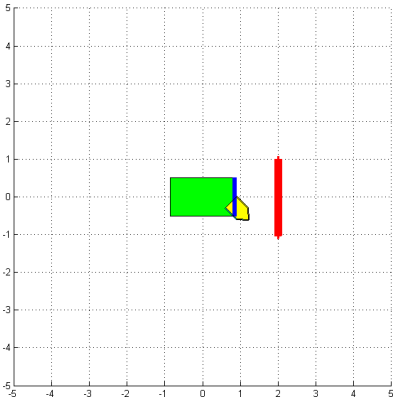
(c) Simulation du balayage effectué par le laser à l'arrière



(d) Calibration : résultat du calcul de la position du laser à l'arrière



(e) Simulation du balayage effectué par le laser à droite



(f) Calibration : résultat du calcul de la position du laser à droite

Figure 3.7 Résultat de la simulation de la calibration des trois télémètres lasers

3.4 Intégration des sonars

Bien que les télémètres lasers fournissent une grande précision et un large champ de vision, ils restent insensibles aux matériaux transparents, chose qui peut devenir un très grand défaut si nous naviguons dans un milieu vitré, ce qui est le cas dans l'ensemble des bâtiments actuels.

Pour remédier à ce problème nous allons utiliser des sonars qui sont des télémètes basés sur les ondes ultrason. Ils se caractéristent par un champ de vision restreint et une portée moyenne. Nous disposons du Devantech SRF08 qui a une portée de 3cm à 6m, et une fréquence de balayage d'environ 15Hz.

Le seul problème d'un sonar est que les ondes envoyées ne sont pas ponctuelles, mais se propagent dans l'espace. Dans notre cas, le SRF08 a le modèle de propagation présenté dans la figure 3.8 ; cette forme est habituellement confondue à un cône d'angle 55 degrés. Par conséquent, ce que le sonar mesure n'est qu'une moyenne sur ce cône des obstacles perçus.

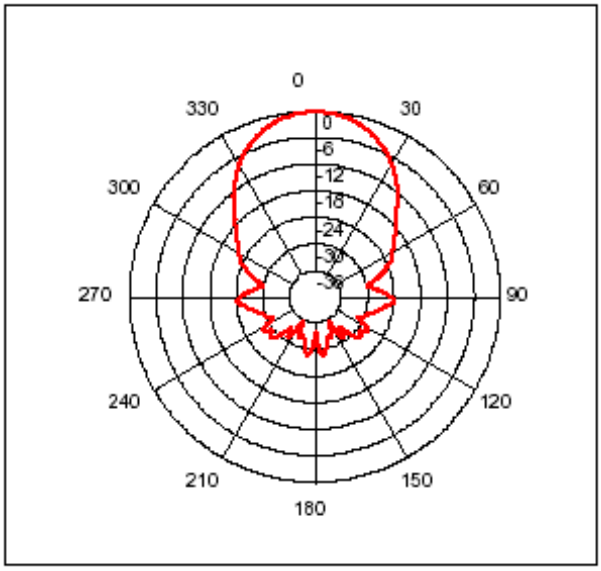


Figure 3.8 Modèle de propagation d'une onde ultra sonore

Sans pilote spécifique du SRF08 pour ROS, nous avons créé un programme permettant la lecture des données envoyées par les sonars en utilisant un protocole de communication série. Le programme a été conçu pour supporter un nombre variable de sonars ; en effet, il prend comme entrée le nombre de sonars relié au FRMI, leurs adresses respectives et les noms de leurs repères de référence [15].

15. Nous rappelons que chaque objet dans ROS doit être associée à un repère, donc les repères de référence de chaque sonar doivent être déclarés en avance dans l'URDF.

Une fois les sonars bien déclarés, l'algorithme suit le cheminement suivant :

- envoi d'une onde ultrason pour chaque sonar ;
- après un temps d'attente [16], lecture de la valeur donnée par les capteurs ;
- envoi de cette donnée sous la forme d'un message ultrason de ROS, puis envoi d'une seule donnée regroupant toutes ces lectures sous forme d'un nuage de points représentant les endroits probables des objets.

La figure 3.9 montre un résultat de lecture des télémètres ultrason, superposé sur les données de télémètre laser. Nous constatons clairement les erreurs commises par les mesures des sonars ; ceci peut induire des problèmes lors de la construction de la carte.

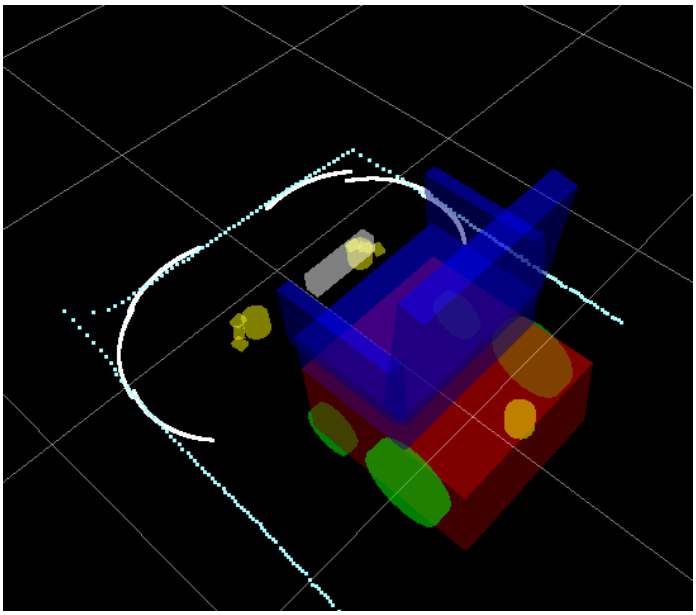


Figure 3.9 Exemple d'une acquisition de données de télémètre ultrason en blanc, superposée sur ceux du laser en bleu

3.5 Intégration de la caméra Kinect

Dans le but d'explorer des solutions fournies par de nouvelles technologies, nous nous sommes intéressés à la caméra Kinect. En plus de son faible coût, cette caméra stéréoscopique donne une mesure de distance aux objets présents dans son champ de vision ; elle peut donc fournir une potentielle alternative aux télémètres classiques présents sur le marché.

ROS contient un pilote en code source libre permettant l'acquisition et le traitement des données envoyées par la caméra Kinect, ainsi, que leur transmission sous une forme de nuages

16. Le temps d'attente doit être supérieur à 65 ms, temps minimal de la lecture de capteur.

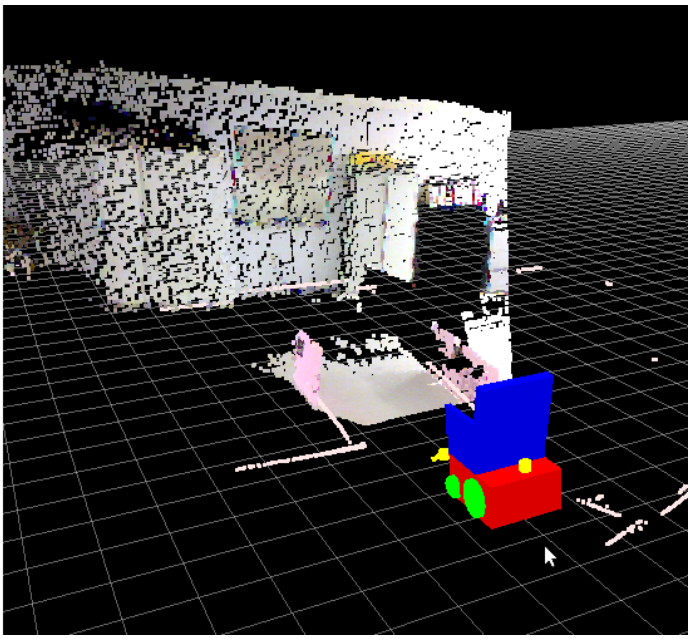


Figure 3.10 Mesure de nuage de points fourni par la caméra Kinect

de points. La figure 3.10 présente un résultat de lecture de données, chaque point dans le nuage est mis avec sa couleur réelle en RGB. Pour illustrer la précision des nuages de points, ils sont superposés à ceux du télémètre laser.

3.6 Génération des données laser et des nuages de points

Nous avons vu que nous disposons de différents types de capteurs de différentes quantités ayant différentes caractéristiques. La question qui se pose maintenant est comment pouvons-nous fusionner toutes ces entrées facilement et efficacement ? Avant de répondre à cette question, examinons de quel type de données nous aurons besoin pour la navigation du FRMI.

D'une part, pour faire une cartographie de l'environnement nous n'avons besoin que des données sur les obstacles sur un plan horizontal parallèle au sol, c'est-à-dire, une donnée de type balayage laser « laser scan ».

D'autre part, pour les évitements d'obstacles, nous avons besoin d'avoir une vision entière sur l'espace ; il est donc intéressant d'avoir une vision en 3D de l'environnement, c'est-à-dire, une donnée de nuage de points « point cloud ».

Nous allons donc présenter un outil permettant la fusion de tous les capteurs pour obtenir au choix l'une de ces données. Ce programme prend comme entrée le type et le nombre de

capteurs que nous voulons fusionner, ainsi que le type de données résultantes et ses caractéristiques. Ceci se fait d'une manière simple dans un fichier de configuration compréhensible par ROS [17]. En effet, l'algorithme effectue la démarche suivante :

- synchroniser la réception des données de chaque capteur avec la fréquence désirée. La fréquence de publication ne peut dépasser la fréquence de la source la plus lente. En général, c'est la caméra Kinect la plus lente : 15Hz.
- transformer toutes les données reçues vers un seul repère, en pratique R_{FRMI}. Pour les transformations nous passons par des quaternions pour accélérer le calcul :

$$P_{out} = q_{capteur} * P_{in} * inverse(q_{capteur}) + v_{capteur} \tag{3.7}$$

avec $P_{in} = \begin{pmatrix} x_{in} \\ y_{in} \\ z_{in} \end{pmatrix}$ le point vu par le capteur dans son repère, $*$ le produit d'un quaternion par un vecteur, calculé comme si le vecteur est un quaternion de partie scalaire nulle, $q_{capteur}$ le quaternion associé à la rotation entre les deux repères, $inverse(q_{capteur})$ l'inverse du quaternion $q_{capteur}$ ou sa conjuguée dans le cas d'un quaternion unitaire (notre cas) et $v_{capteur}$ le vecteur de translation entre les deux repères.

Il se trouve que dans le cas des télémètres lasers et ultrasons cette formule a une forme plus simple. En effet, puisque les télémètres sont disposés sur un plan parallèle à la surface, leurs orientations par rapport au centre du FRMI R_{FRMI} se résume à un angle de lacet ψ_{laser} selon l'axe z. d'où la forme simple du quaternion [18]. :

$$q_{laser} = \begin{pmatrix} 0 \\ 0 \\ sin\left(\psi_{laser}/2\right) \\ cos\left(\psi_{laser}/2\right) \end{pmatrix} \quad \text{et} \quad inverse(q_{laser}) = \begin{pmatrix} 0 \\ 0 \\ -sin\left(\psi_{laser}/2\right) \\ cos\left(\psi_{laser}/2\right) \end{pmatrix}$$

17. ROS peut lire dans un fichier de configuration de type XML.
18. Pour le quaternion nous utilisons la même convention que ROS, à savoir, mettre le terme scalaire à la fin du vecteur : $q = \begin{pmatrix} q_x \\ q_y \\ q_z \\ q_w \end{pmatrix} = \begin{pmatrix} u_x sin(\alpha/2) \\ u_y sin(\alpha/2) \\ u_z sin(\alpha/2) \\ cos(\alpha/2) \end{pmatrix}$ avec $u = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix}$ le vecteur de rotation et α l'angle de rotation.

Avec cette forme de quaternion la formule de transformation devient [19] :

$$x_{out} = x_{in}\left(q_{laser_w}^2 - q_{laser_z}^2\right) - y_{in}q_{laser_z}q_{laser_w} + v_{laser_x} \quad (3.8)$$

$$y_{out} = x_{in}q_{laser_z}q_{laser_w} - y_{in}\left(q_{laser_w}^2 - q_{laser_z}^2\right) + v_{laser_y} \quad (3.9)$$

$$z_{out} = v_{laser_z} \quad (3.10)$$

Dans le cas de la caméra Kinect, les points vus sont inversés par rapport au repère caméra, soit un roulis de $-\pi/2$ et un lacet de $-\pi/2$. De plus, pour modifier son champ de vision la caméra peut tourner avec un angle de lacet ou un angle de tangage par rapport au repère R_{FRMI}, ce qui ne permet pas d'obtenir une forme simplifiée de l'équation 3.7.

- effectuer un filtrage suivant la trace du FRMI (voir la procédure décrite à la section 3.6.1).
- sauvegarder les points perçus pour générer une seule donnée par rapport au repère principal choisi.

Nous notons que pendant la génération d'une donnée de type balayage laser, nous allons considérer la plus petite valeur vue par l'ensemble des capteurs ; ceci signifie que si le FRMI perçoit un obstacle transparent, la donnée fournie par le télémètre ultrason sera prioritaire à celle d'autres capteurs.

La figure 3.11 illustre un résultat de fusion des trois télémètres lasers : à gauche, chacune des données des trois lasers est présentée par une couleur, à droite, la donnée du balayage laser résultante est présentée en blanc. Nous pouvons constater que l'algorithme effectue la fusion en ne gardant que les points proches.

3.6.1 Filtrage suivant la trace du FRMI

Avec la disposition des capteurs laser, ultrason et caméra Kinect, il se peut que ces capteurs perçoivent des parties du FRMI ou de l'usager. Ceci devient un grand problème de navigation puisque le FRMI verra toujours un objet « fictif » (lui-même) dont il ne doit pas se soucier en temps normal. Pour cette raison, nous appliquons un filtre ayant pour objectif d'enlever les points vus à l'intérieur d'un polygone représentant la trace du FRMI.

La figure 3.12 présente le modèle de la trace au sol du FRMI. Cette trace peut changer selon le FRM. Le programme implémenté se base sur un algorithme connu testant si un point est à l'intérieur d'un polygone (MacMartin, 1992). En effet, une façon simple et rapide de trouver si le point est à l'intérieur ou à l'extérieur d'un polygone (convexe, ou concave) consiste à tester combien de fois un rayon partant de ce point et allant dans une direction

19. On remarque que cette formule est exactement la même formule habituelle d'une rotation planaire avec un angle ψ_{laser}

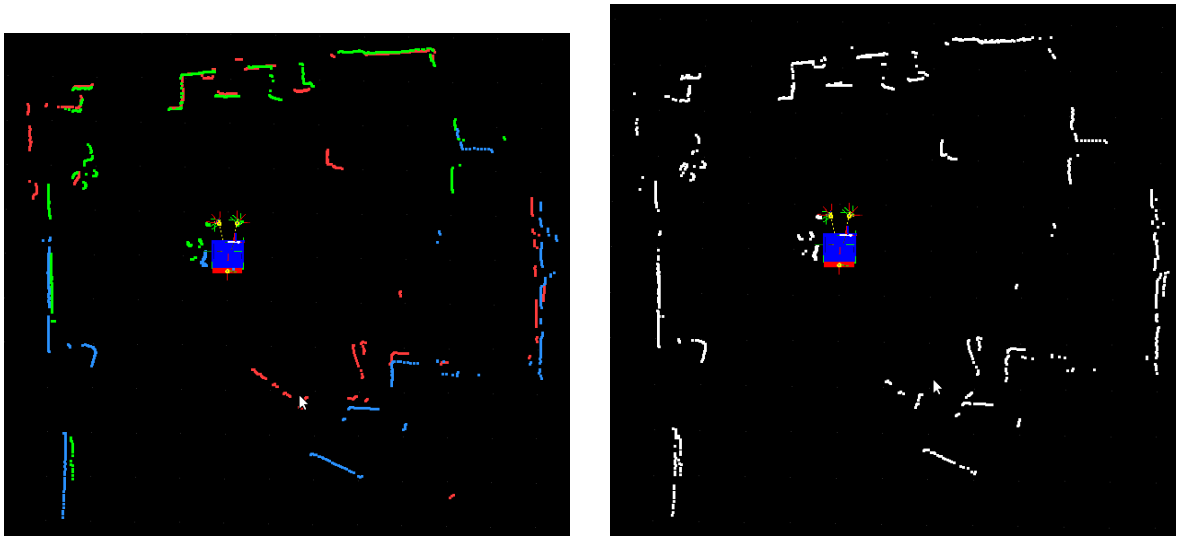


Figure 3.11 Résultats de la fusion des données laser : à gauche les trois données des télémètres lasers superposées, à droite la donnée laser résultante.

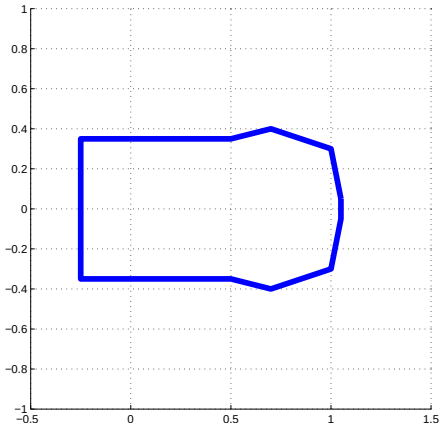


Figure 3.12 Trace au sol du FRMI

fixe (souvent l'un des axes principaux), coupe les bords du polygone. Si le nombre est impair, le point est à l'intérieur sinon il est à l'extérieur [20].

3.6.2 Autres filtres

Outre le filtrage selon la trace du FRMI, d'autres filtres peuvent être appliqués pour enlever d'autres types de bruit. Par exemple :

- nous disposons d'une implémentation d'un filtre enlevant le bruit causé par les côtés des objets. Ce filtre élimine les lectures laser qui sont très probablement causées par la réflexion sur le bord d'un objet. En effet, pour deux points qui se suivent P_1 et P_2, nous calculons l'angle entre la source du laser O, P_1 et P_2 : $\angle OP_1P_2$. Si cet angle est à l'extérieur d'un intervalle donné, nous enlevons tous les voisins proches de ce point. En général cet angle doit être compris entre 10 et 170 degrés.
- nous disposons aussi d'un filtre interpolant la valeur d'un point mesuré en fonction de ces voisins proches.

Nous notons que pendant les expérimentations, nous n'avons pas utilisé ces deux filtres vu leur effet minimal dans le cadre de nos expérimentations.

3.7 Conclusion

Ce chapitre a fourni un aperçu technique du FRMI, ceci en détaillant les composantes constituant le prototype FRMI que nous avons développé. Nous soulignons la façon avec laquelle nous exploitons la maniabilité du modèle du FRM dans ROS pour fournir un prototype modulaire capable de s'adapter aux expériences futures. Aussi, nous avons fourni un service intuitif pour calibrer les positions des capteurs par rapport au repère de base du FRMI R_{FRMI}. Enfin, nous avons présenté notre stratégie de fusion de données de capteurs, permettant d'avoir les deux types de données clés : balayage laser et nuages de points, à partir des capteurs sélectionnés.

20. Nous notons l'existence de quelques cas particuliers traités séparément, e.g. quand la droite choisie passe par un coin de polygone.

CHAPITRE 4

ARCHITECTURE DE CONTRÔLE DU FRMI

Maintenant que nous connaissons les caractéristiques du robot mobile que nous traitons, nous allons pouvoir appliquer et développer des méthodes et algorithmes de navigation. Nous commençons par détailler l'architecture que nous avons prévue pour le module de navigation, une architecture centrée sur un superviseur, puis divergente sur trois grandes branches : un mode manuel permettant la navigation avec une interface de contrôle (par exemple une manette de contrôle), un mode semi-manuel assurant l'exécution de manœvres préprogrammées et un mode automatique pour la navigation globale dans une carte. Par la suite, nous allons introduire le module d'évitement d'obstacle permettant d'interpréter les informations de télémétrie sur l'environnement pour fournir des commandes sécuritaires s'éloignant des obstacles. Enfin, nous allons traiter le problème de cartographie en détaillant la solution que nous avons adoptée.

4.1 Architecture de contrôle globale

Puisque nous travaillons sur un module de contrôle prévu pour l'utilisation d'un large catalogue d'usagers, nous devons respecter un ensemble de contraintes ergonomiques et sécuritaires.

Tout d'abord, pour une utilisation facile, nous avons simplifié les moyens d'interactions avec le FRMI :

- Il faut rendre l'utilisation du contrôleur joystick assez intuitive et facile pour l'usager, ceci en interprétant directement les deux axes du contrôleur comme les deux commandes de vitesses principales : avancer, tourner, mais aussi, en rajoutant une petite corrélation entre ces deux actions pour faciliter les mouvements combinés. Les boutons du contrôleur sont utilisés comme déclencheur des tâches habituelles, mais aussi comme un arrêt d'urgence premier niveau.
- Il faut profiter de l'interface utilisateur sur écran tactile, elle permet d'avoir une vision du FRMI et son environnement ainsi qu'interpréter les directives données par l'usager.

Ensuite, pour une utilisation sécuritaire nous devons prévoir un module maître. Agissant comme un agent de route, ce module aura comme mission de permettre le lancement d'une tâche ou son arrêt, surveiller l'avancement de cette tâche, ainsi qu'interdire le chevauchement de deux actions simultanées. Dans la suite, nous allons appeler ce module : superviseur.

Enfin, pour simplifier le développement du module de navigation nous l'avons construit d'une façon modulaire, chaque action principale aura son module qui sera lié au superviseur, et nous pourrons rajouter d'autres fonctionnalités dans le futur. La figure 4.1 présente l'architecture du module de navigation.

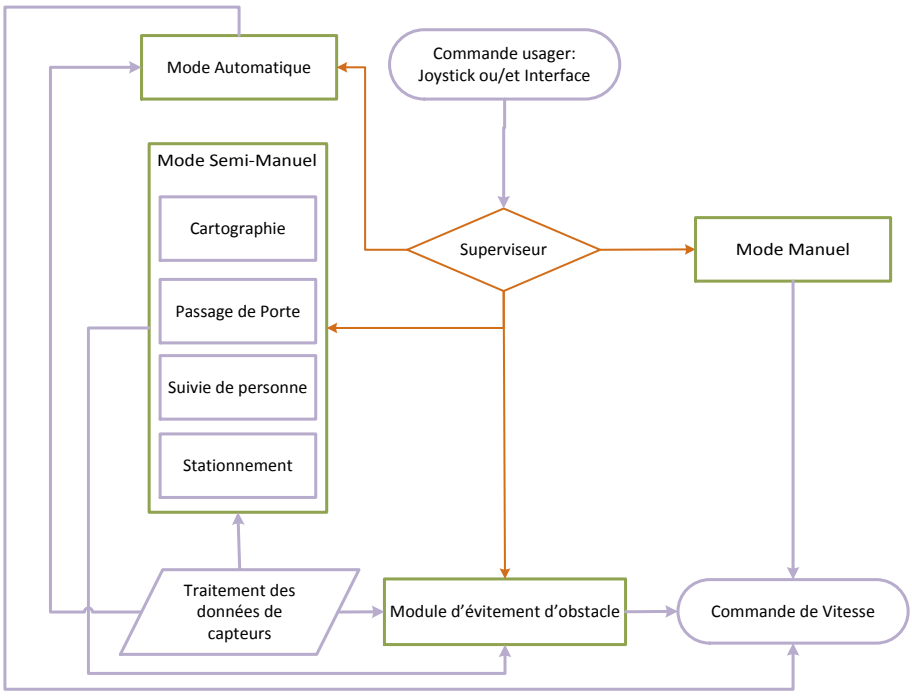


Figure 4.1 Présentation de l'architecture de contrôle du module de navigation pour FRMI

4.1.1 Le superviseur

Le superviseur, en rouge dans la figure 4.1, est l'entité responsable de la gestion de tous les modules participant aux navigations (mode manuel, mode semi-manuel, actions spécifiques et mode automatique). Il se caractérise par un niveau d'autonomie variable dépendamment de la situation :

– l'usager peut demander au superviseur d'effectuer une tâche particulière en utilisant le contrôleur ou l'interface. Chaque bouton du contrôleur peut être configuré pour représenter le déclenchement ou l'arrêt d'une action particulière, de plus, le superviseur est pensé pour qu'un développeur puisse rajouter ou modifier la configuration des boutons avec un simple fichier de données XML (.launch). Dans ce même fichier, nous pouvons

spécifier les limitations de vitesse que le FRMI doit respecter.

– le superviseur peut décider d'annuler une action dangereuse sans le demander à l'usager, par exemple interdire le choix de deux modes instantanément, ou interdire de dépasser une vitesse maximale.

Plus en détail, le superviseur acquiert en continu les commandes envoyées par l'usager, puis met à jour un message interne (voir le tableau 4.1) contenant l'état du module de navigation ; ce message sera publié et mis à disposition de tous les autres modules de contrôle. Ainsi, les autres modules doivent lire l'état de ce message avant de procéder à n'importe quelle manipulation. Pour plus d'efficacité nous avons choisi de l'implanter comme un service dans ROS [1].

Tableau 4.1 Forme du service fourni par le superviseur

Booléenne : activation du mode manuel
Booléenne : activation du mode semi-manuel
Booléenne : activation du mode automatique
Booléenne : activation du service : suivre une personne
Booléenne : activation du service : passage de la porte
Booléenne : activation du service : stationnement
Booléenne : activation du service : suivi de chemin

En plus de publier l'état du module de navigation, le superviseur publie un message de commande de vitesse standard à ROS. En fait, le superviseur écrira dans deux messages différents : un message destiné pour l'envoi direct au FRMI et un autre message qui sera traité par le module de l'évitement d'obstacle avant de l'envoyer au FRMI (voir figure 4.2).

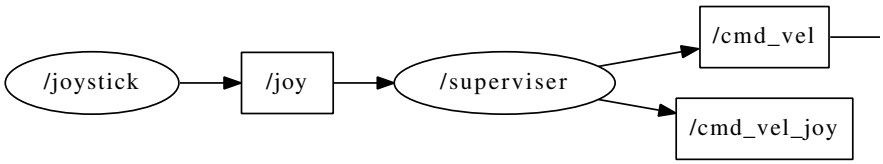


Figure 4.2 Présentation du nœud superviseur

1. Un service sous ROS est un message spécial qui permet de gérer rapidement les réponses automatiques à des questions prédéfinies, par exemple, un service peut être configuré pour fournir le nom ou l'état du nœud si un autre nœud en fait la demande.

4.1.2 Le mode manuel

Le mode manuel permet la navigation normale dans l'environnement à l'aide des valeurs de commande de vitesse linéaire et angulaire fournie par l'usager. Dans ce mode nous visons à commander le FRMI sans nous soucier de l'environnement extérieur. Le module de navigation manuelle est concentré dans le nœud « virtualFRMI » (robot virtuel). En plus de contenir les informations sur l'URDF du FRMI (voir section 3.2.3) ce nœud permet de :

- acquérir les valeurs d'encodeurs optiques montés sur les roues motrices, ce qui est assuré avec une communication Ethernet avec le contrôleur Galil [2] relié au réseau local ;
- assurer la communication avec le microcontrôleur responsable d'envoyer les commandes de couple à exercer sur les moteurs. Les commandes de vitesse sont envoyées directement à un microcontrôleur par une liaison usb-série. Ce microcontrôleur programmé au laboratoire permet d'interpréter des valeurs de gains pour les vitesses linéaires et angulaires ;
- calculer (voir les équations 4.1-4.2-4.3) et publier les valeurs de l'odométrie en se basant sur les valeurs de vitesses (linéaire (v_{lin}) et angulaire (v_{ang})) mesurées. En fait, avec ROS nous allons publier deux messages : un premier message d'odométrie classique « *odom* » utilisé pour la visualisation et pour d'autres algorithmes de navigation (cartographie par exemple), et un deuxième message contenant l'information sur le mouvement (déplacement et orientation) entre le repère principal du FRMI R_{FRMI} et le repère fixe R_0.

$$x_k = x_{k-1} + v_{lin} cos(\theta_{k-1}) dt \tag{4.1}$$

$$y_k = y_{k-1} + v_{lin} sin(\theta_{k-1}) dt \tag{4.2}$$

$$\theta_k = \theta_{k-1} + v_{ang} dt \tag{4.3}$$

- estimer la rotation ϕ des deux roues folles présentes sur l'avant du FRMI. Selon Campion *et al.* (1996) les contraintes cinématiques sur ces deux roues imposent que :

$$\begin{bmatrix} cos(\alpha+\beta) & sin(\alpha+\beta) & d+lsin(\beta) \end{bmatrix} \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{lin} cos(\theta) \\ v_{lin} sin(\theta) \\ v_{ang} \end{bmatrix} + d\dot{\beta} = 0$$

$$-\frac{v_{lin} cos(\alpha+\beta) + (d+lsin(\beta)) v_{ang}}{d} = \dot{\beta}$$

avec α, l et d des constantes présentées sur la figure 4.3. Nous remarquons que si

2. Galil est le contrôleur des moteurs de base du FRM, c'est un ancien modèle de contrôleur permettant un ensemble de fonctionnalité de base comme la lecture des entrées, calcul d'un PID et applications des filtres.

$v_{ang} = 0$ et $v_{lin} \neq 0$, β atteint le point d'équilibre prévu $\pm\frac{\pi}{2} - \alpha$.

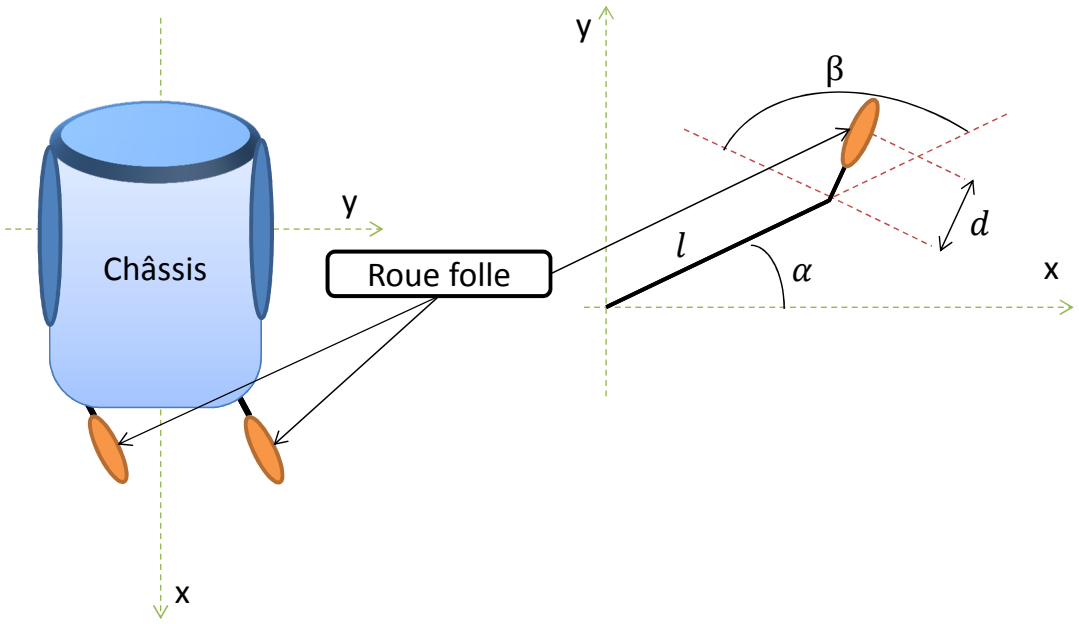


Figure 4.3 Présentation des roues folles

- appliquer un contrôle PID visant à ramener le FRMI aux vitesses demandées tout en limitant les accélérations linéaires et angulaires (voir section 4.1.2).
- autoriser l'influence de la valeur de la vitesse linéaire sur la vitesse angulaire, i.e. nous allons changer le rayon de courbure en virage dépendamment de la vitesse linéaire. Ceci permet de faciliter le contrôle de l'usager sur le FRMI.
- fournir un service de calibration des gains envoyés aux roues motrices (voir section 4.1.2)

Nous notons que toutes les commandes envoyées au mode manuel passent d'abord par le superviseur, ce qui permet de contrôler en tout temps le flux et le type de commandes envoyées par l'usager.

Contrôle PID

Plusieurs paramètres incertains peuvent causer des comportements imprévisibles sur le FRMI, par exemple les roues folles, la forme et l'inclinaison du sol ou la masse de l'usager. Malgré ces contraintes nous souhaitons répondre conformément et instinctivement à la commande de l'usager. Pour cela, nous implémentons un contrôleur PID sur les vitesses linéaire et angulaire. Nous nous sommes largement inspirés du PID proposé par Boucher (2010). Ce

PID visait simplement à compenser les effets des roues folles ainsi que générer une commande croissante forçant l'obtention de la valeur de vitesse désirée ; par contre, nous avons rajouté une saturation sur les accélérations linéaires et angulaires, ce qui nous permettra d'éviter les mouvements brusques et de rajouter un temps d'établissement de la commande. La figure 4.4 présente une forme résumée du PID.

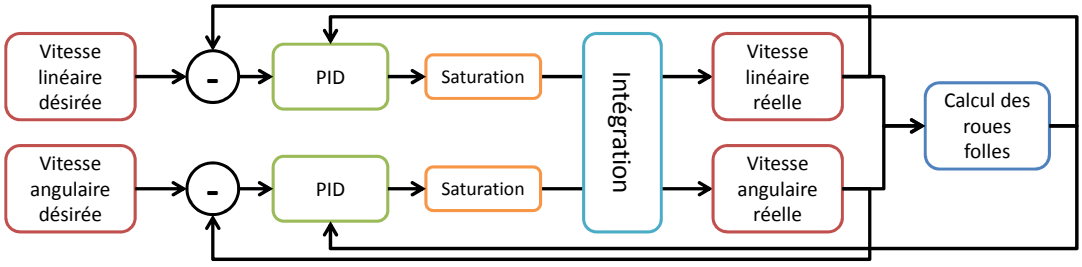


Figure 4.4 Schéma du contrôleur PID

Nous avons simulé le comportement du PID sur MATLAB avant de l'implémenter sur ROS ; ceci nous a permis d'obtenir une valeur approximée des gains PID. Les figures 4.5 montrent l'ensemble des résultats obtenus. Nous appliquons une commande échelon instantanée de vitesse et nous traçons les valeurs de vitesse obtenue. Pour avoir un tel résultat, nous avons choisis les gains pour assurer un temps de réponse dans les alentours d'une seconde, avec une accélération limite de $1m/s^2$.

Calibration de l'odométrie

Lorsqu'un usager envoie une commande de vitesse, il s'attend à une réaction proportionnelle du FRMI. Nous voulons quantifier cette proportionnalité et lui donner un sens plus explicite.

À cause de la non-linéarité du comportement du FRMI dû à la présence des frictions statiques et cinématiques, engendré par les roues folles et le type du sol, il est difficile de préciser un modèle dynamique reliant la force envoyée aux roues motrices [3] et la vitesse effective du FRMI (en m/s). Pour résoudre ce problème, nous le linéarisons autour plusieurs points d'intérêt ; en effet, chaque gain envoyé correspond à une vitesse du FRMI, une fois

3. Lorsque nous parlons d'une force ou d'un couple exercés sur les roues motrices, nous parlons de l'intensité de courant que le contrôleur GALIL enverra aux roues matrices, plus précisément c'est un gain entre -100 et 100.

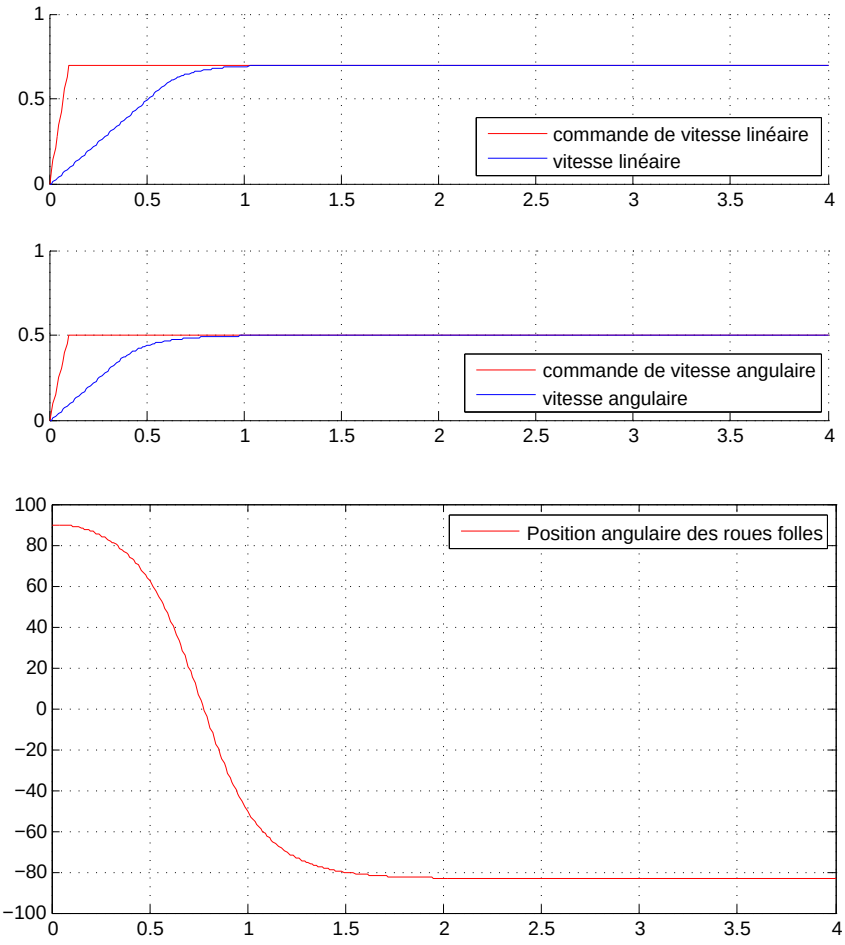


Figure 4.5 Résultat de la simulation de la commande PID de la vitesse

cette relation établie nous pouvons faire une identification inverse pour déterminer la force à appliquer pour aller à la vitesse précise.

Il suffit alors de construire une table reliant la vitesse réelle et la commande (en force) à envoyer au contrôleur GALIL. Pour ce faire, nous allons effectuer une procédure de calibration. La calibration consiste simplement à avancer, reculer puis tourner à gauche et à droite avec des vitesses croissantes, un programme permet de sauver automatiquement les valeurs de vitesses mesurées par odométrie dans un fichier MATLAB[4]. En plus, nous avons implémenté cette méthode sous la forme d'un service : une fois le service appelé, le FRMI rentre en mode calibration et effectue la procédure.

4.1.3 Le mode semi-manuel

Le mode semi-manuel consiste en un ensemble de manœuvres prédéfinies visant à satisfaire une tâche spécifique. Chaque mode peut être associé à un bouton du contrôleur. Bien que ce mode ait été prévu dans notre architecture de base, il a été conçu pour y inclure les anciens services présents dans Acropolis et pour qu'il soit enrichi dans le futur par d'autres développeurs. À l'heure actuelle, ce mode contient les fonctionnalités suivantes :

Traverser une porte : Ce service consiste à automatiser les traversées de porte considérées comme une tâche répétitive et assez délicate pour l'usager. Largement inspiré par l'ancien module de navigation semi-autonome proposé par (Boucher, 2010) ce service reprend les grandes lignes de son algorithme de détection de porte et de génération de trajectoire.

Stationner : Ce service permet de se rapprocher intelligemment d'un obstacle à proximité, pour stationner sur son bord.

Suivre une personne : Ce service permet de détecter la présence des personnes à proximité, choisir une personne devant, derrière, à gauche ou à droite, puis essayer de suivre cette personne.

Toutes ces manœuvres sont caractérisées par les deux étapes suivantes : d'abord un algorithme spécifique à la tâche permet de générer une trajectoire effectuant la manœuvre demandée, puis une fois la trajectoire générée, le module d'évitement d'obstacle s'occupe d'assurer son suivi. Nous nous sommes plus intéressés à la dernière étape dans le module d'évitement d'obstacle.

4. Le fichier MATLAB sera traité séparément dans un deuxième temps, pour extraire la table des gains pour GALIL.

4.1.4 Le mode automatique

Le dernier mode dans l'architecture de contrôle est le mode automatique qui s'occupe de la planification et la navigation dans une carte. Il permet à l'usager de choisir un endroit sur la carte pour que le FRMI s'y rende. Ceci n'étant pas un objectif final de notre recherche, nous n'avons pas développé les interfaces nécessaires pour ce mode. Par contre, le cœur actuel de ce mode est basé sur l'algorithme de Monte-Carlo pour la navigation globale (voir revue de littérature, section 2.4).

Effectivement, nous arrivons à estimer efficacement la position du FRMI à chaque instant dans une carte. La figure 4.6 montre à gauche l'estimation de la position avec odométrie seule ; nous notons que le point d'arrivée coïncide normalement dans la carte avec le point de départ. L'erreur commise à chaque coin en rotation rend très difficile l'estimation de la position ; par contre, une fois combiné avec la carte connaissant les mesures de télémétrie l'algorithme démontre une très bonne performance.

4.2 Module d'évitement d'obstacles

L'une des fonctionnalités essentielles dans une navigation locale sécuritaire est la disposition d'un moyen pour éviter les objets et les personnes présentes dans l'environnement. C'est dans cette perspective que nous avons conçu notre module d'évitement d'obstacles. Plus en détail, notre module fournit les services suivants :

- générer une commande de vitesse sécuritaire dépendamment des obstacles perçus par les capteurs et en respectant au maximum la direction voulue par l'usager : navigation assistée.
- suivre un chemin prédéfini tout en évitant les obstacles aux alentours : suivi d'une trajectoire.
- générer une commande de vitesse permettant de suivre un chemin local définie par rapport au FRMI, e.g. avancer de 1 mètre en continu : suivi d'une commande haut niveau.

Pour pouvoir fournir ces services, le module d'évitement d'obstacles procède en quatre temps. D'abord, il attend la réception d'une consigne d'une des trois formes supportées (une consigne de vitesse, un chemin global ou un chemin local) et des données de télémétries sur l'environnement. Ensuite, pour chaque entrée, il génère une trajectoire représentant une estimation de chemin désirée par l'usager. Cette trajectoire est renvoyée par la suite à une fonction permettant de juger sa faisabilité en fonction des mesures de télémétrie [5]. Enfin, une

5. Cette fonction est une implémentation d'un algorithme d'évitement d'obstacle classique. Dans notre cas nous utilisons la méthode des fenêtres dynamiques (Fox *et al.*, 1997), par contre, cette méthode peut être

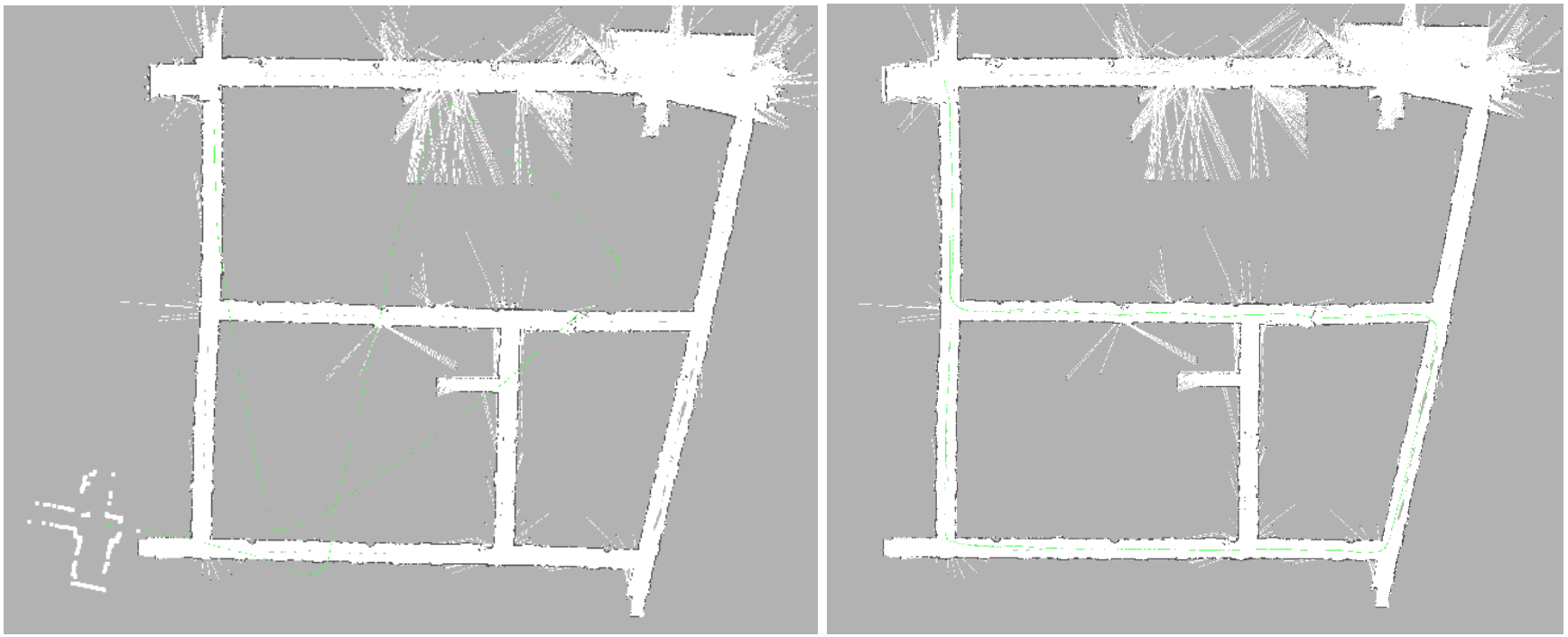


Figure 4.6 Navigation globale avec MCL : À gauche : Navigation basée sur l'odométrie. À droite : Navigation globale avec MCL basée sur la fusion de l'odométrie avec la carte disponible. L'odométrie est en vert et la position de départ et arrivée coïncide avec le coin haut/gauche. La carte est une cartographie du corridor M du cinquième étage du bâtiment Lassonde, 50 par 40 mètres.

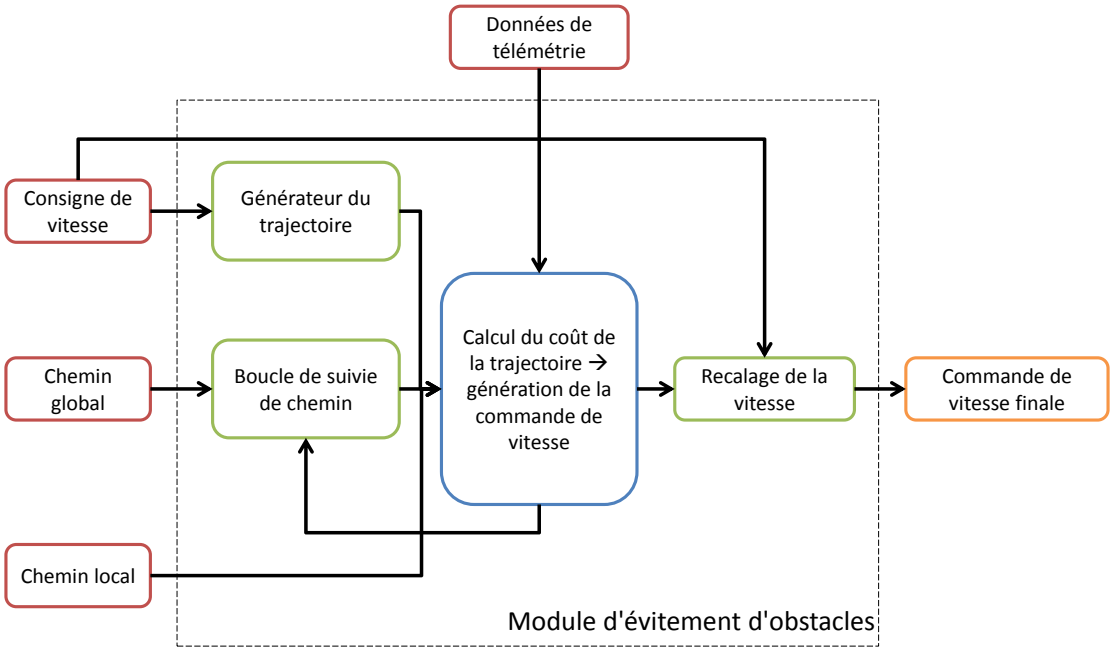


Figure 4.7 Présentation du module d'évitement d'obstacles

commande de vitesse fuyant les obstacles et suivant le chemin est générée (voir la figure 4.7).

Nous allons présenter la méthode des fenêtres dynamiques, puis nous allons détailler chacune des fonctionnalités fournies par le module d'évitement d'obstacles.

4.2.1 Méthode de la fenêtre dynamique

Plusieurs algorithmes, basés sur différents concepts (champs potentiel, histogramme de champs de vecteur ...) existent dans la littérature. Nous avons choisi d'utiliser la méthode de la fenêtre dynamique conseillée par ROS pour sa rapidité de calcul et son efficacité dans les situations difficiles : cul-de-sac, obstacle mouvant ... (voir revue de littérature section 2.2.3).

Le concept est de discrétiser l'espace de vitesse linéaire et angulaire, puis de garder la meilleure combinaison de vitesses pour satisfaire un critère de coût. Nous avons dû choisir les coefficients associés à la fonction coût, notre choix favorisant l'éloignement des obstacles et la limitation d'accélération sur le suivi de direction du mouvement assignée par l'usager. Ceci nous assure, d'une part, une grande sécurité vis-à-vis aux collisions possibles et un mouvement lisse sans des coups brusques. D'autre part, une grande flexibilité permettant de choisir une autre route, pas très évidente, dans le cas d'un minimum local. Par exemple, si le FRMI rencontre un obstacle frontal et l'usager demande d'avancer, notre choix nous permet de tourner doucement pour nous aligner sur le mur puis de continuer à avancer.

4.2.2 Navigation assistée

Cette fonctionnalité nous permet de faire une navigation assistée avec le FRMI, c'est-à-dire, laisser la priorité des consignes de vitesses à l'usager, puis d'intervenir dans le cas de manœuvres brusques ou dangereuses. Nous allons détailler l'algorithme utilisé pour fournir ce service.

À chaque instant, l'usager envoie une commande de vitesse (à l'aide du contrôleur) pour effectuer un mouvement désiré ; le module d'évitement d'obstacle commence par prévoir le mouvement du FRMI sur un cône déterminé par un rayon r_{max} et un angle θ_{max} (voir figure 4.8). Cette estimation du mouvement suppose que le FRMI garde une vitesse fixe désignée par l'usager à l'instant d'envoi de la commande t sur un intervalle de temps δt_{sim}. Cette hypothèse est valable la plupart du temps si nous considérons de petits intervalles de temps (2-3 secondes).

Ce chemin est envoyé, par la suite, au corps de l'algorithme qui lui assigne un coût dépendamment des données de télémétrie (voir section 2.2.3). Si la trajectoire a un coût supérieur à un seuil, le module autorise la commande de l'usager, sinon, le module d'évitement d'obstacle

---

modifiée selon le choix

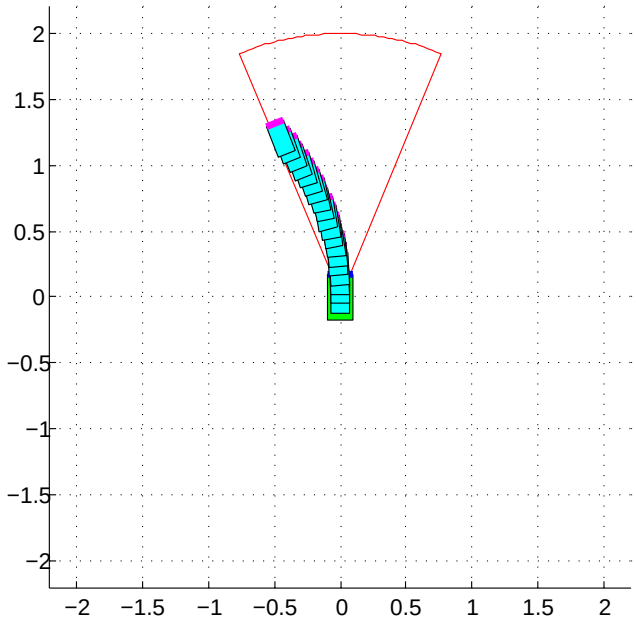


Figure 4.8 Simulation du générateur des trajectoires sous MATLAB avec une vitesse linéaire de $0.7m/s$ et une vitesse angulaire de $0.5rad/s$

propose une nouvelle commande pour fuir les obstacles détectés, ou tout simplement s'arrêter et attendre une nouvelle commande de l'usager.

Nous notons que dans le cas de la génération d'une nouvelle commande de vitesse, cette vitesse à tendance à s'éloigner le plus possible des obstacles. Nous allons donc coupler cette vitesse avec la consigne de vitesse donnée premièrement par l'usager, ceci pour garder le sentiment de contrôle à l'usager.

Les paramètres influant la navigation assistée sont configurable à l'aide d'un fichier configuration ROS, et peuvent changer selon le comportement de l'usager. Par contre, nous allons garder la configuration dans le tableau 4.2 pour les expérimentations futures.

Tableau 4.2 Configuration de la navigation assistée

rayon maximal	3.0
angle maximal	$\pi/6$
temps de simulation	3.0
nombre de points dans la trajectoire	30
corrélation de la vitesse angulaire [6]	0.2
corrélation de la vitesse lineare	0.1

4.2.3 Suivi d'une trajectoire

Nous voulons fournir un outil qui permet de suivre fidèlement une trajectoire fixe par rapport au FRMI. Cette trajectoire est assignée soit à partir d'un fichier de configuration sous la forme d'un vecteur de coordonnés x_{R_0}, y_{R_0}, soit d'une façon automatique par l'intermédiaire d'un service, par exemple, le passage d'une porte.

Une fois la trajectoire reçue, elle est envoyée au module d'évitement obstacle pour juger sa faisabilité, le programme s'arrête quand le module d'évitement d'obstacle lui signale l'arrivée à la fin de la trajectoire, ou si l'usager décide d'interrompre l'action.

4.2.4 Suivi d'une commande haut niveau

Le dernier service de suivi d'une commande de haut niveau consiste à suivre une trajectoire locale définie par rapport au repère associé au FRMI R_{FRMI}. Cette trajectoire est envoyée en continu au module d'évitement d'obstacle, et ce dernier ne fait que juger sa faisabilité et assigner une vitesse de commande pointant vers la direction de la trajectoire.

Cette façon de faire nous permet de demander au FRMI de faire des tâches intéressantes : avancer, reculer ou tourner ou suivre une personne.

4.3 Module de cartographie

Nous avons présenté dans la revue de littérature 2.5 les méthodes utilisées pour la cartographie avec des données laser, par contre, nous avons adopté l'algorithme proposé par (Grisetti *et al.*, 2007) incluant l'ensemble de ces notions [7]. L'algorithme suit le cheminement suivant :

- une première estimation $x_t^{'(i)} = x_{t-1}^{(i)} \oplus u_{t-1}$ de la pose du robot représenté par la particule i est obtenue à partir de la pose précédente $x_{t-1}^{(i)}$ de la particule par intégration des mesures d'odométrie u_{t-1} recueillies depuis la dernière mise à jour du filtre ;
- nous appliquons un algorithme d'analyse de correspondance entre la carte $m_{t-1}^{(i)}$ et la nouvelle mesure z_t à partir de l'estimation initiale $x_t^{'(i)} : \hat{x}_t^{(i)} = argmax_x p(x|m_{t-1}^i, z_t, x_t^{'(i)})$. Si l'analyse de correspondance échoue, les nouvelles poses et poids de particules sont calculés en fonction du modèle de mouvement (et les étapes 3 et 4 sont ignorés). Nous utilisons l'algorithme du plus proche point (ICP) proposé par (Censi, 2008) ;
- un ensemble de points d'échantillonnage x_k est choisi dans un intervalle autour de la pose $\hat{x}_t^{(i)} : x_k \ \left\{x_j || x_j - \hat{x}^{(i)}| < Q\right\}$ avec Q une estimation du bruit du modèle du FRMI.

6. Une corrélation de vitesse de 0.2 signifie que la vitesse finale a 20% de la vitesse initiale plus 80% de la vitesse propose par le module d'évitement d'obstacles.

7. Une implémentation libre de droits de l'algorithme « GMapping » est disponible sur le site `http://www.openslam.org/`.

À partir de ces points, la moyenne et la covariance de la proposition de distribution du filtre particulaire sont calculées pour estimer un poids w_i par $\hat{x}^{(i)}$;
- ajout de la nouvelle mesure de télémétrie z_t dans la carte ;
- échantillonnage si besoin. Normalement nous devons reéchantillonner si les poids associés à chaque particule dégénèrent rapidement, ceci se caractérise par un nombre effectif de particules $N_{eff} = \frac{1}{\sum_p log(w_p)}$.

Malgré la robustesse de cet algorithme, sa réussite dépend de plusieurs paramètres propres au FRMI :
- la valeur choisie pour la matrice de corrélation Q. En effet, le FRMI accumule des erreurs continues d'odométrie, dues essentiellement aux glissements des roues et les frottements avec les roues folles. La valeur de Q doit quantifier ces erreurs. Dans notre cas :
$$\begin{bmatrix} cov(v,v) & cov(v,w) \\ cov(w,v) & cov(w,w) \end{bmatrix} = \begin{bmatrix} 0.03 & 0.015 \\ 0.015 & 0.05 \end{bmatrix} \tag{4.4}$$
- le nombre de particules choisies. Un grand nombre augmente le temps de calcul, mais un petit nombre ne permet pas de propager suffisamment d'informations sur l'environnement. Le fait d'avoir des particules avec des poids permet de profiter des environnements avec des circuits fermés ; il suffit qu'une particule survive jusqu'au moment où le FRMI revient à un endroit déjà visité, pour que cette particule voie son poids augmenter rapidement. Nous utilisons 50 particules ;
- la qualité d'analyse de correspondance entre les données de télémétrie : le FRMI doit bouger lentement durant le processus de construction de carte pour augmenter son taux de réussite ;
- la qualité des données laser : si les données de laser sont erronées, à cause de la présence des milieux transparents ou à cause de la présence d'autres obstacles mouvants, la qualité de la carte construite est largement affectée.

4.3.1 Fusion avec une carte existante

Bien que nous disposons d'un moyen pour construire les cartes de nouveaux espaces, nous voulons profiter de l'existence de carte CAO [8] de plusieurs bâtiments. De plus, puisque nous nous basons sur les nuances de noir pour distinguer obstacle et espace libre, utiliser une carte CAO est tout à fait possible.

De plus, cette façon de faire nous permet de nous concentrer plus sur l'extraction des nouveaux détails de chaque pièce pendant la cartographie et ainsi de profiter de l'exactitude

8. Les cartes CAO sont des cartes conçues par ordinateur pour les plans de sécurité ou pour les architectes, CAD en anglais

des traits rectilignes (murs) présents dans les cartes CAO.

Sur la figure 4.9, nous présentons une carte existante et une carte construite du cinquième étage du bâtiment Lassonde de l'École Polytechnique de Montréal. Nous supposons que les cartes fournies peuvent être mises à l'échelle 5cm/pixel, bicolores noir et blanc et contiennent une information utile sur les murs.

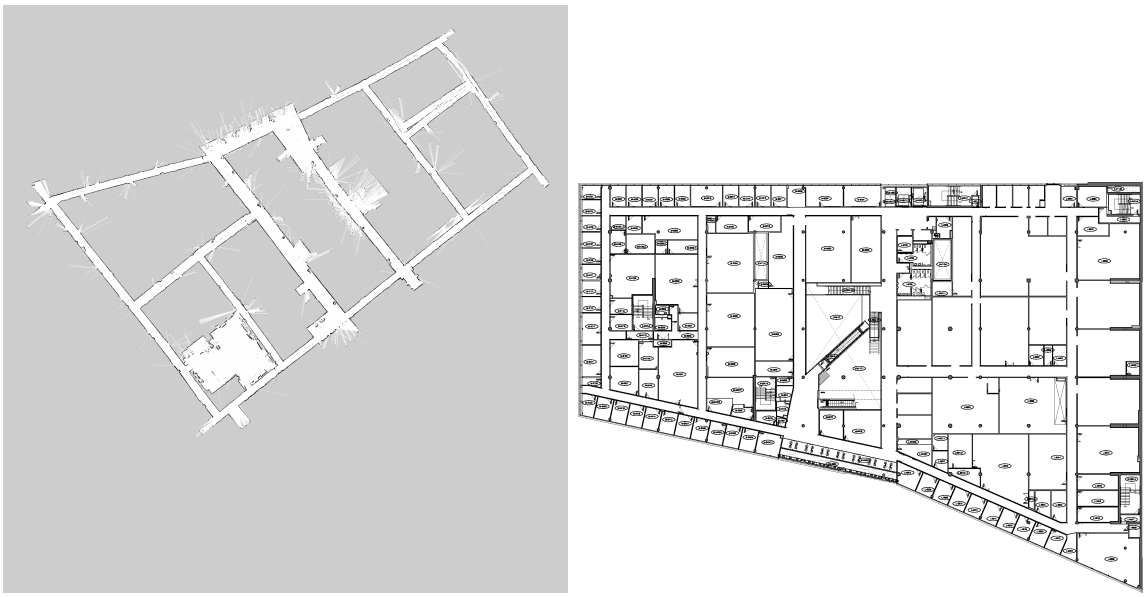


Figure 4.9 Exemple d'une carte obtenue par cartographie à gauche, une carte fournie à droite

Pour effectuer la fusion des deux cartes nous suivons l'algorithme suivant :

- nous allons commencer par orienter la carte construite par cartographie. Pour ce faire nous utilisons le transformée de Hough (Ballard, 1981) pour détecter les lignes dans la carte (voir figure 4.10).
- une fois les lignes extraites, nous cherchons les lignes orthogonales pour détecter l'orientation α de la carte. Nous appliquons alors cette orientation sur la carte (voir figure 4.11).
- par la suite, nous calculons une valeur de corrélation (Lewis, 1995) entre les deux cartes 4.5. Cette valeur atteint son maximum lorsque les deux cartes coïncident.

$$corr(u,v) = \frac{\sum_{x,y}\left[f(x,y) - \bar{f}_{u,v}\right]\left[t(x-u,y-v) - \bar{t}\right]}{\left\{\sum_{x,y}\left[f(x,y) - \bar{f}_{u,v}\right]^2 \sum_{x,y}\left[t(x-u,y-v) - \bar{t}\right]^2\right\}^{1/2}} \tag{4.5}$$

avec f la carte CAO, t la carte originale, $\bar{t}$ la moyenne de t, et $\bar{f}_{u,v}$ la moyenne de f sur la région sous la carte originale de taille (u,v).

Figure 4.10 Résultat de la transformée de Hough pour détection de ligne

– enfin nous superposons les deux cartes (voir figure 4.12).

4.4 Conclusion

Dans ce chapitre nous avons présenté l'ensemble du travail accompli dans le module de navigation pour FRMI.

En résumé, nous fournissons un mode manuel simple et efficace, supporté par un contrôle PID pour limiter les accélérations brusques et compenser les roues folles. Nous avons conçu un module de navigation locale basé sur un algorithme d'évitement d'obstacle de qualité. Ce module est conçu pour subvenir à différents besoins : navigation assistée avec un contrôleur, suivi d'un chemin local et suivi d'une commande instantanée. Le module d'évitement d'obstacle se voit plus utile lorsqu'il est associé à l'ensemble de manœuvres automatiques regroupé dans le mode semi-manuel. Enfin, un mode automatique qui se caractérise par un système d'autolocalisation dans une carte très performant.

Nous notons qu'à côté de cette architecture de navigation, nous avons fourni un moyen pour construction de cartes permettant de naviguer dans les nouveaux milieux non explorés.

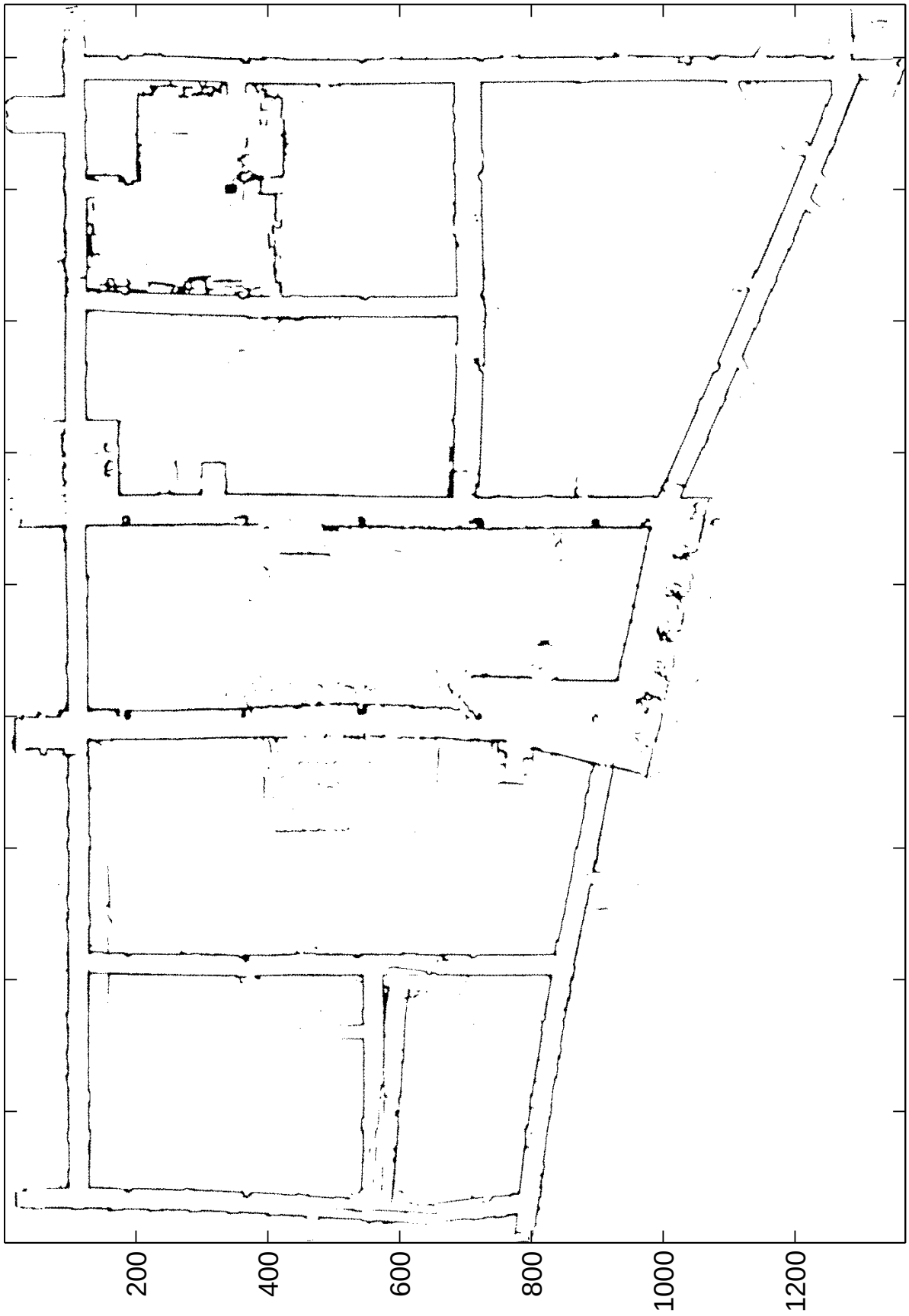


Figure 4.11 Orientation de la carte avec transformée de Hough

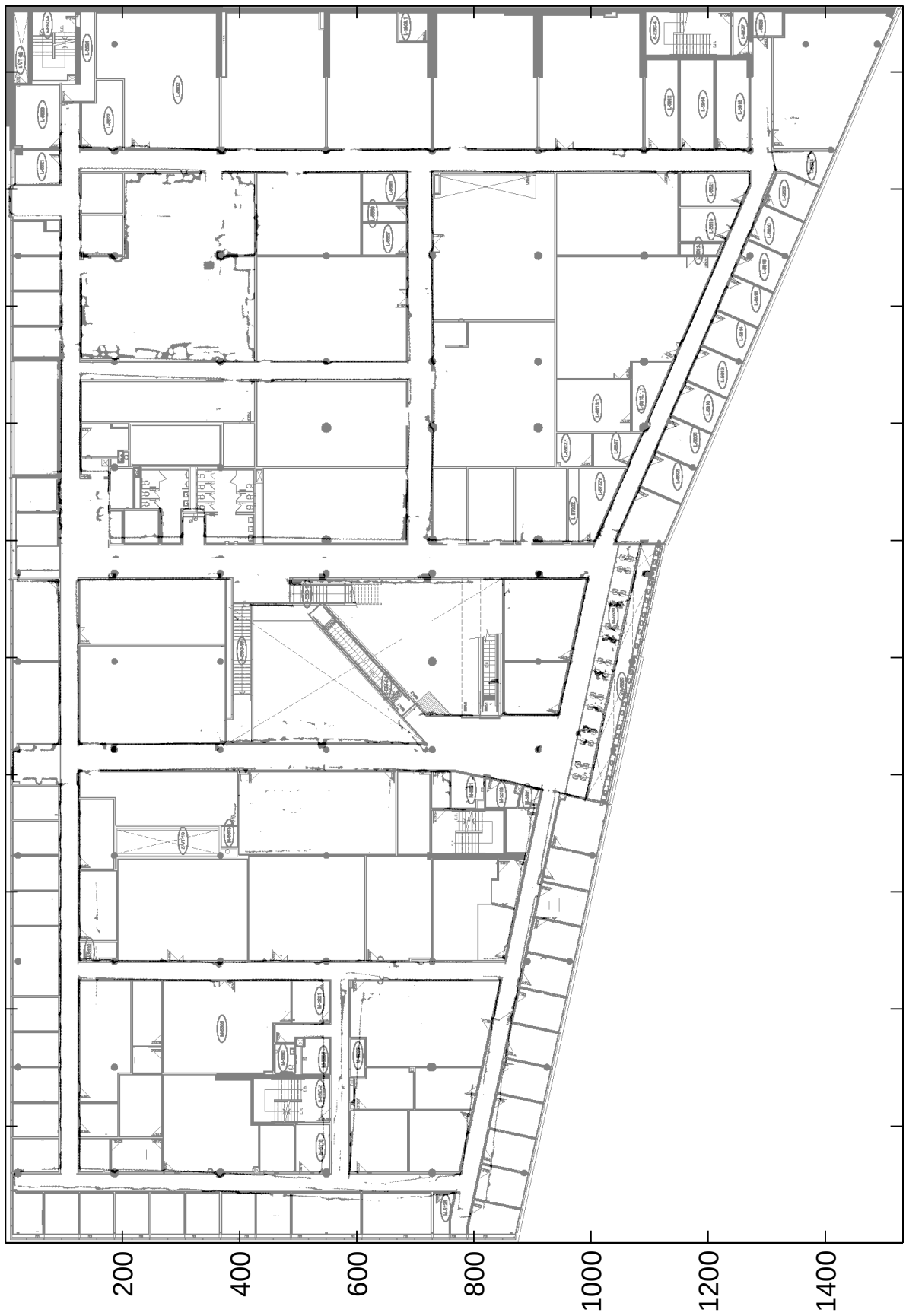


Figure 4.12 Résultat de la fusion des deux cartes

CHAPITRE 5

RÉSULTATS ET VALIDATION EXPÉRIMENTALE

Dans ce chapitre, nous allons valider les algorithmes que nous avons présentés au long du mémoire. Certes, nous savons que le module de navigation incluant, la navigation manuelle, l'évitement d'obstacle et la cartographie, démontre une bonne robustesse et fonctionne normalement, mais nous allons chercher à pousser ces algorithmes à leurs limites, en les testant dans différents environnements, contrôlé et non contrôlé, avec différents types d'obstacle et différentes configurations de capteurs. Ceci dans le but de conclure sur l'ultime objectif de notre travail, à savoir qu'elles sont les capteurs minimaux qu'un FRMI doit avoir pour réussir une navigation efficace et sécuritaire.

Nous soulignons que pendant l'expérimentation nous allons utiliser des données de type balayage laser pour tout ce qui touche à la cartographie, et les données de type nuage de points pour la navigation assistée.

5.1 Analyse et validation de la cartographie

Pour pouvoir valider et confirmer notre algorithme de cartographie, nous avons choisi de le soumettre à deux types d'expérience. Dans un premier temps, nous allons lui faire suivre un circuit prédéfini avec différents types de difficulté (obstacles) et sans aucune autre intervention extérieure (environnement contrôlé). Dans un deuxième temps, nous allons utiliser les résultats du premier test pour cartographier un grand espace arbitraire (environnement non contrôlé).

5.1.1 Environnement contrôlé

Présentation du circuit

Nous avons choisi un circuit qui nous permet de confirmer un ensemble de points précis, à savoir :

– la fiabilité de l'algorithme de la cartographie : vérifier si l'algorithme implémenté permet la faisabilité de la carte dans les deux cas extrêmes : manque de données de l'odométrie ou manque des données de télémétrie. Ces deux cas extrêmes peuvent être vérifiés, respectivement, avec l'utilisation d'un circuit circulaire [1] et en limitant la résolution des

1. Il se trouve que l'erreur d'odométrie en rotation est bien plus visible.

capteurs ;
– la sensibilité à l'environnement : vérifier si nous pouvons cartographier des espaces avec différents types d'obstacles, en forme : objet transparent, semi-transparent ou plein, et en style : coins et couloirs.

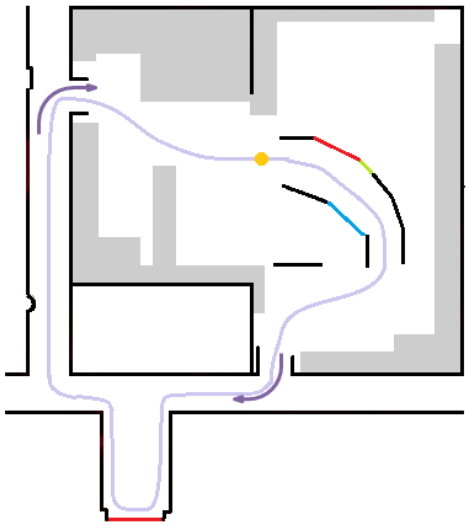


Figure 5.1 Présentation du circuit pour cartographie : schéma détaillé

Avec ces objectifs en tête, et en utilisant l'espace disponible dans le laboratoire de robotique, nous avons décidé de construire le circuit à l'intérieur du laboratoire, mais en rajoutant un ensemble d'obstacles spécifiques. La figure 5.1 présente une vision schématisée de cet espace. Les traits en noir présentent des obstacles pleins (murs, planches . . .), les espaces grisés indiquent la présence aléatoire d'objet dans la pièce (tables, chaises . . .), les traits en rouge représentent des vitres teintées, le trait en vert indique une plaque de plexiglas transparente et le trait en bleu indique un panneau de grillage (voir figure 5.2).

Configuration

Pour le test du module de la cartographie, nous avons décidé de ne pas utiliser le mode semi-manuel, ceci dans le but d'éviter les erreurs combinées provoquées par le module d'évitement d'obstacle ; toutes les expériences sont faites alors en mode manuel. Aussi, pour nous rapprocher du cas général, à chaque fois nous laissons tous les capteurs en marche même si nous n'en avons pas besoin. En somme, nous utilisons le nœud de modélisation du FRMI « *virtualFRMI*, les nœuds d'acquisition de donnée « *laserFRMI* », « *sonarFRMI* » et « *kinectFRMI* », le nœud de conversion de donnée brute en données de télémétrie : un balayage

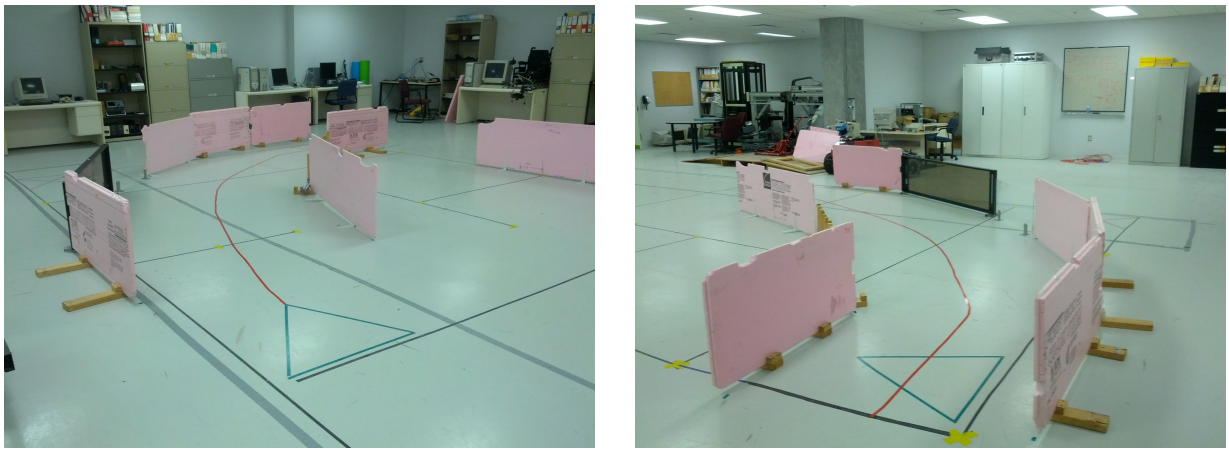


Figure 5.2 Présentation du circuit pour cartographie : vue générale

laser avec une résolution choisie, « *merge_scan* », et le nœud de cartographie « *mapping-FRMI* ». De plus, nous effectuons le processus de construction de carte sans arrêt avec une vitesse moyenne de $0.3m/s$.

Test de cartographie avec un télémètre laser

La première série de tests, aussi la plus intuitive, est celle utilisant les données des télémètres lasers pour construire une carte ; nous utilisons alors les deux lasers pour construire un balayage de laser d'environ 360 degrés. Le module de fusion des balayages laser */merge_scan* nous permet d'échantillonner un balayage pour changer sa résolution [2], donc en variant la résolution nous allons essayer de trouver l'optimum nous permettant d'obtenir une carte lisible avec un minimum de points.

Nous rappelons que le prix excessivement cher des télémètres lasers rend impossible d'imaginer un produit commercial avec ces capteurs ; en effet, chaque télémètre coûte le quart du prix du FRM. Par contre, un système de télémètres lasers ponctuels dispersés sur l'ensemble du FRMI serait une très bonne alternative future.

Les 3 images de la figure 5.3 illustrent le résultat de la cartographie avec différentes résolutions. Nous remarquons que même en allant chercher une résolution de 10 degrés, c'est-à-dire, 36 point par balayage, nous arrivons à reconstituer le chemin circulaire que nous avons suivi sans arrêt. Aussi, il n'y a pas une très grande différence visuelle pour les 2 premières cartes ; ceci s'explique par le fait que nous avons choisi de construire une carte avec une résolution de 5cm par pixel [3]. Cependant, nous soulignons que comme prévu les vitres ne

2. Une résolution de α degré signifie que l'angle entre deux rayons laser successifs est égale à α

3. La résolution de 5cm par pixel est une résolution très raisonnable pour l'évitement d'obstacle et la planification du chemin, considérant la taille du FRMI 135cmx80cm.

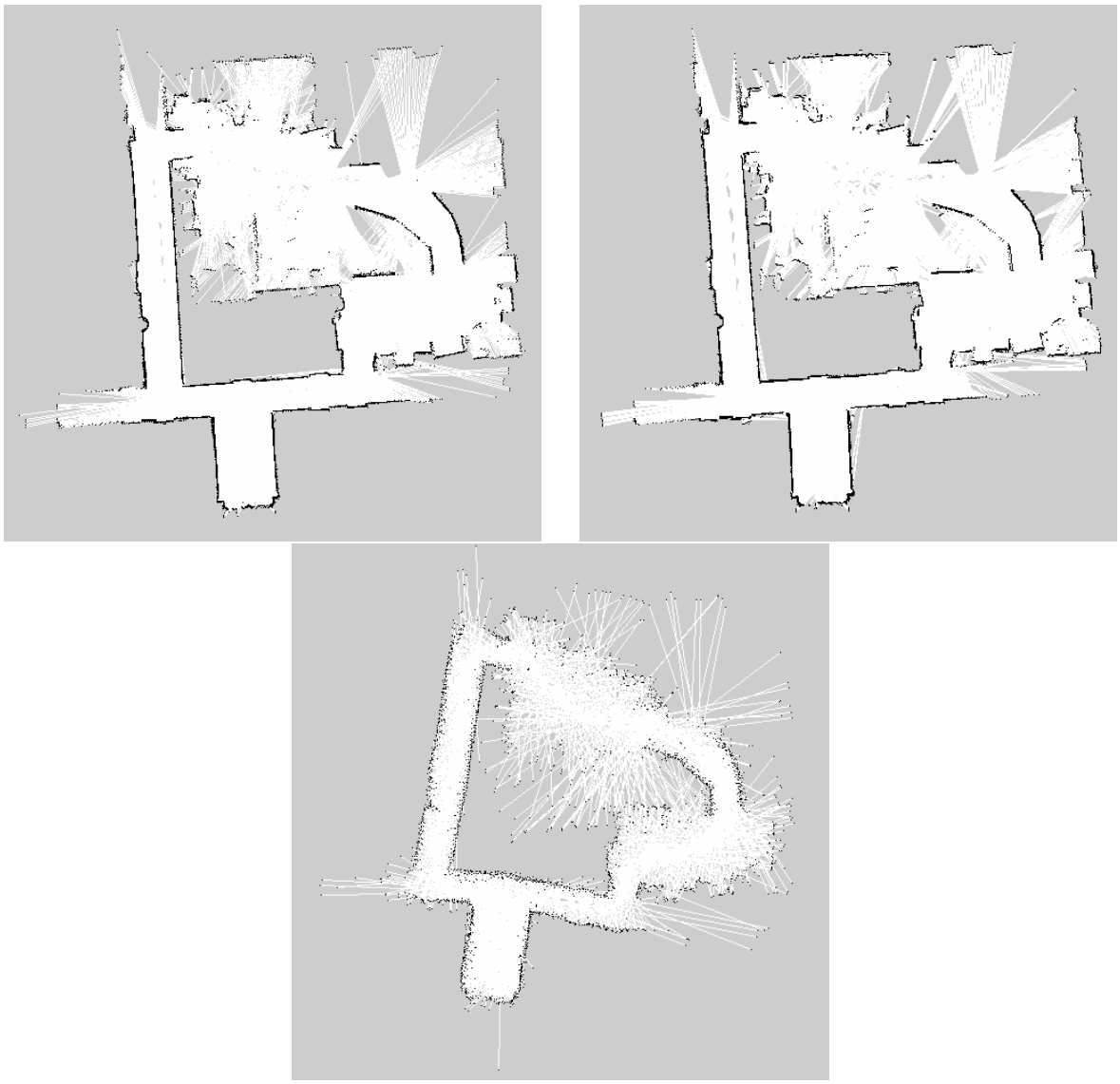


Figure 5.3 Résultat de la cartographie avec télémètre laser, de droite à gauche, du haut en bas, la résolution est égale à 1, 5 et 10 degrés

sont pas détectées.

Test de cartographie avec ultrasons

En combinant les mesures des télémètres ultrasons sur un plan, nous pouvons simuler un balayage laser. Cependant, le problème majeur des ultrasons est leur imprécision angulaire et leur champ de vision restreint. De plus dans notre cas particulier, les ultrasons dont nous disposons donnent des valeurs totalement erronées à partir de 80cm. La figure 5.4 démontre le résultat que nous avons obtenu en essayant une cartographie avec sonar seulement : nous remarquons que les vitres sont détectées, ainsi que quelques détails du corridor, mais la carte n'a aucun sens, chose qui était prévisible vu le manque d'information de télémétrie.

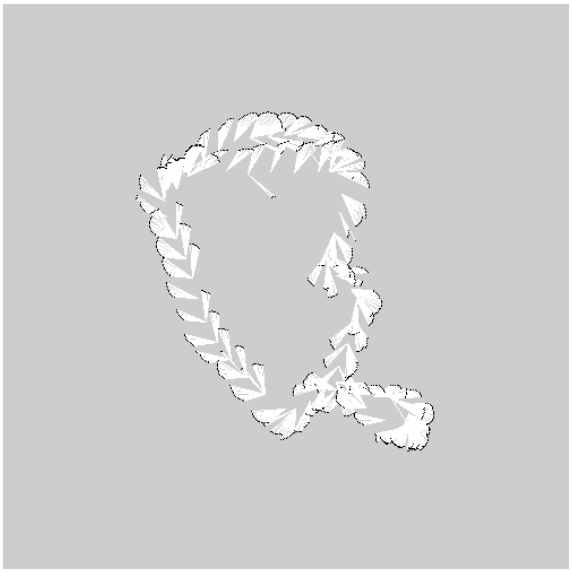


Figure 5.4 Résultat de la cartographie avec télémètre ultrason seul

Test de cartographie avec kinect

Comme nous l'avons vu dans la section 3.5, nous pouvons générer une donnée équivalente à un balayage laser à partir des nuages de points fournis par la caméra Kinect, ceci en nous limitant sur un plan horizontal. C'est ce que nous avons fait pour construire la carte à la figure 5.5. Vu l'utilisation d'une seule caméra Kinect, nous nous sommes restreints à un champ de vision de moins de 140 degrés ; par contre, malgré cette limitation, nous arrivons à un résultat très correct partout sauf dans le corridor ; en effet, nous remarquons une erreur dans la rotation au moment où le FRMI prend le deuxième couloir (cette erreur est justifiée par le manque des données de télémétrie sur le plan).

Nous rappelons que dans l'algorithme de cartographie, nous nous appuyons sur une analyse de correspondance entre le nouveau balayage et la carte pour corriger les erreurs possibles d'odométrie. Pendant ce test au moment de prendre le virage du couloir à droite l'analyse de correspondance a dû échouer à cause du manque de données et donc l'erreur d'odométrie en rotation a subsisté. Nous notons que cette erreur n'est pas très grande au début, mais la longueur du couloir la rend plus visible.

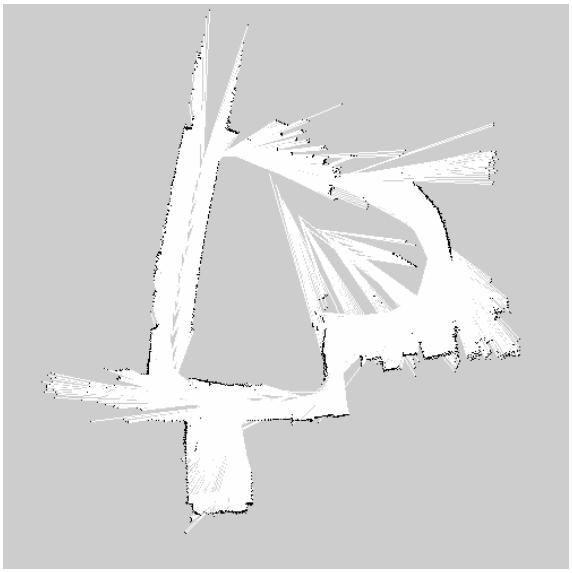


Figure 5.5 Résultat de la cartographie avec la caméra Kinect seule

Test de la cartographie avec une combinaison des télémètres laser et ultrasons

Ce test consiste à prendre les données des télémètres laser échantillonnées à 5 degrés puis les combiner avec des données ultrasons pour construire une carte de l'espace.

La figure 5.6 montre le résultat obtenu. Nous remarquons que la qualité de la carte n'a pas diminué comparée à celle obtenue avec seulement des données laser. Par contre, nous aurions espéré voir plus apparaître les vitres.

Il se trouve que les sonars voient bien la vitre donc la probabilité d'avoir un obstacle dans la position de la vitre augmente. Cependant vu leur erreur angulaire nous associons une probabilité plus faible à ces points. Puis en avançant, ça sera au tour des télémètres lasers (et seulement les télémètres laser grâce à leur grand champ de vision) d'observer la vitre, malheureusement, les lasers voient ce qui est derrière et donc l'algorithme de la cartographie diminue la probabilité d'avoir un obstacle dans cette position. Ce problème peut être corrigé en rajoutant des sonars derrière le FRMI et aux côtés.

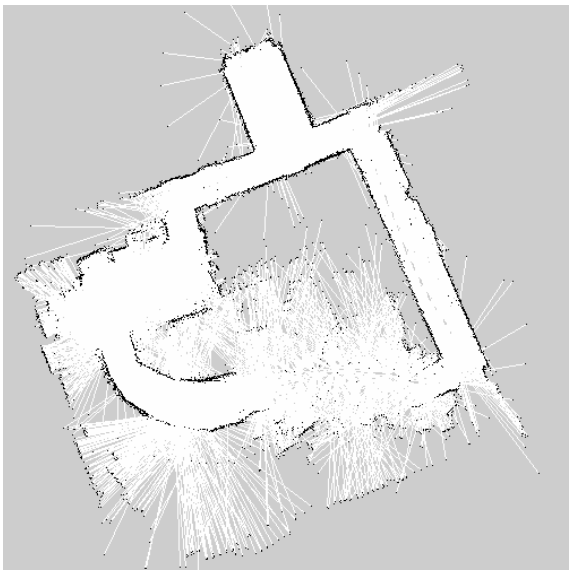


Figure 5.6 Résultat de la cartographie avec télémètres ultrason et laser

Test de cartographie avec une combinaison des télémètres laser et capteur Kinect

Un test intéressant consiste à fusionner les données de télémètre laser et ceux de la camera Kinect. Nous avons gardé le balayage fourni par la camera Kinect pour l'avant du FRMI, et nous avons limité les données laser sur un angle de 135 degrés pour chaque côté du FRMI.

La figure 5.7 montre les résultats obtenus avec un échantillonnage du laser de 1, 5 et 10 degrés, nous remarquons que l'erreur de rotation que nous avions avec la caméra Kinect seulement est corrigée avec les nouvelles données de la télémétrie sur les côtés.

Test de cartographie avec une combinaison des télémètres laser, ultrasons et capteur Kinect

Enfin nous avons conclu avec un test de cartographie incluant tous les capteurs, à savoir : les télémètres lasers avec un balayage limité sur les cotées du FRMI, la caméra Kinect sur le devant et le télémètre ultrason pour la détection des milieux transparents.

La figure 5.8 illustre le résultat obtenu. Comme prévu la caméra fournit, *a priori*, une quantité suffisante de données pour le devant de la chaise, les télémètres laser permettent de rajouter l'information suffisante pour que l'algorithme de cartographie corrige l'erreur d'orientation, et les sonars permettent la détection des vitres. Nous notons que les vitres apparaissent mieux cette fois, puisque nous avons utilisé moins de points issus du télémètre laser dans le balayage laser fusionné.

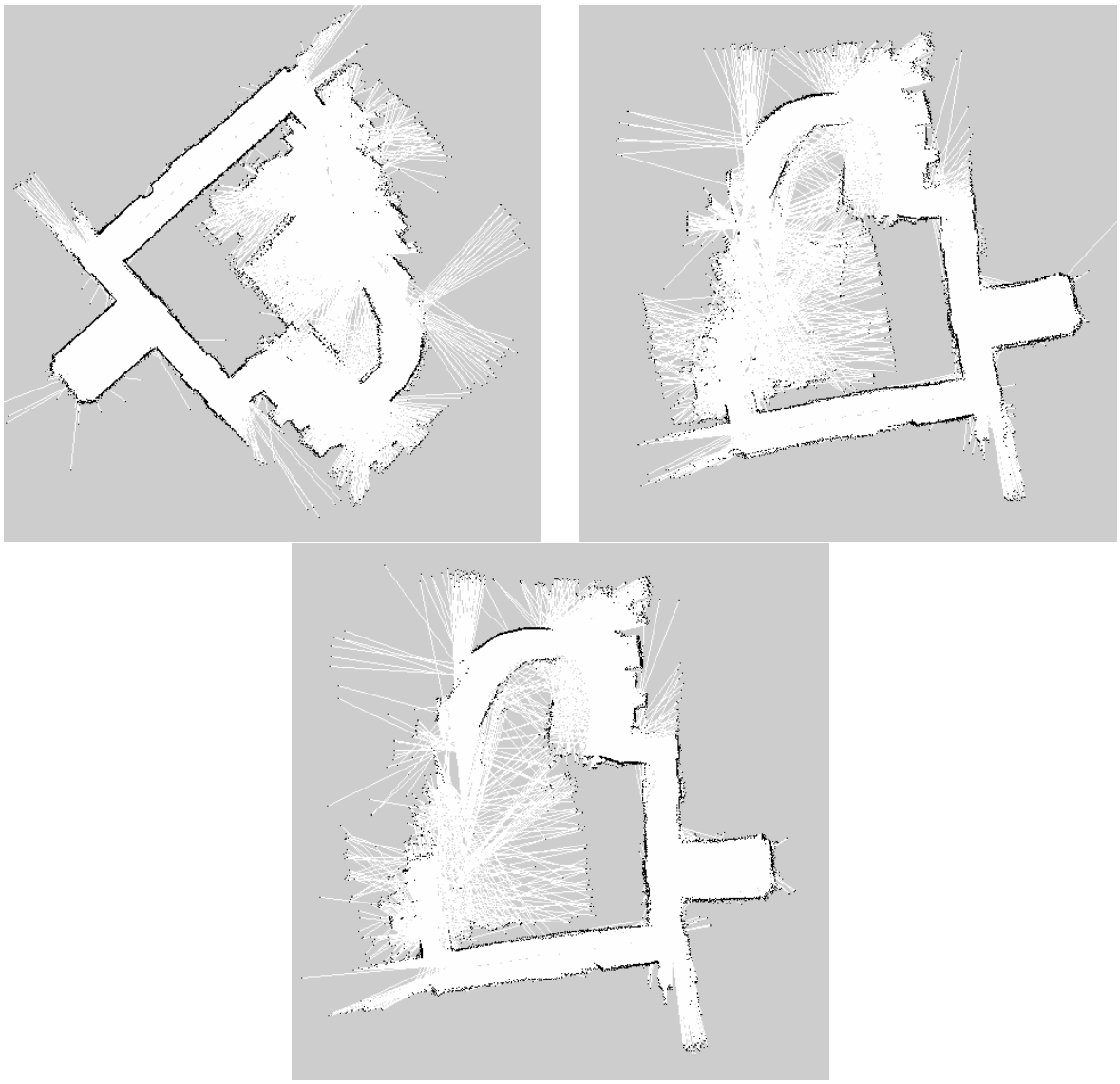


Figure 5.7 Résultat de la cartographie avec combinaison de télémètre laser et camera Kinect de droite à gauche, d'en haut en bas, la résolution du laser change de 1, 5 à 10 degrés

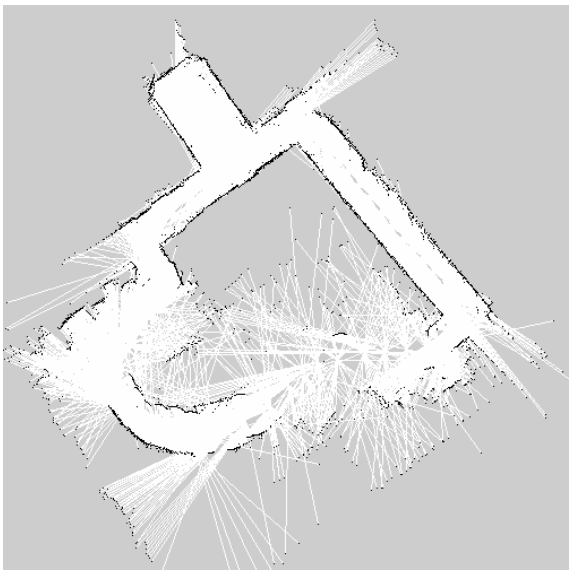


Figure 5.8 Résultat de la cartographie avec une fusion de tous les capteurs

Comparaisons et conclusions

Afin de juger les performances de chaque capteur, nous faisons une étude quantitative des résultats obtenus dans chaque test. Nous avons choisi les critères suivants :

- le coût du matériel sur le marché : en moyenne les télémètres ultrasons coûtent dix fois moins cher qu'une caméra Kinect, qui coûte dix fois moins cher qu'un télémètre laser ;
- la sensibilité à l'environnement détecté par les capteurs : un gain sur une échelle de 10 est attribué au capteur s'il arrive à percevoir un type d'obstacle ;
- la cohérence de la carte résultante : ce critère inclut la présence du bruit dans la carte, la continuité des traits rectiligne et le degré de découverte de l'environnement.

Nous faisons l'hypothèse que si nous diminuons la résolution des télémètres laser nous pouvons trouver un équivalent laser permettant de donner la même résolution à un prix plus bas. Ceci est justifié du fait que pour construire un télémètre laser ponctuel nous n'avons besoin que d'une source laser ponctuel et un capteur CMOS positionné de telle façon pour détecter la réflexion de l'onde optique émise par le laser. Le reste est une simple triangulation pour calculer la distance.

Le tableau 5.1 présente la synthèse de ces critères pour les différentes expérimentations effectuées. Le gain total est normalisé sur un maximum de 100, sachant qu'un gain de 100 est atteint si un capteur ne coûte rien et qu'il donne un score parfait (10) dans tous les autres gains.

Sans perdre de généralité, ce tableau nous permet de bien orienter nos choix de capteurs selon nos critères. Si nous sommes dans un environnement avec seulement des obstacles

Tableau 5.1 Synthèse des différents tests de cartographie dans un environnement contrôlé

Résolution des capteurs			Coût matériel	Sensibilité à l'environnement			Cohérence de la carte			Gain de la carte
Télémètre laser	Télémètre ultrason	Caméra Kinect		Opaque	Semi-opaque	Transparence	Continuité	Bruits	Découverte	
360	Non	Non	-100	10	2	0	10	9	9	17
72	Non	Non	-20	10	2	0	9	8	8	52
36	Non	Non	-10	10	2	0	8	7	7	52
Non	Oui	Non	-1	10	10	10	1	2	2	58
Non	Non	Oui	-10	10	4	0	7	8	4	50
72	Oui	Non	-21	10	9	7	8	5	8	68
180	Non	Oui	-60	10	3	0	9	7	9	33
36	Non	Oui	-20	10	2	0	9	6	8	48
18	Non	Oui	-15	10	2	0	8	5	7	46
36	Oui	Oui	-21	10	10	9	8	6	8	75

opaques et si nous nous soucions moins des coûts, il nous suffit de travailler seulement avec des télémètres laser pour avoir un meilleur résultat. Par contre, pour un produit commercial relativement moins cher fournissant un ensemble de données différentes, mais homogènes, notre choix sera de faire une fusion de tous les capteurs.

Ceci dit, ces tests ont été effectués dans un environnement bien cadré et non bruité. Dans la suite nous allons soumettre notre prototype à de grands espaces non contrôlés.

5.1.2 Environnement non contrôlé

Pour cette expérimentation nous voulons faire une cartographie d'un nouvel espace ressemblant à des espaces d'aéroport ou des centres d'achats. Nous avons choisi le premier étage du bâtiment Lassonde de l'École Polytechnique de Montréal : il contient une grande porte vitrée de plus de deux façades vitrées, un couloir restreint et un autre couloir assez large ; enfin, c'est un espace assez peuplé puisque tous les étudiants traversent la porte principale pour entrer ou sortir de l'école.

Nous avons choisi dans un premier temps d'effectuer une cartographie avec une fusion de données : télémètre laser avec une résolution de 10 degrés (36 points), ultrason et caméra Kinect. La figure 5.9 illustre le résultat obtenu. Malgré la précision de la carte obtenue, la carte est inexploitable à cause de l'erreur incorrigible de l'odométrie (surtout en rotation). En effet, un simple calcul montre qu'une erreur de 1 degré en rotation provoque 0.87m d'erreur en translation au bout de 50 mètres, ceci devient plus grave puisque l'erreur en rotation est supérieure à 1 degré et elle est commise en continu (voir figure 4.6) ! Normalement la cartographie corrige ces erreurs en se basant sur les données de télémétrie, mais dans ce cas particulier, vu la largeur de l'espace et la présence des gens, nos données fusionnées semblent insuffisantes (surtout sur les cotées) ou très bruitées pour réussir une bonne analyse de correspondance.

Nous avons alors effectué une deuxième cartographie, mais cette fois en utilisant les télémètres laser avec une grande résolution (environ 360 degrés), la figure 5.10 illustre le résultat obtenu : une carte avec une très grande précision avec des traits rectilignes continus et un minimum de bruit, nous n'avons jamais été si proches de la perfection. Par contre, nous soulignons que ce résultat n'a pas été direct, puisque nous avons effectuée plusieurs aller-retour à mi-chemin avec de faibles vitesses. Ceci dans l'objectif d'utiliser au maximum la consistance des données laser pour corriger l'odométrie. N'oublions pas que la correction de l'odométrie avec des données lasers n'est efficace que si le taux de réussite de l'analyse de correspondance est assez haut, et donc que si l'estimation de la position basée sur les données d'odométrie, bruitée avec une matrice de covariance, est bonne.

En conclusion, le module de cartographie permet la construction de carte efficacement,

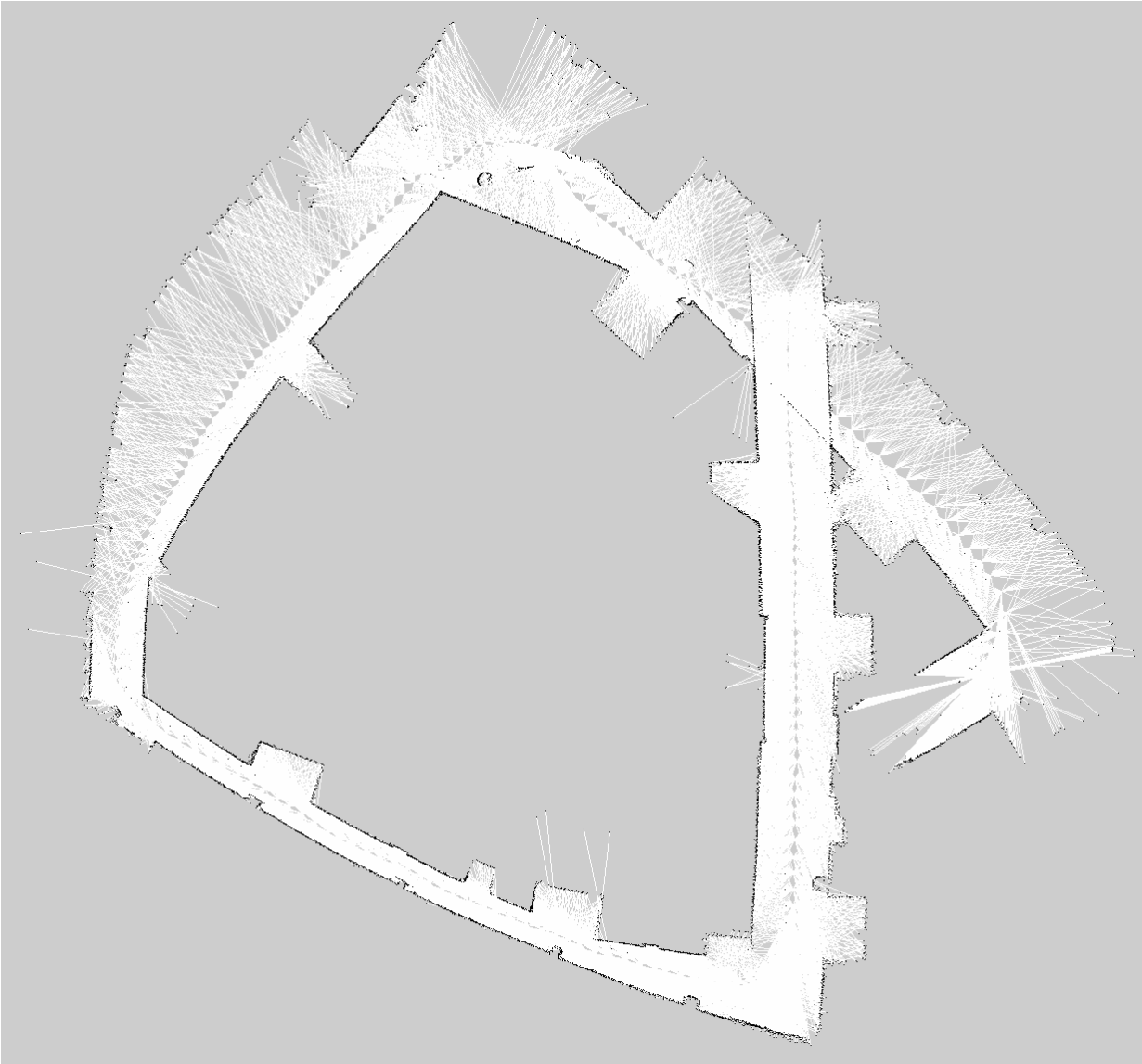


Figure 5.9 Cartographie du premier étage du bâtiment Lassonde avec une fusion de donnée : télémètre laser (10%), ultrason et caméra Kinect

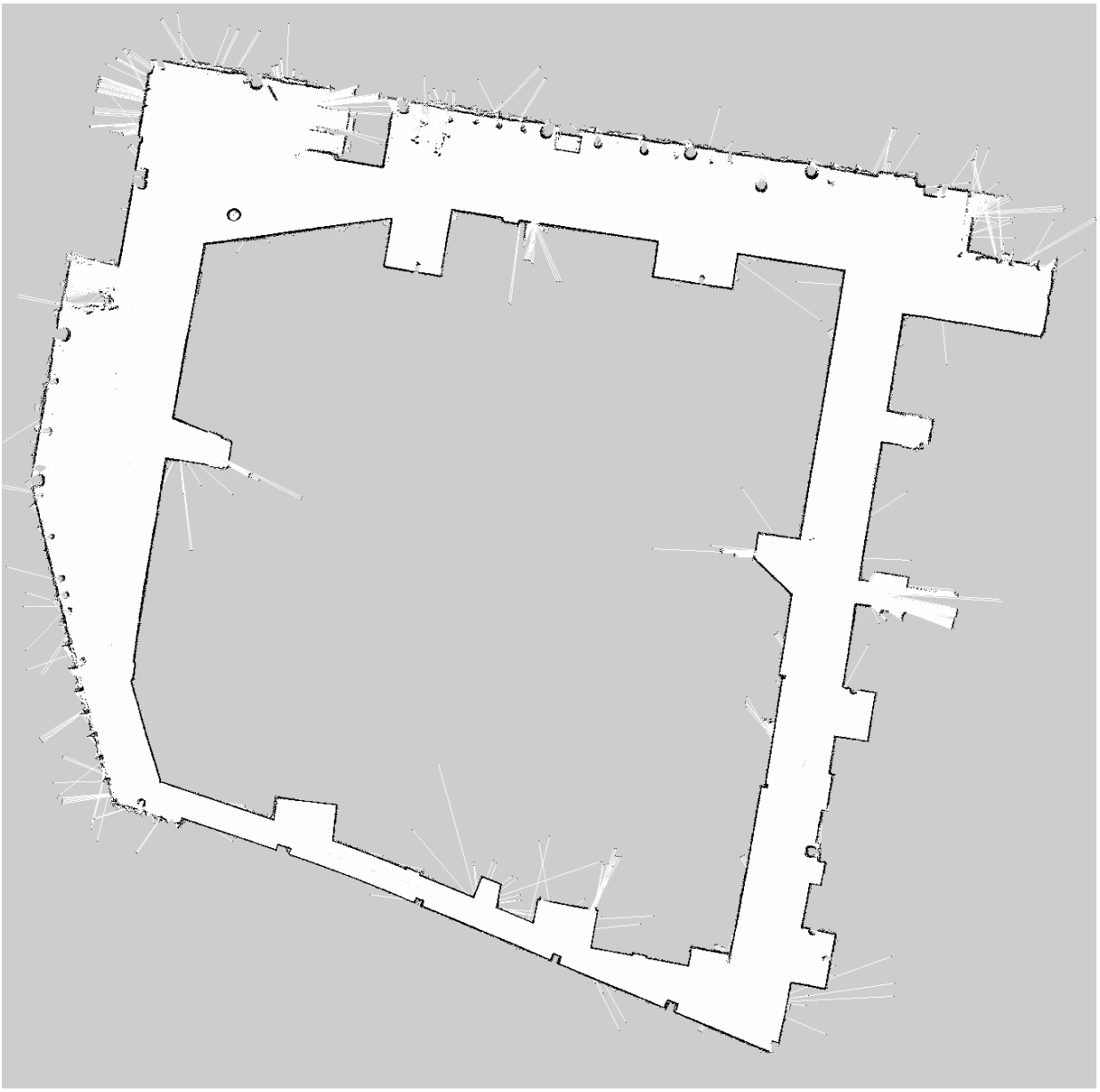


Figure 5.10 Cartographie du premier étage du bâtiment Lassonde avec télémètre laser seulement

mais, dans des conditions précises :

- nous fournissons assez de données sur l'environnement, dans les environs de 320 points par balayage laser ;
- nous naviguons avec une faible vitesse pour maximiser la réussite d'analyse de correspondance ;
- nous naviguons de façon à rester lié au maximum avec la carte construite, par exemple, faire des boucles circulaires dans l'environnement.

Bref, nous devons nous assurer que les particules du filtre particulaire responsable de la cartographie restent regroupées et ne dispersent pas rapidement. Ceci peut être quantifié par une mesure d'entropie de particule que nous fournissons en continu sous la forme d'un message ROS.

5.2 Analyse et validation du module de navigation locale

Dans cette partie, nous allons effectuer une expérimentation visant à valider le module de navigation locale, incluant l'évitement d'obstacle et le suivi de chemin.

Pour cette expérimentation, nous utilisons des données sous forme de nuages de points résultantes de la fusion de tous les capteurs (télémètre laser, télémètre ultrason et caméra Kinect). Nous notons que nous avons construit le chemin à suivre à l'avance en choisissant un ensemble de points dans la carte (voir figure 5.11). Lors de la navigation, la carte est utilisée uniquement pour la localisation globale et comme support visuel. L'évitement d'obstacle et le suivi de chemin sont assurés à chaque instant en se basant seulement sur les données perçues en temps réel par tous les capteurs.

La carte sur la figure 5.11 est une reconstitution sur MATLAB de la carte obtenue précédemment avec seulement des télémètres laser avec une résolution maximale.

La difficulté de cette trajectoire consiste dans les deux points suivant :

- pour effectuer cette trajectoire le module d'évitement d'obstacle doit assurer deux passages de portes (voir figure 5.12). Les passages de portes représentent un défi au module de suivi de chemin puisque le point objectif se situe derrière deux obstacles très rapprochés. Il se trouve qu'avec les paramètres choisis, la traversée de porte ne pose aucun problème ;
- le chemin à suivre contient une grande partie dans un couloir serré (voir figure 5.13). Nous remarquons de petites oscillations lors de l'avancement dans le couloir qui est dû à notre choix d'éloignement aux obstacles.

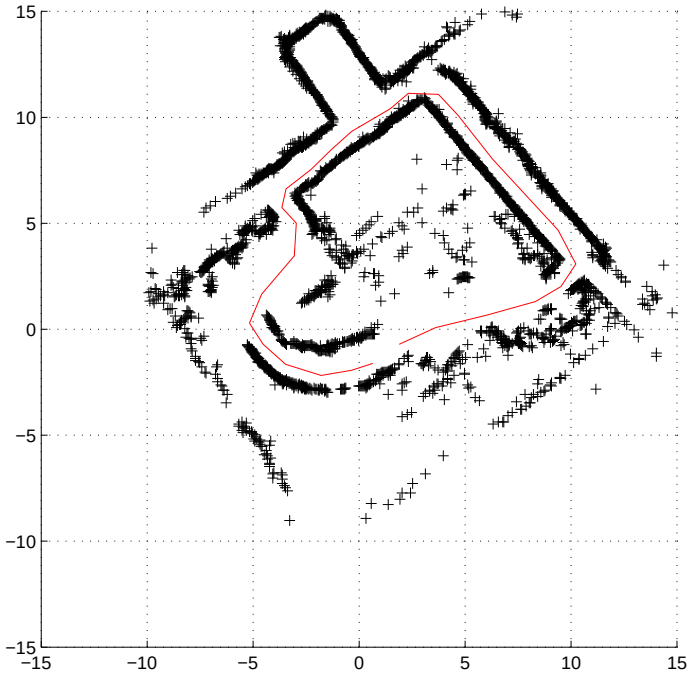


Figure 5.11 Trajectoire utilisée pour l'expérimentation du suivi de chemin

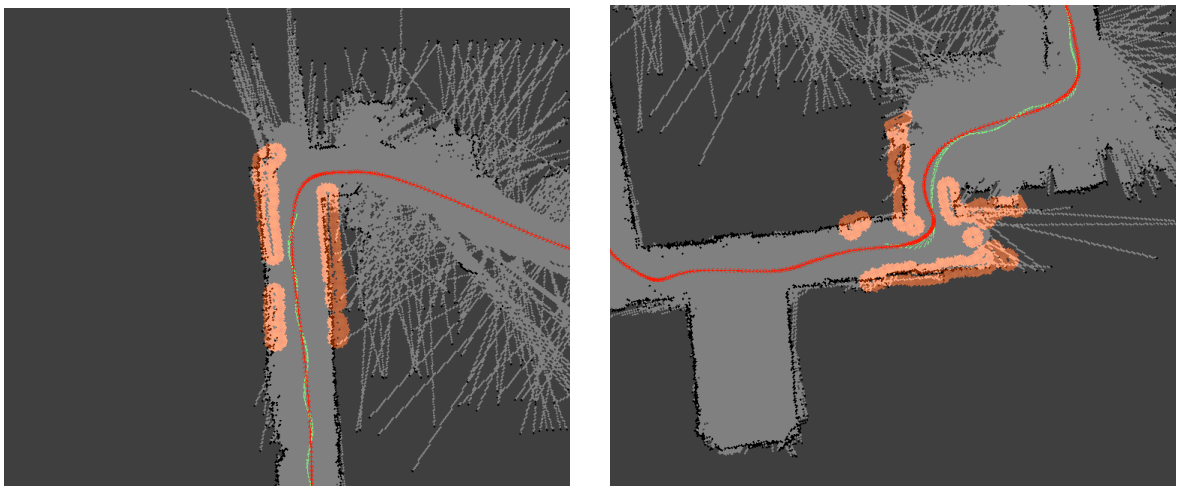


Figure 5.12 Suivi de la trajectoire : instant de traversé de porte

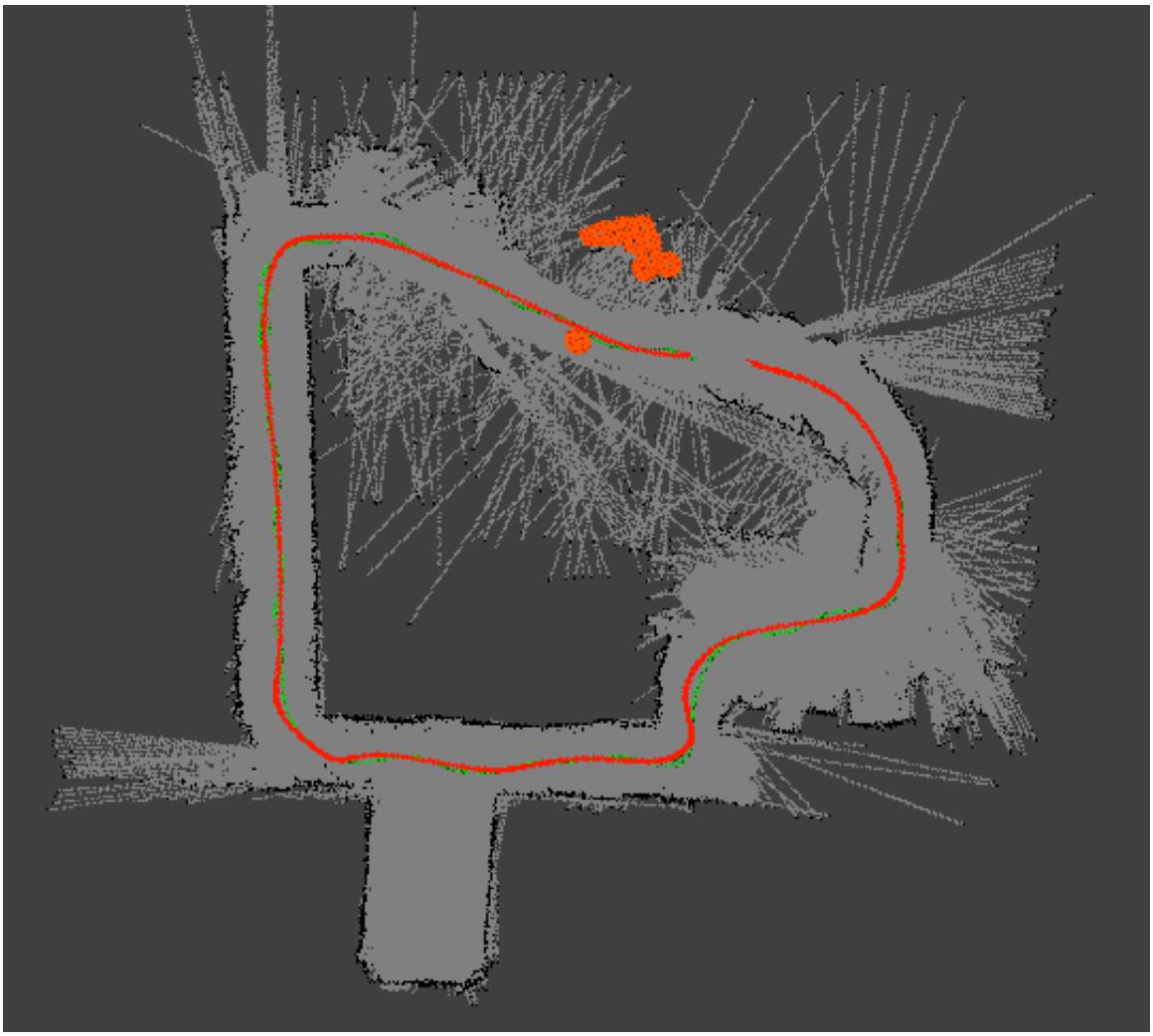


Figure 5.13 Résultat du suivi de trajectoire

5.3 Conclusion

En somme, ce chapitre nous a permis de comprendre les limites de notre stratégie de navigation conçue pour le FRMI. D'une part, nous pouvons conclure que l'évitement d'obstacle basé sur un nuage de points fusionnant tous les capteurs est une solution simple et efficace dans ce cas. D'autre part, nous pouvons effectuer une cartographie sur de petites durées de temps (avant de nous perdre à cause du manque des données) à l'aide d'un balayage laser fusionnant tous les capteurs. Par contre, pour effectuer une cartographie efficace sur de larges espaces intérieurs, nous avons toujours besoin d'utiliser les télémètres lasers à leur maximum. Ceci ne posera pas nécessairement un problème, si nous séparons le problème de construction de carte du problème d'évitement d'obstacles.

CHAPITRE 6

CONCLUSION

Tout au long de ce mémoire, nous avons présenté le module de navigation pour FRMI dans sa globalité. Ce module propose de nouveaux éléments de solution au problème de navigation dans les grands espaces avec un fauteuil roulant motorisé.

6.1 Synthèse des travaux

En résumé, notre travail a contribué dans le domaine de recherche par l'apport d'un ensemble de nouveautés :

ROS : Une grande différence entre l'ancien système Acropolis et notre nouveau module de navigation est l'environnement de travail. Nous avons relevé le défi de mettre en place avec le matériel existant un nouveau système d'exploitation modulaire, facile et ouvert, qui nous a énormément aidés pour concevoir le module de navigation :
- distribution de calcul sur plusieurs coeurs ou ordinateurs ;
- visualisation rapide des données ;
- disponibilité d'un large catalogue d'algorithmes et ressources en accès libre.

Fusion de donnée : Nous avons testé et exploré un bon ensemble de capteurs : télémètre laser, télémètre ultrason et caméra Kinect. De plus, nous fournissons un algorithme générique pour fusionner ces données en deux types :
- un balayage laser : une donnée de télémétrie présentant les obstacles perçus avec leur distance et leur angle respectif, par rapport au FRMI. En présence de conflit, on considère l'obstacle le plus proche.
- un nuage de points : une représentation en trois dimensions de tous les obstacles comme perçus par les capteurs.

Nous avons fait cette distinction entre ces deux types à cause de leurs objectifs différents : les balayages lasers sont destinés pour la cartographie et la localisation globale, tandis que les nuages de points sont utilisés pour l'évitement d'obstacle.

Module de navigation locale : Nous avons proposé une nouvelle architecture de contrôle robuste basée sur un superviseur et des services. Cette architecture se caractérise par sa modularité et donc sa capacité à continuer de s'agrandir. Aussi, ce module a été conçu pour répondre aux objectifs de recherche en proposant :

- un mode manuel, permettant le contrôle précis et sécuritaire du FRMI par n'importe quelle manette de contrôle compatible Linux ;
- un mode semi-manuel, centré sur le module d'évitement d'obstacle et proposant des services complémentaires tels que le passage de porte et le suivie de chemin ;
- un mode automatique, qui reprend une implémentation de l'algorithme MCL pour la localisation globale sur une carte avec les données de balayage laser.

Cartographie : L'une des finalités essentielles de cette recherche, permettant à ce projet de vivre dans le futur, était de fournir un service fiable de construction de carte basée sur les capteurs disponibles sur le FRMI. La méthode que nous proposons, basée sur une implémentation d'un algorithme de cartographie, permet d'atteindre cet objectif de deux façons :

- en nous basant sur les télémètres lasers, nous arrivons à cartographier de nouveaux grands espaces avec précision ;
- en nous basant sur des données fusionnées (en majeure partie provenant de la caméra Kinect et des télémètres ultrason), nous arrivons à construire des cartes pour de petites pièces.

Ces cartes peuvent être fusionnées plus tard avec une carte connue de l'environnement (carte CAO).

Enfin les expérimentations ont permis de conclure sur les deux objectifs principaux de cette recherche :

Identification des capteurs minimaux : Concernant le problème des obstacles transparents, nous répondons à cette finalité de deux façons. D'une part, dans le cas d'évitement d'obstacle, nous profitons de la donnée du télémètre ultrason brute, incluse dans le nuage de points pour assurer la navigation locale. D'autre part, pour la cartographie, la méthode de fusion de donnée proposée permet de générer un balayage laser, favorisant les données ultrasons, pour la construction de carte. Par contre, vu les limites de portée d'ultrason et les bruits de mesure, la construction de carte avec des données fusionnées devient difficile pour les larges espaces. Enfin, la camera Kinect fournit un très bon rendu, mais reste limitée en terme de champ de vision.

Module de navigation : Le module de navigation que nous avons conçu se distingue du précédent par : l'intégration de la localisation globale dans une carte et le rajout du module de cartographie, donnant tous les deux des résultats très satisfaisants. Cependant, la navigation assistée proposée par la navigation locale, même s'il est très robuste et sécuritaire, peut être moins maniable.

6.2 Limitations de la solution proposée

Même si l'architecture du module de navigation proposée présente plusieurs points forts, l'algorithme d'évitement d'obstacle reste moins réactif que le précédent sur Acropolis. Ceci est dû au simple fait qu'il n'a pas été conçu spécialement pour un FRMI, mais adapté d'un algorithme connu d'évitement d'obstacle. Ceci n'empêche pas qu'il y ait possibilité de porter l'ancien module d'évitement d'obstacle sur le nouveau prototype.

Un autre problème qui reste non exploré est celui des objets dynamiques et des obstacles négatifs. Cependant, avec l'ajout de la caméra Kinect, il y a une grande possibilité de faire un traitement spécial dans des développements futurs pour ces cas particuliers.

Pendant les expérimentations, nous avons effectué une comparaison subjective des cartes obtenus, basée sur un critère humain. Pour mieux juger les performances de différents capteurs nous aurions pu utiliser une métrique pour comparer ces cartes.

Enfin, nous n'avons pas développé une interface adaptée à la navigation à la carte, chose qui nous aurait permis de tester plus efficacement la localisation globale dans la carte.

6.3 Améliorations futures

Ce travail de recherche nous permet de nous ouvrir vers d'autres perspectives de recherche futures permettant d'améliorer et de perfectionner le module de navigation proposé pour le FRMI.

Nous proposons d'automatiser ou semi automatiser le processus de génération des cartes (retours sur nos pas) en fonction des mesures d'entropies. Ceci permet de faciliter le processus de construction de carte.

Nous pouvons aussi utiliser plusieurs cartes ou une seule carte non binaire pour tenir compte des objets opaques vs transparents. En d'autres termes, profiter de la précision des télémètres laser pour s'autolocaliser et garder des références sur la position des vitres. Ces références seront rajoutées plus tard.

D'autres stratégies d'autolocalisation récentes démontrant de bonnes performances dans les bâtiments peuvent être explorées telles que la localisation avec le champ magnétique terrestre (Vallivaara *et al.*, 2011) ou avec imagerie.

La fusion des cartes n'a pas été explorée profondément ce qui laisse place à l'amélioration, surtout si nous arrivons à faire une fusion automatique avec les cartes issues, par exemple, de Google via internet.

RÉFÉRENCES

BALLARD, D. H. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13, 111–122.

BORENSTEIN, J. et KOREN, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on*, 7, 278 –288.

BOUCHER, P. (2010). *Navigation semi-autonome d'un fauteuil roulant motorisé en environnements restreints.* Mémoire de maîtrise, École Polytechnique de Montréal.

BURGARD, W., CREMERS, A. B., FOX, D., HÄHNEL, D., LAKEMEYER, G., SCHULZ, D., STEINER, W. et THRUN, S. (1998). The interactive museum tour-guide robot. 11–18.

CAMPION, G., BASTIN, G. et DANDREA-NOVEL, B. (1996). Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *Robotics and Automation, IEEE Transactions on*, 12, 47 –62.

CENSI, A. (2008). An icp variant using a point-to-line metric. *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 19–25.

COX, I. (1991). Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7.

CRANDALL, J. W. et GOODRICH, M. A. (2001). Experiments in adjustable autonomy. *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, 3, 1624–1629 vol.3.

DOUCET, A., FREITAS, N., MURPHY, K. et RUSSELL, S. (2000). Raoblackwellised particle filtering for dynamic bayesian networks. *in The 16th Annual Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers, 176–183.

EDEN, C. (1992). On the nature of cognitive maps. *Journal of Management Studies*, 29, 261–265.

ELFES, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22, 46–57.

ENDRES, H., FEITEN, W. et LAWITZKY, G. (1998). Field test of a navigation system : autonomous cleaning in supermarkets. *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*. vol. 2, 1779 –1781 vol.2.

ENGELSON, S. (1994). *Passive Map Learning and Visual Place Recognition.* Thèse de doctorat, Dept. of Computer Science, Yale University.

FEHR, L., LANGBEIN, W. E. et SKAAR, S. B. (2000). Adequacy of power wheelchair control interfaces for persons with severe disabilities : a clinical survey. *Journal of rehabilitation research and development*, 37, 353–360.

FOX, D., BURGARD, W., DELLAERT, F. et THRUN, S. (1999). Monte carlo localization : Efficient position estimation for mobile robots. *IN PROC. OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI*. 343–349.

FOX, D., BURGARD, W. et THRUN, S. (1997). The dynamic window approach to collision avoidance. vol. 4.

FRESE, U. (2006). Treemap : An $o(\log n)$ algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21, 103–122. 10.1007/s10514-006-9043-2.

GELB, A. (1974). *Applied optimal estimation*. MIT Press.

GRISETTI, G., STACHNISS, C. et BURGARD, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23, 2007.

HAVERINEN, J. et KEMPPAINEN, A. (2009). Global indoor self-localization based on the ambient magnetic field. *Robotics and Autonomous Systems*, 57, 1028 – 1035. 5th International Conference on Computational Intelligence, Robotics and Autonomous Systems (5th CIRAS).

KHATIB, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5, 90–98.

KITAGAWA, L., KOBAYASHI, T., BEPPU, T. et TERASHIMA, K. (2001). Semi-autonomous obstacle avoidance of omnidirectional wheelchair by joystick impedance control. *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*. vol. 4, 2148 –2153 vol.4.

KOREN, Y. et BORENSTEIN, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. 1398 –1404 vol.2.

LATOMBE, J.-C. (1990). *Robot Motion Planning :*. Kluwer international series in engineering and computer science : Robotics. Kluwer Academic Publishers.

LEWIS, J. (1995). Fast normalized cross-correlation. *Vision Interface*, 10, 120–123.

LU, D. (2012). Urdf and you. *ROSCON2012 Conference*.

MACMARTIN, STUART, E. A. (1992). Fastest point in polygon test. *Ray Tracing News 5(3)*.

MITTELSTAEDT et MITTELSTAEDT, H. (1980). Homing by path integration in a mammal. *Naturwissenschaften*, 67, 566–567.

MORAVEC, H. (1988). Sensor fusion in certainty grids for mobile robots. *AI Mag.*, 9, 61–74.

NEUMANN, J. V. et MORGENSTERN, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press.

NOURBAKHSH, I. (1998). Artificial intelligence and mobile robots. MIT Press, Cambridge, MA, USA, chapitre Dervish : an office-navigating robot. 73–90.

RÖFER, T. et LANKENAU, A. (1998). Architecture and applications of the bremen autonomous wheelchair. *Proc. of the Fourth Joint Conference on Information Systems*. 365–368.

SIEGEL, J. S. et (U.S.), N. A. I. C. (1996). *Aging into the 21st century*. National Aging Information Center, Administration on Aging, Bethesda, MD (7830 Old Georgetown Rd, Suite 100, Bethesda 20814-2434) :.

SIMPSON, R., LOPRESTI, E., HAYASHI, S., GUO, S., DING, D., AMMER, W., SHARMA, V. et COOPER, R. (2005). A prototype power assist wheelchair that provides for obstacle detection and avoidance for those with visual impairments. *Journal of NeuroEngineering and Rehabilitation*, 2.

SIMPSON, R. C. (2005). Smart wheelchairs : A literature review. *J Rehabil Res Dev*, 42, 423–36.

SIMPSON, R. C. (2008). How Many People Would Benefit From a Smart Wheelchair ? *Journal of Rehabilitation Research and Development*, 45, 53–71.

SMITH, R., SELF, M. et CHEESEMAN, P. (1990). Autonomous robot vehicles. Springer-Verlag New York, Inc., New York, NY, USA, chapitre Estimating uncertain spatial relationships in robotics. 167–193.

STENTZ, A., FOX, D., MONTEMERLO, M. et MONTEMERLO, M. (2003). Fastslam : A factored solution to the simultaneous localization and mapping problem with unknown data association. *In Proceedings of the AAAI National Conference on Artificial Intelligence*. AAAI, 593–598.

THRUN, S., BÜCKEN, A., BURGARD, W., FOX, D., FRÖLINGHAUS, T., HENNIG, D., HOFMANN, T., KRELL, M. et SCHMIDT, T. (1997). AI-based Mobile Robots : Case studies of successful robot systems. D. Kortenkamp, R. P. Bonasso et R. R. Murphy, éditeurs, *AI-based Mobile Robots*, MIT Press.

THRUN, S., BURGARD, W. et FOX, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. Intelligent robotics and autonomous agents. The MIT Press.

VALLIVAARA, I., HAVERINEN, J., KEMPPAINEN, A. et RONING, J. (2011). Magnetic field-based slam method for solving the localization problem in mobile robot floor-cleaning task. *Advanced Robotics (ICAR), 2011 15th International Conference on*. 198 –203.

WOLF, D. F. et SUKHATME, G. S. (2005). Mobile robot simultaneous localization and mapping in dynamic environments. *AUTONOMOUS ROBOTS*, 19, 53–65.

YIN, L. et YIN, Y. (2008). An improved potential field method for mobile robot path planning in dynamic environments. *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*. 4847 –4852.

ZALZAL, V., E. A. (2009). Acropolis : A fast protoyping robotic application. *Journal of NeuroEngineering and Rehabilitation 6(1)*, 1–6.

ANNEXE A

Cartographie et fusion de carte au premier étage de Lassonde

A.1 Cartographie

Nous allons détailler la procédure à suivre pour cartographier un nouvel espace à l'aide du FRMI. Le nœud responsable de la cartographie est « mappingFRMI », il contient un paquetage [1] de l'algorithme de cartographie fournie par (Grisetti *et al.*, 2007) « gridslam ». Le paquetage que nous proposons permet d'afficher et traquer les particules du filtre particulaire et d'initialiser de façon spécifique les variables. L'algorithme suit la procédure suivante :

- chaque particule est associée à une carte et une position. Notre paquetage initialise un nombre fixe de particules ; dans notre cas nous en utilisons 50. Pour chaque particule, nous associons une carte initiale. L'algorithme original est écrit de façon à initialiser les cartes de départs par une carte vide. Si nous voulons commencer la cartographie sur une carte existante, nous devons éditer l'implémentation originale [2].
- si nous avons une nouvelle mesure de laser et que le FRMI a bougé suffisamment, nous allons commencer le corps de l'algorithme. Dans notre cas nous considérons que le FRMI a bougé s'il translate sur 1 mètre ou s'il tourne de 0.5 rad.
- chaque particule x_p est mise à jour par intégration du modèle du mouvement avec une covariance sur les vitesses Q (voir les équations A.1 - A.6).

$$Q = \begin{bmatrix} 0.03 & 0.015 \\ 0.015 & 0.05 \end{bmatrix} \tag{A.1}$$

$$d = [v + |v|\mathcal{N}(0, Q(1,1)) + |w|\mathcal{N}(0, Q(1,2))]\, dt \tag{A.2}$$

$$\delta = [w + |v|\mathcal{N}(0, Q(2,1)) + |w|\mathcal{N}(0, Q(2,2))]\, dt \tag{A.3}$$

$$x_p = x + d\cos(\theta + 0.5\delta) \tag{A.4}$$

$$y_p = y + d\sin(\theta + 0.5\delta) \tag{A.5}$$

$$\theta p = \theta + \delta \tag{A.6}$$

1. En informatique, le paquetage de bibliothèques (en anglais, wrapper library) est la couche du code source qui expose l'interface d'une bibliothèque donnée en une interface compatible.

2. Le code source de l'implémentation du « gridslam » disponible sur l'adresse `http://openslam.informatik.uni-freiburg.de/data/svn/gmapping/trunk/gridfastslam/gridslamprocessor.cpp` devra être modifié à partir de la ligne 282.

- pour chaque particule, nous calculons un poids basé sur un critère de correspondance entre le balayage laser et la carte actuelle.
- pour chaque particule, en fonction d'une gaussienne centrée sur cette particule et avec une covariance proportionnelle à son poids, nous allons identifier la meilleure position respectant les deux critères de mesure du laser et de l'odométrie. Puis, nous mettons à jour la position de cette particule.
- intégrer le nouveau balayage laser à la carte associée à la particule.
- si les poids des particules dégénèrent, nous allons échantillonner de nouvelles particules. Le critère que nous utilisons est présenté dans l'équation A.7, avec $N_{eff} = 25$.

$$\frac{\left(\sum_{particules} poids(p)\right)^2}{\sum_{particules} poids(p)^2} < N_{eff} \tag{A.7}$$

Les figures A.1 montrent la dispersion des particules à différents instants de cartographie. Nous remarquons qu'au début les particules sont regroupées à la position du FRMI, puis en avançant le FRMI commence à découvrir de nouveaux endroits (la région gauche de la salle). Ceci implique que les positions des particules sont moins corrigées par la carte antérieure, d'où l'effet de dispersion des particules. Lorsque nous revenons à des endroits déjà visités, les particules erronées voient leur poids diminuer et disparaissent. Nous constatons alors un effet de concentration des particules.

A.2 Fusion avec une carte

Une fois la cartographie terminée nous proposons de la fusionner avec une carte connue à priori de l'environnement. Ceci dans le but, d'une part pour évaluer son exactitude, d'autre part, pour rajouter des informations négligées par les balayages laser. Nous avons montré la procédure utilisée dans la section 4.3.1. La figure A.3 illustre un autre résultat obtenu, cette fois du premier étage du bâtiment Lassonde à l'École Polytechnique de Montréal.

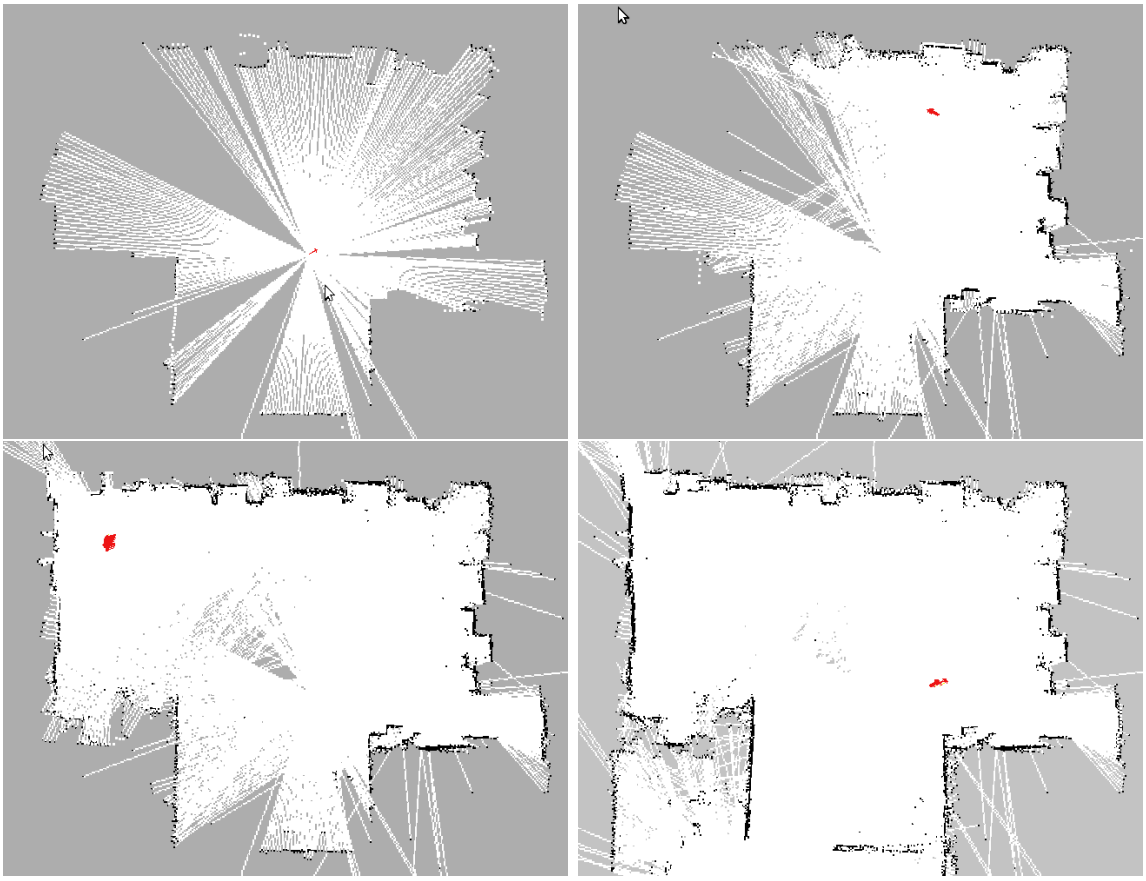


Figure A.1 État des 50 particules (en rouge) à différents instants de la cartographie

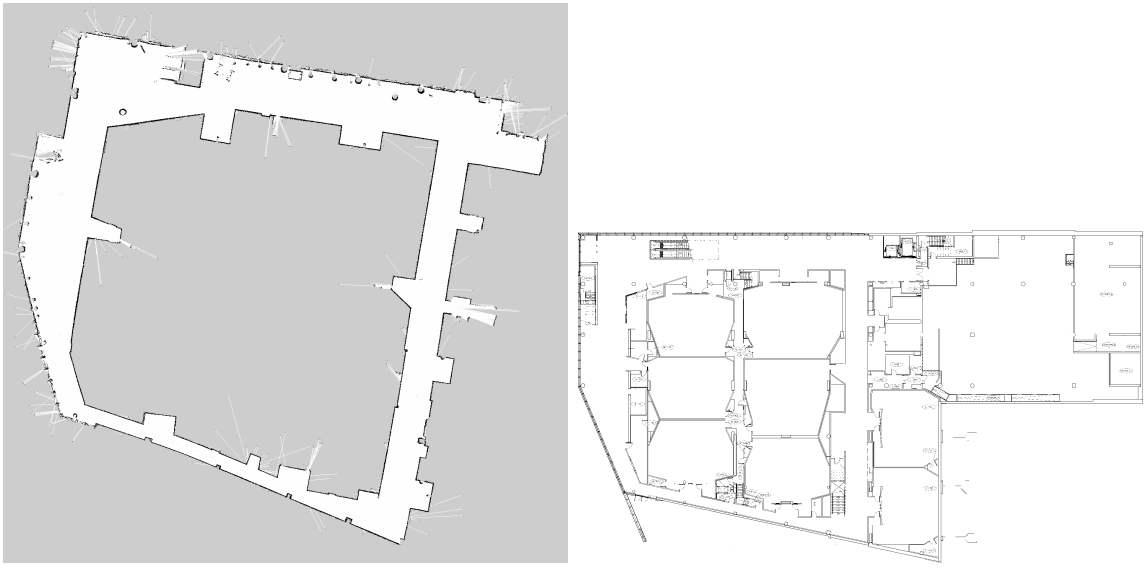


Figure A.2 Carte obtenue par cartographie (à gauche) et une carte fournie (à droite)

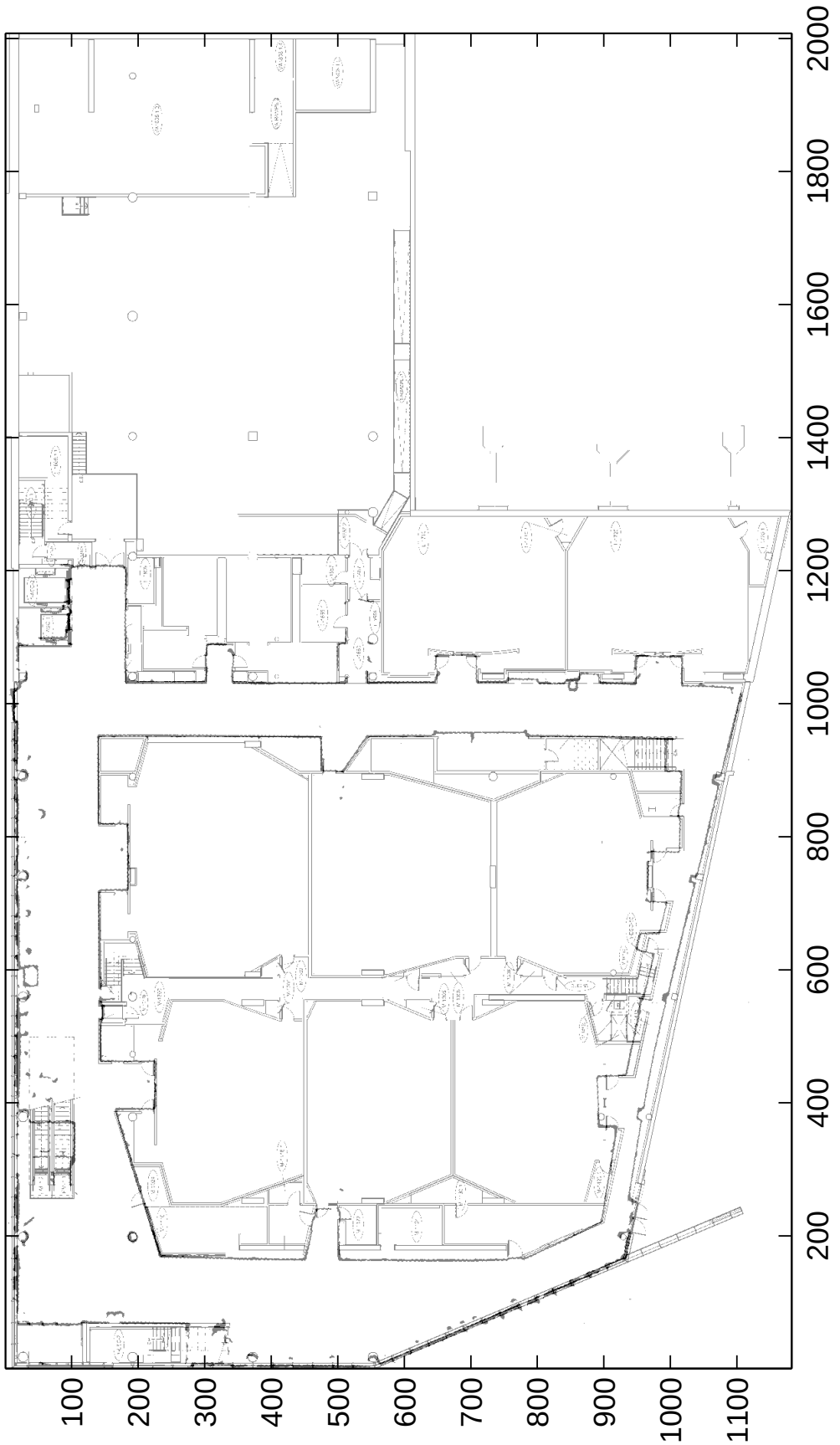


Figure A.3 Résultat de la fusion des deux cartes

ANNEXE B

Résultat d'autolocalisation et navigation point à point

B.1 Localisation globale

La localisation globale se base sur l'implémentation de l'algorithme de Monte-Carlo adaptatif décrit par (Fox *et al.*, 1999). Cet algorithme se caractérise par un nombre de particules variables (dans notre cas entre 500 et 5000 particules) et comme celui de la cartographie par une matrice de covariance sur l'odométrie, construite de façon similaire. Cette matrice, quantifiant les erreurs d'odométrie, dépend de plusieurs paramètres : état du sol, poids de l'utilisateur, état des roues, etc. Nous conseillons alors de la déterminer empiriquement en effectuant des navigations dans différents environnements. Le choix de Q est orienté comme suit :

- si les particules se dispersent rapidement, il faut diminuer la valeur de Q selon la dispersion : rotation, translation ou combiné ;
- si l'erreur persiste selon un mouvement, il faut augmenter la valeur de Q associée à ce mouvement.

Pour illustrer l'efficacité du filtre, nous avons effectué la manipulation suivante : à partir d'un point fixe sur une carte déjà construite nous allons effectuer une trajectoire qui combine une ligne droite, une rotation et un mouvement circulaire, puis retourner au point de départ. La figure B.1 illustre le résultat obtenu ; nous remarquons que l'erreur accumulée en rotation reste considérablement plus grande. Cependant, l'algorithme MCL reste assez stable et fournit une bonne correction de l'odométrie. Nous rappelons que l'algorithme actuel n'est pas très efficace dans les situations d'enlèvement (hijacking), surtout avec l'utilisation des grandes cartes.

B.2 Navigation globale

À chaque fois que l'usager désire se rendre à un endroit déjà visité, il doit choisir une position dans le repère de la carte. Le FRMI doit alors connaître sa position avec précision pour planifier son chemin. En somme, une bonne localisation globale est une condition nécessaire pour assurer la navigation point à point. Nous présentons un exemple de navigation point à point dans une carte donnée :

- nous commençons par assigner la position désirée sur la carte et déclencher la procédure avec un bouton du contrôleur (voir figures B.2) ;

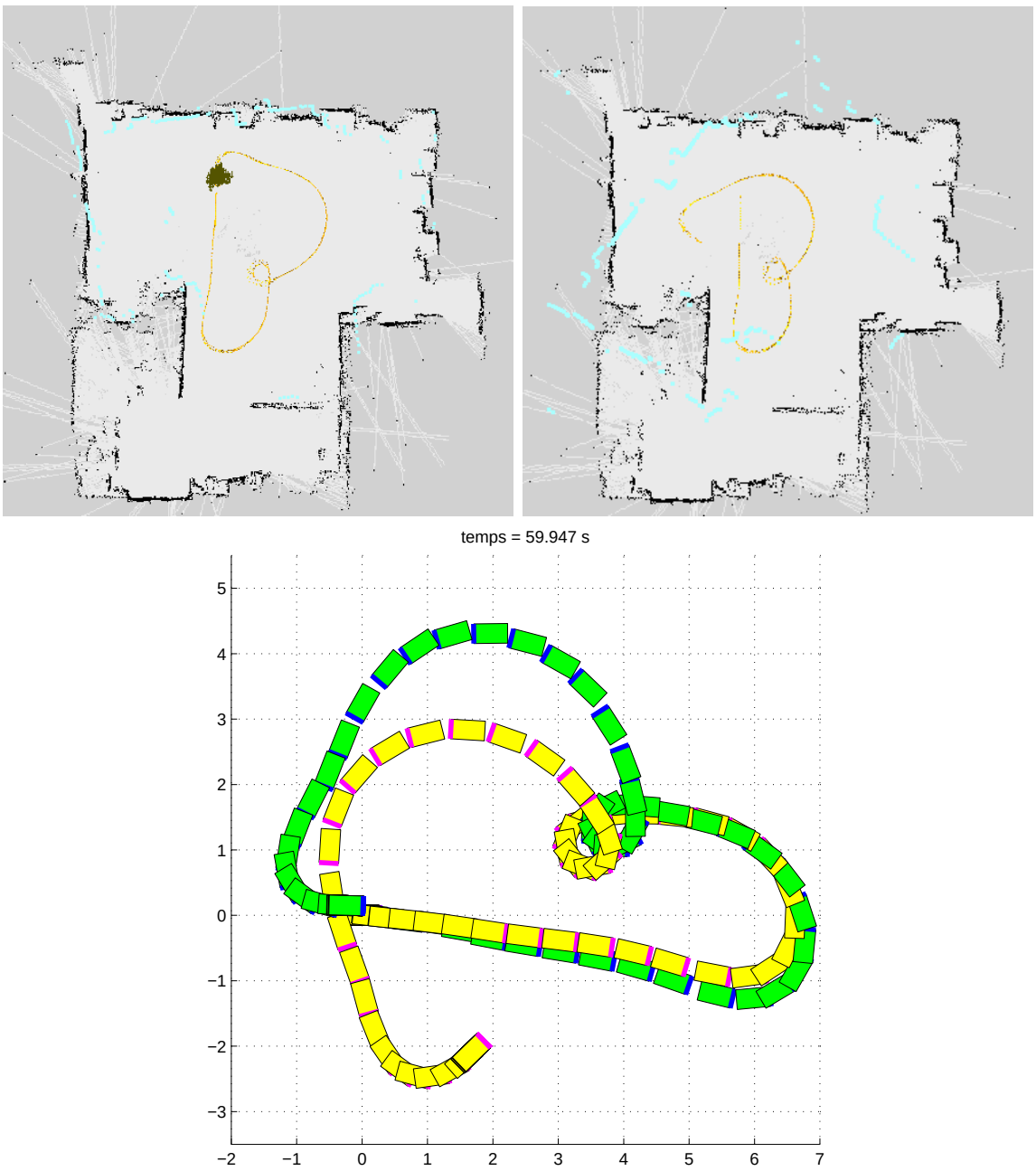


Figure B.1 Test de l'algorithme MCL : À droite, la trajectoire (en jaune) effectuée avec odométrie seulement. À gauche, la trajectoire (en jaune) effectuée avec correction. En bas la superposition des deux trajectoires, en jaune, avec odométrie seule, en vert, avec application du MCL.

- l'algorithme de planification commence par itérer une première trajectoire se basant sur la carte statique ;
- lors de la détection d'un nouvel obstacle perçu par les capteurs, l'algorithme change la trajectoire pour le contourner (voir figures B.3) ;
- l'algorithme s'arrête et le FRMI s'immobilise quand nous arrivons au point désiré (voir figures B.4).

Tout au long de cette manœuvre, nous ne pouvons pas nous permettre d'effectuer une erreur de localisation dans la carte, sinon le FRMI ne pourra jamais suivre la trajectoire assignée.

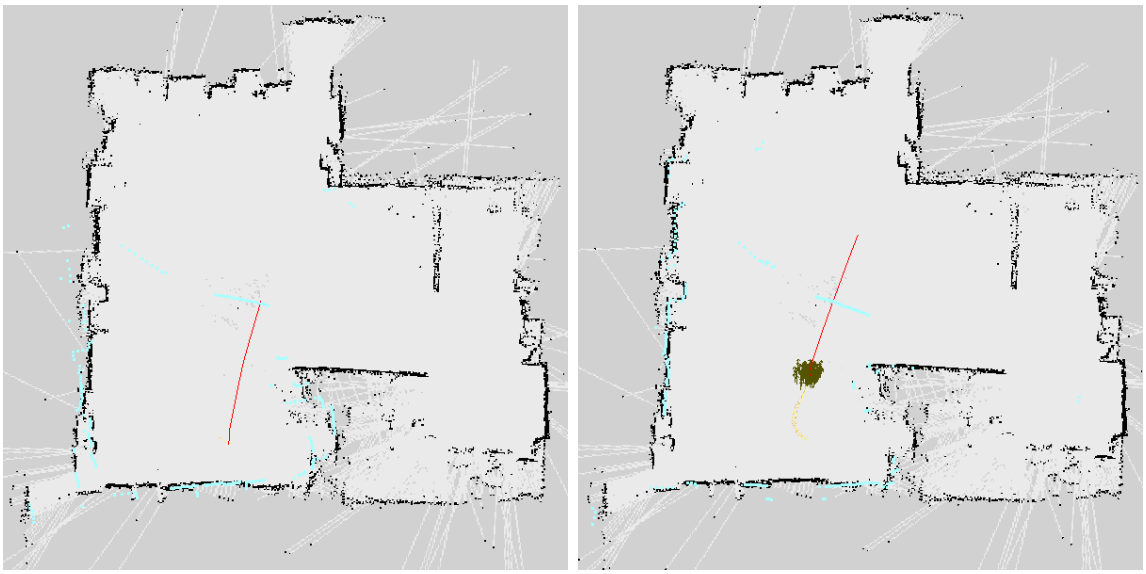


Figure B.2 Navigation globale : à gauche, en vert le point de départ du FRMI et en rouge le chemin initial planifié. À droite, les particules en vert montrent la position estimée avec MCL du FRMI.

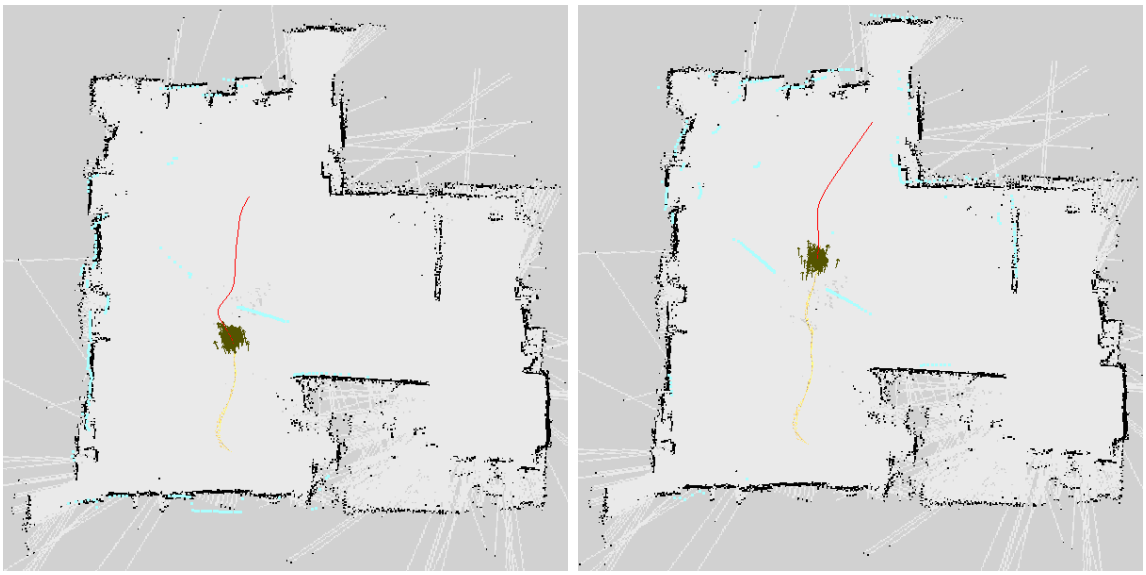


Figure B.3 Navigation globale : évitement d'obstacle statique.

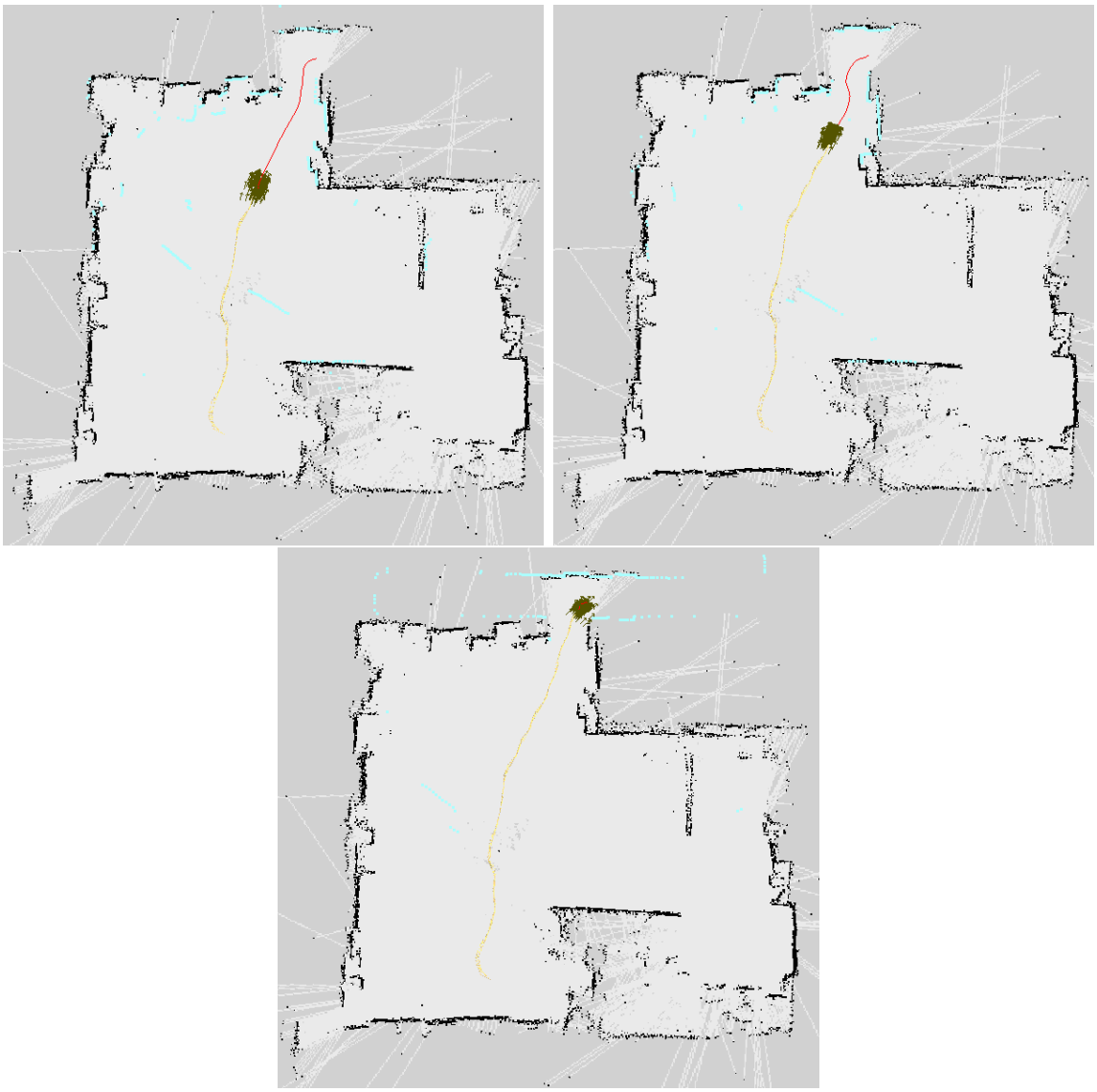


Figure B.4 Navigation globale : traversée de la porte et arrivée a la position demandée