Planification et exécution de mouvements référencés sur des amers

Abed Choaib Malti

THÈSE

présentée au

Laboratoire d'Analyse et d'Architecture des Systèmes

en vue de l'obtention du

Doctorat de l'*Université Paul Sabatier de Toulouse*
Ecole Doctorale Systèmes
Spécialité : **Robotique / Systèmes Informatiques**

par **Abed Choaïb Malti**

PLANIFICATION ET EXÉCUTION DE MOUVEMENTS RÉFÉRENCÉS SUR DES AMERS

Soutenue le 21 décembre 2005, devant le jury composé de :

Raja Chatila	*Président*
Dominique Meizel	*Rapporteur*
Patrick Rives	*Rapporteur*
Michel Courdesses	*Examinateur*
Florent Lamiraux	*Directeur de thèse*
Michel Taïx	*Directeur de thèse*

Avant propos

Ce mémoire porte sur les travaux de recherche effectués au sein du LAAS-CNRS. Je remercie Monsieur Jean Claude Laprie, son ancien directeur, et Monsieur Malik Ghallab, son successeur, de m'y avoir accueilli.

Je remercie Monsieur Raja Chatila, directeur du groupe Robotique et Intelligence Artificielle du LAAS-CNRS, de m'avoir permis de travailler au sein de son équipe de recherche et pour toute son aide durant mes trois années de thèse.

Je voudrais exprimer ma gratitude envers mes directeurs de thèse Michel Lamiraux et Florent Taïx[1] d'avoir su trouver la méthode locale et la commande qui m'ont permis de surmonter mes contraintes non-holonomes et de m'amener à la configuration finale «thèse». L'expérience humaine apprise en les côtoyant n'en est pas moins importante. Merci pour tout.

J'adresse mes remerciements à Dominique Meizel et à Patrick Rives qui ont accepté d'être les rapporteurs de cette thèse soutenue devant le jury auquel ont participé Raja Chatila, son président, Michel Courdesses et mes directeurs de thèse. Qu'ils soient tous assurés de ma gratitude pour avoir jugé et enrichi mon travail.

Je voudrais remercier Viviane Cadenat, Patrick Danès et Frédéric Lerasle pour leur aide durant les TP des IUP-SI2 et pour toutes les discussions que nous avons pu échanger dans le domaine de l'automatique. J'aimerais également remercier Sara Fleury, Matthieu Herrb, David Bonnafous et Olivier Lefebvre de m'avoir aidé à mener à bien les expérimentations sur Hilare2.

Je voudrais remercier les membres du consulat d'Algérie à Toulouse et les membres du service étudiant étranger du CROUS de Toulouse pour tous leurs efforts pour rendre notre séjour en France des plus agréables.

Carré réservé aux thésards RIA : Léonard avec qui je partage les frontières de bureau (nous n'avons jamais eu de conflit territorial et sans avoir à signer de traité), et avec qui j'étais dans la même barque pour le sprint final. Je tiens à le remercier pour avoir contribué à la correction orthographique du manuscrit de son plein gré. Un ex-thésard mais toujours concerné par le sort de ses anciens collègues de bureau, Jérémi bien sûr. Daniel Sidobre, le permanent (mais aussi l'in-

[1] idée : c.f thèse Julien Pettré

trus dans ce paragraphe) plein de bonne humeur qui tient la liste des thésards qui sont passés dans le bureau B-159. Le groupe de l'école d'été à Sophia Antipolis 2004 où «été» et «école» ont fait bon ménage. L'équipe omni-sport de Saubion avec ses managers Aurélie, Thierry et Vincent. Dommage qu'il n'y ait pas de compétition sportive inter-labo, nous aurions cartonné. Un clin d'oeil à tous les autres thésards pour leurs sympathies : Sylvain, Nico, Nacio, Abdellatif, Thomas, Wael, ...une pensée à nos ancêtres : Delphine, Fabien, Fred, Solange, William ...

Avant de continuer, je voudrais présenter toutes «mes excuses» à Hilare2 pour toutes les fois, où par ma faute, il a fini contre le mur.

Carré Personnel :

A tous les latécoèriens[2], latécoèriennes, ex-latécoèriens et ex-latécoèriennes pour tous les moments de partages, de gaieté ...

A tous mes frères et amis toulousains pour tout ce qu'on a partagé dans la joie comme dans la difficulté, à tous mes frères et amis nancéiens, messins et tlemceniens[3]. Sans citer leur noms, de peur d'en oublier certains, qu'ils trouvent ici l'expression de ma gratitude pour leurs soutiens et encouragements.

A mes grands frères Saad-Eddine et Zakaria, pour leurs conseils, encouragements et soutiens.

A mes petites soeurs et mes petits frères, Ilhem et Dounyazed, Samir et Tarik, pour leur soutien affectif durant mes quatre années d'absence.

A mes parents pour tout leur amour, leurs sacrifices et leur soutien et encouragements qui m'ont permis d'arriver au bout de cette aventure.

A toute ma famille à Tlemcen, Oran et à Paris, merci pour tout.

Aux nombreuses personnes que j'ai côtoyées et que je n'ai pas pu citer. Qu'elles trouvent ici l'expression de ma sympathie.

Toute ma gratitude, et plus encore, va à celui sans qui tout ceci n'aurait pas été possible.

[2]habitants de la résidence Latécoère

[3]habitants de Tlemcen (Algérie)

Table des matières

I Introduction **1**

II État de l'art : de la planification de trajectoire à l'exécution du mouvement **5**
II.1 Introduction 5
II.2 Planification de chemins géométriques 6
II.2.1 Formulation du problème 6
II.2.2 Méthodes de planification de chemin géométrique 9
II.3 Localisation sur amers 16
II.3.1 Localisation probabiliste 16
II.3.2 Localisation non probabiliste 20
II.3.3 Sélection d'amers pour la localisation 21
II.4 Navigation référencée capteur 22
II.5 Idée générale de notre approche 28
II.6 Conclusion 30

III Mouvement asservi sur des amers **33**
III.1 Introduction 33
III.2 Notations et définitions 34
III.2.1 Amer 34
III.2.2 Capteur 34
III.2.3 Le couple capteur-amer 34
III.3 Localisation 35
III.3.1 La localisation autour d'une configuration de référence 37
III.3.2 Exemple didactique de localisation 38
III.4 Pondération des couples capteurs-amers 39
III.5 Mouvement asservi sur des amers 41
III.6 Asservissement sur la localisation 42
III.6.1 Existence de la configuration corrigée 43
III.6.2 Étude de la stabilité autour de la configuration corrigée 46
III.6.3 Calcul de l'expression de la configuration corrigée 48
III.6.4 Calcul du résidu de la perception en configuration corrigée 49
III.7 Conclusion 50

IV Principes et calculs des fonctions de pondération 53
IV.1 Introduction . . . 53
IV.2 Exemple de l'effet statique de la variation des poids . . . 54
IV.3 Fonctions de pondération . . . 54
IV.4 Construction des fonctions de pondération . . . 56
IV.4.1 Principe Général . . . 56
IV.4.2 Fonction de pondération associée à la perception . . . 57
IV.4.3 Fonction de pondération associée à la collision . . . 58
IV.4.4 Fonction de pondération prédéfinie . . . 58
IV.4.5 Calcul générique de la fonction de pondération . . . 58
IV.4.6 Contraintes cinématique et fonction de pondération . . . 59
IV.5 Algorithmes de sélection de couples capteurs-amers . . . 60
IV.5.1 Algorithme classique : "Take All" . . . 63
IV.5.2 Algorithme de sélection des meilleurs amers . . . 65
IV.5.3 Algorithme des plus proches amers le long de la trajectoire . . . 68
IV.6 Résultats de simulation . . . 71
IV.7 Conclusion . . . 75

V Implémentation sur Hilare 2 avec remorque 79
V.1 Introduction . . . 79
V.2 Modélisation . . . 79
V.2.1 Capteurs . . . 79
V.2.2 Amers . . . 80
V.2.3 Équations de localisation . . . 82
V.2.4 Calcul des fonctions de pondération . . . 82
V.3 Intégration software . . . 85
V.3.1 Planification de mouvement asservi sur amers . . . 86
V.3.2 Contrôle du mouvement asservi sur amers . . . 86
V.4 Résultats expérimentaux . . . 90
V.4.1 Manoeuvre dans un couloir . . . 90
V.4.2 Manoeuvre de garage . . . 94
V.5 Amélioration des algorithmes de sélection . . . 96
V.5.1 Sélection tenant compte du conditionnement du système de localisation linéarisé . . . 96
V.5.2 Sélection tenant compte de l'appariement des amers . . . 101
V.6 Réalisation d'une tâche de navigation en environnement structuré . . . 104
V.7 Conclusion . . . 107

VI Conclusion 111

Références bibliographiques 115

Chapitre I

Introduction

Un peu de rêve, de fiction et d'histoire

«Tout ce qu'un homme est capable d'imaginer, d'autres hommes seront un jour capables de le réaliser.»

Jules Verne (1828-1905)

Depuis des siècles, les récits de science-fiction exploitent les robots. En effet, selon le chant 18 de l'Iliade d'*Homère* qui date du 8^e siècle avant l'ère chrétienne, *Héphaïstos* (dieu des forgerons chez les Grecs, plus connu sous son nom romain *Vulcain*) fut le premier fabricant de machines artificielles. *Homère* y décrit des tables circulaires sur trois pieds, munies de roues. Autonomes, elles se rendaient seules vers l'Olympe pour y transporter les produits de la forge d'*Héphaïstos*. Ce même personnage de la mythologie grecque s'était également construit deux servantes en or qui l'assistaient dans ses travaux, *Homère* précise même qu'elles pouvaient parler et penser. La civilisation égyptienne, quant à elle, n'est pas en reste. En effet, certaines statues de dieux égyptiens retrouvées dans les tombeaux sont articulées, comme le masque d'*Anubis* dont la mâchoire est mobile. Certains chercheurs n'hésitent pas à y voir les ancêtres de nos automates même si l'appellation «robot » n'y figure pas encore.

Le terme «robot» n'apparaît qu'en 1920 sous la plume du romancier tchèque *Karel Capek*, terme signifiant *travail forcé* en sa langue maternelle. Dans sa pièce de théâtre, le savant vedette *Rossum* inventait une série de robots si intelligents qu'ils finirent par dominer le monde. Depuis, des romans du célèbre écrivain *Isaac Asimov* (1920-1992) jusqu'au cinéma de science-fiction tel les fameuses trilogies de «*La Guerre des étoiles*», l'imaginaire humain ne cesse d'être productif à ce sujet.

Retour à la réalité

Les découvertes et nombreuses avancées dans des disciplines telles que la mécanique, l'automatique et l'informatique ont permis durant la seconde moitié du siècle précédent de donner les premières lueurs d'espoir à l'homme pour commencer à construire la réalisation de l'un de ses rêves de toujours, concevoir des robots autonomes.

Un robot autonome est capable d'interpréter des descriptions de tâches à un haut niveau et de les exécuter sans une intervention humaine. Les descriptions d'entrées spécifient la volonté de l'utilisateur sans pour autant qu'il ait besoin de spécifier la démarche à suivre afin d'atteindre cet objectif. Pour ce faire, le robot dispose de moyens de planification lui permettant d'établir un plan de son évolution dans son environnement, de moyens de locomotion et d'action lui permettant d'évoluer dans cet environnement. Des outils sensoriels proprioceptifs permettent au robot d'avoir des informations sur son état interne et des capteurs extéroceptifs lui fournissent un retour perceptuel de sa situation par rapport à l'environnement afin de suivre et corriger son évolution par un processus de contrôle.

De nos jours, les robots ne sont pas encore complètement autonomes mais offrent néanmoins des performances satisfaisantes et occupent des secteurs variés dans la vie quotidienne des hommes, allant des robots d'assemblage dans les chaînes de montages industriels, aux robots d'exploration spatiale et sous-marine, en passant par les robots pour l'assistance médicale et chirurgicale, etc.

Cadre de notre travail

Développer les technologies nécessaires pour les robots autonomes est un formidable défi qui englobe le raisonnement automatique, la perception et le contrôle. L'un des aspects autonomes que le robot doit être capable d'effectuer est d'exécuter des mouvements sûrs dans son environnement lui permettant d'une part de faire face aux différents obstacles statiques, changeants et dynamiques qu'il peut rencontrer durant son évolution et d'autre part d'offrir une certaine fiabilité quant à la réalisation de l'objectif de son mouvement.

Durant les trois dernières décennies, les chercheurs en robotique ont produit beaucoup de travaux dans le domaine de la génération automatique de mouvement. Généralement, ce mécanisme se passe en deux étapes ; une première étape où un chemin géométrique reliant une situation initiale du robot à une situation finale est planifié et une deuxième étape où ce chemin est exécuté. La phase de planification du chemin repose sur une connaissance a priori parfaite de la géométrie de l'environnement et de la structure cinématique du robot. La plupart des efforts dans ce domaine ont débuté dans les années 1980. Durant les dix dernières années, la compréhension théorique et pratique de certaines questions liées à la planification de chemin ont permis rapidement de produire des algorithmes capables de résoudre une large classe de problème de planification de chemin. La deuxième étape qui consiste à exécuter un chemin planifié en présence d'obstacles

n'est pas une tâche aisée pour plusieurs raisons.

Premièrement, le plan utilisé pour planifier le chemin géométrique n'est jamais exact d'autant que l'environnement peut parfois être sujet à des changements dynamiques des obstacles modélisés ou bien encore être sujet à des apparitions d'obstacles non modélisés.
Deuxièmement, en raison de l'imperfection des capteurs du robot, la connaissance de la situation géométrique du robot dans son environnement n'est jamais parfaite.

Depuis une quinzaine d'années beaucoup de techniques ont été proposées pour résoudre ces deux questions. Ces techniques se basent généralement sur l'utilisation de capteurs extéroceptifs permettant d'asservir le robot sur des primitives géométriques de l'environnement et ainsi de déterminer sa situation pour rendre l'exécution du chemin robuste à l'égard des imperfections du modèle de l'environnement et des erreurs de mesure des capteurs. Notre travail s'inscrit dans cette thématique.

Objectif de notre travail

Quand on effectue un créneau entre deux voitures pour essayer d'amener son véhicule parallèlement à la bordure d'un trottoir, intuitivement, on prend comme base de départ un chemin appris au préalable (après de nombreuses manoeuvres effectuée chez son moniteur d'auto école) dont on asservit l'exécution sur des primitives de l'environnement, par exemple : le bord du trottoir ou l'arrière et l'avant des véhicules adjacents. Même si dans le domaine de la robotique, la question du chemin planifié est résolue dans ce cas, la question de l'extraction automatique des primitives d'asservissement reste posée.

Dans notre travail nous proposons de fournir pour un chemin géométrique sans collision des primitives de l'environnement qui serviront lors de la phase d'exécution à contrôler la position et l'orientation du robot par une boucle d'asservissement tenant compte des perceptions prévues par le plan de référence et des perceptions mesurées par les capteurs extéroceptifs dans l'environnement réel. Dans notre travail on considère que l'environnement est statique et que tous les obstacles sont modélisés.

L'originalité de notre travail réside dans la spécification de l'asservissement du chemin par l'attribution de fonctions d'importance aux différentes primitives de l'environnement. Ces fonctions servent d'une part à la sélection des primitives au cours de l'exécution du chemin et d'autre part à l'asservissement de la trajectoire. Aussi le formalisme proposé a pour objectif d'être suffisamment générique pour être applicable à tout type de robot, tout de type de capteur extéroceptif et tout type de primitive.

Plan du manuscrit

Le manuscrit va dans le second chapitre aborder les principaux travaux concernant la planification de chemin géométrique sans collision. Nous mentionneront dans ce chapitre les différentes

techniques de localisation de robot et principalement les techniques probabilistes de localisation. Ensuite nous discuterons des principales méthodes de navigation référencées capteurs en présentant quelques travaux qui ont abordé ce sujet. Ceci nous mènera à la fin du chapitre à introduire notre travail et à le positionner par rapport à cette littérature.

Dans le chapitre 3 nous posons la définition formelle de notre approche de *mouvement asservi sur amers* qui constitue la principale contribution de notre travail. Au cours de ce chapitre nous montrerons comment nous localisons notre robot par une méthode purement géométrique qui tient compte de l'importance des primitives de l'environnement. Ce résultat de localisation, qui représente essentiellement un écart de perception entre le plan de référence et les mesures des capteurs extéroceptifs, est utilisé dans un processus d'asservissement afin d'amener le robot, dans l'environnement d'exécution, dans une configuration, dite *configuration corrigée*, qui respecte la configuration correspondante sur le chemin planifié. Nous étudierons alors l'existence de cette configuration et la stabilité du système robotique pour cette configuration. Dans ce chapitre nous supposons les primitives d'asservissement avec leurs fonctions de pondération connues.

Le chapitre 4 s'intéresse aux propriétés des fonctions de pondération des primitives. Dans ce chapitre on propose des méthodes génériques permettant de les construire. La construction de ces fonctions va nous permettre de proposer des algorithmes permettant de sélectionner les primitives d'asservissement et de définir leurs successions le long du chemin géométrique planifié. Nous définissons ainsi la structure d'un *mouvement asservi sur amers*.

Le chapitre 5 est un chapitre de première validation de notre approche. Les résultats expérimentaux qui y sont présentés, ont été testés sur le robot mobile non holonome Hilare2 auquel est attaché une remorque (voir figure II.6). Ce robot est équipé de deux scanners lasers situés à l'avant et à l'arrière sur la remorque. Ce chapitre propose une instanciation des méthodes formelles présentées au cours des deux chapitres précédents. Dans la dernière partie de ce chapitre, on discutera de quelques problèmes survenus lors de l'implémentation de notre approche notamment sur la sélection des primitives d'environnement. On proposera alors des solutions qui amélioreront la production du *mouvement asservi sur amers*.

A la fin du manuscrit, une conclusion générale permettra de mettre en avant des futurs travaux complémentaires.

Chapitre II

État de l'art : de la planification de trajectoire à l'exécution du mouvement

II.1 Introduction

Durant les deux dernières décennies de nombreux chercheurs de la communauté robotique ont traité deux questions qui se posent en robotique, à savoir comment planifier un chemin géométrique sans collision et comment exécuter un tel chemin. Au début des années 80 le travail de Shwartz et Sharir [78] a apporté une réponse à la résolution de la première question, plus connue sous le nom du problème du «déménageur de piano», qui peut se résumer à trouver un chemin géométrique sans collision pour des corps rigides évoluant dans un environnement comprenant des obstacles connus. Au jour d'aujourd'hui on peut considérer que cette question est résolue pour une large classe de problème. La deuxième question qui concerne l'exécution du chemin planifié a aussi connu de nombreux développements. À cause des imperfections des modèles et pour prendre en compte des événements non prévus lors de la phase de planification des chemins, les chercheurs ont développé des techniques permettant de suivre l'évolution du robot dans son environnement, problème plus connu sous le nom de localisation et se sont aussi penchés sur l'élaboration de stratégies basées sur le retour sensoriel des capteurs extéroceptifs du robot pour rendre l'exécution de la trajectoire plus robuste, problème plus connu sous le nom de navigation référencée capteur.

Notre contribution s'inscrit dans la deuxième catégorie de travaux où dans la perspective de rendre l'exécution du chemin planifié plus robuste nous le munissons, dans une phase de planification, de stratégies de mouvement qui rendront plus sûre son exécution.

Dans ce chapitre nous présentons une vision assez large des travaux qui ont traité des différentes questions de planification de chemins géométriques, de localisation et enfin de navigation référencée capteur. Nous terminerons en présentant notre approche de mouvement et en la situant par rapport à la littérature.

II.2 Planification de chemins géométriques

Le problème de la planification de chemin se pose dans divers domaines comme la robotique, l'analyse d'assemblage, le prototypage virtuel, l'usinage et l'animation graphique, en résumé, dès qu'on veut faire bouger un corps dans un environnement encombré d'obstacles.

En effet lorsque on veut faire évoluer un robot dans un environnement connu, la question de la génération d'un mouvement sans collision est alors soulevée. La planification de chemins géométriques constitue la première étape pour la construction de trajectoires exécutables en environnement réel. Ici seul le problème géométrique de la non collision du robot avec l'environnement est pris en compte. Même si la construction de ce chemin géométrique est simple dans le cas de robots ponctuels évoluant dans des environnements peu contraints, elle devient très difficile dès que le robot a une structure complexe et/ou l'environnement contraint fortement les déplacements. D'où la nécessité et toute la difficulté de développer des techniques génériques permettant d'automatiser la construction d'un tel chemin.

Le problème «basique » de la planification de chemin implique le calcul d'un chemin sans collision entre deux situations géométriques dans un environnement composé d'obstacles connus. Dans ce cas, les contraintes sur le chemin solution relève de la géométrie des obstacles et de celle du robot. Si ce dernier peut être représenté comme un unique objet rigide pouvant effectuer des rotations et des translations, alors le problème peut être assimilé au «problème du déménageur de piano» [78].
Habituellement, on distingue la «planification de mouvement» de la «planification de chemin» lorsque ce dernier est paramétré en temps.

Nous allons voir dans ce paragraphe les grandes familles de méthodes permettant de résoudre ce problème après avoir introduit sa formulation.

II.2.1 Formulation du problème

Un robot R est un ensemble de solides reliés entre eux par des liaisons mécaniques. Il est de plus muni de capteurs extéroceptifs lui permettant d'avoir un retour sensoriel de l'environnement et d'actionneurs lui permettant de se déplacer. Ces différents solides évoluent dans l'*espace de travail* $\mathscr{W}$, il s'agit en général du plan $(\mathbb{R}^2)$ ou de l'espace $(\mathbb{R}^3)$. La position et l'orientation des différents solides sont déterminées par rapport à un repère de base fixe associé à $\mathscr{W}$.

La situation géométrique du robot dans son espace de travail est déterminée par un vecteur multidimensionnel $\mathbf{q}$ appelé *configuration* du robot. Ainsi on peut noter $R(\mathbf{q})$ la position et l'orientation du repère lié à R par rapport au repère de base lorsque le robot se trouve à la configuration $\mathbf{q}$. L'ensemble des configurations possibles du robot est appelé *espace des configurations* $\mathscr{C}$. Certaines parties de l'espace de travail du robot sont occupées par des obstacles. D'un point de vu formel, ces obstacles peuvent être représentés par des modèles semi-algébriques ou bien par un ensemble de polygones et de polyhèdres.

On appelle *espace des configurations libre*, $\mathscr{C}_{libre}$, l'ensemble des configurations pour lesquelles l'intersection entre le robot et les obstacles est vide. L'introduction de la notion d'espace des configurations [55] a permis de substituer la recherche d'un chemin sans collision pour un robot R évoluant parmi des obstacles de $\mathscr{W}$ par la recherche de la courbe d'un point évoluant dans l'espace des configurations libres du robot. Si le déplacement d'un point dans un espace est simple il va nécessiter le calcul de la transformée des obstacles dans ce nouvel espace $\mathscr{C}$. Nous verrons que la complexité de ce calcul est évitée par les méthodes récentes de planification. Ainsi un chemin est une fonction continue dans l'espace de configuration $\mathscr{C}$:

$$\begin{array}{rccc} \gamma : & [0,U] & \longrightarrow & \mathscr{C} \\ & u & \longmapsto & \gamma(u) \end{array}$$

avec u appelé paramètre du chemin géométrique. $\gamma(0) = \mathbf{q}_i$ est la configuration initiale du robot et $\gamma(U) = \mathbf{q}_f$ est la configuration finale du robot. Un chemin est dit *libre* s'il est entièrement contenu dans $\mathscr{C}_{libre}$. On dit qu'un chemin est *admissible* s'il respecte les contraintes intrinsèque au robot (chaîne fermée, non-holonomie, etc). Nous appelons un chemin *faisable*, un chemin libre et admissible.

Ainsi le problème de la planification de chemin peut être formulé par *la recherche d'un chemin faisable entre une configuration initiale* $\mathbf{q}_i$ *et une configuration finale* $\mathbf{q}_f$ *pour un robot R évoluant dans un espace de travail* $\mathscr{W}$.

Typiquement le problème de planification de chemin est soumis à des contraintes qui dépendent de l'environnement et de la structure cinématique du robot. Une première classe de contraintes concerne les contraintes dites *holonomes*. Elles imposent des conditions d'ordre géométrique en interdisant certaines régions de l'espace de configuration. Une sous classe de contraintes holonomes appelées *contraintes d'inégalité holonomes* [47] concerne des régions interdites de l'espace des configurations en plus des régions en collision avec les obstacles de l'environnement. Dans le cas des chaînes articulées, cette zone correspond aux configurations d'auto-collision entre les corps constituant cette chaîne. Un autre exemple de contrainte d'inégalité holonome s'observe dans le cas d'interaction homme-robot où une zone de sécurité est imposée au robot. Les contraintes holonomes concernent aussi les contraintes géométriques sur les chaînes fermées et la manipulation d'objet.

Si les contraintes citées jusqu'à présent réduisent l'ensemble des configurations libres, il

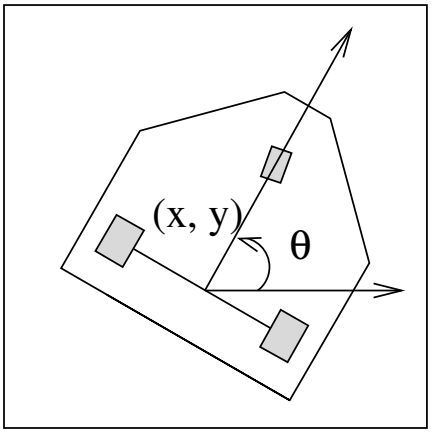

FIG. II.1 – Exemple de plate-forme mobile non-holonome avec une roue folle à l'avant.

existe d'autres types de contraintes parmi lesquelles la contrainte dite *non-holonome* à laquelle nous consacrons le paragraphe suivant.

Contrainte non-holonome

Une contrainte *non-holonome* s'exprime par une équation linéaire sur l'espace des vitesses qui est non intégrable. Dans ce cas l'espace des vitesses accessibles a une dimension inférieure à celle de l'espace des configurations. Typiquement, dans le domaine de la robotique ce problème se pose dans le cas des robots mobiles équipés de deux roues motrices parallèles portées par un même axe et une ou plusieurs roues folles (voir figure II.1). Qui dit roue, dit contrainte de roulement sans glissement. Ainsi la vitesse des points de l'axe des roues motrices est parallèle à l'orientation du robot empêchant ainsi tout déplacement latéral. Un changement de configuration dans le sens latéral n'est alors possible qu'avec un certain nombre de manoeuvres.

Formellement ceci se traduit par le fait que la vitesse du robot est contrainte dans un sous espace de dimension deux de l'espace tangent de l'espace des configurations $\mathscr{C}$ du robot qui est de dimension trois ($\mathscr{C} = \mathbb{R}^2 \times \mathbb{S}^1$). Même si l'espace des vitesses a une dimension inférieure à celle de l'espace des configurations, ces systèmes disposent d'une propriété intrinsèque dite de *commandabilité locale* leurs permettant d'atteindre tout point de l'espace des configurations libres.

Un système est dit localement commandable si pour tout voisinage $\mathscr{V}$ d'une configuration **q**, l'ensemble des configurations accessibles par un chemin inclus dans $\mathscr{V}$ est encore un voisinage $\mathscr{V}'$ de **q** [51, 45]. Cette notion permet d'approximer un chemin faisable quelconque par une séquence finie de chemins respectant des contraintes non holonomes [50]. Ces résultats ont été étendus au cas plus complexe d'un véhicule tractant une remorque étudié par [43]. Ce type de véhicule est la plateforme sur laquelle nous avons mené les expériences réelles de notre travail.

Malgré sa complexité, ces résultats ont permis de formuler et de résoudre le problème de la planification de mouvement pour des systèmes soumis à des contraintes non-holonomes dans l'espace des configurations. Il est alors possible d'élaborer des techniques génériques de planification pouvant être appliquées aussi bien aux systèmes holonomes qu'aux systèmes non-holonomes. Dans ce qui suit nous allons présenter les principales techniques de planification de chemin géométrique.

II.2.2 Méthodes de planification de chemin géométrique

La première difficulté de la planification de trajectoire provient de la contrainte de continuité du chemin à travers l'espace des configurations. Les approches proposées se basent essentiellement sur la décomposition de l'espace des configurations et/ou sur la discrétisation du chemin géométrique modélisant ainsi le problème sous la forme d'un graphe.

Depuis la définition de l'espace des configurations comme espace de recherche durant le début des années 80, la planification de chemin est devenue rapidement un domaine de recherche très actif qui a connu principalement l'émergence de trois approches :
- la décomposition cellulaire,
- les champs de potentiel,
- les méthodes aléatoires.

Pour une méthode donnée, si le planificateur peut répondre à la question de l'existence d'une solution alors la méthode est dite *complète*.

Ces trois méthodes se déroulent généralement en deux étapes : La première est une étape de pré calcul, ou encore d'apprentissage, où la connexité de l'espace $\mathscr{C}_{libre}$ est représentée par l'intermédiaire d'un graphe. La seconde étape, étape de requête, consiste à rechercher un chemin libre entre deux configurations en se basant sur le graphe construit au préalable. D'un point de vu de la complexité algorithmique, la planification de chemin prend un temps exponentiel suivant la dimension de l'espace des configurations et un temps polynomial suivant la complexité des obstacles (nombre de surfaces et degré des équations algébriques qui les modélisent). Voyons à présent les principaux travaux concernant ces méthodes durant les 25 dernières années.

Techniques de décomposition cellulaire

Dans les méthodes de décomposition cellulaire [78, 22], $\mathscr{C}_{libre}$ est décomposé en un ensemble de cellules dont la connexité est représentée par un graphe d'adjacence. Ces cellules sont convexes et ne se chevauchent pas. À l'étape de requête et après avoir identifié les cellules contenant $\mathbf{q}_i$ et $\mathbf{q}_f$, une recherche dans le graphe permet de trouver, si elle existe, une séquence de cellules adjacentes connectant la cellule initiale à la cellule finale. Si une telle séquence est produite elle est alors transformée en un chemin. Cette méthode possède deux variantes majeures : la décomposition cellulaire exacte et la décomposition cellulaire approchée.

Dans une décomposition cellulaire exacte l'ensemble des cellules représente exactement l'espace des configurations libres ou son complémentaire. L'inconvénient de ces méthodes est la difficulté algorithmique de calcul des cellules exactes dans un espace des configurations de grande dimension.

Dans les méthodes approchées les cellules de forme plus simple ne sont qu'une approximation de l'espace des configurations libres.

Les méthodes exactes sont complètes tandis que les méthodes approchées ne sont que *complètes en résolution*, c'est à dire qu'elles garantissent de trouver un chemin solution s'il existe mais uniquement pour une résolution donnée. La complexité algorithmique exponentielle suivant la dimension de $\mathscr{C}$ réduit l'utilisation de ces méthodes à des systèmes pour lesquels la dimension de l'espace des configurations est petite.

Méthodes des champs de potentiel

Autour du milieu des années 80, apparaissent les techniques basées sur les champs de potentiels. Une des applications à la planification globale de chemin consiste à définir une fonction (le champs de potentiel) à travers une grille représentant l'espace des configurations avec pour minimum global la configuration finale [47]. L'étape de requête consiste à chercher sur la grille un chemin utilisant le champs de potentiel comme heuristique (e.g. suivre la descente du gradient). Comme toute méthode discrète, la complexité algorithmique limite cette approche à des espaces des configurations de dimension limitée. Définir une fonction de potentiel pour un espace des configurations de haute dimension reste un problème ouvert.

Durant les quinze dernières années et devant la difficulté des méthodes précédentes, plusieurs approches non déterministes sont apparues, elle ne reposent plus sur une construction préalable d'une représentation exacte ou approchée de l'espace de configurations. Certaines de ces méthodes se basent sur l'exploration de l'espace des configurations libres alors que d'autre visent à capturer sa connexité. Ces nouvelles méthodes de planification dites *méthodes aléatoires*, permettent de faire face à la complexité exponentielle du problème.

Ce type d'approche est caractérisé par une *complétude en probabilité*. Le terme complet en probabilité désigne le fait que ces algorithmes sont capables de trouver un chemin solution si un temps suffisant leur est accordé. Bien que cette propriété soit moins forte que la complétude déterministe vérifiée par les méthodes exactes précédemment citées, les nouvelles approches probabilistes sont moins sensibles à la dimension de l'espace de recherche et s'avèrent très efficaces en pratique. Nous allons consacrer le reste de la section à apporter quelques explications sur ces méthodes. Pour plus de détail sur l'ensemble des contributions, notamment sur les autres méthodes, nous renvoyons les lecteurs vers le livre de référence dans ce domaine [47] ainsi qu'à l'ouvrage plus récent [48].

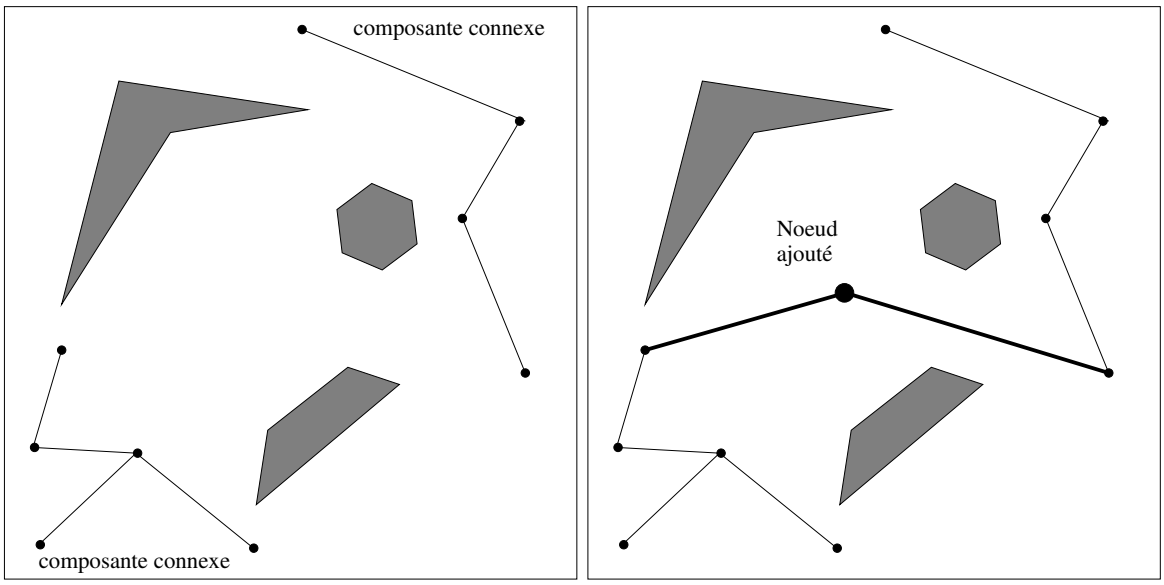

FIG. II.2 – Phase d'apprentissage pour une technique PRM classique. La figure de gauche représente deux composantes connexes construites après un certain nombre de tirages aléatoires de configurations conduisant à la connexion entre certains noeuds. Cependant aucune d'entre elles n'a pu relier ces deux composantes. La figure de droite représente le tirage d'un noeud qui réussit à relier les deux composantes connexes.

Techniques d'échantillonnage aléatoire : méthodes PRM et RRT

L'aube des années 90 a vu apparaître des méthodes aléatoires de planification de chemin. Elles furent initialement introduites par [4] sous le nom de RPP (Random Path Planning) qui propose d'introduire un mouvement aléatoire pour résoudre le problème des minima locaux de la méthode des champs de potentiel. Quelques années plus tard une nouvelle approche aléatoire a vu le jour sous le nom de PRM (Probabilistic RoadMap) ou encore PPP (Probabilistic Path Planner) [36]. Depuis, plusieurs variantes de cette méthodes ont été proposées dans la littérature se basant sur le même concept. Avant de présenter l'essentiel de ces algorithmes nous allons d'abord énoncer quelques principes ayant trait à ces méthodes.

Le principe des méthodes probabilistes est simple. Il consiste à capturer la connexité de l'espace libre par un graphe (ou un arbre) appelé *réseau* construit à partir de configurations choisies aléatoirement dans l'espace des configurations. Une *méthode locale* permet de tester l'existence d'un chemin faisable entre deux noeuds donnés du graphe et ainsi créer des arcs entre les noeuds. Elle définit un chemin d'un robot entre ces deux configurations, i.e. un changement continu dans l'espace des configurations. La méthode locale utilisée prend en considération les différentes contraintes mécaniques et cinématiques du système (e.g. contrainte non holonome). Une méthode locale est dite *non orientée* si l'ensemble des configurations qu'elle parcourt d'un noeud A à un noeud B est le même que celui permettant d'aller du noeud B au noeud A.

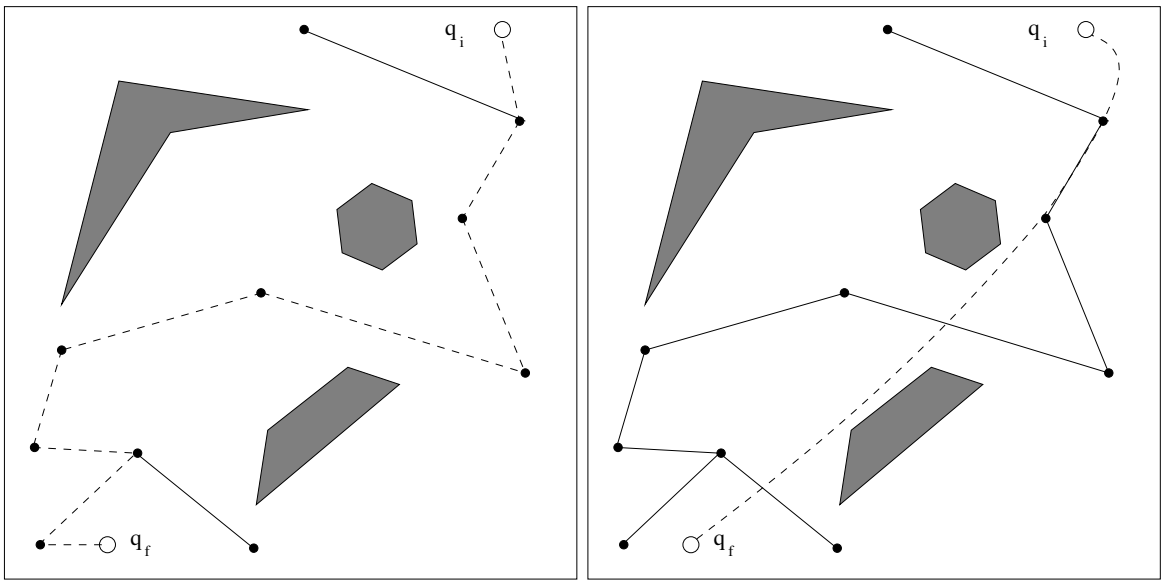

FIG. II.3 – Phase de requête pour un PRM classique. La figure de gauche représente deux configurations $\mathbf{q}_i$ et $\mathbf{q}_f$ qu'on veut relier par un chemin faisable s'il existe. L'algorithme réussit à connecter ces configurations au réseau et trouver ainsi un chemin faisable les reliant. Une procédure d'optimisation permet de lisser le chemin obtenu, figure de droite.

Un *réseau* (roadmap) est un graphe orienté ou non suivant la nature de la méthode locale, dont les noeuds correpondent à des configurations choisies aléatoirement et les arcs correspondent aux méthodes locales.
Un *détecteur de collision* permet de vérifier si les configurations ainsi que les chemins locaux des arcs sont libres (sans collision). Le calcul des collisions ne se fait pas dans l'espace $\mathscr{C}$ puisque celui-ci n'est pas calculé explicitement mais dans l'espace de travail $\mathscr{W}$ qui est de dimension 2 ou 3 (des tests de collisions entre polygones dans le cas bidimensionnel et des tests de collisions entre polyhèdres dans le cas tridimensionnel). Ainsi deux noeuds d'un réseau sont reliés par un arc lorsque le chemin calculé est sans collision et respecte les contraintes cinématiques du robot. Une *composante connexe* est un sous-ensemble de noeuds d'un réseau qui sont connectés directement ou indirectement les uns aux autres. La figure II.2 représente le déroulement de la phase d'apprentissage pour un PRM classique. Le réseau capturant la topologie de l'espace libre peut être construit lors d'une étape d'initialisation pour résoudre efficacement plusieurs requêtes de planification de chemin. Il peut également être enrichi au fur et à mesure que le planificateur résout un problème. La phase de requête (figure II.3 de gauche) consiste à connecter dans un premier temps les configurations initiales et finales aux noeuds du réseau en utilisant la méthode locale. Puis, en cas de succès, à rechercher un chemin dans le réseau entre les noeuds auxquels on s'est connecté. Ce chemin, quand il existe, est une séquence de chemins locaux. Un algorithme de type A^* peut être utilisé pour la recherche du chemin dans le réseau. Si les deux configurations de départ et d'arrivée n'appartiennent pas à la même composante connexe, la phase d'appren-

tissage du réseau peut être effectuée dans le but d'accroître le réseau et de pouvoir relier ces deux configurations dans une même composante connexe. Si cette phase échoue, le planificateur répond qu'il n'y a pas de solution.

En raison du caractère probabiliste du planificateur, le chemin solution est généralement loin d'être optimal. Il peut être long et irrégulier, une phase d'optimisation locale, ou de lissage, s'avère donc nécessaire (figure II.3 de droite).

Une des caractéristiques des méthodes probabilistes est liée à la génération automatique des noeuds. Les méthodes classiques (PRM ou PPP) échantillonnent de façon uniforme l'espace libre. Ce type d'échantillonnage produit une couverture admissible de l'espace libre si les différents obstacles sont répartis de façon uniforme dans l'environnement. Dans certains cas, l'espace des configurations admissibles se compose d'espaces dégagés connectés par des passages étroits. Générer des noeuds dans ce type de passage est très coûteux en terme de temps de calcul, ce qui ce traduit aussi par un réseau de grande taille (un grand nombre de noeuds générés) car la probabilité de tirer une configuration dans un passage étroit, dans un temps petit (raisonnable) est quasiment nulle (voir figure II.4 de gauche). De nombreux auteurs se sont penchés sur cette problématique. Une première approche [36], concentre l'échantillonnage aléatoire autour des composantes isolées du réseau afin de capturer plus efficacement les passages étroits. L'idée principale étant de conserver le nombre d'échecs du planificateur local lors de la connexion entre certains noeuds. Quand ce nombre devient grand pour un noeud donné, ceci signifie que ce dernier se trouve dans une zone *difficile* d'accès. Une exploration de cette zone est alors favorisée (voir figure II.4 de droite). Une autre technique d'échantillonnage gaussien proposée dans [6] tire aléatoirement des paires de configurations séparées par une distance selon une distribution gaussienne et ne conserve un échantillon situé dans l'espace libre que lorsque le second échantillon est en collision. Cette méthode d'échantillonnage présente l'inconvénient de générer des configurations le long des obstacles et pas uniquement dans les passages étroits. La solution envisagée pour contourner ce problème consiste à faire un tirage de trois configurations au lieu de deux et de ne garder une des configurations que si les deux autres sont en collision. La méthode de tirage gaussien est très sensible au réglage des paramètres de la distribution gaussienne. D'autres approches visant à minimiser soit la taille du réseau construit lors de la phase d'apprentissage, soit le nombre d'appels au détecteur de collision lors de la même phase, ont été proposées dans la littérature. L'approche appelée *PRM par Visibilité* [83] définit une condition d'arrêt pour un PRM classique. Le principe de cet algorithme est d'ajouter des échantillons en tant que noeuds du réseau seulement s'ils servent à connecter différentes composantes connexes ou s'ils ne sont pas «visibles» par les autres noeuds. Une configuration est dite visible d'une configuration donnée pour une méthode locale donnée si, et seulement si, le chemin local est sans collision. La taille du réseau produit par cette méthode est nettement plus petite que celle générée par un PRM classique.

D'autres méthodes aléatoires, appelées méthodes par diffusion, combinant les phases d'apprentissage et de recherche pour un problème donné ont vu le jour ces dernières années. Le réseau construit par ces techniques dépend étroitement de la configuration initiale et de la confi-

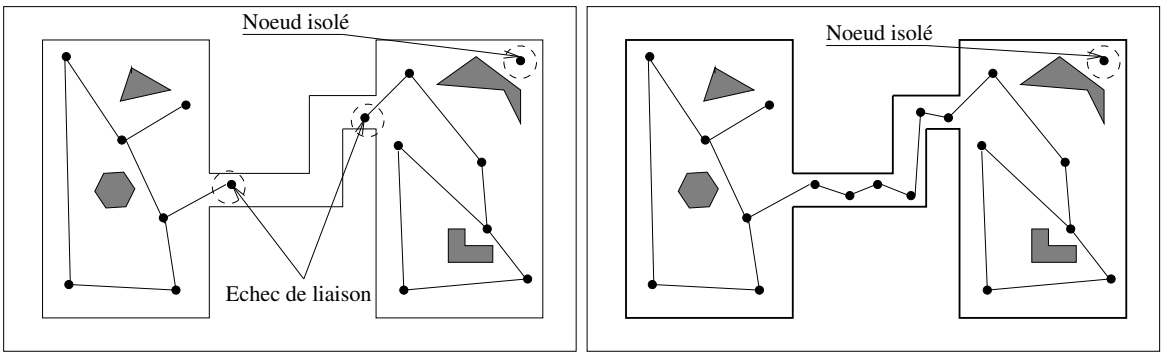

FIG. II.4 – Figure de gauche : Une première construction du réseau permet d'obtenir deux composantes connexes que le planificateur n'a pas pu relier en un premier temps. Cette étape permet de reconnaître les noeuds qui sont les plus difficiles à relier (noeuds isolés ou noeuds ayant un grand nombre d'échecs de tentative de connexion à une composante du réseau). Figure de droite : en concentrant les tirages autour de ces noeuds le planificateur réussit à relier les deux composantes connexes. Le noeud isolé n'a pas pu être connecté à une composante.

guration finale du problème de planification de chemin. Ceci donne des réseaux qui ne peuvent souvent pas être réutilisés pour traiter d'autres requêtes de recherche de chemin. La méthode RRT (Rapidly-exploring Random Trees) est la plus connue d'entre elles. Initialement décrite dans [52], elle consiste à faire évoluer un arbre partant de la configuration initiale afin d'atteindre la configuration finale. A chaque itération, un nouveau noeud est aléatoirement généré et rajouté à l'arbre s'il n'est pas en collision (voir figure II.5 gauche). Une extension de cette méthode a été présentée dans [41] sous le nom de *RRT-bidirectionnel* qui consiste à faire évoluer deux arbres simultanément ayant comme racine respective la configuration initiale et finale. Les deux arbres progressent l'un vers l'autre selon une heuristique simple (voir figure II.5 droite).
La méthode RRT ainsi que ses variantes ont montré des performances meilleures que les méthodes aléatoires classiques lorsque les positions initiales ou finales sont fortement contraintes. Par contre elles sont peu efficaces pour trouver les passages étroits, et moins performantes sur des problèmes type labyrinthe que les algorithmes PRM. Elles sont généralement utilisées pour résoudre des problèmes d'assemblage mécanique où la distance aux obstacles reste très faible autour du chemin solution.

Ces méthodes ont connu un tel intérêt dans la communauté robotique qu'à l'heure actuelle on ne compte plus les travaux et les scientifiques qui se penchent sur le développement et l'amélioration des méthodes aléatoires pour résoudre des problème de planification de chemin. Leurs domaines d'application ne cessent de s'étendre, allant de la manipulation [75] à la planification de chemin pour personnages animés [70], en passant par les applications aux chaînes cinématiques fermées, le problème de docking de molécules [18] et la planification multi-robots [27], etc.

Nous verrons que notre approche utilise en donnée d'entrée une trajectoire sans collision qui

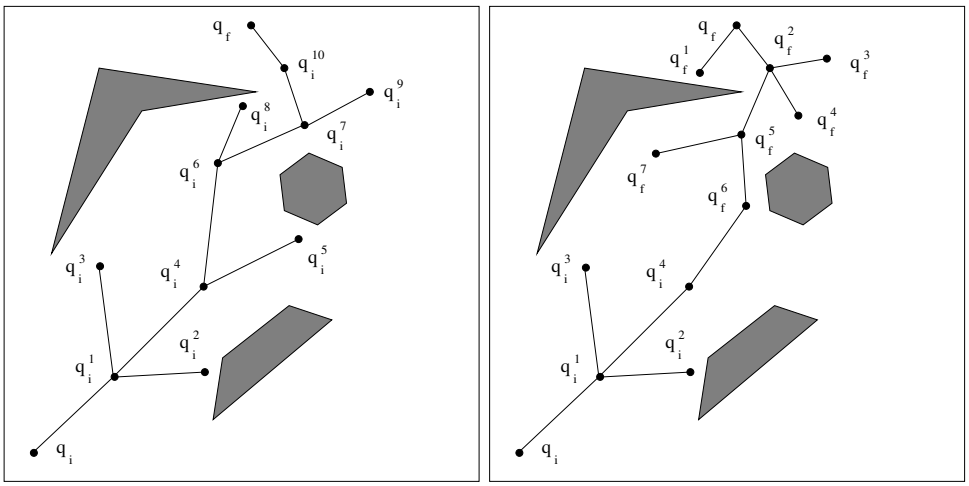

FIG. II.5 – Représentation de l'évolution des méthodes RRT et RRT-bidirectionnel. La figure de gauche représente l'évolution d'un algorithme RRT. Etant données une configuration initiale $\mathbf{q}_i$ et une configuration finale $\mathbf{q}_f$, cette technique consiste à faire progresser une recherche à partir de $\mathbf{q}_i$ en générant aléatoirement des noeuds sans collision pouvant être reliés entre eux $(\mathbf{q}_i^1, \ldots, \mathbf{q}_i^{10})$ jusqu'à pouvoir atteindre la configuration finale $\mathbf{q}_f$. L'algorithme RRT-bidirectionnel lance deux séquences de recherche à partir de $\mathbf{q}_i$ et de $\mathbf{q}_f$ jusqu'à la connexion des deux composantes connexes (par les noeuds $\mathbf{q}_i^4$ et $\mathbf{q}_f^6$ dans ce cas).

est calculée par une méthode probabiliste basée soit sur un RRT soit sur un PRM par visibilité.

La planificateur retourne un chemin géométrique sans collision construit à partir d'un plan de référence de l'environnement. Ce chemin planifié est amené à être exécuté par le robot dans l'environnement réel en tant que chemin de référence. De manière générale, pour faire cette exécution, le robot doit savoir où se trouve la configuration de départ, la configuration d'arrivée et avoir un suivi des configurations exécutées par rapport aux configurations du chemin planifié. À cause des erreurs de modélisation de l'environnement, des imperfections des capteurs et des actionneurs du robot, il est fort probable que l'exécution du chemin planifié échoue. Pour savoir si le robot a bien commencé l'exécution de son chemin à la configuration initiale et/ou qu'il n'a pas dévié de son chemin planifié il faut pouvoir le localiser. La section suivante présente le problème de la localisation pour le robot en environnement réel. Ce problème concerne principalement les structures robotiques de type plate-forme mobile naviguant librement dans l'environnement. Dans ce cas le risque de faire échouer l'exécution est favorisé par la grande étendue du domaine de locomotion possible qui met à rude épreuve les erreurs de modélisation, les imperfections des systèmes de perception et de locomotion. Le problème est particulièrement critique dans les environnements fortement contraints et/ou quand le système robotique a une structure complexe de telle sorte que l'exécution précise du chemin planifié devient une forte contrainte.

II.3 Localisation sur amers

La localisation est un problème fondamental en robotique mobile (voir chapitre 3 dans [49] et [7]). L'objectif de la localisation est d'estimer la position et/ou l'orientation d'un robot dans son environnement, étant donné un plan de l'environnement et des données capteurs. La connaissance de la situation géométrique du robot est essentielle pour le succès de la plupart des tâches de robotique. Les techniques de localisation développées peuvent se distinguer en deux grandes familles suivant le problème qu'elles traitent. Pour les premières, les techniques locales compensent les erreurs odométrique durant la navigation du robot. Elles nécessitent une connaissance au moins approximative de la position initiale du robot et cette localisation peut échouer si le robot perd le suivi de sa position courante. L'autre famille d'approches de localisation concerne les techniques globales. Celles ci estiment la position du robot sans connaissance a priori. Les techniques de localisations globales sont plus robustes que les techniques locales alors que ces dernières offrent un résultat de localisation généralement plus précis. Certaines méthodes tentent de combiner les deux approches pour bénificier des avantages de chacune d'elles.

Il est très important de localiser le robot afin de suivre son état et de pouvoir mener à bien la tâche qui lui est attribuée. Toutefois, la localisation est un problème en robotique principalement à cause des imperfections des systèmes et des modèles de robots et de capteurs. De ce fait, ni les mesures de capteurs, ni l'exécution de configurations de référence ne sont exactes, ce qui peut conduire à un grand écart entre le résultat obtenu et celui désiré. Une notion de processus stochastique vient alors suppléer cette grosse lacune de conception et de modélisation. [69] donne un aperçu général des méthodes de localisation. Dans ce qui suit nous proposons un survol des principaux travaux sur la localisation probabiliste. Ensuite nous consacrerons un paragraphe sur les méthodes de localisation non probabilistes qui mettent l'accent sur la situation géométrique du robot dans l'espace de travail.

II.3.1 Localisation probabiliste

En termes probabiliste, la localisation peut être décrite comme un problème d'estimation bayésienne. Elle permet de déterminer la probabilité $P(x_k|d_{0,\ldots,k})$ que le robot soit à l'état (situation) $x_k \in \mathscr{E}$ au temps k connaissant toutes les données $d_{0,\ldots,k}$ jusqu'à cet instant, avec $\mathscr{E}$ l'ensemble des états (situations) possibles du robot. Pour pouvoir mettre à jour cette probabilité lors du mouvement du robot il faut exprimer les informations de mesure en terme de probabilité. De plus, il est nécessaire de définir le modèle de mouvement qui donne la probabilité que le robot se trouve à l'état x_k sachant qu'à l'instant $k-1$ il était à l'état x_{k-1} et que le robot a effectué une action $u_{k-1} \in U$, ce qui s'exprime par :

$$P(x_k|x_{k-1}, u_{k-1}) \tag{II.1}$$

avec U l'espace des actions pouvant être effectuées par le robot. Aussi il faut définir le modèle de perception en terme probabiliste. Soit Σ l'espace des mesures provenant d'un capteur, et soit σ_k un élément de Σ observé à l'instant k. Le modèle de perception détermine la probabilité

d'observer σ_k depuis un état x_k, il s'exprime comme suit :

$$P(\sigma_k|x_k) \tag{II.2}$$

A la différence de la probabilité de transition (II.1), cette probabilité est difficile à calculer à cause de la grande dimension de l'espace des mesures (e.g. pour une caméra la densité de probabilité doit retourner une probabilité pour chaque image possible et à chaque état possible ce qui représente une grande masse de calcul).

Partant d'une probabilité initiale, à chaque instant $k-1$ et sous l'effet d'une action u_{k-1}, le robot passe de l'état x_{k-1} à l'état x_k suivant la relation de densité (II.1). Une fois dans cet état il peut obtenir des informations du capteur suivant la relation de densité (II.2). Les formules suivantes décrivent l'enchaînement du processus de localisation :

– A l'initialisation, le robot a une probabilité $P(x_0)$ sur l'état dans lequel il se trouve. Dans le cas idéal $P(x_0)$ est une distribution avec un pic pour l'état dans lequel il se trouve réellement. Dans le cas général, $P(x_0)$ est une distribution uniforme.

– La probabilité sur l'état du robot après avoir incorporé l'action u_{k-1} à l'instant $k-1$ et avant d'obtenir une nouvelle observation σ_k est donnée par :

$$P(x_k|\sigma_1, u_1, \ldots, \sigma_{k-1}, u_{k-1}) \tag{II.3}$$

– Une fois que le robot reçoit une mesure σ_k à l'instant k, il l'inclut pour obtenir une probabilité postérieure :

$$P(x_k|\sigma_1, u_1, \ldots, \sigma_{k-1}, u_{k-1}, \sigma_k) \tag{II.4}$$

Sous l'hypothèse que le processus est markovien, qui dans ce cas permet de supposer que l'état présent du robot ne dépend que de l'état précédent et que l'observation actuelle ne dépend que de l'état actuel on obtient les formules suivantes :

$$P(x_k|u_{k-1}) = \int_{\mathscr{E}} P(x_k|x_{k-1}, u_{k-1})\, P(x_{k-1}|\sigma_{k-1})\, d\, x_{k-1} \tag{II.5}$$

$$P(x_k|\sigma_k) = \frac{P(\sigma_k|x_k)\, P(x_k|u_{k-1})}{P(\sigma_k|\sigma_1, u_1, \ldots, \sigma_{k-1}, u_{k-1})} \tag{II.6}$$

L'équation (II.5) permet d'obtenir la probabilité *a priori* d'être en x_k après avoir fini l'action u_{k-1} du temps précédent $k-1$, en intégrant à travers tous les états précédent x_{k-1}, la probabilité que l'action u_{k-1} amène le robot de x_{k-1} à x_k, multiplié par la probabilité *a posteriori* que le robot était en x_{k-1} à l'étape précédente. La régle de Bayes (équation II.6) permet d'obtenir la probabilité *a posteriori* qui est le résultat du produit de la probabilité conditionnelle d'observer σ_k (modèle de perception) par la probabilité *a priori* d'être à l'état x_k normalisé par la probabilité d'observer la mesure σ_k conditionnée par toutes les informations précédentes. Le résultat de la localisation est donné par la probabilité *a priori* $P(x_k|\sigma_k)$. Cette méthode d'estimation probabiliste de la localisation représente le canevas de base des méthodes de localisation probabiliste. Si $P(x)$ est représenté par une fonction linéaire par morceau, nous obtenons une localisation markovienne basée-grille [26]. En représentant $P(x)$ comme un ensemble de particules, on obtient une localisation de Monté Carlo [94]. On obtient un filtrage de Kalman lorsque toutes les densités sont représentées par des gaussiennes. Les paragraphes suivants offrent une vision globale de ces méthodes avec quelques travaux qui les ont traitées.

Localisation markovienne discrétisée

La localisation Markovienne discrétisée traite du problème de l'estimation d'états discrets à partir des données capteurs. Au lieu de maintenir une seule hypothèse sur la position du robot, elle maintient une distribution de probabilité à travers l'espace de toutes les hypothèses. Ces probabilités sont mises à jour à chaque mesure de données capteurs. L'hypothèse de la localisation Markovienne stipule que, connaissant la situation du robot, les mesures futures sont indépendantes des mesures passées et ne dépendent que de l'état actuel. Hypothèse qui s'avère être inexacte si l'environnement contient des obstacles dynamiques (e.g. environnement peuplé de personnes). Mais la localisation Markovienne est robuste vis à vis des changements occasionnels de l'environnement tel que l'ouverture et la fermeture de porte. On distingue parmi les techniques de localisation markoviennes deux grandes familles :

– des méthodes de localisation topologique [15, 81] où l'espace d'état est organisé selon la structure topologique de l'environnement et

– des méthodes de localisation markovienne basées sur une décomposition en grille permettant de discrétiser l'espace d'état et d'utiliser les cellules résultantes comme une base pour l'approximation du modèle probabiliste du système [14]. Les méthodes utilisant cette représentation sont très puissantes mais souffrent de la forte complexité calculatoire et de la difficulté de fixer une taille pour la résolution des cellules.

Fox et al [26] proposent une technique de localisation markovienne dans des environnements dynamiques tels que les environnements peuplés de type musées. Ils proposent une discrétisation de l'espace d'état en grille pour un robot mobile holonome. Pour le problème des obstacles dynamiques, ils ajoutent au modèle de perception une probabilité de mesure des obstacles dynamiques et utilise une couche de filtrage à deux niveaux. Un filtrage entropique qui permet de rejeter les mesures capteurs qui baissent la certitude sur la position du robot et un filtrage de distance dont le principe est le suivant : sachant l'état du système, toute mesure inférieure à la plus petite mesure issue de l'objet le plus proche du plan provient d'un objet non modélisé. Cette approche permet d'étendre l'utilisation des modèles de Markov à des environnements non-statiques.

Localisation par le maximum de vraisemblance

Olson et al [68] décrivent des techniques d'estimation de maximum de vraisemblance pour la localisation de rovers en environnement naturel. Un plan de l'environnement est d'abord généré en utilisant une stéréo vision (plan global). La position globale du rover est calculée en comparant l'image perçue (plan local) par rapport au plan global en utilisant une formulation probabiliste de l'appariement des images. Les mesures utilisées sont formulées par la distance entre les voxels de l'image du plan local et ceux du plan global. La fonction de vraisemblance est alors le produit des distributions de probabilité de ces mesures. La position maximisant la fonction de vraisemblance est prise comme la position du rover. Dans ce travail on suppose que l'orientation est déterminée par un capteur externe et la localisation ne détermine que la position en x et y. Cet espace d'état est alors discrétisé en grille. Cette technique permet de trouver la position du robot dans l'espace discrétisé et ne nécessite pas une estimation initiale de la position du rover. Cependant le résultat dépend beaucoup de la résolution de la discrétisation de l'espace d'état.

Localisation par le filtre de Kalman

Il est l'un des outils les plus connus et les plus souvent utilisés pour l'estimation d'état de processus stochastiques à partir de mesures bruitées. Un filtre de Kalman a les propriétés suivantes :

– Il permet d'utiliser toutes les informations disponibles avec le modèle dynamique du système, sa description aléatoire et toute information sur l'état initial du système.

– Il est récursif, ce qui lui confère la propriété que toute les données n'ont pas à être stockée et recalculées à chaque fois que de nouvelles données apparaissent. Les informations obtenues lors des étapes successives sont toutes incorporées dans le dernier résultat en cours.

– C'est un algorithme de filtrage car seule la connaissance des variables d'entrées et de sorties du système sont utilisées pour l'estimation. Les variables d'intérêt ne peuvent pas être mesurées directement, mais peuvent être connues en utilisant les données disponibles. Il permet d'obtenir une estimée optimale de ces variables à partir de données bruitées.

Pour les applications en robotique, le FK peut être utilisé pour combiner les informations des capteurs pour estimer l'état du robot (sa situation). Le FK prend en compte les différentes incertitudes et erreurs qui perturbent le modèle du robot et les mesures capteurs. En fusionnant les données de mesures de capteurs du robot (encodeurs de roues, gyroscopes, accéléromètres, etc) on obtient une estimée de la position et de l'orientation du robot. Les techniques se basant sur le filtrage de Kalman [77, 32] sont robustes pour le suivi de la position du robot. Il existe principalement deux approches dans la littérature : le filtre de Kalman utilisé pour l'estimation d'état de système linéaire et le filtre de Kalman étendu utilisé pour les systèmes qui ont un modèle dynamique non linéaire. Dans les applications réelles d'ingénierie le filtre de Kalman étendu est l'estimateur d'état le plus utilisé et le plus fiable pour les systèmes non linéaire [89].

Le filtre de Kalman étendu est le filtre de Kalman du modèle linéaire approximé au premier ordre par un développement de Taylor du système non linéaire. Parmi les nombreux travaux on peut citer par exemple : [33, 89] qui proposent de fusionner les données odométriques et les mesures ultrasoniques avec un filtre de Kalman étendu pour localiser la position d'un robot mobile. [66] utilise un filtre de Kalman étendu pour la localisation en fusionnant des données télémétriques et des données visuelles.

L'approximation au premier ordre du système linéaire peut introduire des erreurs dans l'estimation des paramètres qui peuvent amener à des performances sous optimales et parfois à une divergence du filtre. Une approche de filtrage de Kalman proposée initialement par [34] et utilisée dans [2] permet d'obtenir une estimée de la moyenne et de la covariance dont l'ordre de précision correspond à un développement de Taylor du troisième ordre. Les inconvénients des approches de filtrage de Kalman sont d'une part la restriction sur l'hypothèse de densité de probabilité gaussienne pour la modélisation, d'autre part elles ne donnent pas de représentation probabiliste sur les autres état du système (la probabilité que le robot se trouve dans d'autres situations de son espace d'état) et enfin dans le cas où la localisation échoue on ne peut pas (re)-localiser le robot dans le repère global de l'environnement.

[74] propose de compléter le filtrage de Kalman par une couche supérieure d'estimation bayésienne afin de permettre un suivi de position à travers des hypothèses multiples sur l'espace. [30] propose une approche similaire pour combiner des méthodes de localisation markovienne basée-grille avec filtrage de Kalman. L'idée de l'approche est de compenser les lacunes de chaque méthode par les avantages de l'autre. En effet, la localisation markovienne est utilisée pour une recherche globale de la position du robot permettant une grande robustesse aux bruits des mesures capteurs, alors que le filtrage de Kalman est utilisé localement pour un calcul précis de la position du robot.

Localisation par la méthode de Monté Carlo

[19, 94] représentent la densité de probabilité en maintenant un ensemble d'échantillons qui sont aléatoirement générés à partir de cette fonction. La méthode proposée est capable de localiser le robot sans la connaissance de sa configuration initiale. Cette méthode est rapide et nécessite moins d'espace mémoire que les méthodes basées sur une décomposition en grille. Les méthodes du type de Monté Carlo offrent l'avantage qu'offrent les méthodes de localisation markovienne basée-grille avec l'efficacité des techniques de filtrage de Kalman. Comme dans les méthodes basées-grille, on représente une densité de probabilité à travers l'espace d'état, la localisation avec la méthode Monté Carlo est capable de traiter les ambiguïtés de position localisée (plusieurs localisation possible) et donc permet une localisation globale du robot. En concentrant le tirage des échantillons sur la partie pertinente de l'espace d'état, cette méthode offre une exactitude d'estimation comparable au résultat d'un suivi de position avec un filtrage de Kalman. L'inconvénient majeur de cette méthode provient du fait qu'une localisation globale temps-réel nécessite un petit nombre d'échantillons et cette localisation peut échouer si ce nombre est trop faible et ne couvre pas assez de régions dans l'espace d'état. [93, 42] proposent de résoudre ce problème de dégénérescence par une approche d'optimisation permettant d'adapter le nombre d'échantillons et de faire des tirages autour de la région de plus grande vraisemblance.

Dans [31] on trouve une comparaison expérimentale détaillée entre les méthode de localisation markovienne basée-grille, les méthodes de localisation utilisant un filtrage de Kalman et des méthodes de localisation plus récentes qui utilisent le filtrage particulaire tel que la méthode de Monté Carlo.

II.3.2 Localisation non probabiliste

Il est vrai qu'à l'heure actuelle il est difficile d'imaginer de se passer des méthodes probabilistes dans la localisation. Toutefois dans la littérature peu de travaux s'intéressent à l'aspect géométrique du positionnement du robot. On peut citer néanmoins le travail de [5] qui décrit une méthode de localisation géométrique non probabiliste dans le cas de mesures capteurs bruitées. Sur la base d'une triangulation et d'une représentation en nombre complexe, ils construisent un système linéaire sur-contraint qu'ils résolvent par une méthode des moindres carrés.

Construire des systèmes sur-contraints est un bon moyen pour contourner le problème d'incertitude des mesures des capteurs. Dans notre travail, comme nous le verrons dans le chapitre suivant, nous nous basons sur cette idée pour déterminer la situation géométrique lors du processus de localisation.

II.3.3 Sélection d'amers pour la localisation

Plusieurs auteurs ont traité de stratégie de sélection d'amers pour la localisation et la navigation. Dans cette section on s'intéresse uniquement à la sélection d'amers pour la localisation. La sélection d'amers pour la navigation étant une question sous-jacente de la navigation référencée capteur, elle sera traitée dans la prochaine section. Cependant beaucoup de techniques de sélection des amers pour la localisation sont utilisées pour la sélection d'amers pour la navigation.

En général on peut distinguer dans la littérature deux approches de sélection d'amers pour la localisation :

– La première approche regroupe les auteurs qui considèrent le problème où les amers sont sélectionnés à partir d'un ensemble prédéterminé d'amers. Elle consiste à utiliser les données des capteurs (stéréo vision, scanner laser, etc) pour construire un modèle de l'environnement [3, 13, 40]. Dans cette approche un effort considérable est nécessaire pour maintenir la consistance et la précision de la représentation géométrique de l'espace de travail [16].

– La seconde approche initialement proposées par [12] ne prend pas d'ensemble prédéterminé d'amers et base la localisation du robot sur les données capteurs immédiats plutôt que sur des données sauvegardées. Ainsi une large étape de reconstruction est évitée et comme le monde n'est pas statique, il est difficile de maintenir une représentation consistante qui s'avère inutile dans la plupart des tâches de navigation.

En ce qui concerne la première catégorie [84, 85] furent parmi les premiers à développer une méthode de sélection d'amers. Ils ont appliqué des fonctions heuristiques pour sélectionner un triplet d'amers, pris d'un ensemble de triplet, qui vraisemblablement produit une bonne localisation. [28] apprend une fonction pour la sélection d'amers qui minimise l'erreur de localisation. Une technique similaire est proposée par [88], qui fait l'apprentissage d'un réseau de neurones pour apprendre les amers qui optimisent l'incertitude de la localisation. [25] décrit une stratégie pour déterminer le mouvement du robot et la direction de perception dans l'objectif d'améliorer la localisation du robot.

[95] sélectionne un sous ensemble de caractéristiques possibles qui minimisent un coût bayésien de localisation. [65] sélectionne et classe des candidats de triplets d'amers utilisant des heuristiques. Les meilleurs amers sont alors pris pour la navigation. [79] s'intéresse aux amers issus de perception par une caméra, il prend de chaque image les points les plus pertinents comme amers candidats et extrait une sous-fenêtre autour de chacun d'eux. Ensuite, il procède à une analyse de composantes principales sur ces sous fenêtres pour produire une description sous dimensionnée

de l'apparence des amers candidats. Lors de la localisation, une phase d'appariement permet de retrouver les amers candidats et de calculer par rapport à chacun d'eux une estimation de la localisation. Finalement le résultat de la localisation est la moyenne de toutes les estimations. [54] recherche des amers stables en détectant les coins dans l'environnement. [67] utilise un modèle d'erreur de capteur pour estimer la distribution de probabilité de l'environnement que les capteurs s'attendent à percevoir depuis la position courante du robot. La distribution estimée est comparée à un modèle de l'environnement et les amers optimaux sont sélectionnés en minimisant la prédiction sur l'incertitude de localisation. [35] sélectionne les séquences d'amers permettant au robot d'atteindre son objectif final en un minimum de temps. [39] propose de prendre les amers les plus convenables en évaluant et en comparant les distributions de l'estimation de la position du robot issue de l'utilisation des différents amers disponibles. Le calcul de l'estimation de la position est effectué par un filtre de Kalman étendu.

Les travaux de recherche qui ne considèrent pas un ensemble prédéterminé d'amers incluent les travaux de [80]. Ils proposent de choisir des régions dans l'environnement où peuvent s'appliquer des mesures métriques locales suivant un critère de distinction des mesures. Ce critère est évalué comme une fonction de la stratégie de localisation et de l'environnement. Les régions de l'environnement ont une haute distinction des mesures si elles proposent des structures spatiales suffisantes et de bons retours pour la perception des capteurs. [29] font l'apprentissage d'une fonction entre une image issue d'une caméra et la position du robot. Ils utilisent un algorithme d'apprentissage sans connaissance a priori sur le modèle de structure pour la construction de l'environnement. [91] propose, pour un système de vision, d'utiliser un suivi affine pour sélectionner les amers dans un environnement naturel.

Si la localisation permet au robot de connaître sa situation statique dans l'environnement d'exécution et avoir ainsi une information nécessaire sur l'état de l'exécution de sa trajectoire planifiée, le problème qui se pose concerne l'aspect dynamique de l'exécution du mouvement. En effet il est très important de définir des stratégies de navigation permettant d'utiliser les amers de l'environnement de manière optimale par les capteurs extéroceptifs du robot. Ce problème est pris en charge par les méthodes de navigation référencée capteur que nous présentons dans la section suivante.

II.4 Navigation référencée capteur

Au lieu de planifier un chemin et de le suivre en essayant de localiser le robot, la «navigation référencée capteur» introduit les informations des capteurs extéroceptifs, reflétant l'état courant de l'environnement, dans un processus de planification de mouvement du robot. Elle est importante pour des raisons liées principalement à l'environnement d'évolution du robot. En effet le robot peut être supposé n'avoir aucune connaissance a priori sur le monde qui l'entoure, il peut aussi avoir une représentation très sommaire ou bien «complète» de ce monde. Dans tous ces cas, le plan de cet environnement est inexact par rapport à la réalité. De plus, le monde peut parfaitement être sujet à des changements de situation inattendus. La taille du robot et la com-

plexité de sa structure articulaire est aussi un facteur important qui doit être pris en compte pour la navigation. Un robot de taille relativement petite par rapport à l'environnement et de structure articulaire simple (robot qui peut être assimilé à un corps ponctuel après avoir augmenté la taille des obstacles suivant celle du robot) peut naviguer aisément dans son environnement sans un grand risque de collision avec les obstacles qui l'entourent. Un robot qui est amené à traverser des passages étroits ou à se positionner précisément par rapport à un certain nombre d'obstacles est soumis à davantage de contraintes. Le problème est d'autant plus important lorsque le robot présente une structure articulaire ou cinématique complexe.

Beaucoup de méthodes qui ont été proposées dans le domaine de la navigation référencée capteur traitent surtout du problème de l'inexactitude de l'environnement et assimile le robot à un corps ponctuel orienté. Dans ce travail on s'intéresse plus particulièrement aux approches qui se basent sur la construction de chemins qui seront par la suite exécutés par le robot. Ces méthodes peuvent être divisées en trois grandes classes : la première regroupe les techniques qui proposent de construire des réseaux directement en phase d'exploration avant de planifier des chemins, la seconde et la troisième quant à elles se basent d'abord sur un chemin construit à partir d'un plan de référence de l'environnement. Alors que la seconde classe munit l'exécution du chemin avec des techniques d'évitement d'obstacle, la troisième tente avant l'exécution du chemin de fournir des primitives issues du plan de l'environnement afin de les utiliser dans une boucle d'asservissement lors de la phase d'exécution. Notre travail se situe dans le cadre de cette dernière approche.

Parmi les techniques se basant sur la construction de graphe en phase d'exploration, [92] proposent des fonctions de navigation définies sur la base des propriétés du graphe de Voronoï. L'originalité de cette approche réside dans le fait que, grâce à un schéma de commande référencée capteur utilisant les informations issues du capteur (un laser dans ce cas) ils garantissent que le robot se déplace sur le diagramme de Voronoï sans que celui ci soit construit explicitement. Ceci permet de fournir des trajectoires dans l'espace libre tout en évitant les obstacles de façon autonome. Le schéma de navigation ainsi construit consiste en une phase d'exploration, une étape de localisation et une étape de commande référencée capteur. [53] propose une approche de construction de réseau, appelé «convex-HGVG» (convex hierarchical generalized Voronoi graph), pouvant être utilisée pour explorer un espace de travail plan par un corps convexe. Une caractéristique de ce graphe est qu'il est défini en terme de distance dans l'espace de travail ce qui rend facile l'exploitation des informations métriques issues des capteurs. Utilisant un convex-HGVG, le robot peut explorer des configurations inconnues sans explicitement construire l'espace de configuration.

De leur côté [96] ont développé un algorithme PRM référencé capteur. Cette méthode fait une expansion incrémentale de la roadmap au fur et à mesure que le capteur métrique fournit des informations sur les nouvelles régions explorées de l'espace de travail. Le calcul de la roadmap représentant l'espace de travail peut s'avérer très coûteux si l'environnement (ou bien la scène actuellement visible) et/ou la géométrie du robot sont complexes. Cette méthode est basée sur l'utilisation d'un seul capteur métrique embarqué sur le robot.

Se focalisant sur le contexte du mouvement, une stratégie commune est de calculer un chemin et de l'utiliser pour diriger un module réactif à l'exécution [9, 63, 1]. Ces techniques requièrent beaucoup de ressources de calcul, mais utilisent des planificateurs complets. D'autre techniques calculent un chemin qui est déformé lors de l'exécution basé sur l'évolution de l'environnement dans l'espace de travail [11] ou dans l'espace de configuration [72]. Néanmoins, ces méthodes nécessitent une replanification lorsque le chemin n'est plus valide ou lorsqu'il est trop loin de la configuration initialement planifiée à cause des obstacles non modélisés. [90] présente une stratégie permettant de générer des arbres de chemins obtenus en exécutant un algorithme prédictif-réactif quelques étapes avant l'exécution, afin de calculer l'ensemble intermédiaire de commandes de mouvement possibles et prendre celle qui sera la meilleure. Ce système obtient de bons résultats avec un moindre coût de ressource calculatoire mais il n'assure pas une convergence au but (défaut classique des méthodes réactives). Une autre possibilité est de calculer un canal d'espace libre qui contient des ensembles de chemins, laissant le choix s'effectuer lors de l'exécution [17].

Toutes ces stratégies utilisent un planificateur pour obtenir un chemin pour guider le contrôle réactif. Le planificateur est habituellement un outil assez efficace pour obtenir un chemin permettant de guider le contrôle réactif. Cependant ces méthodes réactives sont d'utilité limitée dans les applications réelles lorsque le scénario de la tâche rend difficile la manoeuvre du robot, ceci se produit généralement lorsqu'il y a une grande densité d'obstacles. Récemment [62] ont identifié certaines limitations liées à ces méthodes, comme les situations de blocage local («local trap»), les mouvements irréguliers et les mouvements oscillatoires, ou l'impossibilité d'amener le robot à une région avec une grande densité d'obstacle.

Ces comportements sont très pertinents dans le développement d'applications robustes pour naviguer indépendamment de la complexité de l'environnement. Enfin dans ce type d'architecture, les modèles sont normalement construits pour servir de base dans le planificateur dans l'optique de réaliser des comportements réactifs en temps réel.

La troisième classe de techniques de navigation référencée capteur suppose un environnement parfaitement connu où tout en se basant sur un plan de cet environnement pour construire un chemin sans collision, elle tente d'associer à ce chemin des primitives d'asservissement issues de l'environnement lors d'une phase de planification afin de rendre plus robuste l'exécution de la trajectoire en environnement réel. La sélection des primitives se fait généralement en définissant une fonction de coût. Le contrôle de l'exécution de la trajectoire se fait généralement par la définition d'une fonction de tâche [76] garantissant le suivi du chemin par un asservissement sur les primitives sélectionnées.

Des travaux initiés dans notre groupe de recherche [8] proposent une approche de planification de mouvement pour un robot mobile qui prend en compte les incertitudes des mesures des capteurs et les incertitudes de contrôle du mouvement dès la phase de planification. En supposant un environnement parfaitement connu, une première méthode produit automatiquement un plan en terme de primitives référencées capteurs. Ce plan permet d'atteindre un but fixé avec

une incertitude bornée à partir d'une position initiale incertaine. Pour réduire l'incertitude sur la position du robot ce dernier peut se déplacer dans des zones de localisation absolue (localisation avec un moyen de perception qui n'est pas monté sur le robot). Une seconde technique proposée dans ce travail traite du problème de la planification basée sur le déplacement asservi du robot dans des régions de validité des mesures capteurs à partir d'un modèle incertain de l'environnement. Les plans sont générés sous la forme de listes de fonctions d'exécution prenant en compte les contraintes de non-holonomie du robot.

À la même époque, [24] introduit une planification de déplacement pour un robot mobile basé sur le concept de tâche robotique [76]. Il décompose la génération de cette tâche en trois étapes : définir un chemin comme une séquence discrète de configurations sur une grille de cellules qui décompose l'espace des configurations du robot. Chaque configuration est attachée à une liste de primitives du plan (amers du plan). Il utilise un algorithme A^* pour la recherche de trajectoires dans la grille de l'espace de configuration. La fonction de coût pour chaque noeud (cellule) correspond à une combinaison d'un coût de localisation et d'un coût de changement de primitive. La deuxième étape groupe les configurations utilisant les mêmes listes de primitives, cette liste est alors appelée *plan local*. Une classification binaire permet de grouper les configurations qui ont les mêmes primitives du plan local. La troisième étape consiste à définir un chemin (avec une interpolation utilisant les courbes de Bézier) pour chaque groupe de configuration dans le plan local. Le chemin global planifié apparaît comme une séquence de sous chemin, chacun étant défini dans son propre plan local. Il utilise le suivi des morceaux de chemin comme fonction de tâche de l'asservissement avec la localisation comme retour d'état. La condition de changement de plan local s'effectue lorsque tous les nouveaux segments du plan local suivant sont détectés et si tous les segments actuels ne le sont pas. Au départ le robot ne commence pas si tous les segments du plan local initial ne sont pas détectés. A la fin le véhicule s'arrête à condition qu'il soit dans le voisinage de la configuration d'arrivée. Finalement chaque tâche robotique est définie par un plan local, un chemin référencé dans ce plan, et un algorithme de suivi de chemin.
[20] décrit une classe d'algorithmes pour établir un plan d'exécution pour un chemin donné dans un environnement connu. Le plan d'exécution consiste en une liste d'amers qui sont visibles et qui doivent être détectés et suivis aux différents segments d'un chemin donné. Tous les amers visibles depuis un segment de chemin ne sont pas inclus dans la liste d'amers associés à ce segment. Les ensembles d'amers et de segments de chemin sont définis de manière à minimiser l'effort de détection et de suivi des amers. Ce travail ne considère que des robots ponctuels.

Parce que les échecs de navigation en robotique mobile sont souvent causés par les erreurs de localisation du robot dans l'environnement, [87] explorent l'idée que ces erreurs peuvent être considérablement réduites en planifiant des chemins amenant le robot à travers des positions où des caractéristiques pertinentes de l'environnement peuvent être perçues par les capteurs du robot. Utilisant un modèle probabiliste des capteurs du robot, ils simulent des perceptions en supposant le robot dans une configuration $\mathbf{q}$ donnée et exécutent un algorithme d'appariement sur ces perceptions par rapport au plan modèle. En répétant cette étape ils construisent un modèle de la distribution probabiliste P à travers l'espace de configuration libre du robot. Une discrétisation en cellules de cet espace est construite en la munissant d'une distribution de probabilité sur la

configuration du robot. Une fois cette grille de cellules obtenue elle est utilisée par un planificateur afin de chercher un chemin sans collision qui optimise le coût d'une fonction. Cette fonction peut être choisie pour retourner les cellules les plus fiables en terme de probabilité. Cette méthode pouvant amener à des chemins de longueur assez conséquente, la fonction de coût peut être une combinaison de minimisation d'erreur d'appariement et de longueur de chemin. Les résultats donnent des chemins fiables avec une longueur raisonnable. Les chemins produits peuvent alors être suivis avec une plus grande précision.
Si ces différents travaux se basent essentiellement sur l'utilisation de capteurs extéroceptifs de proximité (ultrasonique, scanner laser) l'utilisation de caméras comme moyen de perception pour le développement de techniques d'asservissement visuel a connu une avancée depuis une quinzaine d'année dans la communauté robotique.

L'asservissement visuel a connu une popularité croissante en tant que méthode de spécification des tâches robotiques référencées capteurs. Il représente la capacité d'extraire à partir d'une image des informations suffisantes et pertinentes pour effectuer une tâche qui est généralement une tâche de positionnement. L'algorithme de contrôle est une boucle d'asservissement qui procure une robustesse acceptable par rapport aux incertitudes issues des capteurs et de l'environnement. Dans son schéma classique, un asservissement visuel se déroule généralement en deux étapes : d'abord on définit dans l'image un ensemble de points caractéristiques qui constituent l'objectif à atteindre. Ensuite une procédure de contrôle assure la convergence vers la configuration correspondant à l'image but à partir d'une configuration initiale différente [21].

Typiquement le problème de l'asservissement visuel [21] est basé sur l'approche de fonction de tâche initialement introduite par [76]. Lorsque le problème de contrôle visuel implique la succession de certaines primitives à suivre, le problème d'enchaînement de tâches visuelles est alors soulevé. Dans le cas d'un enchaînement d'asservissement sur deux primitives [71, 59] proposent une fonction de tâche issue de la combinaison convexe des deux fonctions de tâche associées à chaque primitive. Une telle combinaison permet de lisser le passage d'une primitive à une autre. Nous verrons plus tard dans ce manuscrit (notamment dans les chapitres IV et V) comment, dans notre approche, nous gérons l'enchaînement des primitives d'asservissement par la construction de fonctions de coût qui seront associées à ces primitives.

Les tâches de contrôle basées sur les asservissements référencés image sont définies en termes de caractéristiques extraites de ces images et suivies dans une séquence d'image. Toutefois, avec les méthodes classiques d'asservissement visuel, la tâche d'évitement d'obstacle est rarement prise en charge par la tâche de contrôle de l'asservissement visuel du fait de la difficulté de reconnaissance d'objets perçus en temps réel pour pouvoir exécuter des algorithmes d'évitements d'obstacles. [86] propose une approche multi-capteurs où la tâche est spécifiée en terme d'objectif visuel est effectuée en utilisant des techniques d'asservissement visuel alors que les évitements d'obstacles sont effectués en utilisant un scanner laser. Dernièrement [60] proposent une méthode d'évitement d'obstacle avec un asservissement visuel.

Généralement les méthodes d'asservissement visuel se basent surtout sur la régulation d'une

consigne qui s'exprime dans l'espace image de la caméra. Un des inconvénient des méthodes de contrôle visuel classiques est qu'elles ne tiennent pas compte de certaines contraintes, par exemple l'objet observé doit rester entièrement dans le champ de vision de la caméra et le mouvement du robot est limité par ses butées articulaires. Un autre inconvénient est que les tâches d'asservissement visuel sont définies de manière locale alors que le déplacement qui doit être effectué par la caméra peut être grand et engendrer un risque d'échec de la tâche d'asservissement. Récemment [61] proposent de coupler la planification de chemin dans l'espace image avec le contrôle référencé image, cette méthode de planification de chemin proposée est basée sur la notion du champ de potentiel. L'approche proposée permet d'étendre la robustesse locale et la stabilité des techniques d'asservissement visuels à un champ d'application plus global. Les contraintes de visibilité de la cible et des limites articulaires sont introduites. Le schéma de l'approche se résume en trois étapes : pour planifier directement des chemins dans l'espace image il faut introduire les contraintes afin que les images calculées correspondent à des positions valides de la caméra. L'approche consiste à planifier d'abord le chemin de la caméra et ensuite à en déduire le chemin correspondant dans l'espace image. Le chemin de la caméra est calculé comme une séquence de N positions intermédiaires où chaque morceau s'approche autant que possible d'une translation linéaire. Ces configurations sont calculées par les méthodes du champ de potentiel. Ici la force répulsive est la combinaison de deux forces ; une force qui croit lorsque le robot a des configurations qui tendent vers les limites articulaires et une seconde fonction répulsive qui tend vers l'infini lorsqu'au moins une caractéristique de l'image devient proche des limites de perception. La force d'attraction permet d'amener le robot à la configuration finale. Ainsi dans cette phase les contraintes mécanique et de visibilité sont introduites. Dans l'objectif d'utiliser un contrôle référencé sur image, le chemin géométrique discret de la cible dans l'image est alors déterminé à partir du chemin de la caméra. Afin de réaliser des chemins continus et avoir des courbes dérivable pour améliorer le comportement dynamique du système ils utilisent des méthodes de B-spline pour interpoler une séquence de configurations. Le chemin continu ainsi obtenu est alors suivi de manière efficace en exploitant la robustesse locale et la stabilité des méthodes de contrôle basées sur image.

Les méthodes de navigation référencées capteur présentées dans cette section s'appliquent pour des types particuliers de capteurs et des robots de structure articulaire assez simple. Bien qu'elles définissent des stratégies de mouvement basées sur des primitives de l'environnement elles ne définissent pas de moyen permettant d'automatiser l'enchaînement entre plusieurs primitives pour l'asservissement du mouvement alors que cet enchaînement est très important pour maintenir une continuitée dans l'estimation de la localisation comme nous le verrons dans la suite de ce travail.

Ayant présenté ces différentes techniques de mouvement référencé capteur nous allons au cours de la section suivante faire une présentation de notre approche de «mouvement asservi sur amers» en nous positionnant par rapport à littérature traitant de ce thème.

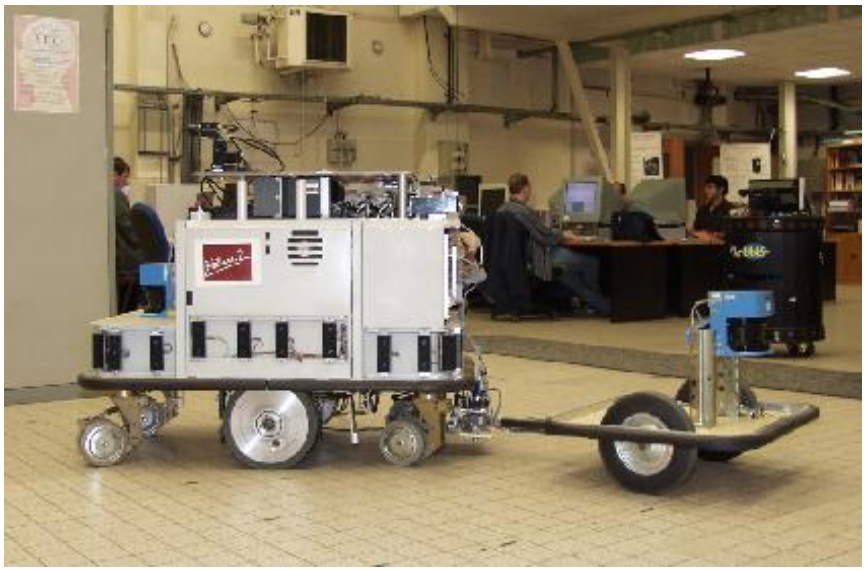

FIG. II.6 – Robot mobile Hilare 2 avec une remorque. Cette structure robotique dispose de deux scanners laser ; le premier est monté à l'avant du véhicule et le second est monté sur la remorque.

II.5 Idée générale de notre approche

En ce début de $21^{éme}$ siècle, on peut raisonnablement considérer que les chercheurs en robotique ont produit un ensemble suffisant de techniques et d'algorithmes permettant de résoudre le problème de la planification de chemins géométriques en environnement structuré.

Résoudre le problème de la planification géométrique ne suffit pas pour effectuer une tâche de navigation de manière efficace essentiellement pour trois raisons. Premièrement l'environnement n'est jamais connu exactement et les obstacles ne sont pas exactement où ils sont supposés être dans le plan de l'environnement. Deuxièmement, parce que la localisation du robot n'est pas exacte. En effet, les erreurs de localisation résultent non seulement de l'imprécision du plan mais aussi d'une connaissance obsolète de la situation initiale du robot et de la non fiabilité des capteurs proprioceptifs utilisés généralement pour avoir un modèle d'état de la situation du robot (e.g. capteurs odométriques). Enfin, les obstacles statiques ou dynamiques non modélisés dans le plan peuvent être en collision avec le mouvement planifié.

Ces trois types de perturbations peuvent produire une collision durant l'exécution du mouvement planifié. Pour un robot mobile de petite taille, ces perturbations peuvent être surmontées d'une façon relativement simple. En localisant le robot sur des amers locaux de l'environnement, les imprécisions du plan sont automatiquement corrigées par le processus de localisation. En plus, augmenter la taille du robot dans la phase de planification du mouvement permet de réduire les risques de collision après localisation en produisant un chemin loin des obstacles. Pour prendre en compte des obstacles dynamiques non modélisés dans le plan, des méthodes réactives d'évitement d'obstacles peuvent être utilisées [73, 37, 10, 62]. Toutefois ces techniques

réactives sont inefficaces dès que le robot présente une structure géométrique complexe ou que l'environnement est fortement contraint. Plus précisément, pour des robots mobiles non holonome navigant à proximité d'obstacles comme par exemple des camions de livraison tractant des remorques et effectuant une manoeuvre d'amarrage le problème est plus compliqué comme ceci sera expliqué dans l'exemple suivant.

Soit le robot mobile non holonome Hilare 2 tractant une remorque représenté en figure II.6. La figure II.7(a) représente un chemin planifié permettant au robot d'effectuer une tâche d'amarrage. Le robot et la remorque sont équipés de deux scanners laser. Durant le mouvement le robot se localise en utilisant des segments de droite sauvegardés dans un plan construit au préalable. Dans cet exemple l'environnement est composé de trois obstacles polygonaux notés O_1, O_2 et O_3. On suppose que la position de l'obstacle O_2 dans le plan est différente de sa position réelle dans l'environnement (figure II.7(b)). Dans ce cas, si le robot se localise sur l'obstacle O_2 uniquement, l'estimation de l'erreur sur la position du robot sera la même que l'erreur dans la position de l'obstacle O_2. Ainsi, si cette erreur est plus grande que la distance entre le chemin planifié et O_1 et si le robot se localise sur O_2 alors qu'il est encore proche de O_1 il entrera en collision avec ce dernier. Ce raisonnement est valable pour des robots sans contrainte cinématique. Cependant le problème est plus critique dans le cas de robots mobiles non-holonomes constitués de plusieurs corps. En effet lorsque la procédure de localisation passe de l'obstacle O_1 et O_3 à l'obstacle O_2, une discontinuité apparaît dans la localisation du robot car celui-ci n'est plus sur la trajectoire planifiée. La loi de contrôle du mouvement corrige alors cet écart. Toutefois les systèmes non holonome requièrent un déplacement longitudinal pour corriger des erreurs de mouvement latéral et en plus ce type de correction nécessite des déviations angulaires du système.

Pour résumer, le raisonnement ci-dessus consistant à naviguer à proximité des obstacles avec un robot mobile non holonome constitué de plusieurs corps dans un plan inexact, conduit à des localisations discontinues poussant le robot à quitter sa trajectoire de référence. Même si la trajectoire de référence n'est pas en collision et est adaptée dynamiquement comme expliqué dans [44], le robot pourra percuter éventuellement un obstacle. De plus comme on a vu précédemment, les techniques d'évitement d'obstacle cherchent davantage à éviter des collisions qu'à définir une configuration à suivre pour le robot ce qui peut être un inconvénient dans le cas de structures robotiques complexes.

Dans un contexte plus général, les techniques de navigation référencée capteur, basées sur la sélection de primitives issues du plan de l'environnement ne définissent pas une stratégie de passage d'un ensemble de primitives à un autre qui permettrait d'éviter un changement brutal dans les amers utilisés lors du processus de localisation. Concrètement dans notre exemple, ceci se traduit par une technique permettant au robot de passer progressivement d'un asservissement par rapport à l'ensemble de primitive $\{O_1,\ O_3\}$ à un asservissement par rapport à l'ensemble $\{O_2\}$.

L'approche de mouvement asservi sur amers que nous proposons repose sur un aspect purement géométrique. Notre formalisme a pour objectif d'être générique, i.e. applicable pour

tout type de robot, tout type de capteur et tout type d'amer. À partir d'un chemin géométrique sans collision nous attribuons aux différents capteurs du robot un ensemble d'amers tout au long du chemin géométrique, construisant ainsi des séquences de couple capteur-amer. Pour chaque paire nous attribuons une fonction de coût continue sur l'ensemble du chemin. Les paires capteur-amer avec leurs fonctions de poids associées constituent les primitives d'asservissement de contrôle de l'exécution du chemin planifié.

Cette fonction de poids qu'on appellera *fonction de pondération* représente, pour chaque configuration du chemin planifié, l'importance que le robot doit accorder à l'amer perçu par le capteur de la paire capteur-amer lors de la phase d'exécution du chemin. Cette importance va se répercuter sur le résultat du processus de localisation. La configuration de localisation calculée, on définit la situation géométrique du robot dans l'environnement réel qui satisfait le critère de localisation en fonction des primitives d'asservissement et de la configuration du chemin planifié. On utilise l'écart de localisation pour exprimer une propriété importante de cette situation géométrique. Ainsi tout au long de l'exécution du chemin on définit des configurations dans l'environnement réel permettant une exécution plus robuste du chemin planifié par rapport aux primitives d'asservissement. En dehors de l'intérêt des fonctions de pondération dans le processus de localisation, elle vont permettre d'assurer un enchaînement lisse des différentes primitives d'asservissement pendant l'exécution du mouvement afin d'éviter des discontinuités indésirables dans la localisation.

II.6 Conclusion

Dans ce chapitre nous avons présenté une vue générale de la littérature très fournie qui traite du problème du mouvement d'un robot, de la planification géométrique de chemin sans collision à la navigation référencée capteur en passant par les techniques de localisation. A la fin nous avons introduit l'idée générale de notre approche de mouvement asservi sur amers. Le chapitre suivant sera consacré à la présentation des bases de notre formalisme.

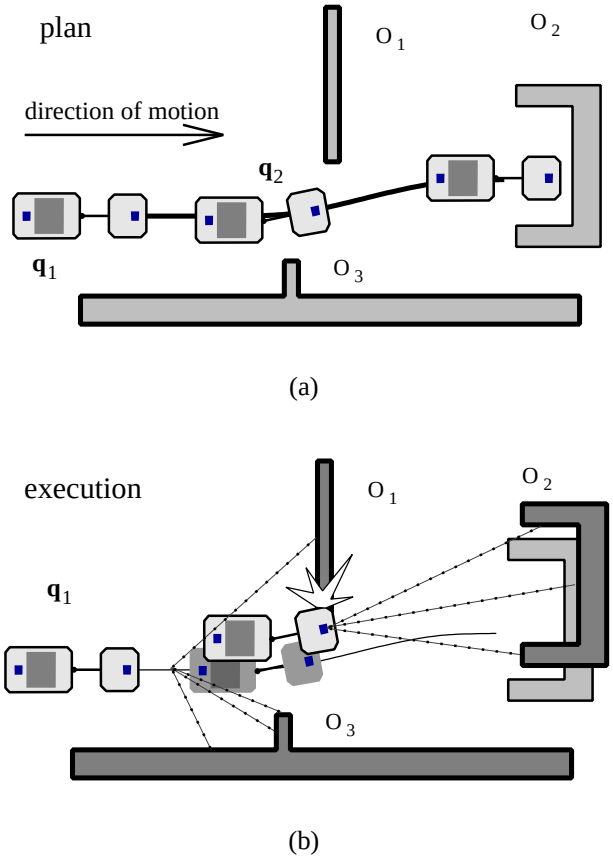

FIG. II.7 – Le robot Mobile Hilare 2 avec une remorque est équipé de 2 scanners laser : le premier est monté sur le robot et le second est monté sur la remorque. Le robot se localise en utilisant un plan polygonale 2D de l'environnement. Dans cet exemple, e robot doit effectuer un amarrage à proximité de l'obstacle O_2. Le chemin géométrique planifié correspond au mouvement en marche arrière (figure de haut). Les lignes en pointillé représentent les caractéristiques utilisées par le robot pour la localisation (figure du bas). Si la position de l'obstacle O_2 est inexacte dans la carte, l'utilisation de cet obstacle pour la localisation lors du passage à proximité de O_1 peut conduire à une collision.

Chapitre III

Mouvement asservi sur des amers

III.1 Introduction

Dans ce chapitre nous abordons les bases formelles de notre approche de mouvement asservi sur amers. Notre objectif est d'établir un formalisme générique applicable pour tout type de robot, tout type d'amer et tout type de capteur. En prenant comme point de départ un chemin géométrique sans collision nous construisons tout au long de ce dernier des primitives d'asservissement issues du plan de référence. L'approche développée dans ce travail ne prend pas en compte les obstacles de l'environnement non modélisés dans ce plan. Au cours de l'exécution de la trajectoire planifiée dans l'environnement réel, pour faire face à l'inexactitude du plan de référence et/ou à d'éventuelles dérives du robot, ce dernier s'appuie sur ces primitives pour se localiser et corriger sa configuration. Dans cette perspective les primitives en question doivent être prises en compte selon leur degré d'importance vis-à-vis de l'exécution du mouvement. La question du choix des primitives sera traitée au cours du chapitre suivant. Dans ce chapitre nous supposons ces primitives déjà connues pour un chemin donné. Nous verrons alors comment le robot va les utiliser pour se localiser et corriger sa configuration vis à vis des erreurs de perceptions pour respecter le chemin planifié. Nous utilisons pour cela le formalisme établi dans [21] qui se base sur le principe de la fonction de tâche en robotique introduit par [76]. Notre approche se distingue par l'utilisation de fonctions de pondération des primitives dont l'étude est l'objet du chapitre suivant. Nous pourrons alors donner la définition du mouvement asservi sur amers, ce qui représente la contribution principale de notre travail.

Afin de pouvoir définir précisément notre approche, il est préférable de commencer par introduire quelques notations et définitions.

III.2 Notations et définitions

III.2.1 Amer

De manière générale, le terme d'amer fait référence à une caractéristique géographique utilisée par les explorateurs afin de trouver leur chemin à travers une région lors du retour d'un voyage. En robotique nous adoptons la définition suivante :

Definition III.2.1 *Un amer L est une caractéristique géométrique dans l'espace de travail détectable par un capteur. Soit $\mathbb{L}$, l'espace des configurations de L et $l \in \mathbb{L}$ la configuration de L.*

Par exemple, si L est un point dans l'espace, $\mathbb{L} = \mathbb{R}^3$ et si L est un segment de droite dans le plan, $\mathbb{L} = \mathbb{R}^4$.

III.2.2 Capteur

Dans sa dénomination courante un capteur désigne un appareil technologique ou un organe biologique capable de détecter un signal ou une propriété physique et/ou une composition chimique. En ce qui nous concerne nous adoptons la définition suivante :

Definition III.2.2 *Un capteur S est un objet mobile qui fait correspondre à un ou plusieurs amers une caractéristique dans son espace image. On note $\mathbb{S} = SE(2)$ ou $SE(3)$, l'espace des configurations du capteur S et $s \in \mathbb{S}$ la configuration de S.*

III.2.3 Le couple capteur-amer

Un capteur peut être amené à percevoir plusieurs types d'amers et réciproquement un amer peut être perçu par plusieurs types de capteurs. Ainsi nous adoptons la définition suivante pour la perception d'un amer par un capteur :

Definition III.2.3 *La perception d'un amer L par un capteur S est caractérisée par une application de classe C^1 :*

$$\begin{array}{rccc} P_{S,L} : & \mathbb{S}\times\mathbb{L} & \longrightarrow & I_{S,L} \\ & (s,l) & \longmapsto & P_{S,L}(s,l) \end{array}$$

qui fait correspondre à chaque configuration du capteur et de l'amer une caractéristique dans l'espace image $I_{S,L}$. Cet espace correspond en pratique à un espace euclidien de dimension p.

Remarque III.2.1 *On considère que P est défini sur un sous ensemble de $\mathbb{S}\times\mathbb{L}$ correspondant aux paires de configurations (capteur, amer) pour lesquelles l'amer est dans le champ de perception du capteur. En dehors de ce sous ensemble de $\mathbb{S}\times\mathbb{L}$ l'image $P_{S,L}(s,l)$ n'est pas définie.*

Par exemple, dans le cas d'un scanner laser percevant une ligne droite (Δ) l'espace image de perception est de dimension $p = 2$ (voir figure (III.1)). En effet si pour un référentiel global

(o,x,y) on définit cette droite par le point $L(l^1,l^2)$ issu de la projection orthogonale du point o sur (Δ), l'équation de la droite est donnée par :

$$(\Delta) \ : \ l^1 x + l^2 y = (l^1)^2 + (l^2)^2$$

Soit $s(s^1,s^2,s^3)$ la configuration du scanner laser où (s^1,s^2) est la position de son centre o_S dans le référentiel global et s^3 son orientation par rapport à l'axe x du même référentiel. L'image de cette droite dans l'espace image du capteur est donnée par les coordonnées du point *im*, issu de la projection du centre du capteur sur cette droite, exprimées dans le référentiel capteur. Soit (x',y') les coordonnées de ce point exprimées dans le référentiel global (figure (III.1) de gauche). Nous avons alors :

$$\begin{cases} im \in (\Delta) \Rightarrow & l^1 x' + l^2 y' = (l^1)^2 + (l^2)^2 \\ (o_S\, im) \perp (\Delta) \Rightarrow & -l^2 (x' - s^1) + l^1 (y' - s^2) = 0 \end{cases}$$

Ce système linéaire de deux équations et de deux inconnues permet d'obtenir les coordonnées (x',y') du point image *im* exprimées dans le repère global (o,x,y) :

$$x' = \frac{l^1 \left[(l^1)^2 + (l^2)^2\right] + (l^2)^2 s^1 - l^1 l^2 s^2}{(l^1)^2 + (l^2)^2}, \quad y' = \frac{l^2 \left[(l^1)^2 + (l^2)^2\right] + (l^1)^2 s^2 - l^1 l^2 s^1}{(l^1)^2 + (l^2)^2}$$

Exprimées dans le référentiel capteur, ces coordonnées retournent l'image de la droite dans le scanner laser (figure (III.1) de droite) :

$$im^1 = cos(s^3)\frac{l^1 \left[(l^1)^2 + (l^2)^2\right] - (l^1)^2 s^1 - l^1 l^2 s^2}{(l^1)^2 + (l^2)^2} + sin(s^3)\frac{l^2 \left[(l^1)^2 + (l^2)^2\right] - (l^2)^2 s^2 - l^1 l^2 s^1}{(l^1)^2 + (l^2)^2}$$

$$im^2 = -sin(s^3)\frac{l^1 \left[(l^1)^2 + (l^2)^2\right] - (l^1)^2 s^1 - l^1 l^2 s^2}{(l^1)^2 + (l^2)^2} + cos(s^3)\frac{l^2 \left[(l^1)^2 + (l^2)^2\right] - (l^2)^2 s^2 - l^1 l^2 s^1}{(l^1)^2 + (l^2)^2}$$

Cette image n'est pas définie lorsque la droite (Δ) passe par le centre du référentiel global. Dans son espace de définition on retrouve bien une expression de $P_{S,L}$ correspondant à la définition (III.2.3) : connaissant la configuration $l(l^1,l^2)$ de l'amer L et la configuration $s(s^1,s^2,s^3)$ du capteur S on peut calculer la perception $im(im^1,im^2)$ de cet amer par le capteur. Ici l'espace image est de dimension $p = 2$.

Ayant introduit ces quelques définitions, nous abordons dans ce qui suit la question de la localisation dans notre approche de mouvement asservi sur amers.

III.3 Localisation

Soit un robot équipé de n capteurs $S_1,\ldots,S_n$ et soit $\mathscr{C}$ son espace des configurations supposé euclidien de dimension r. La configuration s_i de chaque capteur S_i, $i = 1,\ldots,n$ est uniquement définie par la configuration $\mathbf{q}$ du robot. Soit m amers $L_1,\ldots,L_m$ de configurations de référence connues $l_{01},\ldots,l_{0m}$. Chaque amer L_j est visible par un capteur S_{i_j} où $i_1,\ldots,i_m$ est une séquence d'indices avec des valeurs dans $1,\ldots,n$, définissant quel capteur perçoit quel amer.

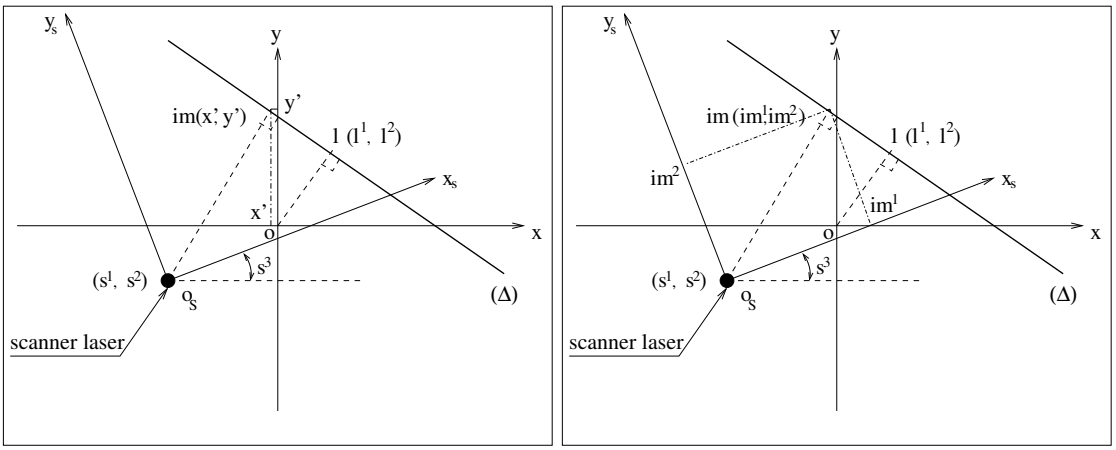

FIG. III.1 – Image d'une droite par un scanner laser. A gauche les coordonnées du point image exprimées dans le référentiel global. A droite les coordonnées du point image exprimées dans le référentiel capteur.

Ainsi chaque paire (S_{i_j}, L_j) de capteur et d'amer donne lieu à une *équation de localisation* :

$$P_{S_{i_j},L_j}(s_{i_j}(\mathbf{q}), l_{0\,j}) \quad = \quad im_j \tag{III.1}$$

où $im_j \in I_{S_{i_j},L_j}$ est l'image de L_j dans S_{i_j}.

L'équation III.1 représente une *liason virtuelle* [21] du fait qu'elle contraint un sous espace de l'espace des configurations du robot par rapport à la perception du capteur.

Ce système contient M équations et r inconnues. La configuration $l_{0\,j}$ est supposée connue et im_j est mesurée. L'inconnue dans cette équation est la configuration $\mathbf{q}$ du robot.

Si le nombre d'amers n'est pas suffisant, le système d'équations (III.1) peut admettre une infinité de solutions. La localisation est dite *sous déterminée*. Si le nombre d'amers est trop grand, le système (III.1) n'a pas de solution. La localisation est dite *sur-contrainte*.
Certaines des notations mentionnées ci dessus nécessitent d'être justifiée. Les deux remarques suivantes sont destinées à cet effet.

Remarque III.3.1 *Afin d'alléger les notations, on suppose que chaque amer ne peut être perçu que par un seul capteur à la fois. Dans le cas où un amer serait visible par plus d'un capteur à la fois on admet alors, pour maintenir la validité des équations, que cet amer est présent dans la liste $L_1, \ldots, L_m$ autant de fois que le nombre de capteurs le percevant.*

Remarque III.3.2 *En supposant qu'à la configuration $\mathbf{q}$ les m amers sont visibles, si chaque fonction de perception $P_{S_{i_j},L_j}$ retourne un vecteur image de dimension p_{i_j} alors pour les m amers nous avons un vecteur image de dimension $\sum_{j=1}^{m} p_{i_j}$. Pour alléger les notations nous notons cette dimension par M. Dans l'exemple précédent de la droite perçue par un scanner laser, si on*

essaye de localiser ce capteur avec trois droites nous aurons un système à $M = 6$ *équations et* $r = 3$ *inconnues.*

Le système d'équations de localisation obtenu est non linéaire. Pour le résoudre nous proposons dans une première étape de linéariser ses équations autour d'une configuration de référence. C'est ce que nous verrons dans la section suivante.

III.3.1 La localisation autour d'une configuration de référence

Si le robot est amené à suivre une trajectoire de référence, ou si la localisation est effectuée avec une fréquence assez élevée , on peut supposer que le résultat du système (III.1) est au voisinage d'une configuration de référence qui peut être le résultat de la dernière localisation ou bien une configuration appartenant à la trajectoire planifiée [57]. Soit $\mathbf{q}_0$ cette configuration de référence. La linéarisation de chaque équation de localisation (III.1) autour de $\mathbf{q}_0$ donne lieu à l'équation linéaire suivante :

$$im_{0\,j} + \frac{\partial P_{S_{i_j},L_j}}{\partial s}(s_{i_j}(\mathbf{q}_0), l_{0\,j})\frac{\partial s_{i_j}}{\partial \mathbf{q}}(\mathbf{q}_0)(\mathbf{q}-\mathbf{q}_0) + o(\mathbf{q}-\mathbf{q}_0) = im_j \tag{III.2}$$

En négligeant le résidu de la linéarisation nous obtenons l'équation ci-dessous :

$$\frac{\partial P_{S_{i_j},L_j}}{\partial s}(s_{i_j}(\mathbf{q}_0), l_{0\,j})\frac{\partial s_{i_j}}{\partial \mathbf{q}}(\mathbf{q}_0)(\mathbf{q}-\mathbf{q}_0) = im_j - im_{0\,j} \tag{III.3}$$

Cette équation exprime l'approximation d'ordre 1 de la relation entre la variation de configuration autour de $\mathbf{q}_0$ et la variation de l'image de chaque amer dans le capteur correspondant. $\frac{\partial P_{S_{i_j},L_j}}{\partial s}(s_{i_j}(\mathbf{q}_0), l_{0\,j})$ est une matrice Jacobienne qui représente la variation de l'image en fonction de la variation de la configuration du capteur. Cette matrice Jacobienne dépend des propriétés géométriques du capteur et de l'amer. $\frac{\partial s_{i_j}}{\partial \mathbf{q}}$ représente la variation de la configuration du capteur en fonction de la variation de la configuration du robot. Cette matrice Jacobienne dépend de la structure cinématique du robot. im_j et $im_{0\,j}$ représentent respectivement l'image de l'amer L_j dans le capteur S_{i_j} et l'image prévue, *i.e.* l'image qui serait perçue depuis la configuration $\mathbf{q}_0$ si le plan de référence était exact.
Pour l'ensemble des perceptions nous avons le système :

$$\begin{pmatrix} \frac{\partial P_{S_{i_1},L_1}}{\partial s}(s_{i_1}(\mathbf{q}_0), l_{0\,1})\frac{\partial s_{i_1}}{\partial \mathbf{q}}(\mathbf{q}_0) \\ \vdots \\ \frac{\partial P_{S_{i_m},L_m}}{\partial s}(s_{i_m}(\mathbf{q}_0), l_{0\,m})\frac{\partial s_{i_m}}{\partial \mathbf{q}}(\mathbf{q}_0) \end{pmatrix} (\mathbf{q}-\mathbf{q}_0) = \begin{pmatrix} im_1 \\ \vdots \\ im_m \end{pmatrix} - \begin{pmatrix} im_{0\,1} \\ \vdots \\ im_{0\,m} \end{pmatrix} \tag{III.4}$$

La matrice ci-dessus est appelée *matrice d'interaction* [21] du fait qu'elle exprime comme une transformation linéaire les variations de perception des primitives d'asservissement en fonction des variables de configuration du robot.

Avant de poursuivre le développement sur la résolution du système (III.4), on se propose d'illustrer le problème de localisation par l'exemple didactique suivant.

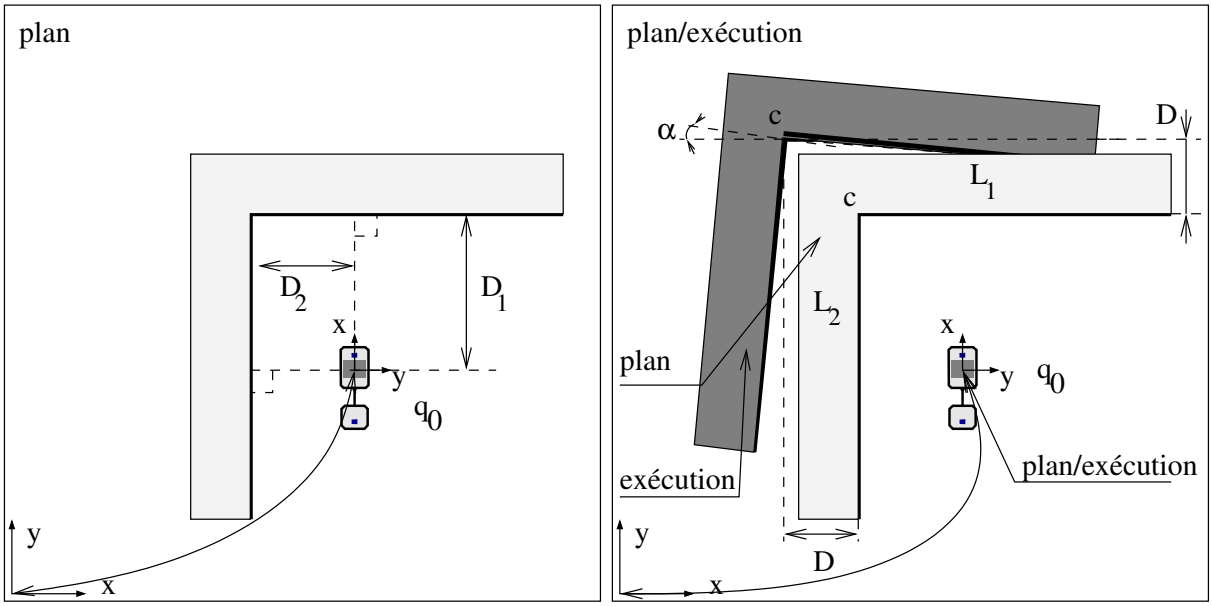

FIG. III.2 – Le robot Hilare 2 et sa remorque, figure de gauche, se trouvent dans un plan de référence contenant deux murs L_1 et L_2 perpendiculaire l'un à l'autre constituant un corps rigide en forme de L. Le robot se trouve en une configuration de référence $\mathbf{q}_0$. Soit D_1 et D_2 respectivement les distances de L_1 et de L_2 au centre du robot. L'axe x du robot est perpendiculaire à L_1 et est parallèle à L_2. L'axe longitudinal de la remorque est parallèle à L_2. Dans l'environnement réel, figure de droite, le corps rigide constitué des deux murs est décalé vers le haut et vers la gauche d'une même distance D. Il subit ensuite une rotation, suivant l'axe parallèle à z et passant par le coin c, d'un angle α dans le sens des aiguilles d'une montre. Le plan de référence devient inexact. Dans cet exemple, par rapport au référentiel global de l'environnement d'exécution le robot se trouve à la même configuration que celle dans le plan de référence.

III.3.2 Exemple didactique de localisation

Soit le robot Hilare 2 et sa remorque dans un plan de référence donné (voir figure (III.2) de gauche). Ce plan contient deux murs L_1 et L_2 perpendiculaires l'un à l'autre constituant un corps rigide en forme de L. Le robot se trouve en une configuration de référence $\mathbf{q}_0$. On suppose que ces murs sont perceptibles par le scanner avant du véhicule. Il est considéré dans ce cas que la fonction de perception du scanner laser avant retourne la distance ainsi que l'orientation des murs par rapport au robot tandis que la fonction de perception du scanner laser arrière retourne uniquement l'orientation des murs par rapport à la remorque. Soit D_1 et D_2 respectivement les distances de L_1 et de L_2 au centre du robot. L'axe x du robot est perpendiculaire à L_1 et est parallèle à L_2. La remorque est parallèle au mur L_2.
L'environnement réel, figure (III.2) de droite, est décalé vers le haut et vers la gauche d'une même distance D par rapport au plan de référence. On suppose que cette transformation en translation est suivie d'une transformation en rotation dans le sens des aiguilles d'une montre d'un angle α.

Cette rotation est effectuée suivant l'axe perpendiculaire au plan et passant par le coin intérieur reliant les deux murs. Ainsi le plan de référence devient inexact. Pour simplifier l'exemple on considère dans ce cas que le robot se trouve dans l'environnement d'exécution à la même configuration de référence $\mathbf{q}_0$.

Lorsque le robot procède à une opération de localisation, dans le cas de petite variation en rotation (α petit), il s'aperçoit que d'une part le mur L_1 est plus loin que prévu d'une distance D suivant son axe x et que L_2 est plus loin d'une même distance D par rapport à son axe y. D'autre part il s'aperçoit qu'il existe un écart angulaire d'ordre α entre sa configuration de référence et sa perception. Ainsi, en supposant que les murs n'ont pas bougé, le robot va croire que dans son plan de référence il s'est décalé d'une distance D vers la droite et décalé d'une distance D vers le bas par rapport à sa configuration prévue $\mathbf{q}_0$ et qu'ensuite il a subit une rotation d'un angle α dans le sens opposé de l'aiguille d'une montre (voir figure (III.3)).

Cet exemple représente un cas simple où le robot se localise de manière unique dans son plan étant données les mesures de perception. Dans un cadre plus général où le système d'équations (III.4) est sur-contraint il serait plus intéressant de paramétrer ces équations par des pondérations qui permettrons d'ajuster le résultat de la localisation en donnant plus d'importance à certains écarts de perception qu'à d'autres.

III.4 Pondération des couples capteurs-amers

Si le système (III.4) est sur-contraint, une approche commune consiste à modéliser les configurations des capteurs et des amers comme des variables aléatoires gaussiennes indépendantes et à déterminer la configuration du robot qui maximise la fonction de vraisemblance des mesures. Dans ce cas chaque amer est pondéré en fonction de la variance sur sa configuration.

Dans notre approche, le poids associé à chaque amer est une partie de la spécification de la tâche de contrôle du mouvement. Ainsi, à partir de l'équation (III.4) nous construisons un système linéaire d'équations où en affectant à chaque paire capteur-amer (S_{i_j}, L_j) un réel positif w_j on pondère chaque équation par cette valeur. Nous expliquerons plus tard dans ce manuscrit le rôle de ces coefficients. Nous obtenons ainsi le système linéaire suivant :

$$W(\mathbf{q}-\mathbf{q}_0) = IM - IM_0 \tag{III.5}$$

où W est la matrice :

$$\begin{pmatrix} w_1 \frac{\partial P_{S_{i_1},L_1}}{\partial s}(s_{i_1}(\mathbf{q}_0), l_{01}) \frac{\partial s_{i_1}}{\partial \mathbf{q}}(\mathbf{q}_0) \\ \vdots \\ w_m \frac{\partial P_{S_{i_m},L_m}}{\partial s}(s_{i_m}(\mathbf{q}_0), l_{0m}) \frac{\partial s_{i_m}}{\partial \mathbf{q}}(\mathbf{q}_0) \end{pmatrix}$$

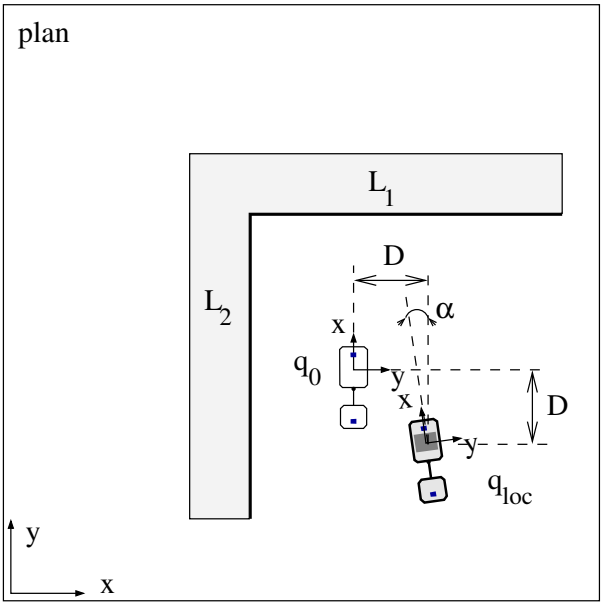

FIG. III.3 – Lors de la localisation, le robot s'aperçoit que les deux murs L_1 et L_2 sont plus éloignés d'une même distance D. Il s'aperçoit aussi qu'il existe un écart angulaire entre l'orientation des murs réels et son orientation de référence. Le résultat de la localisation retourne, dans le plan de référence et pour des petites variations angulaires, une configuration décalée d'une distance D vers la droite, décalée d'une distance D vers le bas et avec une rotation α dans le sens contraire des aiguilles d'une montre par rapport à la configuration de référence $\mathbf{q}_0$.

et

$$IM = \begin{pmatrix} w_1 im_1 \\ \vdots \\ w_m im_m \end{pmatrix} \quad IM_0 = \begin{pmatrix} w_1 im_{0\,1} \\ \vdots \\ w_m im_{0\,m} \end{pmatrix}$$

La solution de ce système de M lignes et de r colonnes est donnée au sens des moindres carrés par l'expression :

$$\mathbf{q}_{loc} = \mathbf{q}_0 + W^+(IM - IM_0) \tag{III.6}$$

Où W^+ est la matrice pseudo inverse de W. Le résultat retourné est soumis à certaines conditions :

1. Si $M > r$ et $rang(W) = r$, le système (III.5) n'admet pas de solution. Le résultat retourné par (III.6) minimise la somme pondérée des carrés de la norme euclidienne de la différence entre l'approximation d'ordre 1 de l'image perçue en $\mathbf{q}$ et l'image réelle im_j :

$$J = \sum_{j=1}^{m} w_j^2 \left\| \frac{\partial P_{S_{i_j},L_j}}{\partial s}(s_{i_j}(\mathbf{q}_0), l_{0\,j}) \frac{\partial s_{i_j}}{\partial \mathbf{q}}(\mathbf{q}_0)(\mathbf{q} - \mathbf{q}_0) - (im_j - im_{0\,j}) \right\|^2$$

2. Si $M < r$ et $rang(W) = M$, le système (III.5) admet une infinité de solutions exactes. Le résultat retourné par (III.6) minimise la norme euclidienne de la différence entre la configuration solution $\mathbf{q}$ et la configuration de référence $\mathbf{q}_0$:

$$\min_{\mathbf{q}} \|\mathbf{q} - \mathbf{q}_0\| \text{ sur } \{\mathbf{q} / \; W(\mathbf{q} - \mathbf{q}_0) = IM - IM_0\}$$

3. Si $rang(W) < \min(M, r)$, le système (III.5) n'a pas de solution. Le résultat retourné par (III.6) minimise le critère J ainsi que la norme euclidienne de la différence entre la configuration solution $\mathbf{q}$ et la configuration de référence $\mathbf{q}_0$.
4. Si $rang(W) = M = r$, le système (III.5) admet une solution exacte unique, dans ce cas la matrice pseudo inverse W^+ correspond à la matrice inverse W^{-1}.

Le développement conduit dans cette section peut être résumé comme suit. Localiser un robot sur des amers consiste à résoudre un système d'équations qui relie la configuration du robot avec les images des amers perçus dans les capteurs du robot. Si le système est sur-contraint, la localisation consiste à trouver une configuration qui minimise une somme pondérée de résidus. Si les amers sont exactement à la même configuration dans le plan de référence que dans la réalité, le choix des poids n'a pas d'influence sur le résultat. Cependant si le plan des amers n'est pas exact, le choix des poids va avoir une grande influence. C'est pourquoi dans notre approche nous suggérons d'utiliser ces poids comme un moyen pour planifier un mouvement asservi sur amers.

III.5 Mouvement asservi sur des amers

Sur la base des développements de la section précédente, nous définissons un mouvement asservi sur amers pour un robot comme suit :

Definition III.5.1 *Pour un robot avec n capteurs $S_1,\ldots,S_n$ et un environnement avec m amers $L_1,\ldots,L_m$, respectivement associés aux capteurs $S_{i_1},\ldots,S_{i_m}$, un mouvement asservi sur amers est défini par :*

1. *Une trajectoire de référence sans collision :*

$$\begin{array}{rccc} \gamma: & [0,U] & \rightarrow & \mathscr{C} \\ & u & \rightarrow & \gamma(u) \end{array}$$

2. *m fonctions réelles continues et positives $w_1,\ldots,w_m$:*

$$\begin{array}{rccc} w_j: & [0,U] & \rightarrow & \mathbb{R}^+ \\ & u & \rightarrow & w_j(u) \end{array}$$

Tel que $w_j(u)=0$ pour tout u où L_j n'est pas visible depuis le capteur S_{i_j} quand le robot est à la configuration $\gamma(u)$.

Finalement, suivre un mouvement asservi sur amers pour un robot consiste à estimer la configuration courante du robot avec une tâche de contrôle en boucle fermée en résolvant le système (III.5) autour d'une configuration de référence $\mathbf{q}_0$ appartenant à la trajectoire planifiée $\gamma(u)$ et avec les valeurs des poids $w_j(u)$ pour l'abscisse u le long de cette trajectoire.

Une fois cette configuration courante estimée, un processus d'asservissement doit amener le système dans une situation d'équilibre qui permettra alors de réduire l'écart de perception entre le plan de référence et l'environnement d'exécution selon la spécification des valeurs de pondération. Le paragraphe suivant est consacré à l'étude de cette situation d'équilibre.

III.6 Asservissement sur la localisation

La configuration de localisation calculée d'après l'équation (III.6) dépend de la configuration de référence $\mathbf{q}_0$, du vecteur de perception de référence IM_0 et du vecteur de perception mesuré IM. Ce dernier est fonction des configurations réelles des amers $l_1,\ldots,l_m$ et de la configuration du robot en environnement réel d'où les mesures ont été acquises, soit $\mathbf{q}$ cette configuration. On peut écrire de nouveau l'expression de la localisation sous la forme :

$$\mathbf{q}_{loc}(\mathbf{q},\mathbf{l}) = \mathbf{q}_0 + W^+(IM(\mathbf{q},\mathbf{l}) - IM_0) \tag{III.7}$$

où $\mathbf{l}=(l_1,\ldots,l_m)^T$ est le vecteur des configurations réelles des amers supposé inconnus.

Asservir le robot sur la configuration de référence $\mathbf{q}_0$ par rapport aux amers $L_1,\ldots,L_m$ revient à modifier la perception mesurée $IM(\mathbf{q},\mathbf{l})$ (en modifiant la configuration du robot en environnement d'exécution) de manière à obtenir une configuration de localisation $\mathbf{q}_{loc}$ égale à la configuration de référence $\mathbf{q}_0$. Le schéma classique d'un tel asservissement se traduit par la formulation suivante :

$$\dot{\mathbf{q}} = -(\mathbf{q}_{loc}(\mathbf{q},\mathbf{l}) - \mathbf{q}_0) \tag{III.8}$$

L'équation (III.8) représente une commande proportionnelle. Nous allons maintenant étudier les propriétés de stabilité de ce système au voisinage de $\mathbf{q}_0$ et montrer qu'il admet une unique configuration *localement asymptotiquement stable* en ce voisinage.

Definition III.6.1 *La configuration corrigée $\mathbf{q}_c$ est l'unique configuration localement asymptotiquement stable au voisinage de $\mathbf{q}_0$ du système (III.8).*

Les candidats à l'équilibre stable du système (III.8) permettent de vérifier $\dot{\mathbf{q}} = 0$. Le remplacement de l'expression de la localisation de (III.7) dans (III.8) permet de déduire la propriété suivante pour la configuration corrigée.

Propriété III.6.1 *La configuration corrigée $\mathbf{q}_c$ vérifie la condition nécessaire suivante :*

$$W^+(IM(\mathbf{q}_c, \mathbf{l}) - IM_0) = 0 \tag{III.9}$$

L'expression (III.9) est un système non linéaire avec autant d'équations que d'inconnues.

Remarque III.6.1 *L'équation (III.8) est une commande proportionnelle classique qui suppose que les variables de configuration sont complètement actionnées sans aucune contrainte cinématique. Elle a pour but de montrer que dans ce cas simple, notre formalisme a un sens, au moins localement.*

Dans ce qui suit nous allons étudier les conditions d'existence et d'unicité des candidats à l'équilibre stable du système (III.8). Ensuite nous passerons à l'étude de la stabilité de ce système dans le cas où la configuration d'équilibre stable existe et est unique. Enfin nous établirons deux propriétés supplémentaires de la configuration corrigé $\mathbf{q}_c$.

III.6.1 Existence de la configuration corrigée

Pour prouver l'existence de la solution de l'équation (III.9), on pose d'abord la fonction Φ tel que :

$$\begin{array}{rccc} \Phi : & \mathbf{L} \times \mathscr{C} & \longrightarrow & \mathscr{C} \\ & (\mathbf{l}, \mathbf{q}) & \longmapsto & \Phi(\mathbf{l}, \mathbf{q}) = W^+(IM(\mathbf{q}, \mathbf{l}) - IM_0) \end{array}$$

avec $\mathbf{L} = \mathbb{L}_1 \times \ldots \times \mathbb{L}_m$ l'espace des configurations des m amers $(L_1, \ldots, L_m)$.
La preuve d'existence de candidats à l'équilibre stable du système (III.8) repose sur le *théorème des fonctions implicites*. En effet la propriété formulée en (III.9) exprime une relation implicite entre $\mathbf{q}$ et $\mathbf{l}$ sous certaines conditions suivant le *théorème des fonctions implicites* dont nous rappelons l'énoncé dans ce qui suit.

Théorème III.6.1 *Soit Φ une application de classe C^1 définie sur le produit de deux ouverts U et V de deux espaces de Banach E et F et prenant ses valeurs dans un troisième espace de Banach G.*

Supposant qu'en un point (a,b) de $U \times V$, l'application linéaire tangente de l'application partielle $\Phi_a : y \mapsto \Phi_a(y) = \Phi(a,y)$ soit un isomorphisme de F sur G. Alors il existe un voisinage

ouvert U' de a ($U' \subset U$), un voisinage ouvert W de $\Phi(a,b)$, un voisinage ouvert Ω de (a,b) ($\Omega \subset U \times V$), et une application $\psi : U' \times W \to V$ de classe C^1 tels que les deux conditions suivantes soient équivalentes :

1. $(x,y) \in \Omega$, $z \in W$ *et* $\Phi(x,y) = z$;
2. $(x,z) \in U' \times W$ *et* $y = \psi(x,z)$.
   En particulier, pour tout $(x,z) \in U' \times W$, $z = \Phi(x, \psi(x,z))$.

Puisque la fonction de perception $P_{S,L}$ est de classe C^1 d'après la définition (III.2.3) il s'en suit que Φ est aussi de classe C^1. Dans notre cas, le point (a,b) correspond au point $(\mathbf{l}_0, \mathbf{q}_0)$ qui vérifie $\Phi(\mathbf{l}_0, \mathbf{q}_0) = z$ avec $z = 0$. Au point $(\mathbf{l}_0, \mathbf{q}_0)$, L'application linéaire tangente de l'application partielle $\Phi_{\mathbf{l}}$ est un isomorphisme de $\mathbf{L}$ vers $\mathscr{C}$ si la derivée partielle de l'application Φ par rapprort à $\mathbf{q}$ est inversible au point $(\mathbf{l}_0, \mathbf{q}_0)$. L'expression de cette dérivée en ce point est donnée par l'expression :

$$\frac{\partial \Phi}{\partial \mathbf{q}}(\mathbf{l}_0, \mathbf{q}_0) = W^+ \frac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_0, \mathbf{l}_0) \tag{III.10}$$

Notons que $W = \frac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_0, \mathbf{l}_0)$, on obtient alors :

$$\frac{\partial \Phi}{\partial \mathbf{q}}(\mathbf{l}_0, \mathbf{q}_0) = W^+ W \tag{III.11}$$

Cette dérivée partielle est une matrice carrée symétrique d'ordre r. Sa condition d'inversibilité dépend du rang de W par rapport à r :

1. Si $rang(W) = r$, alors $\frac{\partial \Phi}{\partial \mathbf{q}}(\mathbf{l}_0, \mathbf{q}_0)$ est égale à la matrice unité d'ordre r et donc inversible. Dans ce cas, il existe un voisinage ouvert $\mathbf{L}'$ de $\mathbf{l}_0$ ($\mathbf{L}' \subset \mathbf{L}$), un voisinage ouvert $\mathscr{C}'$ de $\Phi(\mathbf{l}_0, \mathbf{q}_0)$, un voisinage ouvert Ω de $(\mathbf{l}_0, \mathbf{q}_0)$ ($\Omega \subset \mathbf{L} \times \mathscr{C}$) et une application $\psi : \mathbf{L}' \times \mathscr{C}' \to \mathscr{C}$ de classe C^1 tel que $\mathbf{q}$ s'exprime en fonction de l'unique variable $\mathbf{l}$, i.e. $\mathbf{q} = \psi(\mathbf{l}, 0)$ pour tout $\mathbf{l} \in \mathbf{L}'$. En particulier, si la configuration réelle $\mathbf{l}$ des amers appartient au voisinage $\mathbf{L}'$ de $\mathbf{l}_0$ alors il existe une unique configuration $\mathbf{q}$ du robot fonction de la configuration réelle des amers et qui vérifie $\Phi(\mathbf{l}, \mathbf{q}) = 0$. Ainsi dans ce cas la configuration corrigée $\mathbf{q}_c$ existe et est unique.

2. Si $rang(W) = k < r$, alors $\frac{\partial \Phi}{\partial \mathbf{q}}(\mathbf{l}_0, \mathbf{q}_0)$ n'est pas inversible. Dans ce cas on ne peut pas utiliser le *théorème des fonctions implicite* pour exprimer la variable de configuration $\mathbf{q}$ en fonction de la configuration des amers.

Ainsi l'existence et l'unicité de $\mathbf{q}_c$ dépendent du rang de la matrice W par rapport à la dimension r de l'espace des configurations. Dans la suite de notre étude de la configuration corrigée on se placera toujours dans la première situation.

Pour éclaircir ces situations voyons quelques cas classiques de localisation. Le premier concerne la situation où on veut localiser dans un plan un robot mobile équipé d'un scanner laser S avec deux droites perpendiculaires L_1 et L_2 (voir figure (III.4) de gauche). Considérons que la configuration du robot correspond à la configuration du capteur, i.e. $\mathbf{q} = (s^1, s^2, s^3)$. Soit $\mathbf{q}_0 = (1,0,0)$

la configuration de référence du robot. Soit $l_{01} = (l_{01}^1 = 4, l_{01}^2 = 0)$ et $l_{02} = (l_{02}^1 = 0, l_{02}^2 = 5)$ respectivement les configurations de référence des deux droites L_1 et L_2 (ces configurations correspondent aux coordonnées du point issu de la projection de l'origine du référentiel global sur chacune des deux droites). Soit $w_1 = 1$ et $w_2 = 1$ les poids associés respectivement aux amers L_1 et L_2 visible par S. En tenant compte des formules établies dans la section III.2.3 concernant la perception d'une droite par un scanner laser, nous écrivons la matrice W pour cette situation :

$$W = \begin{pmatrix} w_1 \frac{\partial P_{S,L_1}}{\partial s}(s(\mathbf{q}_0), l_{01}) \frac{\partial s}{\partial \mathbf{q}}(\mathbf{q}_0) \\ \\ w_2 \frac{\partial P_{S,L_2}}{\partial s}(s(\mathbf{q}_0), l_{02}) \frac{\partial s_{i_2}}{\partial \mathbf{q}}(\mathbf{q}_0) \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 5 \\ 0 & -1 & 0 \end{pmatrix}$$

La matrice W est de rang plein colonne, égal à la dimension de l'espace des configuration ($r = 3$). Dans ce cas la configuration corrigée existe et est unique. Dans la situation de la figure (III.4) du milieu, où les deux droites L_1 et L_2 sont parallèle avec les configurations de référence respectives $l_{01} = (l_{01}^1 = 0, l_{01}^2 = -5)$ et $l_{02} = (l_{02}^1 = 0, l_{02}^2 = 5)$, nous obtenons la matrice W suivante :

$$W = \begin{pmatrix} w_1 \frac{\partial P_{S,L_1}}{\partial s}(s(\mathbf{q}_0), l_{01}) \frac{\partial s}{\partial \mathbf{q}}(\mathbf{q}_0) \\ \\ w_2 \frac{\partial P_{S,L_2}}{\partial s}(s(\mathbf{q}_0), l_{02}) \frac{\partial s_{i_2}}{\partial \mathbf{q}}(\mathbf{q}_0) \end{pmatrix} = \begin{pmatrix} 0 & 0 & -5 \\ 0 & -1 & 0 \\ 0 & 0 & 5 \\ 0 & -1 & 0 \end{pmatrix}$$

Ainsi la matrice W n'est que de rang 2, ce qui est inférieur à la dimension de l'espace des configurations. Dans ce cas particulier toutes les configurations à orientation égale appartenant à une droite parallèle aux deux droite L_1 et L_2 produisent la même perception. Ceci est le problème classique de localisation d'un robot mobile dans un long couloir où les bouts ne sont pas perceptibles par les capteurs embarqués. Un autre cas de rang déficient de la matrice W peut se produire lorsqu'on considère par exemple que notre robot mobile, cité dans cet exemple, est équipé d'une remorque dont la position est caractérisée par l'angle α qu'elle fait avec l'axe du robot (figure (III.4) de droite). Dans ce cas le vecteur de configuration $\mathbf{q} = (s^1, s^2, s^3, \alpha)$ est de dimension 4 . En reprenant les hypothèses de la première situation mettant en jeu deux droites perpendiculaires, on obtient la matrice W suivante :

$$W = \begin{pmatrix} w_1 \frac{\partial P_{S,L_1}}{\partial s}(s(\mathbf{q}_0), l_{01}) \frac{\partial s}{\partial \mathbf{q}}(\mathbf{q}_0) \\ \\ w_2 \frac{\partial P_{S,L_2}}{\partial s}(s(\mathbf{q}_0), l_{02}) \frac{\partial s_{i_2}}{\partial \mathbf{q}}(\mathbf{q}_0) \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}$$

L'indépendance de la configuration du capteur par rapport au paramètre de configuration de la remorque rend la matrice W de rang colonne déficient et ceci quel que soit le nombre d'amers utilisés pour la localisation. Alors que la configuration corrigée de la partie mobile du véhicule est unique, la remorque peut adopter n'importe qu'elle situation sans que ceci affecte la perception des amers. D'après l'étude que nous avons faite sur la configuration corrigée, nous ne pourrons rien dire sur son existence et son unicité pour ces deux derniers cas. Lors de la planification on

essaiera d'éviter ces situations en cherchant des amers qui permetteront d'avoir une matrice de localisation de rang égal à la dimension de l'espace des configurations du robot.

Après avoir établi l'existence de la configuration corrigée $\mathbf{q}_c$, nous allons nous intéresser à présent à la stabilité du système (III.8).

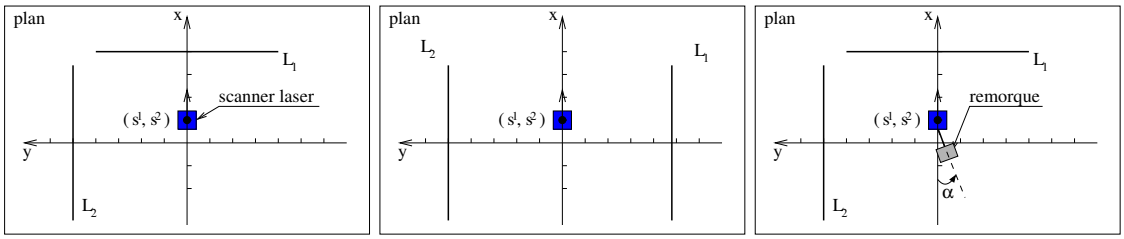

FIG. III.4 – Exemples d'unicité de la configuration corrigée. Figure de gauche, $\mathbf{q}_c$ existe et est unique. La figure du milieu et celle de droite représentent des situations où on ne peut pas se prononcer sur l'existence de $\mathbf{q}_c$. En pratique on évitera ces situations.

III.6.2 Étude de la stabilité autour de la configuration corrigée

Nous allons maintenant montrer que $\mathbf{q}_c$ est une configuration *localement asymptotiquement stable* du système (III.8). Le système défini par l'équation (III.8) est un système non linéaire où $\mathbf{q}_{loc}$ est une fonction non linéaire de la configuration réelle du robot. En utilisant la formulation de (III.7) on écrit notre système sous la forme :

$$\dot{\mathbf{q}} = -W^{+}(IM(\mathbf{q},\mathbf{l}) - IM_0) \tag{III.12}$$

En développant l'expression de $IM(\mathbf{q},\mathbf{l})$ autour du point $(\mathbf{q}_c,\mathbf{l})$ au premier ordre par un développement de Taylor nous obtenons :

$$\dot{\mathbf{q}} = -W^{+}\left(IM(\mathbf{q}_c,\mathbf{l}) - IM_0 + \frac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_c,\mathbf{l})\,(\mathbf{q}-\mathbf{q}_c) + o(\mathbf{q}-\mathbf{q}_c)\right) \tag{III.13}$$

avec $o(\mathbf{q}-\mathbf{q}_c)$ le reste d'ordre 1 de la linéarisation autour de $\mathbf{q}_c$.
D'après la propriété (III.9), le terme $\{W^{+}(IM(\mathbf{q}_c,\mathbf{l}) - IM_0)\}$ s'annule, nous obtenons alors :

$$\dot{\mathbf{q}} = -W^{+}\frac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_c,\mathbf{l})\,(\mathbf{q}-\mathbf{q}_c) + o(\mathbf{q}-\mathbf{q}_c) \tag{III.14}$$

En posant le changement de variable $\tilde{\mathbf{q}} = \mathbf{q} - \mathbf{q}_c$, nous obtenons le système suivant :

$$\dot{\tilde{\mathbf{q}}} = -W^{+}\frac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_c,\mathbf{l})\,\tilde{\mathbf{q}} + o(\tilde{\mathbf{q}}) \tag{III.15}$$

Pour déterminer la stabilité d'un tel système nous utilisons la théorie de *Lyapunov*. Pour cela, nous posons la fonction de *Lyapunov* quadratique suivante :

$$V(\tilde{\mathbf{q}}) = \frac{1}{2}\|\tilde{\mathbf{q}}\|^2 \tag{III.16}$$

Rappelons que le théorème de *stabilité* de *Lyapunov* stipule que si :

$$\exists \alpha > 0,\ \forall \tilde{\mathbf{q}} \in \mathscr{C}, \quad \dot{V}(\tilde{\mathbf{q}}) \leq -\alpha V(\tilde{\mathbf{q}})$$

alors $\tilde{\mathbf{q}} = 0$ est un *équilibre globalement asymptotiquement stable*.
Calculons la dérivée par rapport au temps de notre fonction de *Lyapunov* :

$$\begin{aligned} \dot{V}(\tilde{\mathbf{q}}) &= \tilde{\mathbf{q}}^T \dot{\tilde{\mathbf{q}}} \\ &= \tilde{\mathbf{q}}^T \left(-W^+ \tfrac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_c, \mathbf{l})\, \tilde{\mathbf{q}} + o(\tilde{\mathbf{q}}) \right) \\ &= -\tilde{\mathbf{q}}^T W^+ \tfrac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_c, \mathbf{l})\, \tilde{\mathbf{q}} + o(\|\tilde{\mathbf{q}}\|^2) \end{aligned}$$

Notons que par définition W^+W est une matrice symétrique, elle est égale à la matrice identité d'ordre r dans notre cas. Comme $\mathbf{q}_c$ est au voisinage de $\mathbf{q}_0$ et la configuration réelle $\mathbf{l}$ des amers est dans le voisinage de leur configuration de référence $\mathbf{l}_0$, alors $\frac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_c, \mathbf{l})$ est proche de W. Mais comme $\left(W^+ \frac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_c, \mathbf{l}) \right)$ n'est pas symétrique nous écrivons de nouveau l'expression de $\dot{V}(\tilde{\mathbf{q}})$ de la façon suivante :

$$\dot{V}(\tilde{\mathbf{q}}) = -\tfrac{1}{2}\, \tilde{\mathbf{q}}^T \left[\left(W^+ \tfrac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_c, \mathbf{l}) \right) + \left(W^+ \tfrac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_c, \mathbf{l}) \right)^T \right] \tilde{\mathbf{q}} + o(\|\tilde{\mathbf{q}}\|^2)$$

En posant $A = \frac{1}{2} \left[\left(W^+ \frac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_c, \mathbf{l}) \right) + \left(W^+ \frac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_c, \mathbf{l}) \right)^T \right]$ on obtient :

$$\dot{V}(\tilde{\mathbf{q}}) = -\tilde{\mathbf{q}}^T A\, \tilde{\mathbf{q}} + o(\|\tilde{\mathbf{q}}\|^2) \tag{III.17}$$

la matrice A est une matrice symétrique proche de la matrice symétrique W^+W qui est égale à la matrice identité d'ordre r. Dans ce cas les valeurs propres de A sont suffisamment proches de l'unité. Soit, $\lambda_{min} \geq \frac{3}{4}$, la plus petite de ces valeurs propres qu'on choisit de minorer par $\frac{3}{4}$. Nous savons par ailleurs que :

$$\tilde{\mathbf{q}}^T A\, \tilde{\mathbf{q}} \geq \lambda_{min} \|\tilde{\mathbf{q}}\|^2$$

nous en déduisons alors à partir de (III.17) que :

$$\dot{V}(\tilde{\mathbf{q}}) \leq -\frac{3}{4}\|\tilde{\mathbf{q}}\|^2 + o(\|\tilde{\mathbf{q}}\|^2) \tag{III.18}$$

sachant que :

$$\forall \varepsilon > 0 \text{ il existe un voisinage de } o \text{ sur lequel } \tilde{\mathbf{q}} \text{ satisfait : } -\varepsilon \|\tilde{\mathbf{q}}\|^2 \leq o(\|\tilde{\mathbf{q}}\|^2) \leq \varepsilon \|\tilde{\mathbf{q}}\|^2$$

En prenant $\varepsilon = \frac{1}{4}$ et en remplaçant dans (III.18) on obtient :

$$\dot{V}(\tilde{\mathbf{q}}) \leq -\frac{3}{4}\|\tilde{\mathbf{q}}\|^2 + \frac{1}{4}\tilde{\mathbf{q}}^T\, \tilde{\mathbf{q}}$$

d'où

$$\dot{V}(\tilde{\mathbf{q}}) \leq -V(\tilde{\mathbf{q}}) \tag{III.19}$$

Finalement, d'après le théorème de *Lyapunov*, la configuration corrigée $\mathbf{q}_c$ qui existe dans un voisinage de $\mathbf{q}_0$ est le point d'*équilibre localement asymptotiquement stable* pour le système (III.8).

Reprenons les trois situations de l'exemple précédent. Pour le premier cas où la configuration corrigée est unique (figure (III.4) de gauche), le robot converge vers la configuration planifiée. Dans les deux dernières situations (figures (III.4) du milieu et de droite), on ne peut rien dire. On essaiera lors de la phase de planification d'éviter ces situations.
Dans la suite de cette section, nous allons établir deux propriétés vérifiées par la configuration corrigée $\mathbf{q}_c$.

III.6.3 Calcul de l'expression de la configuration corrigée

Dans le cas où $rang(W) = r$, la configuration $\mathbf{q}_c$ existe et est unique. Elle est au voisinage de $\mathbf{q}_0$ et la configuration réelle $\mathbf{l}$ des amers est dans le voisinage de leur configuration de référence $\mathbf{l}_0$. Linéarisons autour du point $(\mathbf{l}_0, \mathbf{q}_0)$ l'expression (III.9). Nous obtenons alors par un calcul immédiat :

$$W^+ \left(\frac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_0, \mathbf{l}_0)\,(\mathbf{q}_c - \mathbf{q}_0) + \frac{\partial IM}{\partial l}(\mathbf{q}_0, \mathbf{l}_0)\,(\mathbf{l} - \mathbf{l}_0) + o(\mathbf{q}_c - \mathbf{q}_0) + o(\mathbf{l} - \mathbf{l}_0) \right) = 0 \quad \text{(III.20)}$$

avec : $\frac{\partial IM}{\partial l}(\mathbf{q}_0, \mathbf{l}_0)$, une matrice Jacobienne représentant la variation de l'image en fonction de la variation de la configuration amer. Cette matrice dépend des propriétés géométriques des capteurs et des amers.
En négligeant les restes d'ordre 1 de cette équation en $(\mathbf{q}_c - \mathbf{q}_0)$ et $(\mathbf{l} - \mathbf{l}_0)$ et en remarquant que $W^+ \frac{\partial IM}{\partial \mathbf{q}}(\mathbf{q}_0, \mathbf{l}_0)$ est égal à la matrice identité, on obtient le système linéaire suivant :

$$\mathbf{q}_c - \mathbf{q}_0 + W^+ W_L (\mathbf{l} - \mathbf{l}_0) = 0 \quad \text{(III.21)}$$

Où W_L est la matrice :

$$\frac{\partial IM}{\partial l}(\mathbf{q}_0, \mathbf{l}_0) = \begin{pmatrix} w_1 \frac{\partial P_{S_{i_1}, L_1}}{\partial l}(s_{i_1}(\mathbf{q}_0), l_{01}) \\ \vdots \\ w_m \frac{\partial P_{S_{i_m}, L_m}}{\partial l}(s_{i_m}(\mathbf{q}_0), l_{0m}) \end{pmatrix}$$

Les inconnues du système linéaire (III.21) sont $\mathbf{q}_c$ et $\mathbf{l}$. En se référant au § III.3.1 où nous avons mentionné que si la localisation s'effectue à une fréquence assez élevée, on peut supposer que la configuration de localisation $\mathbf{q}_{loc}$ est au voisinage de la configuration de référence $\mathbf{q}_0$. L'asservissement du robot sur la configuration de référence nous permet de considérer que les perceptions mesurées $IM(\mathbf{q}, \mathbf{l})$ en environnement réel et utilisées pour le calcul de la localisation sont effectuées d'une configuration proche de la configuration de référence. Nous développons cette fonction au premier ordre en $\mathbf{q}$ et en $\mathbf{l}$ autour du point $(\mathbf{q}_0, \mathbf{l}_0)$. L'expression obtenue après avoir négligé les restes de la linéarisation en $(\mathbf{q} - \mathbf{q}_0)$ et $(\mathbf{l} - \mathbf{l}_0)$ s'écrit :

$$IM(\mathbf{q}, \mathbf{l}) = IM_0 + W\,(\mathbf{q} - \mathbf{q}_0) + W_L\,(\mathbf{l} - \mathbf{l}_0) \quad \text{(III.22)}$$

En arrangeant les termes de cette équation et en les multipliant à gauche par W^+ nous avons :

$$W^+ (IM(\mathbf{q},\mathbf{l}) - IM_0) = W^+ W (\mathbf{q} - \mathbf{q}_0) + W^+ W_L (\mathbf{l} - \mathbf{l}_0) \quad \text{(III.23)}$$

En se référent au résultat de localisation de l'équation (III.7) nous déduisons l'expression :

$$\mathbf{q}_{loc}(\mathbf{q},\mathbf{l}) - \mathbf{q}_0 = W^+ W (\mathbf{q} - \mathbf{q}_0) + W^+ W_L (\mathbf{l} - \mathbf{l}_0) \quad \text{(III.24)}$$

Ce qui nous donne :

$$\mathbf{q}_{loc}(\mathbf{q},\mathbf{l}) = \mathbf{q} + W^+ W_L (\mathbf{l} - \mathbf{l}_0) \quad \text{(III.25)}$$

En substituant le terme $W^+ W_L (\mathbf{l} - \mathbf{l}_0)$ issu de l'équation (III.21) dans l'expression ci-dessus nous trouvons après arrangement :

$$\mathbf{q}_c - \mathbf{q} = -(\mathbf{q}_{loc}(\mathbf{q},\mathbf{l}) - \mathbf{q}_0) \quad \text{(III.26)}$$

Ainsi l'écart entre configuration corrigée et configuration de perception est l'opposé de l'écart entre configuration de localisation et configuration de référence. Ce résultat exprime une propriété locale de la configuration corrigée.

Propriété III.6.2 *L'écart entre la configuration corrigée et la configuration de perception est l'opposé de l'écart entre la configuration de localisation et la configuration de référence suivant la formule (III.26).*

Une propriété de la configuration corrigée qui concerne les résidus de perception découle de ce résultat. Ceci est l'objet du paragraphe suivant.

III.6.4 Calcul du résidu de la perception en configuration corrigée

Dans un cadre général, il est impossible que le robot se localise dans son plan de référence de manière à parfaitement respecter la perception de son environnement. De façon analogue une fois dans la configuration corrigée il est impossible que le robot ait satisfait complètement la perception prévue par le plan dans la configuration de référence.

Nous rappelons ici l'expression du résidu de localisation (c.f. § (III.4)) qui représente la norme carrée de l'écart entre la perception des amers dans la configuration de localisation et la perception mesurée.

$$J = \|IM(\mathbf{q}_{loc}(\mathbf{q},\mathbf{l}),\mathbf{l}_0) - IM(\mathbf{q},\mathbf{l})\|^2 \quad \text{(III.27)}$$

Le résidu de correction est formulé par la norme carrée de l'écart entre la perception des amers dans la configuration corrigée et la perception de référence.

$$J' = \|IM(\mathbf{q}_c,\mathbf{l}) - IM(\mathbf{q}_0,\mathbf{l}_0)\|^2 \quad \text{(III.28)}$$

Développons au premier ordre la perception au point $(\mathbf{q}_c,\mathbf{l})$ autour du point de référence $(\mathbf{q}_0,\mathbf{l}_0)$ et en négligeant le résidu de linéarisation nous trouvons :

$$IM(\mathbf{q}_c,\mathbf{l}) = IM_0 + W (\mathbf{q}_c - \mathbf{q}_0) + W_L (\mathbf{l} - \mathbf{l}_0) \quad \text{(III.29)}$$

Remplaçons ce résultat et le résultat de l'équation (III.22) dans l'expression (III.28) du résidu J' nous obtenons après calcul :

$$J' = \|W(\mathbf{q}_c - \mathbf{q})\|^2 \tag{III.30}$$

En utilisant le résultat de la propriété (III.6.2) nous exprimons le résidu de perception en $\mathbf{q}_c$ en fonction de la configuration de localisation :

$$\begin{aligned} J' &= \| - W(\mathbf{q}_{loc}(\mathbf{q},\mathbf{l}) - \mathbf{q}_0)\|^2 \\ &= \|W(\mathbf{q}_{loc}(\mathbf{q},\mathbf{l}) - \mathbf{q}_0)\|^2 \end{aligned} \tag{III.31}$$

Par ailleurs si nous développons la perception au point $(\mathbf{q}_{loc}(\mathbf{q},\mathbf{l}),\mathbf{l}_0)$ autour de la référence $(\mathbf{q}_0,\mathbf{l}_0)$ en négligeant le résidu d'ordre 1 de la linéarisation,

$$IM(\mathbf{q}_{loc}(\mathbf{q},\mathbf{l}),\mathbf{l}_0) = IM_0 + W(\mathbf{q}_{loc}(\mathbf{q},\mathbf{l}) - \mathbf{q}_0) \tag{III.32}$$

En remplaçant cette équation dans l'expression (III.27) du résidu de localisation J', on obtient après simplification des termes opposés :

$$J = \|W(\mathbf{q}_{loc}(\mathbf{q},\mathbf{l}) - \mathbf{q}_0)\|^2 \tag{III.33}$$

Finalement, en comparant (III.31) et (III.33) nous trouvons que :

$$J' = J \tag{III.34}$$

D'où la propriété suivante :

Propriété III.6.3 *Le résidu de la perception dans la configuration corrigée $\mathbf{q}_c$ exprimé en (III.28) est égal au résidu de perception dans la configuration de localisation $\mathbf{q}_{loc}(\mathbf{q},\mathbf{l})$ exprimé en (III.27).*

Revenons à l'exemple III.3.2 pour lequel la configuration corrigée est unique. Celle-ci est décalée en translation par rapport à la configuration exécutée $\mathbf{q}_0$ d'une distance D vers le haut et d'une même distance vers la gauche. En rotation, le robot subit une transformation d'un angle α dans le sens des aiguilles d'une montre (voir figure (III.5)). Pour cette configuration, le robot a une perception dans son environnement d'exécution des murs L_1 et L_2 identique à la perception des mêmes murs dans le plan de référence. De ce fait, la localisation retourne la configuration planifiée $\mathbf{q}_0$.

III.7 Conclusion

La définition III.5.1 constitue l'élément le plus important de notre approche de mouvement asservi sur amers. Elle présente ce mouvement comme un chemin géométrique auquel est attribué un ensemble de primitives pondérées issues de l'association des amers de l'environnement et des capteurs du robot. La pondération de chaque primitive est une spécification intrinsèque du mouvement asservi sur amers. La construction de ces fonctions est l'objet du chapitre suivant.

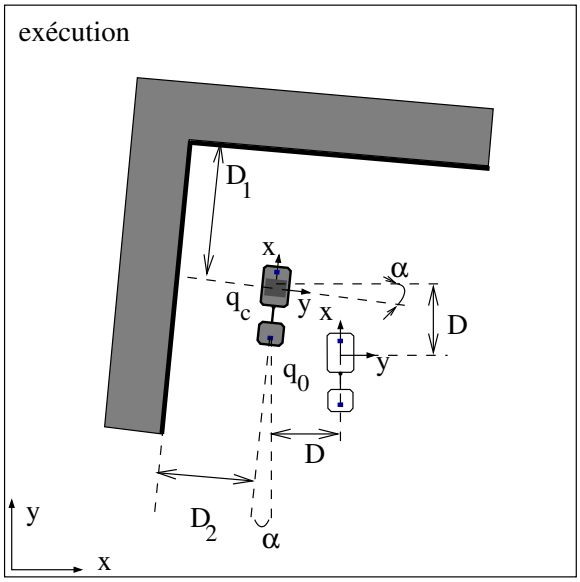

FIG. III.5 – Robot se trouvant à la configuration corrigée. Cette configuration est décalée vers la gauche d'une distance D et vers le haut d'une même distance par rapport à la configuration actuelle du robot dans l'environnement d'exécution. En rotation, le robot subit une transformation d'un angle α dans le sens des aiguilles d'une montre. Dans ce cas, la configuration corrigée est unique.

Chapitre IV

Principes et calculs des fonctions de pondération

IV.1 Introduction

A la fin du chapitre précédent nous avons introduit la définition du mouvement asservi sur amers où l'on propose à partir d'un chemin géométrique planifié de fournir une liste de fonctions de pondération associées aux amers de l'environnement, afin de les utiliser pour asservir l'exécution de la trajectoire dans l'environnement réel. L'introduction des fonctions de pondération comme caractéristique intrinsèque du mouvement asservi sur amers soulève essentiellement trois grandes questions :

- Comment construire ces fonctions ?
- Comment sélectionner les amers les plus pertinents pour l'exécution du mouvement ?
- Comment enchaîner la succession des amers choisis au cours du mouvement ?

Dans un premier temps, ce chapitre tente de répondre à chacune de ces questions en proposant des méthodes de calcul génériques des fonctions de pondération, tout en établissant certaines propriétés. Dans un deuxième temps, on propose des algorithmes permettant de définir des critères de sélection par rapport à la trajectoire planifiée et nous montrons comment construire la structure algorithmique d'un mouvement asservi sur amers afin de pouvoir prendre en compte à la fois l'aspect sélection des amers et l'aspect enchaînement des séquences d'amers choisis. Nous verrons dans le chapitre dédié aux expérimentations que cette approche peut grandement être améliorée en prenant en compte des critères liés à l'exécution du mouvement tels que le conditionnement de la matrice de localisation pondérée et l'appariement des primitives planifiées en environnement réel. A la fin de ce chapitre, on fournit des résultats de simulations effectuées sur

le robot Hilare2 avec sa remorque.
Avant d'aborder cette partie du chapitre nous allons d'abord voir sur un exemple le problème de la variation de la pondération des amers.

IV.2 Exemple de l'effet statique de la variation des poids

Pour voir l'influence de la variation des poids sur le résultat de la localisation on considère l'exemple de la figure IV.1. Soit le robot Hilare2 avec sa remorque s'apprêtant à passer en ligne droite au milieu du passage entre deux obstacles L_1 et L_2. Tout au long du chemin on considère deux couples capteur-amer issus de l'association du scanner laser avant S_1 et des deux amers L_1 et L_2 (voir figure IV.1(a)). Soit w_{11} et w_{12} les deux fonctions de pondération associées respectivement aux couples (S_1, L_1) et (S_1, L_2). Ainsi nous avons dans ce cas un mouvement asservi sur amers constitué d'un chemin géométrique sans collision (une ligne droite passant au milieu d'une porte entre L_1 et L_2) et de deux fonctions de pondération w_{11} et w_{12}.

Dans un premier temps, figure IV.1(b), on suppose que l'amer L_1 en environnement réel est décalé vers le bas d'une distance D de sa position de référence dans le modèle, ce qui n'est pas le cas de l'amer L_2 qui garde sa configuration de référence. On obtient alors un environnement d'exécution localement incohérent par rapport au plan de référence. En attribuant, le long du chemin, la valeur 1 à la fonction w_{11} et la valeur 0 à w_{12}, le robot sera décalé vers le bas par rapport au chemin de référence d'une distance D. S'il n'est pas en collision avec L_2 ce chemin solution ne traversera pas la porte par le milieu.

Dans un second temps, figure IV.1(c), on met la fonction w_{12} à 1 et donc tout au long du chemin les deux amers auront la même importance vis à vis de la localisation. Le robot ne se décalera vers le bas que de $D/2$ ce qui va lui permettre de traverser la porte par le milieu. La troisième situation, figure IV.1(d), correspond au cas où l'amer L_2 est décalé d'une distance D mais vers le haut par rapport à sa position de référence. En gardant des fonctions de pondération égales, le robot ne se décale pas et reste sur sa trajectoire de référence car l'effet conjugué des erreurs opposés issues des deux murs s'annule avec une pondération égale. On constate donc que le choix des pondérations des couples capteurs-amers aura une grande importance sur la boucle de mouvement référencé sur des amers.

IV.3 Fonctions de pondération

Comparons les figures IV.1(b) et IV.1(c) avec la même représentation de l'environnement d'exécution par rapport à l'environnement de référence (seul L_1 est décalé). De façon triviale, on observe que dans les deux situations et pour une même configuration, deux valeurs différentes du poids produisent deux estimées différentes de la configuration du robot. Ceci entraîne une discontinuité de la localisation à cause du changement brutal de w_{11} de 0 à 1. Pour cette raison on impose une condition nécessaire de continuité aux fonctions de pondération.

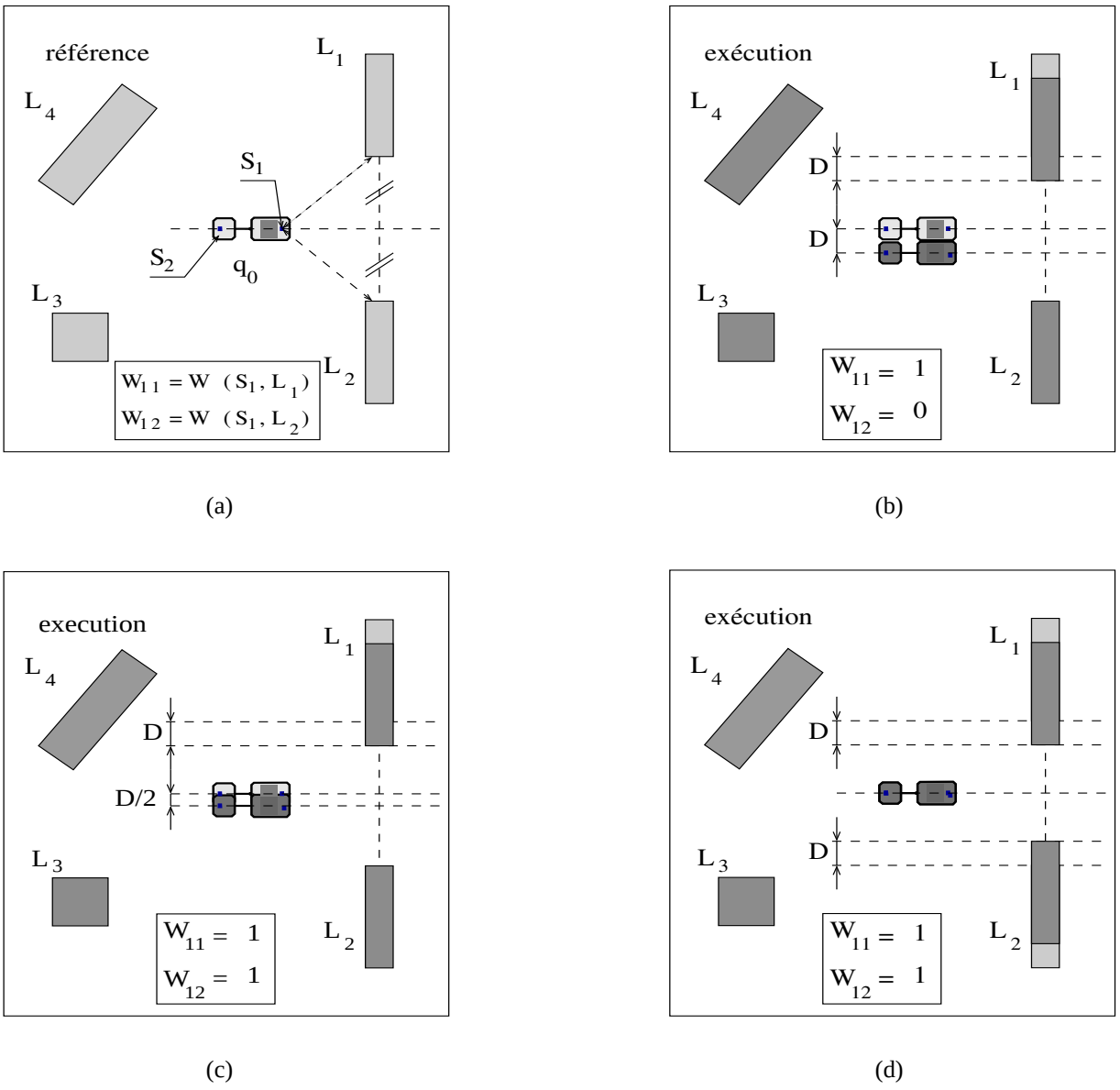

FIG. IV.1 – Exemple de l'effet de la variation statique des fonctions de pondération. Fig IV.1(a) : Dans un plan de référence donné, le robot Hilare2 avec sa remorque s'apprête à traverser une porte par le milieu suivant une ligne droite. $\mathbf{q}_0$ est une configuration appartenant au chemin géométrique planifié. Tout au long de ce chemin le scanner laser avant S_1 est associé aux deux côtés de la porte L_1 et L_2 avec les poids w_{11} et w_{12} respectivement. Fig IV.1(b) : Dans l'environnement d'exécution, L_1 est décalé vers le bas d'une distance D par rapport à sa position de référence. Tout au long du chemin on prend $w_{11} = 1$ et $w_{12} = 0$. Pour atteindre la configuration de correction le robot se décale d'une distance D vers le bas par rapport à $\mathbf{q}_0$. Fig IV.1(c) : Même situation que précédemment sauf que cette fois ci $w_{12} = 1$. L'estimation de la configuration de correction retourne une position décalée seulement de $D/2$ vers le bas par rapport à $\mathbf{q}_0$. Fig IV.1(d) Par rapport aux deux cas précédents, à présent on considère que l'amer L_2 est décalé vers le haut d'une distance D. Avec des fonctions de pondération égales, le robot ne se décale pas de sa configuration de référence après localisation.

En ne prenant pas en compte les amers non perçus par les capteurs dans le processus de localisation on impose aux fonctions de pondération d'être nulles lorsque l'amer en question n'appartient pas au champ de perception du capteur.

De manière générale, on admet que pour une petite variation de la configuration du robot, la fonction de pondération varie peu pour ne pas produire des écarts dynamiques importants dans l'estimation de la localisation. Par exemple dans la situation de l'environnement d'exécution des figures IV.1(b) et IV.1(c). En supposant que le robot parcourt une distance δ suivant sa trajectoire rectiligne et que durant cette distance le poids w_{12} associé à l'amer L_2 varie de 0 à 1, alors le robot aura à rattraper cette erreur de $D/2$ sur une distance proche de δ. Suivant le type de contraintes cinématiques auquel le robot est soumis, deux cas se présentent. Si le robot est holonome, il peut facilement rattraper l'écart de localisation car il peut se mouvoir instantanément dans toutes les directions de l'espace de travail. Mais si le robot est non-holonome et que l'écart à corriger est suivant la direction transversale à sa vitesse (comme le montre la figure IV.1(c)), alors le problème est plus critique. En effet pour que ce type de robot réalise un déplacement transversal il doit parcourir une certaine distance longitudinale et doit modifier son orientation. Pour cela la distance δ durant laquelle le poids w_{12} passe de 0 à un 1 doit être suffisamment grande pour permettre au robot de réaliser l'écart transversal de $D/2$ qui va lui permettre de corriger sa configuration. L'erreur de perception induite par l'angle de braquage peut être réduite si la distance δ est grande.
Ainsi on peut définir les propriétés nécessaires de nos fonctions de pondération :

Propriété IV.3.1 *Une fonction de pondération d'une paire (S,L) doit obéir aux trois propriétés suivantes :*

1. *Elle doit être continue et positive tout au long du chemin géométrique planifié $\gamma(u)$, $u \in [0,U]$.*
2. *Elle doit être nulle lorsque l'amer L n'appartient pas au champ de perception du capteur S qui lui est associé.*
3. *$\forall u, u\prime \in [0,U]$, $|w(u) - w(u\prime)| \leq \beta |u - u\prime|$, avec β une constante qui dépend des contraintes cinématiques du robot et de ces paramètres cinétiques et dynamiques. Dans ce cas la fonction de pondération w est dite $\beta -$ Lipschitz.*

Plus tard dans ce chapitre on reviendra sur la propriété 3. Pour le moment on s'intéresse au calcul formel des fonctions de pondération.

IV.4 Construction des fonctions de pondération

IV.4.1 Principe Général

Dans les méthodes d'estimation probabiliste utilisant des moindres carrés pondérés, les valeurs des poids utilisés correspondent à l'inverse de la matrice de variance covariance associée aux mesures. Dans ce cas, plus les mesures ont une grande variance moins elles seront prises en

compte. Ce principe est le même pour la majeure partie des méthodes probabiliste d'estimation qui supposent une représentation gaussienne des variables aléatoires associées au modèle (e.g. filtrage de Kalman).

Dans notre travail, on considère les poids comme une propriété intrinsèque de la tâche robotique associée à l'exécution du chemin géométrique planifié. Ainsi pour une configuration donnée le long d'une trajectoire, l'attribution d'un poids à un couple capteur-amer doit refléter son importance par rapport à l'environnement et à la trajectoire, afin d'éviter toute collision. En ce sens il est semblable au module de la force de répulsion utilisée dans les méthodes de déformation de trajectoire par bande élastique [73] et il doit permettre de respecter le chemin planifié tout en tenant compte des changements de l'environnement. Ces fonctions de pondération peuvent être assimilées à des modules de forces tentant d'amener le robot à la position d'équilibre (configuration de référence dans le plan de référence) dont l'expression d'équilibre est régie par l'équation de localisation pondérée (III.6). Il est difficile d'établir une méthode générique et précise permettant le calcul formel de telles fonctions pour tout type de robot, tout type de capteur et tout type d'amer. Cependant nous présentons dans ce qui suit un formalisme général pouvant être adapté suivant les différents cas qui peuvent se présenter pour calculer ces fonctions.

L'idée est de décomposer la fonction de pondération en un produit de sous fonctions où chacune d'elle représente une caractéristique précise, essentiellement la perception de l'amer par le capteur et le risque de collision que cet amer représente pour une configuration donnée du robot tout en respectant les propriétés établies en IV.3.1. Ces sous fonctions de pondération sont définies dans l'espace des configurations. Comme on le verra plus tard dans ce chapitre, pour un chemin géométrique donné, on calcule ces fonctions pour les différents couples capteurs-amers extraient de l'environnement par les algorithmes de sélection. Le choix des amers se fera essentiellement sur les amers de plus grandes valeurs de poids calculées après construction de ces fonctions de pondération.

IV.4.2 Fonction de pondération associée à la perception

Definition IV.4.1 *Dans un plan de référence donné, soit $P_{S,L}$ une fonction de perception de classe C^1 définie sur une partie de $\mathbb{S}\times\mathbb{L}$ lorsque L est visible depuis S. Pour le couple (S,L) la fonction de pondération associée à la perception $w_{visi_{S,L}}$ est définie par :*

$$\begin{array}{rccc} w_{visi_{S,L}} : & \mathbb{S}\times\mathbb{L} & \rightarrow & \mathbb{R}^+ \\ & (s,l) & \mapsto & w_{visi_{S,L}}(s,l) \end{array}$$

où

$$w_{visi_{S,L}}(s,l)\begin{cases} > 0 & \textit{si à la configuration s, L est visible depuis S.} \\ = 0 & \textit{sinon.} \end{cases}$$

En multipliant $w_{visi_{S,L}}$ avec les sous fonctions de pondération continues définies ci-dessous, on va permettre d'assurer la deuxième propriété (c.f. § IV.3), à savoir que le poids associé à un couple capteur-amer sera nul en dehors du champs de visibilité du capteur. $w_{visi_{S,L}}$ est une fonction décroissante lorsque l'amer tend à disparaître du champ de visibilité du capteur.

IV.4.3 Fonction de pondération associée à la collision

Cette sous fonction de pondération est inspirée des méthodes utilisés pour l'évitement réactif d'obstacle proposées initialement dans [38]. En effet, le long d'un chemin géométrique planifié, sa valeur correspond au module de la force de répulsion entre l'amer, vu ici comme un obstacle, et le robot. Ainsi nous adoptons la définition suivante pour la fonction de pondération associée à la collision :

Definition IV.4.2 *Dans un plan de référence donné, pour chaque amer L la fonction de pondération w_{Col_L} associée à la collision est définie par :*

$$\begin{array}{rccc} w_{Col_L} : & \mathscr{C} \times \mathbb{L} & \rightarrow & \mathbb{R}^+ \\ & (\mathbf{q}, l) & \mapsto & w_{Col_L}(\mathbf{q}, l) \end{array}$$

En utilisant la plus courte distance entre le robot et l'amer L, w_{Col_L} peut avoir une valeur inversement proportionnelle à cette distance. Dans la littérature traitant des évitements réactifs d'obstacle, on définit une distance limite de l'influence du champs de potentiel de répulsion. Dans notre cas, il n'est pas nécessaire de définir cette distance car l'influence de la fonction de pondération associée à la collision sera limitée par la fonction de pondération associée à la perception. Dans une configuration donnée, le robot sera sujet à des poids (ou forces) représentant le danger de collision potentiel issue de chaque amer sélectionné pour la localisation. Ainsi la correction de la configuration du robot sera fonction du danger de collision et de l'écart entre plan de référence et plan d'exécution.

IV.4.4 Fonction de pondération prédéfinie

En plus du calcul automatique des sous fonctions de pondération telles que les fonctions associées à la collision et à la perception servant au calcul du poids d'un couple capteur-amer, on peut ajouter une fonction continue prédéfinie par l'utilisateur servant à réhausser ou à rabaisser l'importance qu'on veut attribuer à un amer particulier par rapport à la valeur calculée automatiquement. Ceci peut être, par exemple, le cas d'un environnement équipé de balises de localisation ou encore des obstacles immobiles comme par exemple les piliers d'un bâtiment. Ces amers sont des primitives d'asservissement très fiables pour un bon positionnement du robot et il est nécessaire qu'ils soient pris en compte en priorité. Ainsi ce type de fonction force la sélection d'amers spécifiques dans le cas où ils ne constituent pas un danger de collision ou quand ils sont moins visibles que d'autres amers de l'environnement. Cette fonction sera notée :

$$\begin{array}{rccc} w_{user} : & \mathscr{C} \times \mathbb{L} & \rightarrow & \mathbb{R}^+ \\ & (\mathbf{q}, l) & \mapsto & w_{user}(\mathbf{q}, l) \end{array}$$

IV.4.5 Calcul générique de la fonction de pondération

Definition IV.4.3 *Soit un robot amené à suivre un chemin géométrique $\gamma(u)$, $u \in [0, U]$. Une fonction de pondération w_{S_L} associée à un couple capteur-amer constitué d'un capteur S monté*

sur ce robot et d'un amer L est le produit de trois fonctions continues :

$$\forall u \in [0,U], \quad w_{S_L}(u) = w_{visi_{S,L}}(s(\gamma(u)),l) \times w_{Col_L}(\gamma(u),l) \times w_{user}(\gamma(u),l),$$

Bien evidemment il est difficile d'arrêter la décomposition d'une fonction de pondération au produit de ces uniques trois fonctions. D'autres critères de pondération peuvent également être développés et ajoutés à ceux ci. La fonction de pondération ainsi obtenu respecte les deux premières propriétés établies en IV.3.1 c'est à dire qu'elle est continue et nulle en dehors du champs de perception du capteur. Toutefois elle ne tient pas compte des contraintes de mouvement du robot. Ce point est traité dans la section suivante.

Notons également que la fonction de pondération étant définie dans l'espace de configuration $\mathscr{C} \times \mathbb{L}$, il faut construire cette fonction pour chaque chemin planifié $\gamma(u)$, $u \in [0,U]$.

IV.4.6 Contraintes cinématique et fonction de pondération

Dans notre approche on veut que la fonction de pondération w_{S_L} associée à une paire capteur-amer (S,L) tienne compte des contraintes cinématiques du robot sur lequel est monté le capteur. Pour cela la dérivée des variables de configuration du robot impose une contrainte sur la dérivée de w_{S_L}, qui s'exprime par la fonction :

$$\frac{\partial w_{S_L}}{\partial u} = f(\frac{\partial \mathbf{q}}{\partial u}) \tag{IV.1}$$

La production du mouvement associé à ce chemin permet de paramétrer l'abscisse curviligne u de la trajectoire en fonction du temps $u = g(t)$. Ainsi on exprime cette contrainte sur l'espace temporel en tenant compte des contraintes des vitesses du robot.

$$\frac{\partial w_{S_L}}{\partial t} = h(\frac{\partial \mathbf{q}}{\partial t}) \tag{IV.2}$$

Dans le cas d'un robot non-holonome (c.f section (IV.3)), cette contrainte permet de réduire les grandes variations de l'erreur de localisation durant l'exécution du mouvement afin d'éviter des corrections latérales trop importantes et aussi pour éviter de générer des erreurs de perception importantes issues de la correction (changement d'orientation du robot pour corriger des erreurs de position latérales).

Ne pouvant pas résoudre ce problème de manière formelle, nous avons opté pour une solution sous-optimale qui consiste à imposer à la fonction de pondération une variation lente le long du chemin planifié. Cette contrainte a donné lieu à la troisième propriété présentée au paragraphe (IV.3).

Les développements conduits jusqu'ici traitent de la première question posée dans l'introduction de ce chapitre. Les algorithmes proposés ci-après traitent des deux dernières questions à savoir la sélection et l'enchaînement des paires capteurs-amers.

IV.5 Algorithmes de sélection de couples capteurs-amers

Savoir comment calculer les fonctions de pondération pour un couple capteur-amer donné n'est pas suffisant pour pouvoir construire un mouvement asservi sur amers et ceci principalement pour trois raisons :

1. Pour un couple capteur-amer donné, la courbe de poids qui lui est associé dépend du chemin géométrique planifié alors que les fonctions de pondération sont construites sur le produit cartésien des espaces des configurations $\mathscr{C} \times \mathbb{L}$. Ainsi pour chaque trajectoire $\gamma(u),\ u \in [0,U]$ il est nécessaire de calculer ces fonctions.
2. Pour un environnement de référence composé de κ amers et pour un robot à n capteurs, il existe $\kappa \times n$ couples capteurs-amers possibles mais tous les couples ne sont pas forcément nécessaires pour l'asservissement de la trajectoire.
3. Si on sélectionne des couples capteurs-amers pour en faire des séquences utilisées pour l'asservissement de la trajectoire alors il faut pouvoir traiter le passage d'une séquence à l'autre, c'est à dire leur enchaînement. Dans le cadre de l'asservissement visuel, ce problème est généralement résolu en proposant pour le suivi de deux primitives une fonction de tâche qui résulte de la combinaison convexe des deux fonctions de tâche associées à chaque primitive [71, 59].

Il faut maintenant définir un moyen qui va nous permettre de structurer un mouvement asservi sur amers pour un robot sur lequel sont montés n capteurs $\mathscr{S} = \{S_1, \ldots, S_n\}$ évoluant dans un environnement dont le plan de référence comprend κ amers géométriques $\mathscr{L}_{env} = \{L_1, \ldots, L_\kappa\}$.

Soit $\gamma(u) : [0,U] \rightarrow \mathscr{C}$, un chemin géométrique sans collision permettant au robot de se déplacer d'une configuration initiale à une configuration finale. La structure d'un mouvement asservi sur amers, noté *maa*, est constituée d'un chemin géométrique $\gamma(u),\ u \in [0,U]$ et d'un séquencement de paires capteur-amer où pour chaque capteur est défini un ensemble d'amers tout au long de la trajectoire.

Pour une configuration $\gamma(u')$, nous avons vu que l'on peut évaluer les fonctions de pondération associées à chaque couple capteur-amer et déterminer ainsi le meilleur choix de couples qui correspond aux poids les plus élevés. Cependant, nous ne connaissons pas la forme analytique de ces fonctions de poids le long d'un chemin γ. Aussi nous ne savons pas comment varient ces poids au voisinage de u' lorsque le robot se déplace le long de γ.

Enfin, nous ne connaissons pas les amers qui vont devenir non visibles ou bien visibles lorsque $u = u' + \varepsilon$. Cette approche statique peut entraîner des discontinuités dans le calcul de l'erreur de localisation.

Pour éclaircir ce point considérons le cas de la figure (IV.2) qui illustre les fonctions de pondération $w_1(u), \ldots, w_4(u)$ obtenues pour des amers $L_1, \ldots, L_4$, tous associés à un capteur S donné. Ces fonctions sont calculées pour un chemin planifié donné. Si lors de l'exécution, le robot se trouve à la configuration $\gamma(u')$ et si celui-ci s'asservit uniquement sur les deux meilleurs

amers, en l'occurrence L_1 et L_2, on s'aperçoit qu'à l'abscisse curviligne u'', L_3 prend la place de L_2 dans la liste des deux meilleurs amers. Si L_1 et L_2 n'engendrent pas d'erreur de localisation le long de γ et si la configuration réelle de L_3 est très différente de sa configuration de référence, nous aurons une discontinuité de l'erreur de localisation en u''. Pour que l'intervention de l'amer L_3 ne provoque pas de forte discontinuité il doit être pris en compte avant u''.

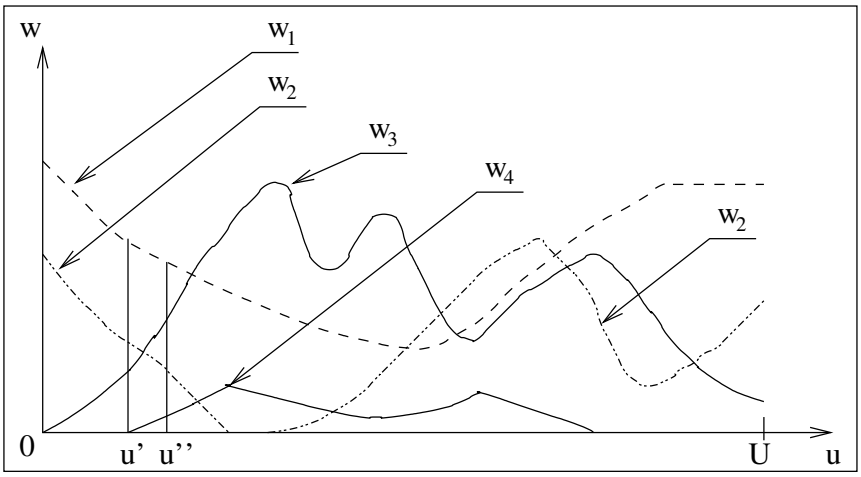

FIG. IV.2 – Fonctions de pondération réelles pour des amers $L_1, \ldots L_4$ le long d'une trajectoire géométrique.

Pour ces raisons il faut déterminer, avant l'exécution d'un chemin donné, les fonctions de pondération des amers. Nous avons fait le choix d'extraire les points caractérisant ces fonctions, essentiellement les maxima et minima locaux, pour reconstituer les fonctions par des méthodes d'interpolation lors de la phase d'exécution du mouvement. Les maxima et minima locaux sont des valeurs importantes de ces fonctions car :

– Lorsqu'on atteint un maximum, la valeur de la fonction va décroître et donc l'amer va perdre de l'intérêt.

– Lorsqu'on atteint un minimum, soit la fonction est nulle car l'amer n'est plus visible soit la valeur de la fonction va croître et l'amer va devenir de plus en plus intéressant pour le calcul de l'erreur de la localisation.

Ne disposant pas de forme analytique de ces fonctions nous calculons celles-ci point par point le long de la trajectoire afin de déterminer leurs valeurs extrémales. N'ayant pas de contrainte sur la continuité de la dérivé de la fonction de pondération nous adoptons une simple méthode d'interpolation linéaire. Il sera ainsi facile de connaître la valeur de ces fonctions pour toute valeur de u.

Ainsi pour chaque capteur, l'ensemble des amers qui lui est associé est décomposé en des sous ensembles de séquences locales d'amers sur des portions $[u_1, u_2]$ de la trajectoire. Ces sous ensembles d'amers possèdent la propriété suivante :

Propriété IV.5.1 *La séquence locale d'amers $\mathscr{L}_{select}$, associée à un capteur S, définie sur un intervalle $[u',u''] \subset [0,U]$ du chemin géométrique $\gamma(u)$ est caractérisée d'une part par la monotonie des fonctions de pondération attribuées aux paires issues de l'association de S et des amers de $\mathscr{L}_{select}$ sur $[u',u'']$ et d'autre part par les valeurs $w_{S_L}(u')$ et $w_{S_L}(u'')$ pour tout $L \in \mathscr{L}_{select}$.*

Dans le cadre de notre exemple, pour les amers $L_1,\ldots,L_4$ associés au capteur *S*, nous avons des séquences locales d'amers définies sur les intervalles $[u_i,u_{i+1}]$, $i=\overline{1,15}$ avec $u_1=0$ et $u_{15}=U$. Ainsi les fonctions de pondération de la figure IV.3 sont approximées par les fonctions de la figure IV.4.

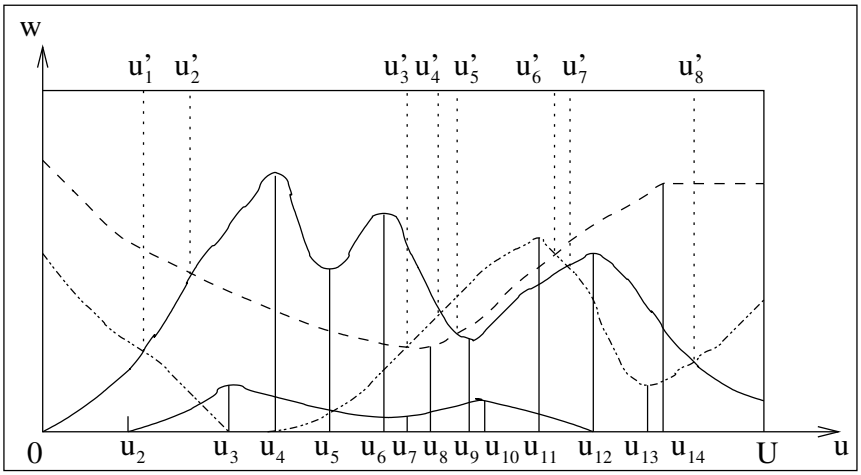

FIG. IV.3 – Extraction des points extrémaux $u_i,\ i=\overline{1,15}$ ($u_1=0$ et $u_{15}=U$) des fonctions de pondération. Extraction des points d'intersection $u'_i,\ i=\overline{1,8}$ entre les fonctions de pondération.

Par conséquent, un mouvement asservi sur amers est structuré selon le schéma ci-dessous :

– Un chemin géométrique $\gamma(u)$, $u \in [0,U]$.

– Pour chaque capteur S_i, $i \in [0,n]$, est attribué une séquence d'enchaînement d'amers notée maa_S_i sur $[0,U]$.

– Une séquence maa_S_i est constituée d'une succession de p_i sous ensemble de séquences locales d'amers $\mathscr{L}^i_{select}$ sur $[u_j,u_{j+1}]$, $j=\overline{0,p_i-1}$, où $u_0=0$ et $u_{p_i}=U$.

– Une séquence locale d'amers $\mathscr{L}^i_{select}$ est entièrement définie sur une portion de trajectoire $[u',u''] \subset [0,U]$ par l'ensemble des amers qu'elle contient et les valeurs des fonctions de pondération, $w(u)$, associées à ces amers aux points u' et u''.

Afin d'obtenir un mouvement asservi sur amers à partir de cette structure, nous proposons trois algorithmes : Le premier prend en considération tous les amers visibles et il est surtout adapté à un environnement contenant un nombre faible d'amers. Les deux autres algorithmes peuvent être utilisés dans des environnements contenant un nombre conséquent d'amers et où tous les amers visibles ne sont pas forcément nécessaires à l'asservissement de la trajectoire. Ces algorithmes définissent ainsi les séquences d'enchaînement de couples capteurs-amers tout au long d'une trajectoire $\gamma(u)$ sans collision.

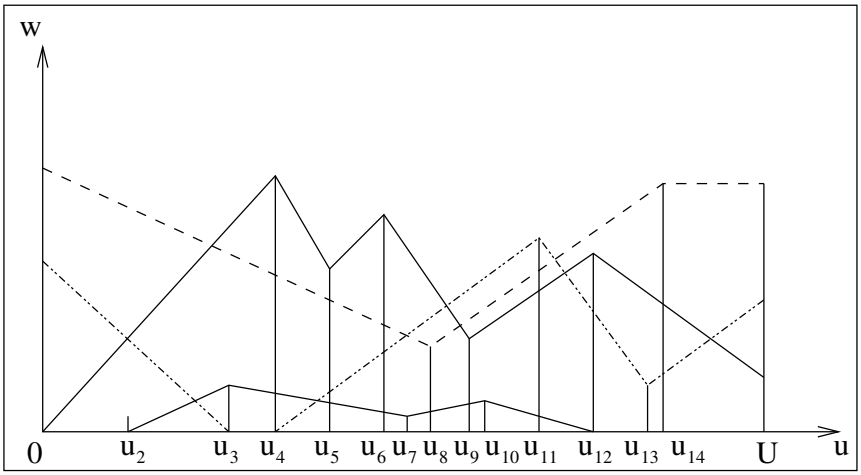

FIG. IV.4 – Approximation linéaire des fonctions de pondération.

IV.5.1 Algorithme classique : "Take All"

Le premier algorithme qu'on propose est très simple. Il produit un mouvement asservi sur amers qui prend en compte tous les amers visibles au cours du chemin géométrique planifié. Il est surtout adapté aux environnements ayant un faible nombre d'amers. Il admet principalement en entrée un ensemble d'amers constituant l'environnement en question $\mathscr{L}_{env}$, l'ensemble des capteurs du robot $\mathscr{S}$ et un chemin géométrique planifié sans collision $\gamma(u)$, $u \in [0,U]$.

À l'initialisation, le robot est à la configuration de départ $\gamma(u_1 = 0)$. Pour un capteur $S_i \in \mathscr{S}$ on détermine, $\mathscr{L}^i_{select}$, l'ensemble des amers visible à la configuration $\gamma(u_1 = 0)$. Cet ensemble $\mathscr{L}^i_{select}$ constitue alors la première séquence locale d'amers associée à la séquence d'enchaînement des amers maa_S_i de ce capteur. Cette séquence s'étend sur un intervalle $[u_1,u_2]$ durant lequel les poids associés aux amers de $\mathscr{L}^i_{select}$ visibles depuis S_i gardent la même monotonie. Aussi, durant cet intervalle seuls les amers de $\mathscr{L}^i_{select}$ sont visibles depuis S_i. Tenant compte des ces contraintes, la fonction Max_Dist() retourne la longueur maximale *distParam* de l'intervalle $[u_1,u_2]$. Ce calcul de u_2 correspond à un changement de monotonie de l'une des fonctions de pondération capteur-amer. Afin de définir entièrement cette première séquence locale, on calcule pour chaque paire issue de l'association du capteur S_i aux amers de $\mathscr{L}^i_{select}$, les valeurs des poids aux points u_1 et u_2. En mettant $u_1 = u_2$ et en répétant cette procédure, on progresse sur le chemin planifié pour trouver la suite des séquences locales d'amers attribuées au capteur S_i jusqu'à la fin du chemin géométrique γ. Les mêmes étapes sont alors répétées pour l'ensemble des n capteurs du robot.

Dans le cas de notre exemple (figure (IV.2)), à la configuration initiale $\gamma(u_1 = 0)$ l'ensemble des amers visibles $\mathscr{L}_{select} = \{L_1,L_2,L_3\}$ constitue la première séquence locale d'amers associée au capteur S. Lors de cette première itération, la fonction Max_Dist() retourne ($distParam = u_2 - u_1$) car en (u_2) apparaît l'amer L_4 dans le champ de perception du capteur. On complète

alors la définition de cette première séquence en calculant les valeurs $w_i(u_1)$ et $w_i(u_2)$ pour les amers L_i, $i = \overline{1,3}$. En sortie l'algorithme retourne une séquence *maa_S* de 14 séquences locales d'amers parmi lesquelles à titre d'exemple : $\{L_1, L_2, L_3, L_4\}$ sur $[u_2, u_3]$, $\{L_1, L_3, L_4\}$ sur $[u_3, u_4]$ qui diffère de la précédente par la disparition de L_2 du champs de perception de S et le changement de monotonie de L_4. Enfin la séquence $\{L_1, L_2, L_3\}$ où on retrouve L_2 qui est réapparu dans le champs de perception de S au milieu du chemin ($u = u_4$) et qui est resté visible jusqu'à la fin du chemin. Les fonctions de pondération considérées dans le mouvement asservi sur amers obtenu sont construites à partir de l'interpolation linéaire des extrema des fonctions de pondération réelles (figure (IV.4)).

Notons que la particularité de cet algorithme réside dans le fait que deux séquences locales d'amers successives définies sur des intervalles successifs $[u_i, u_{i+1}]$ et $[u_{i+1}, u_{i+2}]$ associées à un même capteur S_i possèdent au moins l'une des propriétés suivantes :

– Si les deux séquences contiennent les mêmes amers, alors au moins une fonction de pondération a deux monotonies différentes durant $[u_i, u_{i+1}]$ et $[u_{i+1}, u_{i+2}]$ (e.g. L_3 dans $[u_4, u_5]$ et $[u_5, u_6]$).

– La séquence locale d'amers définie sur $[u_{i+1}, u_{i+2}]$ a au moins un amer qui était visible durant $[u_i, u_{i+1}]$ et qui ne l'est plus.

– Il y a au moins un nouvel amer visible qui apparaît sur l'intervalle $[u_{i+1}, u_{i+2}]$ (e.g. L_4 dans $[u_2, u_3]$ par rapport à $[u_1, u_2]$).

Cet algorithme a un temps de calcul en $(n \times \kappa)$. Si le nombre d'amers du plan de référence est important alors cet algorithme devient très coûteux en terme de temps de calcul. L'algorithme suivant présente une complexité algorithmique moindre que celui étudié dans ce paragraphe car il sélectionne les meilleurs amers visibles de l'environnement.

Données : $\mathscr{L}_{env}$, $\mathscr{S}$, $\gamma(u)$
Résultat : *maa*
début
  pour $i \leftarrow 1$ ***à*** n **faire**
    $u_1 \leftarrow 0$;
    $maa\_S_i \leftarrow Null$;
    tant que $u_1 \leq U$ **faire**
      $\mathscr{L}^i_{select} \leftarrow$ Amers_Visibles$(\mathscr{L}_{env}, S_i, \gamma(u), u)$;
      $distParam \leftarrow$ Max_Dist$(\mathscr{L}_{env}, \mathscr{L}^i_{select}, S_i, \gamma(u), u)$;
      $u_2 \leftarrow u_1 + distParam$;
      pour $L \in \mathscr{L}^i_{select}$ **faire**
        $w_{i_L}(u_1) \leftarrow$Calcul_Poids(S_i, L, u_1);
        $w_{i_L}(u_2) \leftarrow$Calcul_Poids(S_i, L, u_2);
      $maa\_S_i \leftarrow$ Ajouter_Séquence_Locale$(\mathscr{L}^i_{select}, w_i(u_1), w_i(u_2), u_1, u_2)$;
      $u_1 \leftarrow u_2$;
    $maa \leftarrow$Ajouter_*maa*_capteur(maa_S_i);
fin

Algorithme 1 – Take all

IV.5.2 Algorithme de sélection des meilleurs amers

Le second algorithme qu'on propose produit un mouvement asservi sur amers qui prend en compte les meilleurs amers visibles par rapport à leurs poids au cours du chemin géométrique planifié [58]. Il est surtout adapté aux environnements ayant un grand nombre d'amers et où tous les amers visibles ne sont pas forcément intéressants pour la localisation. On prendra les meilleurs amers en fonction de la grandeur de leur poids. En effet, d'après l'équation de localisation (III.5) nous savons que ce sont les amers de poids prépondérant qui ont le plus d'influence sur le calcul de la localisation $\mathbf{q}_{loc}$. Il est alors inutile de prendre tous les amers, les NB_L meilleurs suffisent à calculer une bonne estimée de cette configuration. Les entrées principales de l'algorithme sont un ensemble d'amers associés à l'environnement $\mathscr{L}_{env}$, l'ensemble des capteurs du robot $\mathscr{S}$, un chemin géométrique planifié $\gamma(u)$, $u \in [0,U]$ et un entier NB_L, choisi a priori, représentant le nombre d'amers de plus grandes valeurs retenus pour la localisation.

À l'initialisation le robot est à la configuration de départ $\gamma(u_1 = 0)$. Pour un capteur $S_i \in \mathscr{S}$ on détermine, $\mathscr{L}^i_{select}$ ($Cardinal\{\mathscr{L}^i_{select}\} \leq NB_L$), l'ensemble des amers visibles à la configuration $\gamma(u_1 = 0)$ qui ont les NB_L plus grandes valeurs de poids associés à S_i. Si le nombre d'amers visibles de cette configuration est inférieur à NB_L alors tous les amers visibles sont sélectionnés. Cet ensemble $\mathscr{L}^i_{select}$ constitue la première séquence locale d'amers attribuée à la séquence d'enchaînement des amers maa_S_i de ce capteur. Cette séquence locale s'étend sur un intervalle $[u_1, u_2]$ durant lequel les poids associés aux amers de $\mathscr{L}^i_{select}$ visible depuis S_i gardent

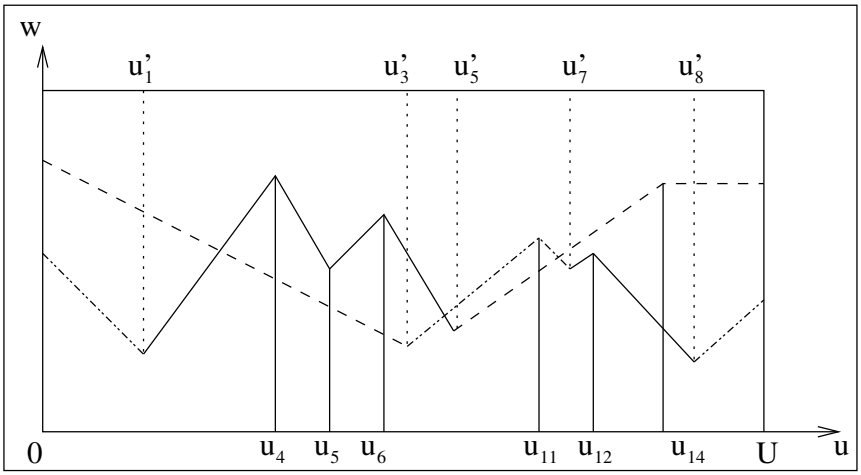

FIG. IV.5 –

la même monotonie. Aussi, durant cet intervalle seules les amers de $\mathscr{L}^i_{select}$ ont les poids les plus grands parmi les amers visibles. Tenant compte des ces contraintes, la fonction Max_Dist() retourne la longueur maximale *distParam* de l'intervalle $[u_1, u_2]$. Afin de définir entièrement cette première séquence locale, on calcule pour chaque paire issue de l'association du capteur S_i aux amers de $\mathscr{L}^i_{select}$, les valeurs des poids aux points u_1 et u_2. Une fois cette séquence définie on l'ajoute aux séquences d'enchaînement maa_S_i du capteur S_i. En mettant $u_1 = u_2$ et en répétant cette procédure, on progresse ainsi sur le chemin planifié pour trouver la suite des séquences locales d'amers attribuées au capteur S_i et ce jusqu'à la fin du chemin géométrique γ. Au final on obtient une séquence d'enchaînement maa_S_i qui contient une successions de séquences locales d'amers sur $[0, U]$ où chacune est composée au maximum de NB_L amers.

Dans l'exemple précédent (figure (IV.2)), on suppose que l'on veut sélectionner les $NB_L = 2$ meilleurs amers durant le chemin planifié. À la configuration initiale $\gamma(u_1 = 0)$ l'ensemble des deux meilleurs amers visibles $\mathscr{L}_{select} = \{L_1, L_2\}$ est retourné par la fonction Meilleurs_Amers(). Cette séquence constitue la première séquence locale d'amers attribuée à la séquence d'enchaînement des amers *maa_S*. Dans cette première itération, la fonction Max_Dist() retourne ($distParam = u'_1 - u_1$) et non pas ($distParam = u_2 - u_1$) comme dans le cas de l'algorithme précédent car même si L_4 apparait en u_2 il n'est pas pris en compte car son poids est moins important que celui des amers actuels de $\mathscr{L}_{select}$. Au point u'_1 le poids de L_3 devient plus important que celui de L_2. On complète alors la définition de cette première séquence en calculant les valeurs $w_i(u_1)$ et $w_i(u'_1)$ pour les amers L_i, $i = \overline{1,2}$. En progressant ainsi, on construit une première séquence d'enchaînement d'amers telle que celle illustrée en figure (IV.5). On obtient à ce stade une *maa_S* de 12 séquences locales d'amers parmi lesquelles à titre d'exemple : $\mathscr{L}_{select} = \{L_1, L_3\}$ sur $[u'_1, u_4]$ et $\mathscr{L}_{select} = \{L_1, L_2\}$ sur $[u'_5, u_{11}]$.

Arrivé à ce point de l'algorithme, deux séquences locales d'amers successives définies sur

des intervalles successifs $[u_i, u_{i+1}]$ et $[u_{i+1}, u_{i+2}]$ et associées à un même capteur S_i possèdent une des propriétés suivantes :

– Si les deux séquences contiennent les mêmes amers, alors au moins une fonction de pondération a deux monotonies différentes durant $[u_i, u_{i+1}]$ et $[u_{i+1}, u_{i+2}]$, c'est à dire que u_{i+1} représente un optimum locale pour cette fonction (e.g. L_3 dans les intervalles $[u_4, u_5]$ et $[u_5, u_6]$).

– La séquence locale d'amers définie sur $[u_{i+1}, u_{i+2}]$ a au moins un amer qui avait un poids faisant parti des NB_L plus grand poids des amers visibles sur $[u_i, u_{i+1}]$ et qui ne l'est plus (e.g. L_2 disparaît dans $[u'_1, u_4]$ alors qu'il est présent dans $[u_1, u'_1]$).

– Il y a au moins un nouvel amer visible dont le poids est devenu significatif sur l'intervalle $[u_{i+1}, u_{i+2}]$ par rapport aux amers de la séquence produite sur $[u_i, u_{i+1}]$ (e.g. L_3 apparaît dans $[u'_1, u_4]$).

L'implémentation et le test de cet algorithme a donné de bons résultats dans des cas simples mais présente des discontinuités de localisation lors du changement de séquences locales d'amers. Même si dans ces séquences d'amers on prend à chaque fois les meilleurs amers visibles, l'inconvenient d'un tel enchaînement est que même si certains amers restent visibles il peuvent disparaître d'une séquence locale d'amers à la suivante parce que leurs poids sont devenus moins importants et que le nombre d'amers par séquence est fixe. De même, certains amers bien que visibles depuis un certain moment apparaîssent brutalement dans une séquence parce que leurs poids sont devenus suffisament grands. Pour des raisons de contraintes sur la localisation évoqués aux paragraphe § IV.4.6, ces apparitions et disparitions brutales des amers peuvent causer de grands écarts dans le résultat de la localisation issue de l'utilisation de deux séquences consécutives. Pour cela la fonction Etendre_Amers() permet pour chaque séquence locale d'amers d'étendre les amers qu'elle contient aux séquences précédentes et suivantes afin de rendre plus lisse le passage d'une séquence à une autre. Finalement, pour construire entièrement notre mouvement asservi sur amers ces mêmes étapes sont alors répétées pour l'ensemble des n capteurs du robot.

Dans l'exemple considéré, le résultat obtenu (voir figure (IV.6)) est similaire à celui obtenu par l'algorithme précédent, sauf que dans ce cas on ne prend jamais en compte l'amer L_4 car son poids n'est jamais classé parmi les deux plus grands poids des amers de l'environnement.

Cet algorithme a un temps de calcul en $(n \times NB_L)$. Même si cet algorithme a une complexité algorithmique inférieure à celle du précédent à condition bien sûr de prendre un NB_L inférieur à κ, nombre d'amers du plan de référence, il est très difficile de fixer le paramètre NB_L pour prendre tous les amers pertinents lors de l'exécution du mouvement. Un des cas critiques est illustré en figure (IV.7). En effet si on considère par rapport à l'exemple précédent un cinquième amers L_5 qui à un certain moment a un poids de même ordre que les amers pertinents par exemple sur $[u', u'']$. Comme il n'est jamais parmi les deux meilleurs amers et si on choisit $NB_L = 2$, alors l'algorithme de sélection ne le prend pas en considération dans le calcul de la localisation $\mathbf{q}_{loc}$ même s'il présente un risque de collision non négligeable. Pour les amers présentant un danger de collision dans des environnements contraints et qui risquent de ne pas être sélectionnés par cet algorithme, nous proposons de sélectionner les amers au voisinage de $\gamma(u)$.

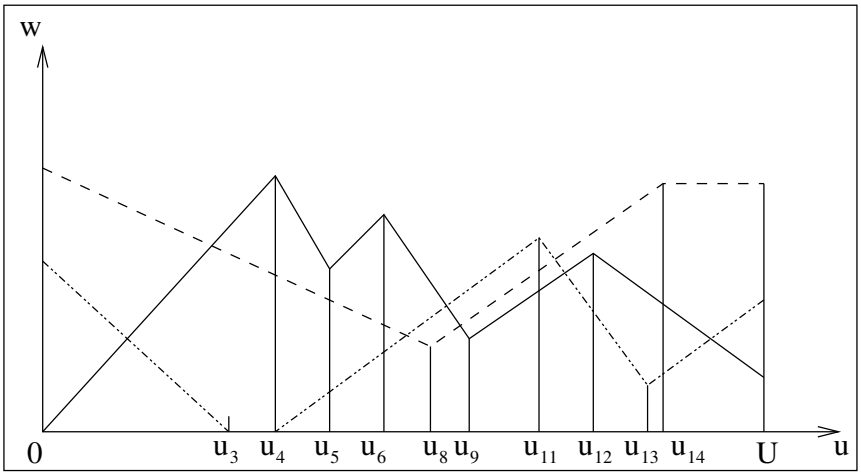

FIG. IV.6 –

Données : $\mathscr{L}_{env}$, $\mathscr{S}$, $\gamma(u)$, NB_L
Résultat : *maa*
début
  pour $i \leftarrow 1$ ***à*** n **faire**
    $u_1 \leftarrow 0$;
    $maa\_S_i \leftarrow Null$;
    tant que $u_1 \leq U$ **faire**
      $\mathscr{L}^i_{select}, Cardinal\{\mathscr{L}^i_{select}\} \leq NB_L \leftarrow$ Meilleurs_Amers$(\mathscr{L}_{env}, S_i, \gamma(u), u, NB_L)$;
      $distParam \leftarrow$ Max_Dist$(\mathscr{L}_{env}, \mathscr{L}^i_{select}, S_i, \gamma(u), u)$;
      $u_2 \leftarrow u_1 + distParam$;
      pour $L \in \mathscr{L}^i_{select}$ **faire**
        $w_{i_L}(u_1) \leftarrow$Calcul_Poids(S_i, L, u_1);
        $w_{i_L}(u_2) \leftarrow$Calcul_Poids(S_i, L, u_2);
      $maa\_S_i \leftarrow$ Ajouter_Séquence_Locale$(\mathscr{L}^i_{select}, S_i, \gamma(u), u_1, u_2)$;
      $u_1 \leftarrow u_2$;
    $maa\_S_i \leftarrow$ Etendre_Amers(maa_S_i);
    $maa \leftarrow$Ajouter_*maa*_capteur(maa_S_i);
fin

Algorithme 2 – Algorithme nombre d'amers maximum

IV.5.3 Algorithme des plus proches amers le long de la trajectoire

Le troisième algorithme proposé produit un mouvement asservi sur amers qui prend en compte les amers visibles situés à une distance inférieure à une distance maximale par rap-

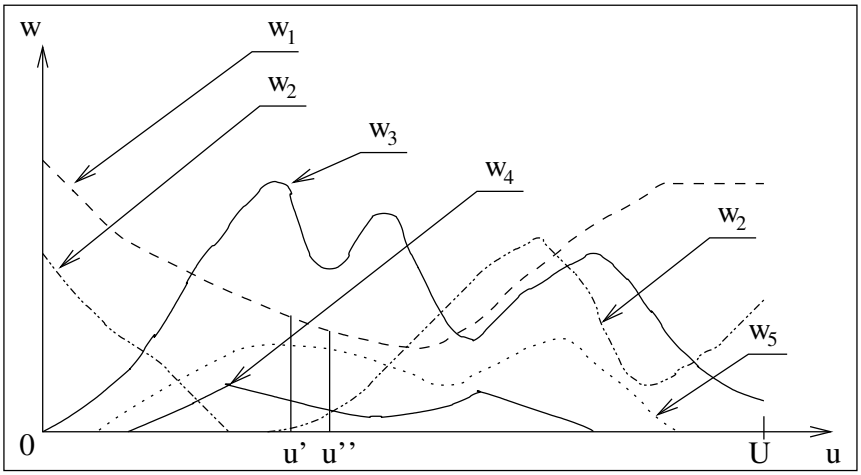

FIG. IV.7 –

port au robot quand celui-ci parcourt le chemin géométrique planifié. Cet algorithme est surtout adapté aux environnements ayant un grand nombre d'amers puisque dès le départ il permet de faire un filtrage sur les amers de l'environnement. N'ayant pas un nombre d'amers imposé par séquence locale d'amers, les amers choisis sont essentiellement liés au chemin géométrique planifié. L'algorithme admet principalement en entrée un ensemble d'amers constituant l'environnement en question $\mathscr{L}_{env}$, l'ensemble des capteurs du robot $\mathscr{S}$, un chemin géométrique planifié $\gamma(u)$, $u \in [0,U]$ et une distance ρ. Cette distance utilisée comme argument par la fonction Amers_proches() sert à sélectionner parmi les amers de $\mathscr{L}_{env}$ ceux qui vont être à une distance inférieure ou égale à ρ par rapport au robot quand celui-ci parcourt le chemin $\gamma(u)$. L'ensemble d'amers obtenu est noté $\mathscr{L}_{proche}$.

Le déroulement de l'algorithme est similaire à celui de l'algorithme présenté au paragraphe § IV.5.1 où les amers de l'environnement concernés sont limités à l'ensemble $\mathscr{L}_{proche}$. Pour un capteur $S_i \in \mathscr{S}$ on détermine, $\mathscr{L}^i_{select} \subset \mathscr{L}_{proche}$, l'ensemble des amers visibles à la configuration $\gamma(u_1 = 0)$. Cet ensemble $\mathscr{L}^i_{select}$ constitue la première séquence locale d'amers attribuée à la séquence d'enchaînement des amers maa_S_i de ce capteur. Cette séquence locale s'étend sur un intervalle $[u_1,u_2]$ durant lequel les poids associés aux amers de $\mathscr{L}^i_{select}$ visibles depuis S_i gardent la même monotonie. Durant cet intervalle, parmi les amers de $\mathscr{L}_{proche}$, seuls les amers de $\mathscr{L}^i_{select}$ sont visibles depuis S_i. Tenant compte de ces contraintes, la fonction Max_Dist() retourne la longueur maximale *distParam* de l'intervalle $[u_1,u_2]$. Afin de définir entièrement cette première séquence locale on calcule pour chaque paire issue de l'association du capteur S_i aux amers de $\mathscr{L}^i_{select}$, les valeurs des poids aux points u_1 et u_2. Une fois cette séquence complètement définie, on l'ajoute aux séquences d'enchaînement maa_S_i du capteur S_i. En remplaçant u_1 par u_2 et en répétant la procédure précédente on progresse sur le chemin planifié pour trouver la suite des séquences locales d'amers attribuées au capteur S_i jusqu'à la fin du chemin géométrique γ. Les mêmes étapes sont ainsi répétées pour l'ensemble des n capteurs du robot. En ne contenant que

des amers appartenant à $\mathscr{L}_{proche}$, les séquences locales d'amers produites partagent les mêmes propriétés que celles déduites pour les séquences locales d'amers présentées en § IV.5.1. Cet algorithme a un temps de calcul en $(n \times \alpha)$.

Données : $\mathscr{L}_{env}$, $\mathscr{S}$, $\gamma(u)$, ρ
Résultat : *maa*
début
  $\mathscr{L}_{proche} = \{L_{\lambda(1)}, \ldots, L_{\lambda(\alpha)}\} \leftarrow$Amers_Proches($\mathscr{L}_{env}$, $\gamma(u)$, ρ);
  pour $i \leftarrow 1$ ***à*** n **faire**
    $u \leftarrow 0$;
    $maa\_S_i \leftarrow Null$;
    tant que $u \leq U$ **faire**
      $\mathscr{L}_{select} \leftarrow$ Amers_Visibles($\mathscr{L}_{proche}, S_i, \gamma(u), u$);
      $distParam \leftarrow$ Max_Dist($\mathscr{L}_{proche}, S_i, \gamma(u), u$);
      $u_2 \leftarrow u_1 + distParam$;
      pour $L \in \mathscr{L}_{select}$ **faire**
        $w_{i_L}(u_1) \leftarrow$Calcul_Poids(S_i, L, u_1);
        $w_{i_L}(u_2) \leftarrow$Calcul_Poids(S_i, L, u_2);
      $maa\_S_i \leftarrow$ Ajouter_Séquence_Locale($\mathscr{L}^i_{select}, S_i, \gamma(u), u_1, u_2$);
      $u_1 \leftarrow u_2$
    $maa \leftarrow$Ajouter_*maa*_capteur(maa_S_i)
fin

Algorithme 3 – Algorithme des plus proches amers le long de la trajectoire

Dans l'exemple de la figure (IV.7), l'algorithme fait d'abord un filtrage sur les amers qui sont situés à une distance inférieure à ρ du robot, le long du chemin planifié. Considérons que les amers gardés sont alors $\mathscr{L}_{proche} = \{L_1, L_2, L_3, L_5\}$. Un processus similaire à celui utilisé dans le premier algorithme est alors effectué sur cet ensemble pour construire les fonctions de pondération associées aux amers pour la trajectoire actuelle (voir résultat figure (IV.8)). Bien que le choix de la distance ρ soit un problème critique pour cet algorithme, il reste moins problématique que la détermination des nombres d'amers pour l'algorithme précédent.

Les trois algorithmes décrits ci-dessus permettent d'obtenir des séquences locales d'amers contenant des amers choisis selon certains critères. Le premier algorithme prend en compte tous les amers visibles. Il est donc peu intéressant dans le cas d'un environnement contenant un grand nombre d'amers étant donné qu'il n'introduit aucun critère de sélection. Ainsi le nombre de couple capteur-amer sera d'autant plus grand que le chemin $\gamma(u)$ sera grand. Les deux derniers algorithmes sont par contre mieux adaptés pour ce type d'environnement, prenant les amers les plus pertinents qui peuvent représenter un danger de collision ou du moins qui sont à une certaine distance de la trajectoire. Ces deux algorithmes peuvent être combinés pour produire un algorithme, privilégiant à la fois les amers importants au vu de leurs poids pour une situation

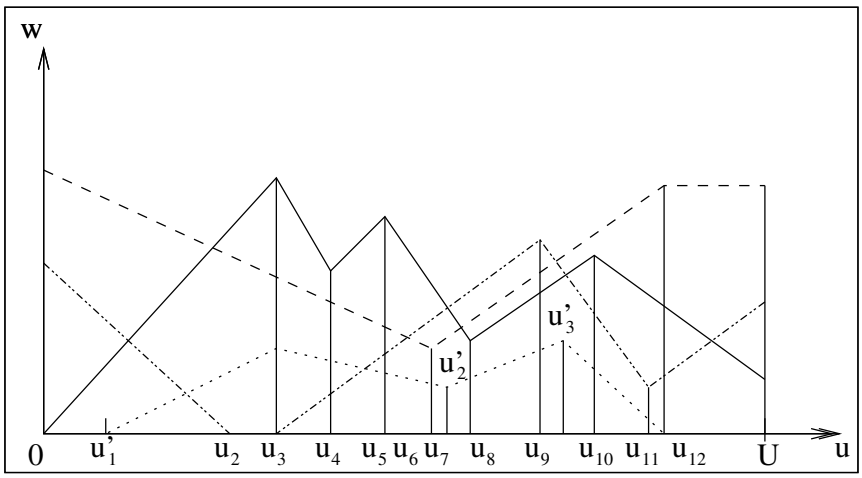

FIG. IV.8 –

donné et les amers qui sont à une certaine distance du robot durant le chemin planifié. On peut ainsi imaginer d'autres combinaisons pour prendre à chaque fois les meilleurs amers du plan de référence selon certains critères de choix. Dans la perspective d'exécuter le mouvement asservi sur amers dans un environnement réel, d'autres questions apparaissent telles que la mise en correspondance entre les amers sélectionnés par ces algorithmes et les amers perçus par les capteurs du robot dans l'environnement réel.

Dans le paragraphe suivant nous allons présenter quelques résultats de simulation obtenus avec *l'algorithme de sélection des meilleurs amers*. Les deux autres algorithmes seront testés durant la partie expérimentale dans le chapitre suivant.

IV.6 Résultats de simulation

Les algorithmes présentés ci-dessus sont intégrés au sein d'une plate-forme software orientée objet que nous avons développé et que nous détaillerons dans le chapitre suivant. Avant de les intégrer dans une plate-forme réelle nous avons développé un environnement de simulation permettant de valider notre approche. Les résultats de simulation présentés dans cette section ont pour objectif d'étudier l'effet du nombre d'amers sélectionnés en utilisant *l'algorithme de sélection des meilleurs amers* (c.f. § IV.5.2). Les simulations sont appliquées au robot mobile Hilare2 avec sa remorque (voir figure (II.6)). Cette structure robotique dispose de deux scanners laser : Le premier est monté à l'avant du véhicule et le second est monté sur la remorque. Pour la simulation des perception des scanners lasers, nous utilisons une librairie développée dans notre laboratoire de manière à avoir une simulation de la fonction de perception qui utilise les même fonctionnalités que la fonction de perception réelle. Cette librairie dispose aussi de fonctionnalité d'appariement que nous utilisons pour apparier les mesures simulées aux mesures de référence. L'environnement de référence a été construit au préalable par l'intermédiaire d'un scanner laser

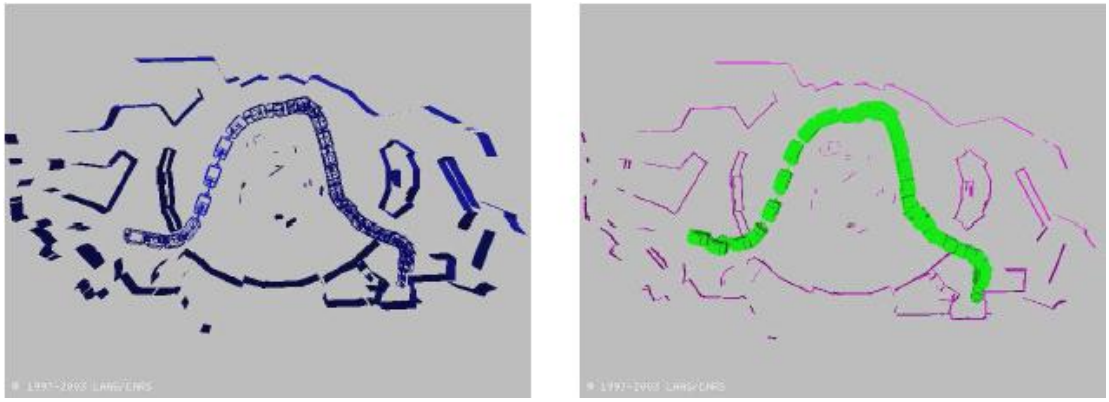

FIG. IV.9 – Figure de gauche, le plan de référence avec une trajectoire planifiée. Figure de droite, environnement réel simulé différent du plan de référence. Exécution de la trajectoire avec un mouvement asservi sur amers construit avec les $NB_L = 3$ meilleurs amers.

dans l'environnement réel[1]. Le plan obtenu est illustré en figure (IV.9) de gauche. Dans ce plan de référence, une trajectoire sans collision est planifiée d'une configuration initiale à une configuration finale à l'aide du planificateur de chemin géométrique **Move3D** [82] développé au sein de notre groupe de recherche. En sélectionnant pour chaque capteur un nombre d'amers maximum $NB_L = 3$ on construit un mouvement asservi sur amers.

À présent nous introduisons dans le plan de référence des perturbations en translation et en rotation pour simuler un environnement réel différent du modèle, voir figure (IV.10). L'exécution de la trajectoire planifiée dans l'environnement réel simulé n'est pas correcte (voir figure (IV.10)) puisqu'elle rentre en collision lorsque le robot entre par la porte de gauche. De plus puisque l'environnement réel simulé est différent du modèle, les perceptions de référence ne sont pas égales aux perceptions réelles simulées. L'exécution du mouvement asservi sur amers construit avec les trois meilleurs amers pour chaque capteur permet d'asservir la trajectoire en fonction des perturbations de l'environnement réel simulé (voir trajectoire en vert dans figure (IV.10)). Durant cette étape d'exécution, les perceptions des capteurs sont simulées par le modèle du scanner laser dont nous disposons. La comparaison entre la trajectoire initiale (figure (IV.9) de gauche) et la trajectoire exécutée en simulation avec un mouvement asservi sur amers (figure (IV.9) de droite) montre que le robot s'adapte parfaitement aux perturbations introduites dans l'environnement en modifiant sa trajectoire.

La figure (IV.11) montre un agrandissement où on peut remarquer la configuration de référence en bleu et la configuration corrigée correspondante en vert. Dans cette configuration, on remarque que le robot s'asservit sur trois amers pour le capteur avant :L_{136}, L_{135} et L_{43} et trois amers pour le capteur arrière : L_{163}, L_{197} et L_{65}. Les valeurs des poids pour cette configuration sont représentées en figure (IV.12). La figure (IV.13) illustre les poids des paires capteurs-amers construits pour les segments L_{136}, L_{43}, L_{76}, L_{213} lorsqu'ils sont perçus par le capteur laser avant.

[1]Le modèle construit représente «la cité de l'espace» à Toulouse (France).

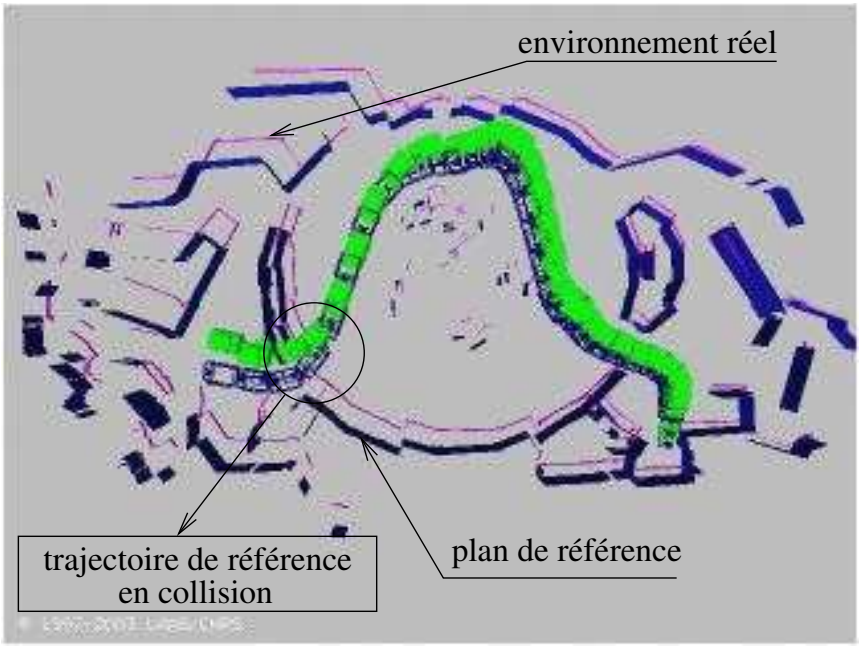

FIG. IV.10 – Le plan de référence et l'environnement réel simulé sont superposés. La trajectoire planifiée est en collision avec le bord supérieure de la porte de gauche. La trajectoire exécutée avec un mouvement asservi sur amers n'est pas en collision et respecte les perceptions de référence.

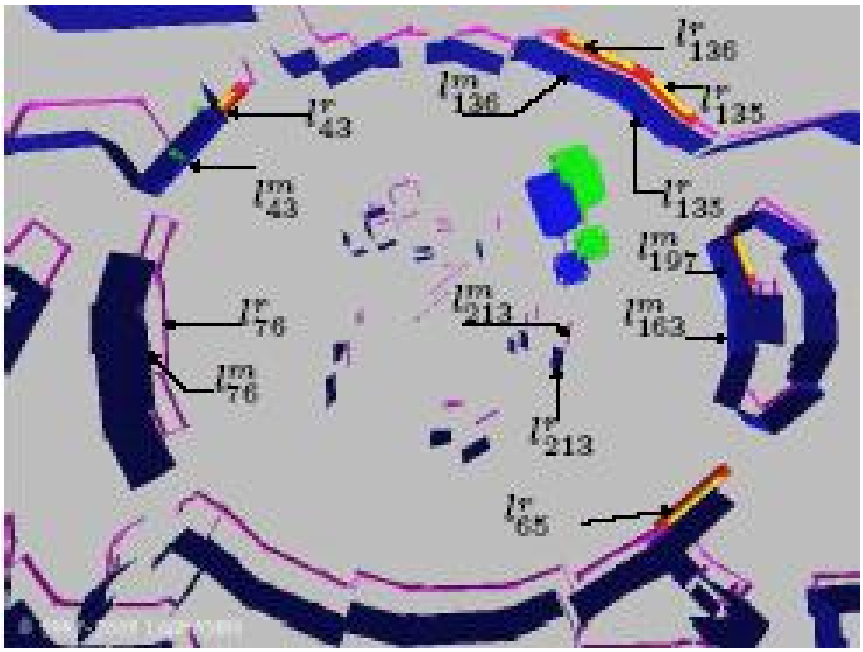

FIG. IV.11 – Agrandissement des plans superposés. Le robot en vert est à la configuration corrigée et le robot en bleu est à la configuration de référence. Les amers en rouge avec des pastilles jaunes représentent les segments sélectionnés que le robot a réussi à apparier avec les segments de l'environnement réel simulé.

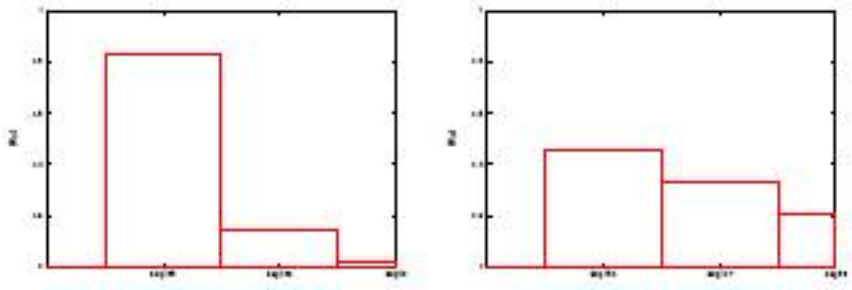

FIG. IV.12 – Poids des segments pour la configuration illustrée en figure (IV.11). La figure de gauche correspond aux segments perçus par le capteur avant et la figure de droite correspond aux segments perçus par le capteur arrière.

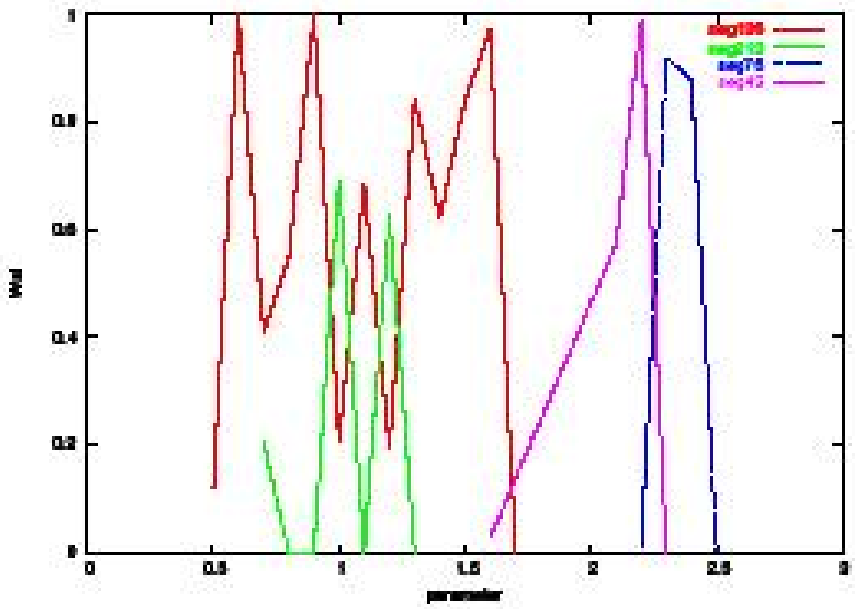

FIG. IV.13 – Poids associés aux amers sélectionnés pour le capteur avant. On observe bien les sauts des fonctions de poids le long de la trajectoire.

Nous avons par ailleurs construit des mouvements asservis sur amers en variant le nombre NB_L des meilleurs amers choisis. La figure (IV.14) montre que pour un nombre $NB_L > 3$, on obtient le même résultat qu'avec un nombre d'amers $NB_L = 3$. Cependant pour $NB_L = 2$ on trouve des sauts dans la localisation si l'environnement est fortement perturbé. Dans ces simulations, nous n'avons pas pris en compte l'extension des meilleurs amers à travers le chemin planifié ce qui explique les sauts dans les fonctions de pondération (figure (IV.13)). On constate sur cet exemple que prendre tous les couples capteurs-amers ne sert à rien lors de la phase d'exécution de la trajectoire. Par contre choisir un nombre d'amers minimum permet de prendre les amers les plus pertinents le long de la trajectoire. Une modélisation fortement entachée d'erreurs induit automatiquement un nombre d'amers minimum NB_L supérieur au nombre minimum nécessaire pour calculer l'erreur de localisation et ainsi corriger la trajectoire du robot. Cette condition rejoint la condition d'existence et d'unicité de la configuration corrigée établie au § III.6. Au cours du chapitre suivant nous verrons comment améliorer les algorithmes par rapport à ce type de problème.

IV.7 Conclusion

Les fonctions de pondérations sont définies comme le produit de deux sous fonctions : une sous fonction qui prend en compte la visibilité des amers et une seconde qui prend en compte le danger de collision que représente cet amer par rapport à la configuration du robot. Ces fonctions doivent être continues, positives, nulles en dehors de l'espace de perception du capteur et doivent

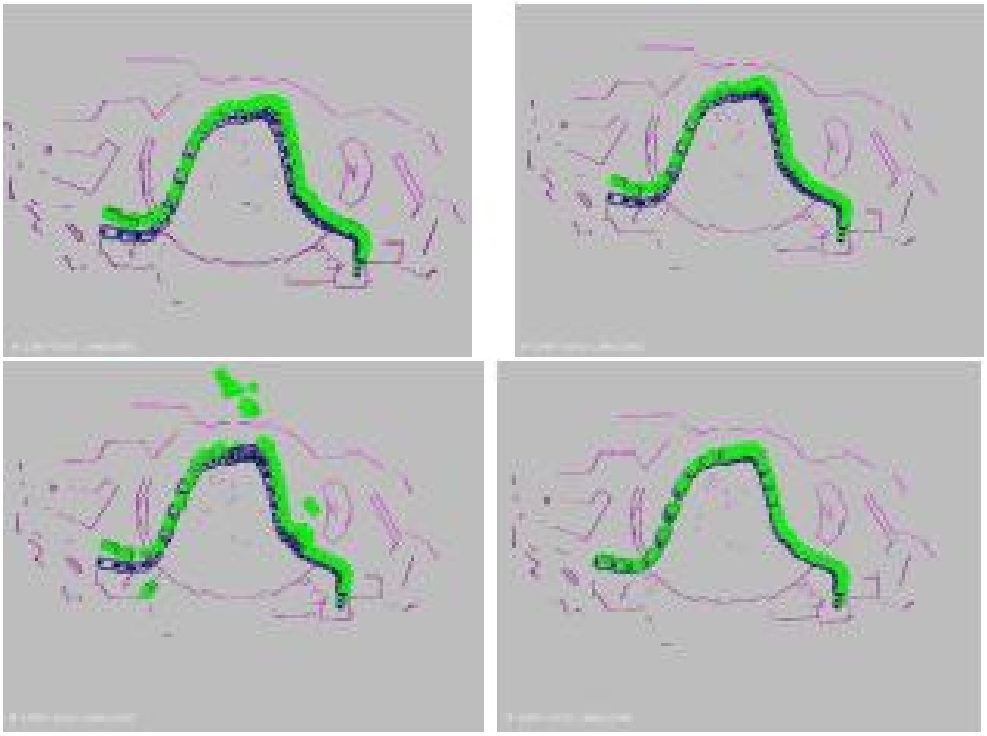

FIG. IV.14 – En haut à gauche $NB_L = 4$, en haut à droite $NB_L = 5$, en bas à gauche $NB_L = 2$ et en bas à droite $NB_L = 2$ avec un environnement faiblement perturbé.

varier lentement afin d'éviter des variations brutales dans l'erreur de localisation entre deux états discrets de la boucle de commande le long de la trajectoire.

Nous avons proposé trois algorithmes : le premier permet essentiellement de construire les fonctions de pondération pour une trajectoire donné. Les deux derniers permettent de poser des critères de sélection pour les amers les plus pertinents. En fin de chapitre des tests de simulation ont permis de montrer d'une part l'intérêt de notre approche et d'autre part le problème de discontinuité de l'erreur de localisation due au saut des valeurs des fonctions de pondération et au mauvais conditionnement de la matrice de localisation pondérée (condition d'existence de la configuration corrigée).

Chapitre V

Implémentation sur Hilare 2 avec remorque

V.1 Introduction

Ce chapitre présente l'implémentation du formalisme du mouvement asservi sur amers à bord du robot Hilare2 avec sa remorque. Dans un premier temps nous allons présenter la modélisation utilisée pour les capteurs, les amers et le calcul des poids. Cette modélisation sera intégrée dans les fonctionnalités bas niveau de notre software dont l'architecture haut niveau définit un cadre générique de "planificateur" de mouvement asservi sur amers pouvant être appliqué à tout type de robot, tout type de capteur et tout type d'amer. Nous présentons également l'architecture de contrôle du robot utilisant le mouvement asservi sur amers issu du planificateur. Finalement nous montrons des résultats expérimentaux réalisés avec notre robot. Il est à noter que ce système est non-holonome et ne peut donc pas être asservi suivant la loi proportionelle définie par l'équation (III.8). Cependant, les résultats expérimentaux montrent que le robot converge toujours vers sa trajectoire de référence.

V.2 Modélisation

V.2.1 Capteurs

Le robot Hilare2 tractant une remorque est équipé de deux scanners laser S_1 et S_2, le premier monté sur le robot mobile et le second monté sur la remorque (voir figure II.6). La configuration des capteurs varie dans un plan horizontal : $\mathbb{S}_1 = \mathbb{S}_2 = SE(2)$. La figure V.1 décrit la géométrie

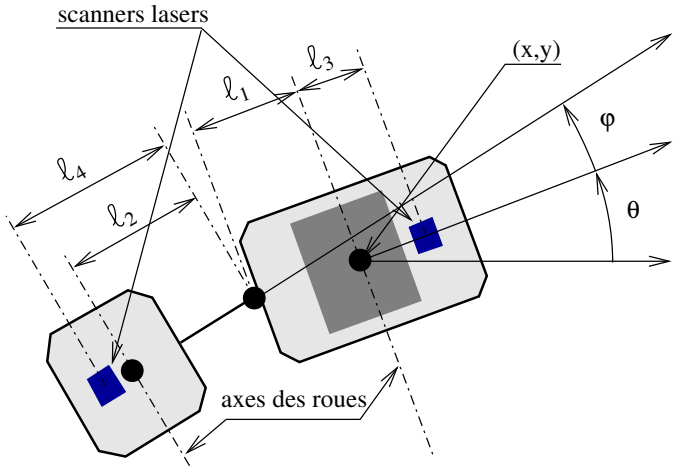

FIG. V.1 – Le robot mobile Hilare2 avec sa remorque équipé de deux scanners lasers. La configuration du robot est paramétrée par (x,y,θ,φ), où (x,y) est le centre, θ l'orientation du robot et φ est l'angle de la remorque par rapport à l'axe de la base. ℓ_1 est la distance entre l'axe du robot et le point de connection de la remorque, ℓ_2 est la distance entre l'axe des roues de la remorque et le point de connection de la remorque. La configuration de capteur S_1, monté sur le robot, est donnée par $s_1 = (x+\ell_3\cos\theta, y+\ell_3\sin\theta, \theta)$. La configuration du capteur S_2 monté sur la remorque est donnée par $s_2 = (x-\ell_1\cos\theta-\ell_4\cos(\theta+\varphi), y-\ell_1\sin\theta-\ell_4\sin(\theta+\varphi), \theta+\varphi+\pi)$. ℓ_3 et ℓ_4 sont des longueurs constantes.

du robot mobile. Soit $\mathscr{R} = (O,x,y,z)$ un repère global fixe. Les configurations des capteurs en fonction de la configuration $\mathbf{q} = (x,y,\theta,\varphi)$ du robot sont données par :

$$s_1(\mathbf{q}) = \begin{pmatrix} x+\ell_3\cos\theta \\ y+\ell_3\sin\theta \\ \theta \end{pmatrix} \tag{V.1}$$

$$s_2(\mathbf{q}) = \begin{pmatrix} x-\ell_1\cos\theta-\ell_4\cos(\theta+\varphi) \\ y-\ell_1\sin\theta-\ell_4\sin(\theta+\varphi) \\ \theta+\varphi+\pi \end{pmatrix} \tag{V.2}$$

V.2.2 Amers

On considère deux types d'amers : des plans verticaux et des lignes verticales. Les lignes verticales sont définies par l'intersection de deux plans verticaux. Ces amers sont projetés dans l'espace image d'un scanner laser respectivement comme des lignes droites et des points.

droite verticale

Une ligne droite L_1 est représentée par ses projections $l_1 = (l_1^1, l_1^2)$ sur (O,x,y). L'image d'une ligne verticale dans un scanner laser se trouvant à la configuration $s \in \mathbb{S} = SE(2)$ est l'image de

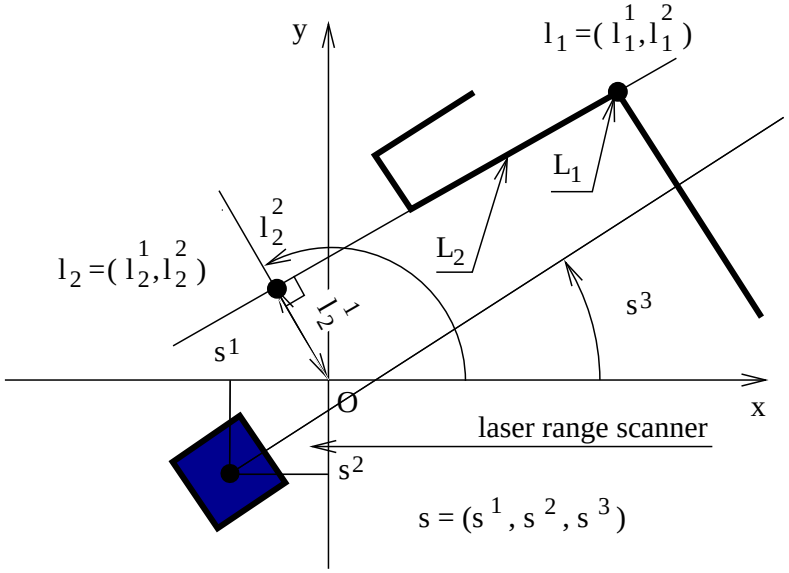

FIG. V.2 – Un scanner laser à la configuration $s = (s^1, s^2, s^3)$ percevant deux amers, avec (s^1, s^2) la position du centre et s^3 l'orientation du capteur dans le plan (O, x, y). L_1 est une ligne verticale représentée par ses projections $l_1 = (l_1^1, l_1^2)$ dans (O, x, y). L_2 est un plan vertical représenté par $l_2 = (l_2^1, l_2^2)$, les coordonnées polaires dans le plan (O, x, y) de la projection de l'origine O dans le plan.

l_1 par s^{-1} :

$$P_{S,L_1}(s, l_1) = \begin{pmatrix} \cos(s^3)\,(l_1^1 - s^1) + \sin(s^3)\,(l_1^2 - s^2) \\ -\sin(s^3)\,(l_1^1 - s^1) + \cos(s^3)\,(l_1^2 - s^2) \end{pmatrix}$$

Plan vertical

Nous représentons un plan vertical L_2 par les coordonnées cartésiennes $l_2 = (l_2^1, l_2^2)$ dans le plan (O, x, y) de la projection de l'origine O dans le plan, voir figure (III.1). L'équation de L_2 dans l'espace s'exprime alors par :

$$(x, y, z) \in L_2 \Leftrightarrow l^1 x + l^2 y = (l^1)^2 + (l^2)^2$$

On exclut du domaine de définition de la fonction de perception la singularité qui apparaît lorsque O appartient au plan vertical. L'image d'un plan vertical dans un scanner laser est une ligne droite dans $\mathbb{R}^2$ définie par les coordonnées cartésiennes $im_2 = (im_2^1, im_2^2)$ de la projection de l'origine du capteur sur la ligne droite. $s = (s^1, s^2, s^3)$ dénote respectivement la position et l'orientation du scanner laser. De manière équivalente, s représente une transformation rigide de corps qui correspond à la composition d'une rotation d'angle s^3 avec une translation de vecteur (s^1, s^2). L'image de L_2 dans S est l'image de l'intersection du plan avec le plan horizontal à travers s^{-1}. Nous rappelons ci dessous l'expression de la fonction de perception déjà établie au § (III.2.3) :

$$im^1 = cos(s^3)\frac{l^1\left[(l^1)^2 + (l^2)^2\right] - (l^1)^2 s^1 - l^1 l^2 s^2}{(l^1)^2 + (l^2)^2} + sin(s^3)\frac{l^2\left[(l^1)^2 + (l^2)^2\right] - (l^2)^2 s^2 - l^1 l^2 s^1}{(l^1)^2 + (l^2)^2}$$

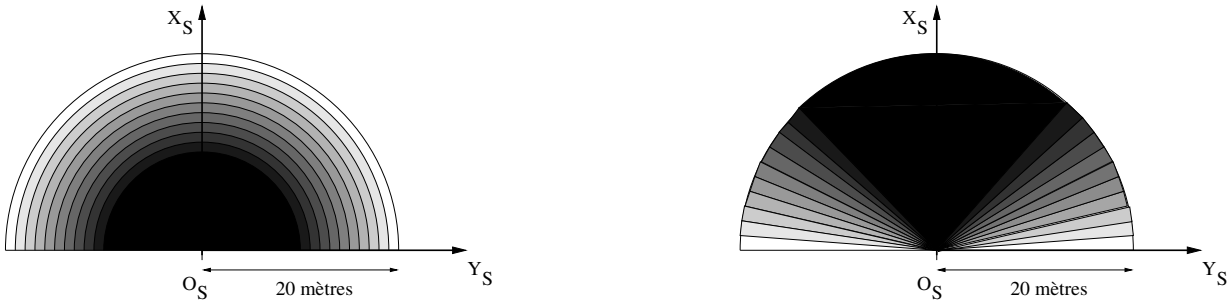

FIG. V.3 – Fonction de coût de visibilité d'un point dans le champ de perception d'un scanner laser. A gauche la fonction f_1 associée aux limites métriques du champ de perception. A droite la fonction f_2 associée aux limites angulaires du champ de perception. Plus le point est proche des limites de perception plus l'effet de la visibilité s'estompe. Dans ce cas on considère que $\alpha_{max} = \frac{\pi}{2}$, que $d_{min} = 0$ et que $d_{max} = 20$ mètres. L'effet de clarté des couleurs montre la diminution de la fonction de visibilité vers les limites du champ de perception.

$$im^2 = -sin(s^3)\frac{l^1\left[(l^1)^2+(l^2)^2\right]-(l^1)^2 s^1 - l^1 l^2 s^2}{(l^1)^2+(l^2)^2} + cos(s^3)\frac{l^2\left[(l^1)^2+(l^2)^2\right]-(l^2)^2 s^2 - l^1 l^2 s^1}{(l^1)^2+(l^2)^2}$$

V.2.3 Équations de localisation

Chaque paire capteur-amer donne lieu à un système d'équations linéarisés de la forme III.4.

Ligne verticale

Pour une ligne verticale L_1 perçue par un scanner laser S,

$$\frac{\partial P_{S,L_1}}{\partial s} = \begin{pmatrix} -\cos s^3 & -\sin s^3 & -\sin s^3(l_1^1 - s^1) + \cos s^3(l_1^2 - s^2) \\ \sin s^3 & -\cos s^3 & -\cos s^3(l_1^1 - s^1) - \sin s^3(l_1^2 - s^2) \end{pmatrix}$$

Plan vertical

Pour un plan vertical L_2 perçu par un scanner laser S,

$$\frac{\partial P_{S,L_2}}{\partial s} = \begin{pmatrix} -\frac{(l^1)^2}{(l^1)^2+(l^2)^2} & -\frac{l^1 l^2}{(l^1)^2+(l^2)^2} & \frac{l^2[(l^1)^2+(l^2)^2]-(l^2)^2 s^2 - l^1 l^2 s^1}{(l^1)^2+(l^2)^2} \\ -\frac{l^1 l^2}{(l^1)^2+(l^2)^2} & -\frac{(l^2)^2}{(l^1)^2+(l^2)^2} & \frac{l^1[(l^1)^2+(l^2)^2]-(l^1)^2 s^1 - l^1 l^2 s^2}{(l^1)^2+(l^2)^2} \end{pmatrix}$$

V.2.4 Calcul des fonctions de pondération

La construction des fonctions de pondération pour les deux types d'amers considérés tient compte des propriétés établie en § IV.3. Dans ce qui suit nous allons voir comment nous avons

choisi de construire ces fonctions.

Fonction de pondération pour un point

Soit un point p l'image d'une ligne verticale visible L_1 depuis un scanner laser S. Soit ξ_ρ et ξ_α deux fonctions continues qui donnent pour une configuration s du capteur et une configuration l_1 de l'amer les coordonnées polaires du point p, ρ et α, dans le repère capteur $\mathscr{R}_S$, soit alors :

$$\begin{array}{rcl} \xi_\rho \ : \ SE(2)\times\mathbb{R}^2 & \rightarrow & [d_{min}, d_{max}] \\ (s, l_1) & \mapsto & \rho = \xi_\rho(s, l_1) \\ \xi_\alpha \ : \ SE(2)\times\mathbb{R}^2 & \rightarrow & [-\alpha_{max}, \alpha_{max}] \\ (s, l_1) & \mapsto & \alpha = \xi_\alpha(s, l_1) \end{array}$$

Où $[d_{min}, d_{max}]$ est l'intervalle de visibilité en distance des scanners lasers montés sur le robot et $[-\alpha_{min}, \alpha_{max}]$ représente le champ de perception angulaire de ces capteurs. En se référent à la section § IV.4, le poids associé à ce point est le produit d'une fonction qui tient compte de la visibilité du point par le scanner laser et d'une fonction qui prend en compte le danger de collision que ce point représente par rapport au robot.

On propose alors la formulation suivante de la fonction de pondération qui concerne la visibilité :

$$\begin{array}{rcl} w_{visi} \ : \ SE(2)\times\mathbb{R}^2 & \rightarrow & [0,1] \\ (s, l_1) & \mapsto & w_{visi}(s, l_1) = f_{visi}(\xi_\rho(s, l_1), \xi_\alpha(s, l_1)) \end{array}$$

Avec :

$$\forall (\rho, \alpha) \in [d_{min}, d_{max}] \times [-\alpha_{min}, \alpha_{max}], \quad f_{visi}(\rho, \alpha) = f_1(\rho)\, f_2(\alpha)$$

Où

$$\forall \rho \in \mathbb{R}^+, \quad f_1(\rho) = \begin{cases} 1, & \text{si } \rho \in [d_{min}, \frac{d_{min}+d_{max}}{2}] \\ -\frac{2}{d_{min}+d_{max}}\,\rho + 2, & \text{si } \rho \in]\frac{d_{min}+d_{max}}{2}, d_{max}] \\ 0, & \text{sinon.} \end{cases}$$

et

$$\forall \alpha \in [-\alpha_{max}, \pi[, \quad f_2(\alpha) = \begin{cases} 1, & \text{si } \alpha \in [-\frac{\alpha_{max}}{2}, \frac{\alpha_{max}}{2}] \\ -\frac{2}{\alpha_{max}}\,|\alpha| + 2, & \text{si } \alpha \in [-\alpha_{max}, -\frac{\alpha_{max}}{2}] \cup [\frac{\alpha_{max}}{2}, \alpha_{max}] \\ 0, & \text{sinon.} \end{cases}$$

Les fonctions $f_1(\rho)$ et $f_2(\alpha)$ permettent de tenir compte de la visibilité d'un point par rapport aux frontières métriques et angulaires du champ de perception des scanners lasers. La figure V.3 montre une vue sur le plan (O_s, x_s, y_s) de l'effet de ces deux fonctions sur le champ de visibilité du scanner laser. La fonction f_{visi} est continue sur $[d_{min}, d_{max}] \times [-\alpha_{max}, \alpha_{max}]$ et nulle en dehors de cet espace. Il s'en suit que la fonction de pondération w_{visi} obtenue est continue sur l'espace de visibilité du scanner laser et nulle à l'extérieur.

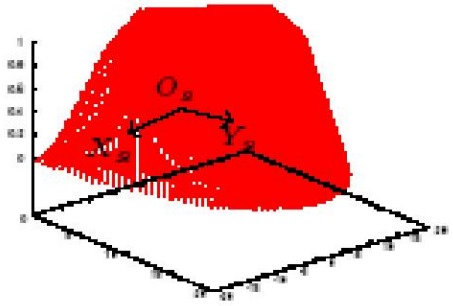

FIG. V.4 – Fonction de pondération associée à un point. Ce tracé correspond à la fonction $w(s,l_1)$ pour une configuration donnée du capteur lorsque les coordonnées polaires du point perçu varient à travers l'espace $[d_{min},d_{max}]\times[-\alpha_{max},\alpha_{max}]$.

Pour un objet quelconque de l'environnement, la fonction de pondération associée à la collision est une fonction inversement proportionnelle à la distance séparant le robot de cet obstacle. Dans le cas d'un amer en ligne verticale perçu par un scanner laser, nous prenons la distance entre le point p, image de cet amer dans le capteur, et le centre du capteur $O_S(s^1,s^2)$. On propose ainsi la formulation suivante de la fonction associée à la collision :

$$\begin{array}{rccc} w_{col} \ : & SE(2)\times\mathbb{R}^2 & \rightarrow & [0,1] \\ & (s,l_1) & \mapsto & w_{col}(s,l_1)=\frac{1}{1+\xi_\rho(s,l_1)} \end{array}$$

Finalement la fonction de pondération associée à l'image point p d'une ligne verticale perçue par un scanner laser est le produit $w_p = w_{visi}\times w_{col}$. L'effet de w_{col} dans la fonction w_p est limité par w_{visi} au champ de perception du scanner laser. Le tracé de la fonction w_p pour une configuration donnée du capteur et un point p dont les coordonnées polaires (ρ,α) varient à travers l'espace de visibilité $[d_{min},d_{max}]\times[-\alpha_{max},\alpha_{max}]$ est illustré à la figure V.4.

Fonction de pondération pour un segment

Soit un segment *seg*, image d'un plan vertical L_2 depuis un scanner laser S. Pour une configuration donnée s du capteur et une configuration donnée l_2 de l'amer, on appelle $\mathscr{V}_{seg}(s,l_2)$ l'ensemble des points visibles du segment défini comme suit :

$$\forall s\in SE(2),\ \forall l_2\in\mathbb{R}^2\ :\ \mathscr{V}_{seg}(s,l_2)=\{\,p_i\in seg/\,\rho_{p_i}\in[d_{min},d_{max}]\quad et\quad \alpha_{p_i}\in[-\alpha_{max},\alpha_{max}]\,\}$$

Où $(\rho_{p_i},\alpha_{p_i})$ sont les coordonnées polaires du point p_i dans le repère $\mathscr{R}_S$. En pratique, l'ensemble $\mathscr{V}_{seg}$ contient un nombre fini de points car on discrétise le segment suivant le pas angulaire de segmentation du scanner laser. On note $N_{\mathscr{V}_{seg}}$ le cardinal de cet ensemble.
En utilisant la fonction f_{visi} définie ci-dessus, nous proposons la formulation suivante de la fonction de pondération associée à la visibilité de $SE(2)\times\mathbb{R}^2$ dans $[0,N_{\mathscr{V}_{seg}}]$ pour un segment :

$$\forall s\in SE(2),\ \forall l_2\in\mathbb{R}^2\ :\ w_{visi}(s,l_2)=\sum_{p_i\in\mathscr{V}_{seg}(s,l_2)} f_{visi}(\rho_{p_i},\alpha_{p_i})$$

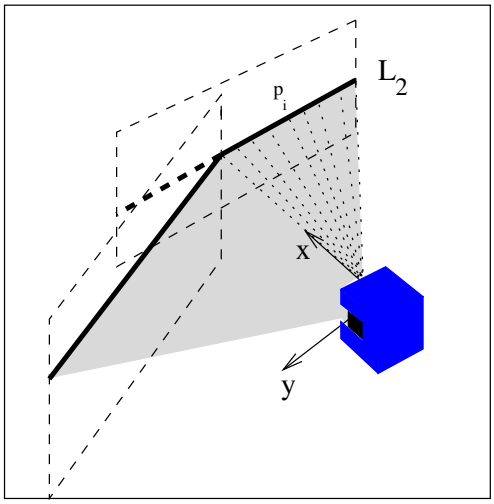

FIG. V.5 – Discrétisation des segments pour le calcul de la fonction de pondération. Dans ce cas une partie du segment L_1 n'est pas prise en compte car elle est masquée par le segment L_2. Une partie de ce dernier n'est pas prise en compte car elle n'appartient pas au champ de perception du scanner laser si on considère que son $\alpha_{max} = \frac{\pi}{2}$.

Pour la fonction de pondération associé à la collision définie de $SE(2) \times \mathbb{R}^2$ dans $[0, N_{\mathscr{V}_{seg}}]$, nous proposons l'expression suivante :

$$\forall s \in SE(2),\ \forall l_2 \in \mathbb{R}^2\ :\ w_{col}(s,l_2) = \sum_{p_i \in \mathscr{V}_{seg}(s,l_2)} \frac{1}{1+\rho_{p_i}}$$

Les fonctions $w_{visi}(s,l_2)$ et $w_{col}(s,l_2)$ sont continues sur $SE(2) \times \mathbb{R}^2$. La fonction de pondération issue du produit $w_{seg}(s,l_2) = w_{visi}(s,l_2)\, w_{col}(s,l_2)$ est continue sur l'espace $SE(2) \times \mathbb{R}^2$ et nulle lorsque le segment n'appartient plus au champ de perception du scanner laser. Lorsque un segment se trouve partiellement caché par un ou plusieurs autres segments, l'ensemble $\mathscr{V}_{seg}$ ne contient que les points appartenant aux parties visibles de ce segment (voir figure V.5).

Les formules de modélisation posées ci-dessus représentent les fonctionnalités bas niveau de la plateforme logiciel développée pour produire un mouvement asservi sur amers pour contrôler le mouvement d'un robot lors de l'exécution d'une trajectoire planifiée.

V.3 Intégration software

Dans cette section, nous allons d'abord décrire l'intégration software basée sur les concepts définis dans les chapitres III et IV. Ensuite nous présentons les résultats expérimentaux sur le robot mobile Hilare2 avec sa remorque.

V.3.1 Planification de mouvement asservi sur amers

Notre planificateur de mouvement asservi sur amers est écrit en langage orienté objet (C++) dans l'objectif de prendre avantage de la généralité de notre approche dans le code source. Le niveau haut de la planification de mouvement et le calcul des poids utilisent des classes abstraites, alors qu'à un niveau plus bas, les calculs spécifiques sont effectués avec des classes dérivées. Les quatre classes abstraites principales sont robot, capteur, amer et paire-capteur-amer. Elles sont décrites comme suit :

– robot modélise la chaîne cinématique qui compose le robot.

– capteur traite de la configuration du capteur en fonction de la configuration du robot. Elle prend en compte les séquences d'enchaînement des amers qui lui sont associés dans le mouvement asservi sur amers.

– amer traite des différentes instantiations des amers et leurs positions dans l'espace de travail.

– paire-capteur-amer traduit les interactions entre amers et capteurs. Cette classe procède au calcul de l'image d'un amer donné dans un capteur donné en fonction de leurs positions respectives. La fonction de calcul des poids est une méthode polymorphe de la classe abstraite paire-capteur-amer.

La figure V.6 présente l'implémentation de notre software pour notre robot mobile Hilare2 avec sa remorque. Dans ce cas, la classe abstraite capteur est dérivée pour obtenir la classe sick correspondant à un scanner laser. La classe amer est dérivée pour obtenir les classes ligne_vert et plan_vert correspondant respectivement aux amers ligne verticale et plan vertical. Ainsi, la classe paire-capteur-amer est dérivée en classes point et segment. Ces deux dernières classes dérivées contiennent des méthodes qui calculent l'image d'une ligne verticale et d'un plan vertical dans un capteur laser.

V.3.2 Contrôle du mouvement asservi sur amers

Notre architecture de contrôle de mouvement asservi sur amers est montée à bord du robot mobile Hilare2 avec sa remorque et utilise des éléments de l'architecture de contrôle décrite dans [46]. Le principe du schéma de contrôle d'un mouvement asservi sur amers est présenté en figure V.7. Notons que ce schéma peut être intégré dans n'importe quelle architecture de contrôle en boucle fermée. Pour cette raison nous n'apportons ici aucun détail sur la loi de contrôle en boucle fermée. Nous invitons le lecteur pour plus de détail à se référer à [56] et [46]. L'architecture de contrôle d'un mouvement asservi sur amers est composée de modules construits avec le logiciel de prototypage temps-réel GenoM [23] développé au *LAAS-CNRS*. Un GenoM-module est un serveur temps-réel multi-thread qui exécute des fonctions définies par l'utilisateur lorsqu'elle sont appelées par un client via des requêtes. Ces modules peuvent aussi se partager des informations. Les principales caractéristiques d'un tel module sont :

– Des requêtes déclenchées par un client qui lancent l'exécution des fonctions définies par l'utilisateur,

– Des tâches d'exécutions avec leurs propres périodes qui exécutent les fonctions lorsqu'une requête est déclenchée.

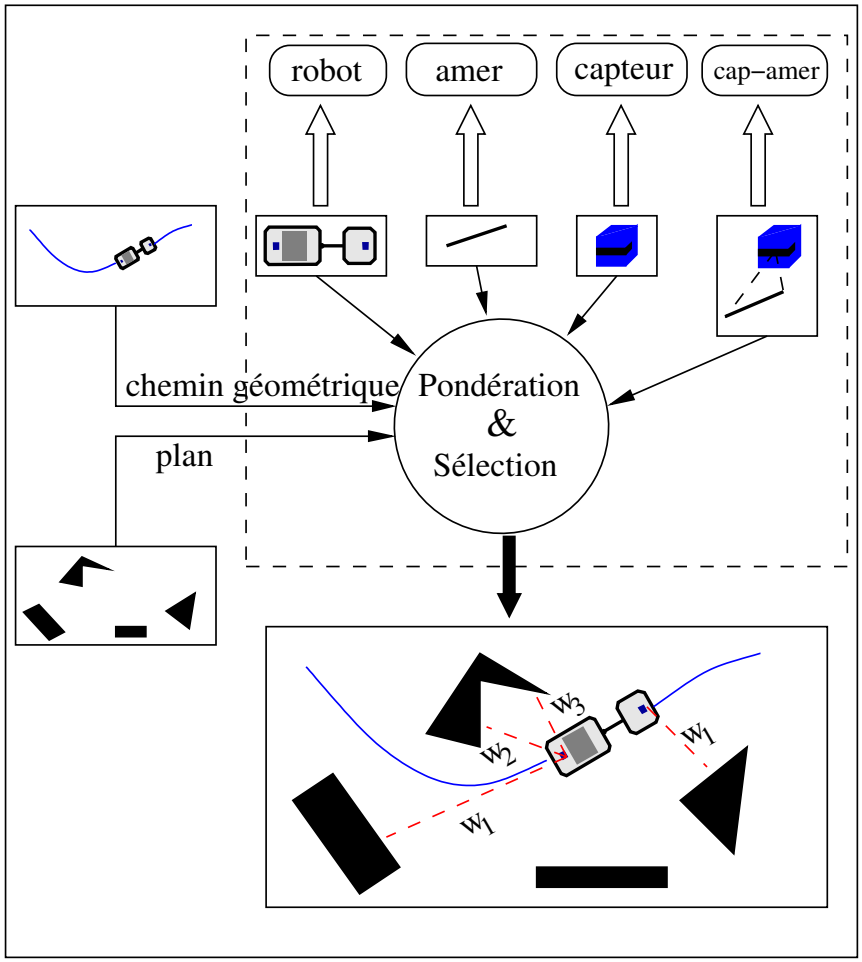

FIG. V.6 – L'architecture software implémentée à bord du robot Hilare2 tractant une remorque. Étant donné un plan de l'environnement, une configuration initiale et une configuration finale, un chemin géométrique sans collision obtenu avec le planificateur de chemin Move3D. Alors, utilisant le plan de l'environnement composé de plans verticaux, notre software retourne un ensemble de paires capteur-amer pondérées qui est utilisé durant l'exécution du mouvement pour localiser le robot dans l'environnement.

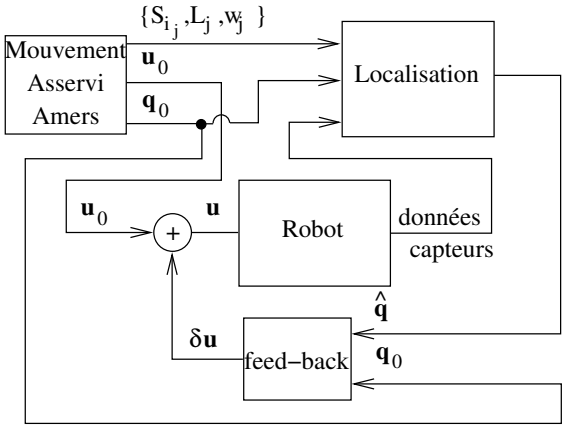

FIG. V.7 – Diagramme d'asservissement du mouvement asservi sur amers. Le robot prend en entrée un vecteur **u** et retourne en sorties des données capteurs (i.e. les images des amers dans les capteurs). Ces images sont des entrées de la fonction de localisation qui les compare avec les images de référence et calcule une estimée de la configuration de localisation $\mathbf{q}_{loc}$ du robot comme un écart par rapport à la configuration de référence $\mathbf{q}_0$, utilisant des paires capteurs-amers préselectionnées $\{S_{i_j}, L_j, \lambda_j\}$. Cette estimation de la configuration du robot est une entrée de la boucle d'asservissement qui la compare avec la configuration de référence et retourne en sortie une correction $\delta\mathbf{u}$. À noter que la fonction d'asservissement n'est pas spécifique à notre approche : toute boucle d'asservissement stabilisant le système peut être utilisée.

– Une structure de donnée interne contenant toutes les informations partagées par les différentes tâches.

– Des posters, i.e. des segments de mémoire partagés dans lesquels le module écrit des informations qui peuvent être lues par d'autre modules ou clients.

De son côté l'utilisateur doit uniquement écrire le prototype du module dans un fichier, définir les tâches, les requêtes, la structure de donnée interne, les posters et les fonctions appelées lorsqu'une requête est déclenchée.

La figure V.8 montre le schéma des modules de contrôle intégrés à bord du robot mobile Hilare2. Le module LOCO s'exécute avec une période de 25ms et contrôle le mouvement du robot. Il correspond à la boite feed-back de la figure V.7. Le module WLOC correspond à la boite de localisation dans la figure V.7. La fréquence de calcul de localisation dans WLOC n'est pas assez rapide pour se synchroniser à la fréquence de LOCO (40Hz). Pour contourner cette limitation, nous procédons comme suit. À chaque calcul de localisation dans WLOC, nous calculons la transformation rigide T_{odo}^{glob} qui permet de passer de la configuration $\mathbf{q}_{odo}$, estimée par l'odométrie à l'instant où les données capteur sont prises, à la configuration $\hat{\mathbf{q}}$ calculé par WLOC avec les données de perception capteur. Entre deux calculs de T_{odo}^{glob}, la localisation globale du robot est donnée par la transformation : $\hat{\mathbf{q}} = T_{odo}^{glob}\mathbf{q}_{odo}$, où à présent $\mathbf{q}_{odo}$ est la mesure odométrique courante et T_{odo}^{glob} est la dernière valeur calculée. Utilisant cette méthode, nous avons une localisation globale à la fréquence odométrique (40Hz). Comme le module LOCO contrôle le robot

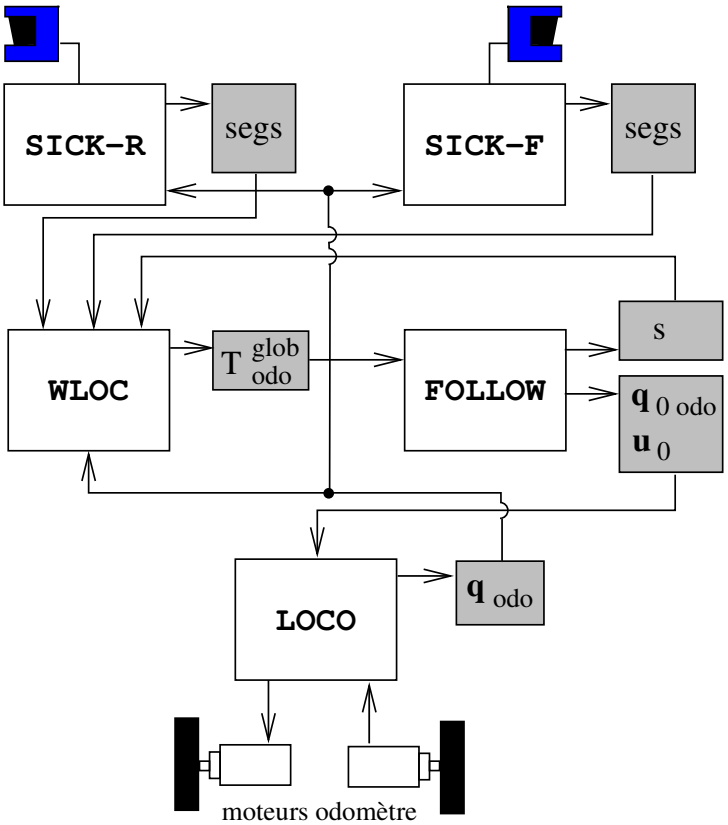

FIG. V.8 – L'architecture de contrôle du mouvement asservi sur amers telle qu'elle est implémentée à bord du robot mobile Hilare2 avec sa remorque. Les boites blanches représentent des modules GenoM, les boites grises représentent des posters. Le module LOCO contrôle les moteurs du robot en utilisant une loi de contrôle classique de suivi de trajectoire. Le module FOLLOW échantillonne la trajectoire de référence et exporte la configuration de référence courante $\mathbf{q}_{0\,odo}$ exprimée dans le repère odométrique. FOLLOW exporte aussi l'abscisse courante s le long de la trajectoire. WLOC lit les données capteurs exportées par les modules SICK-F et SICK-R qui gèrent respectivement les lasers scanners avant et arrière, et calcule la localisation pondérée asservi sur amers $\mathbf{q}_{loc}$ du robot. Cette configuration est exportée comme la transformation rigide T_{odo}^{glob} transformant la configuration estimée par l'odométrie à la localisation pondérée asservi sur amers du robot : $\hat{\mathbf{q}} = T_{odo}^{glob}\mathbf{q}_{odo}$.

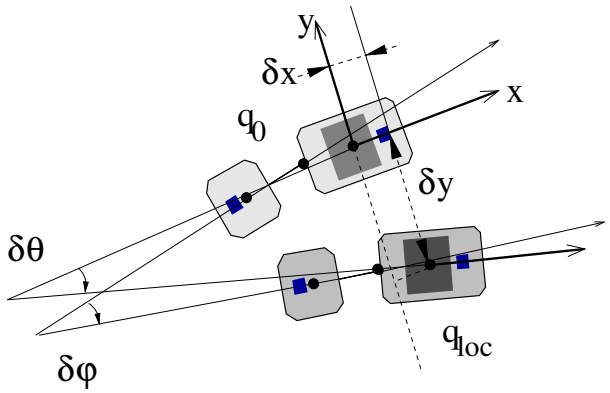

FIG. V.9 – Écart entre une configuration $\mathbf{q}_0$ de la trajectoire de référence et $\mathbf{q}_{loc}$, configuration de la trajectoire exécutée en utilisant un mouvement asservi sur amers. Les variables intervenant dans la loi de commande du robot sont l'écart longitudinale δx, l'écart transversal δy et l'écart angulaire $\delta\theta$.

sur la base des données odométriques, le module follow exprime les configurations de référence le long de la trajectoire dans le repère odométrique : $\mathbf{q}_{0\,odo} = T_{odo}^{glob\ -1}\mathbf{q}_0$.

V.4 Résultats expérimentaux

A présent nous présentons deux expériences. La première illustre comment la pondération des amers permet au robot d'avoir un mouvement robuste vis à vis des incertitudes du plan de l'environnement. La seconde montre comment exécuter une tâche de parking entre deux obstacles dont la position n'est pas connue précisément.

Ces expériences ont été réalisées au sein même de notre laboratoire. Le plan de l'environnement est un ensemble de plan verticaux construit au préalable. La génération d'un chemin géométrique sans collision se fait à l'aide du planificateur géométrique **Move3D** [82] développé dans notre laboratoire. L'appariement des primitives planifiées avec les primitives perçues par les scanners laser se fait par l'intermédiaire d'une librairie «segkit» développée au LAAS sur la base des travaux de [64].

V.4.1 Manoeuvre dans un couloir

Dans cette première expérience, nous planifions d'abord un chemin géométrique sans collision que nous exécutons dans un environnement légèrement différent (Figure (V.10) gauche). La figure (V.10) de droite montre une configuration appartenant à la trajectoire de référence et les quatre segments de droite utilisés pour exécuter le mouvement. Nous mettons une longue planche en face du mur pour simuler une position différente de ce dernier. Le couloir réel est alors plus étroit que le couloir du plan de référence mais l'espace restant est suffisant pour le passage du

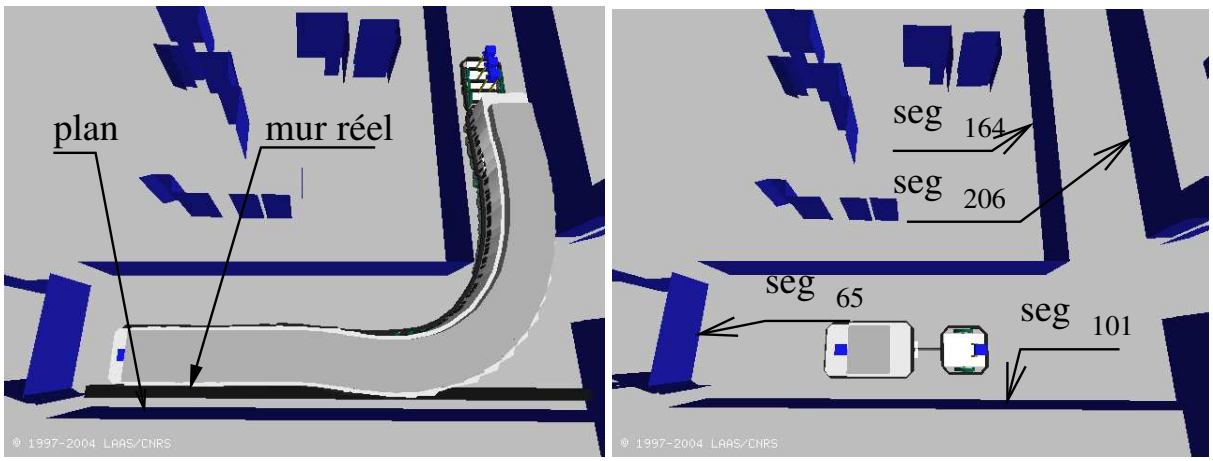

FIG. V.10 – Sur l'image de gauche, un chemin géométrique sans collision est planifié à travers un couloir. Sur l'image de droite, sont représentés quatre segments de droite utilisés dans cette expérience : seg_{101}, seg_{164}, seg_{205}, et seg_{206} et une configuration de référence le long de la trajectoire.

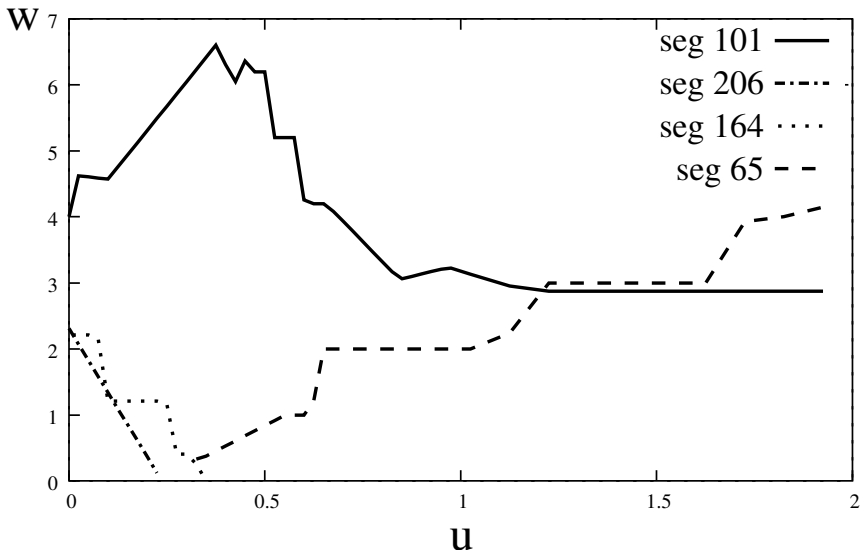

FIG. V.11 – Poids associés à chaque amer.

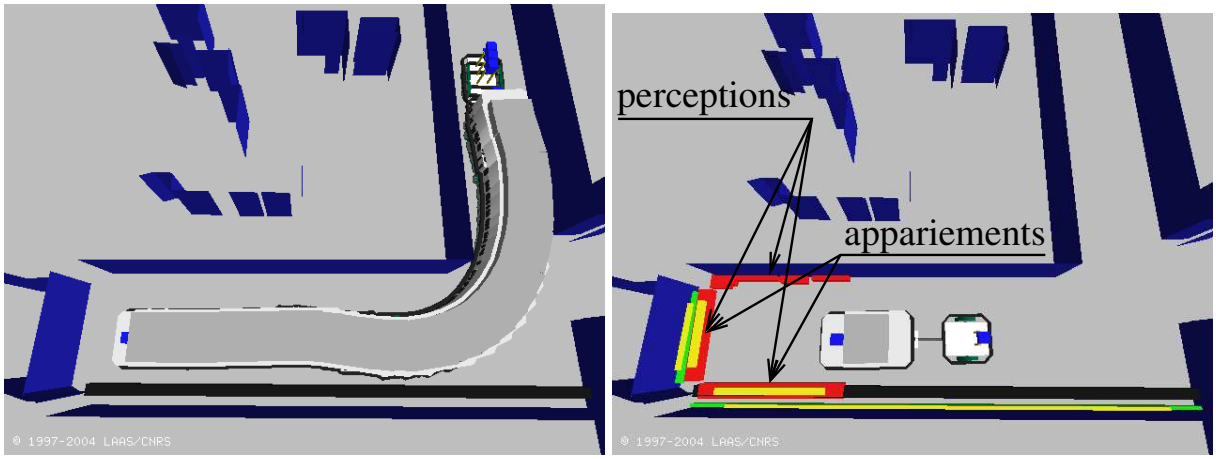

FIG. V.12 – Exécution d'un mouvement asservi sur amers dans un plan de l'environnement inexact. Sur la figure de gauche, la position d'un mur est différente dans le plan et dans l'environnement réel. Malgré cette erreur, le robot adapte la trajectoire pour rester à la distance spécifiée par rapport au mur. Sur la figure de droite, la position du robot correspond à la configuration de référence montré en figure (V.10) à droite. Les segments verts représentent les segments d'asservissement. Les segments rouges sont les segments perçus par le robot dans l'environnement d'exécution. Les segments avec une pastille jaune sont les segments que la procédure d'appariement a réussi à faire correspondre.

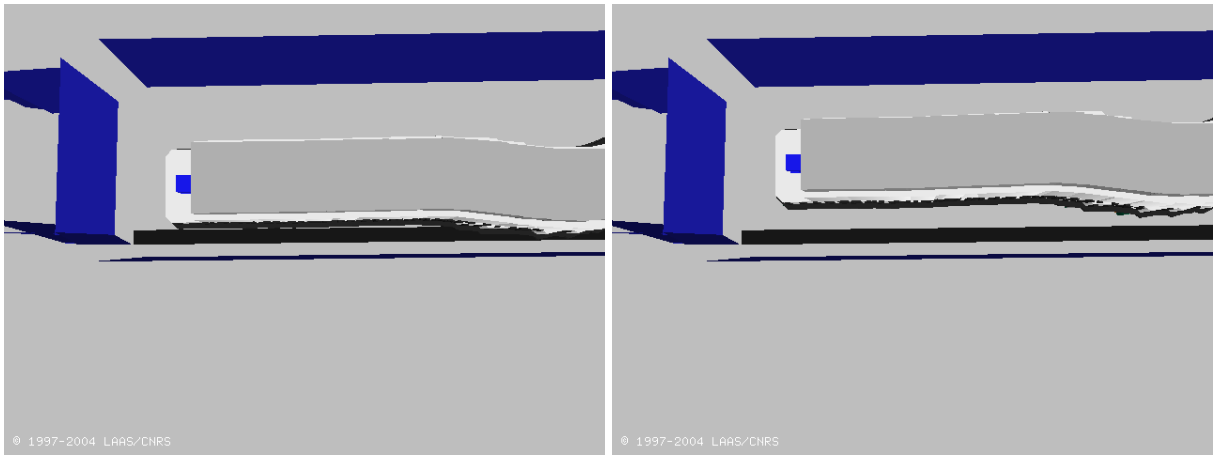

FIG. V.13 – Zoom de la fin de la trajectoire. Sur la gauche, la trajectoire de référence ; sur la droite, la trajectoire exécutée. A noter que la configuration finale est à la distance spécifiée par rapport au segment 101.

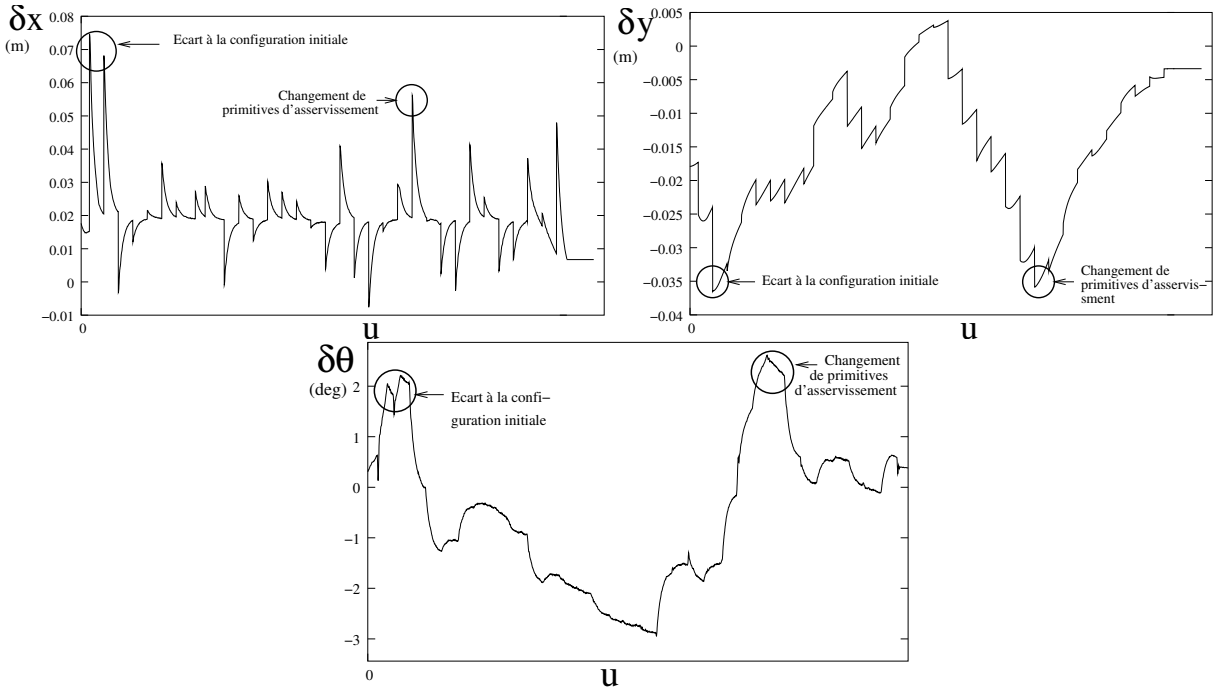

FIG. V.14 – Manoeuvre du couloir : Les écarts de localisation entre configurations de référence et configurations de localisation (voir figure (V.9)) au cours de la trajectoire.

robot. L'objectif de cette expérience est d'illustrer le contrôle de la tâche de mouvement par rapport au mur «déplacé ». Pour cette raison, les poids associés à chaque segment sont modifiés manuellement et montrés dans la figure (V.11). La figure (V.12) montre la trajectoire exécutée dans l'environnement modifié. La figure (V.13) montre la fin de la trajectoire de référence (à gauche) et la fin de la trajectoire exécutée (à droite). Notons que la fin de la configuration est à la distance de référence spécifiée par rapport au segment 101. Dans ce cas la localisation retourne la configuration de référence. Le résidu de perception par rapport aux segment 101 et 65 pour la configuration de localisation finale (configuration de référence) est égal au résidu de perception pour la configuration atteinte et sont tous deux nuls. La figure (V.14) représente les écarts de localisation entre configuration de référence et configuration de localisation tout au long de la trajectoire. On peut observer que les écart δx, δy et $\delta\theta$ convergent vers 0 ce qui montre que le robot a bien atteint sa configuration corrigée. A la configuration initiale le robot est sujet à un grand écart de correction dû à la différence de perception entre le plan et l'environnement d'exécution et cet écart converge progressivement vers 0. Lorsque le segment 65 apparaît et que le segment 206 commence à disparaître (changement de primitives d'asservissement) ceci correspond en même temps à une rotation progressive de 90° du robot ce qui fait apparaître un écart de perception suivant la direction transversale et la direction longitudinale du robot. Cet écart est principalement dû au fait qu'avant de tourner l'erreur du modèle était dans la direction longitudinale du robot (segment 101 modifié) et ensuite elle s'est transformée en une erreur transversale.

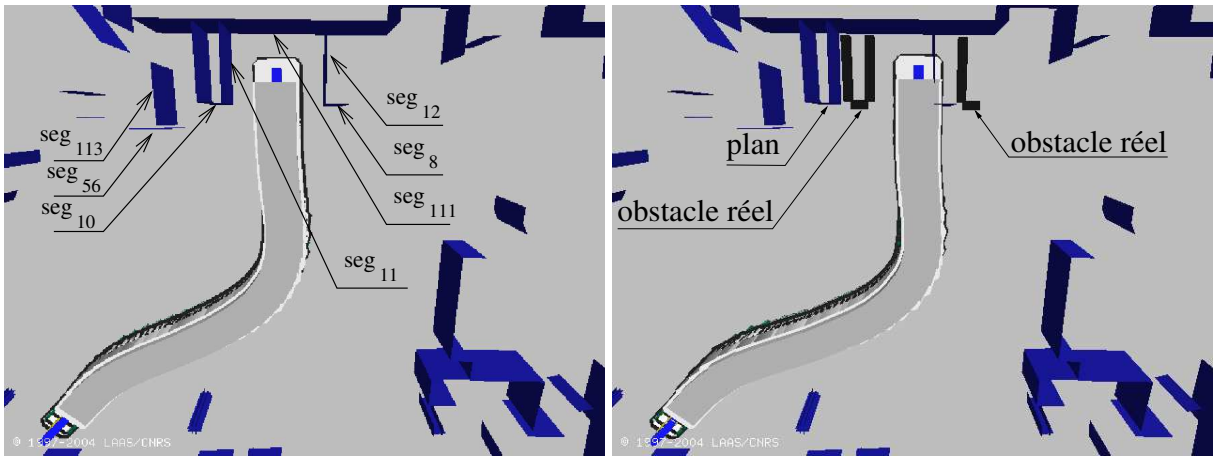

FIG. V.15 – À gauche, la trajectoire de référence planifiée dans le plan de l'environnement et les segments de droite utilisés pour exécuter le mouvement. À droite, les segments 8, 10, 11 et 12 sont décalés à droite. Cependant, le robot adapte le mouvement pour se garer entre ces amers.

V.4.2 Manoeuvre de garage

Dans cette seconde expérience, nous montrons notre approche dans le cadre d'une manoeuvre de garage. Nous avons utilisé dans cette situation le troisième algorithme proposé dans § IV.5.3. Cet algorithme est le plus approprié dans ce cas de figure car le robot effectue une manoeuvre assez proche des côtés du parking et les amers proches de la trajectoire sont les plus pertinents. Au final, le robot est amené à se garer entre deux segments parallèles (11 et 12, sur la figure (V.15) de gauche). La trajectoire de référence planifiée dans le plan de l'environnement est montrée à gauche. Dans cette expérience on ne considère que le scanner laser avant. Les amers sélectionnés et associés à ce capteur sont seg_8, seg_{10}, seg_{11}, seg_{12}, seg_{56}, seg_{111} et seg_{113} (voir figure (V.15) de gauche). Les fonctions de pondération associées aux couples capteur-amer sont illustrées en figure (V.16). Dans la figure (V.15) de droite, les segments $\{seg_8, seg_{10}, seg_{11}, seg_{12}\}$ définissant la configuration du parking sont décalés vers la droite. L'image de droite montre la trajectoire exécutée.

Les figures (V.17) et (V.18) montrent l'évolution du mouvement du robot à travers des configurations (les figures de droite) avec les valeurs des poids associés aux amers pris en charge dans la phase d'asservissement (les figures de gauche). La figure (V.17) du haut montre la configuration initiale de la trajectoire. Les amers en rouge avec une pastille jaune sont les amers de l'environnement réel appariés avec les amers sélectionnés. On remarque que les amers perçus du parking ne coincident pas encore avec les amers réels. Ceci est essentiellement dû au fait que le plan de référence n'est pas exact et aussi au fait que la configuration de départ n'est pas exactement connue dans l'environnement réel. Les poids associés aux amers sont à peu près égaux à l'instant initial. La figure (V.17) du bas montre le robot au milieu de l'exécution de sa trajectoire de référence. Le poids du segment seg_{111} est nettement plus grand que les poids des autres segments parce qu'il occupe une grande place dans l'espace de perception du capteur avant. Dans

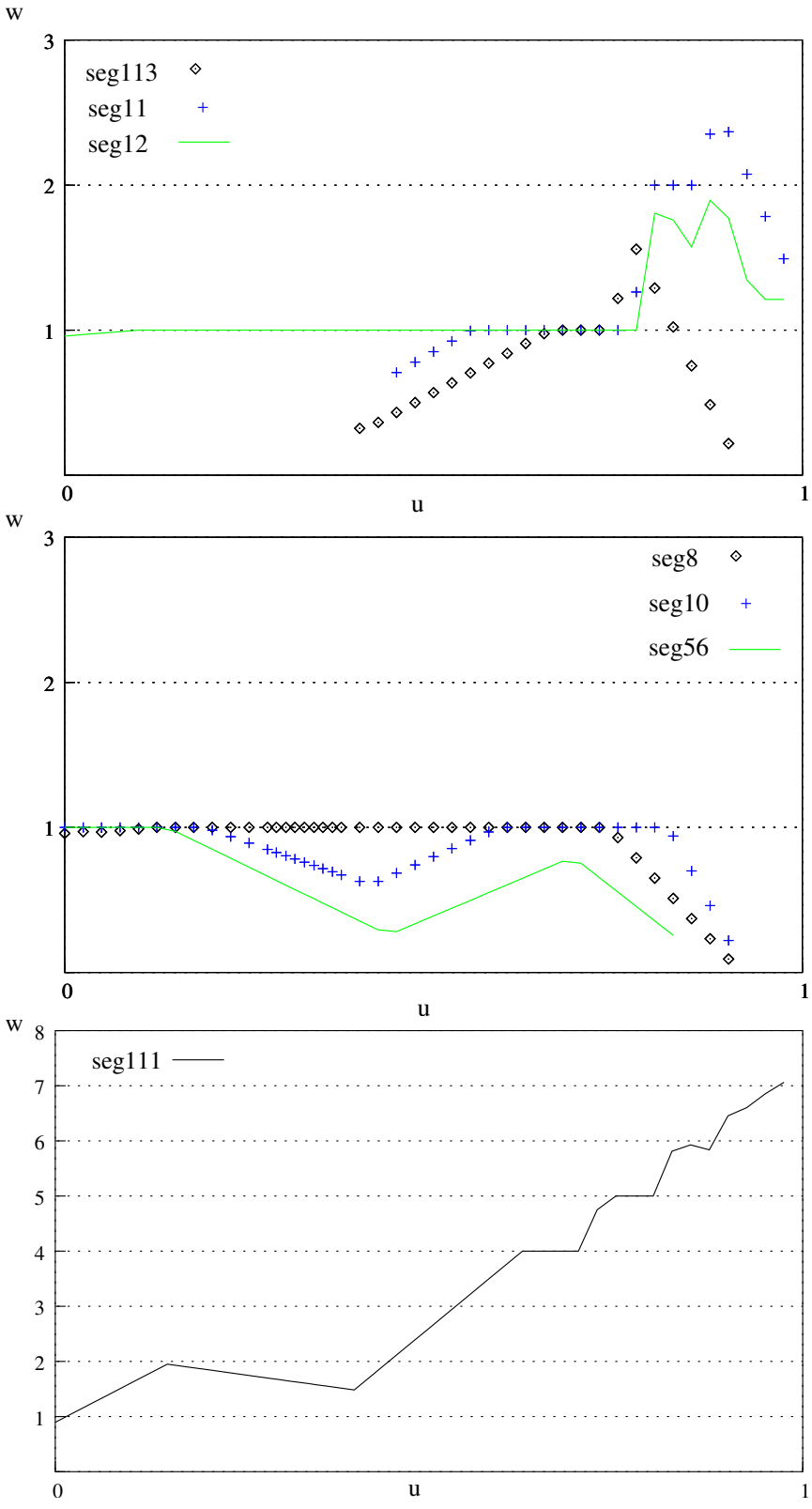

FIG. V.16 – les poids associés à chaque amer.

cette configuration, les amers perçus coincident presque avec les amers du parking.
En figure (V.18) du haut, le poids du segment seg_{111} continue à augmenter car ce dernier est de plus en plus proche. Le même effet est observé sur les segment seg_{12} et seg_{11} représentant les côtés latéraux du parking. Au contraire, les segments seg_{56} et seg_8 voient leurs poids diminuer parce qu'ils commencent à sortir du champs de perception du capteur avant. À la fin (figure (V.18) du bas) seuls les amers seg_{111}, seg_8 et seg_{12} sont utilisés pour asservir la configuration du robot. Remarquons que dans la configuration finale atteinte, le robot a une perception des amers seg_{111}, seg_8 et seg_{12} identique à celle de la configuration finale de référence.

La figure (V.19) représente les écarts de localisation entre configurations de référence et configurations de localisation au cours de la trajectoire. On peut observer que les écart δx, δy et $\delta\theta$ convergent vers 0 ce qui montre que le robot a bien atteint sa configuration corrigée. A la configuration initiale le robot est sujet à un grand écart de correction dû à la différence de perception entre le plan et l'environnement d'exécution, cet écart converge progressivement vers 0. Avant la dernière phase d'entrée dans le parking (lorsque sa direction longitudinale est perpendiculaire au segment 111) le robot corrige sa position transversale en effectuant une légère rotation avant de revenir à une direction perpendiculaire au segment 111.

V.5 Amélioration des algorithmes de sélection

V.5.1 Sélection tenant compte du conditionnement du système de localisation linéarisé

Les amers sélectionnés par les algorithmes du chapitre précédent peuvent ne pas tenir compte du conditionnement du système de localisation (équation III.5). Dans la section III.6 nous avons établi que l'existence et l'unicité de la configuration corrigée $\mathbf{q}_c$ dépend du rang de la matrice de localisation W qui doit être de rang plein colonne.

L'idée est de prendre parmi les amers visibles non sélectionnés celui ou ceux qui améliorent le conditionnement de la matrice du système de localisation. L'amer qui sera introduit est donc un amer n'appartenant à aucune séquence locale d'amers et au vu du critère de sélection de l'algorithme il n'est jamais considéré comme pertinent pour l'asservissement du chemin planifié. C'est pourquoi, si cet ensemble d'amers existe, il faut limiter l'influence de ces éléments sur le résultat de la localisation.

L'algorithme 4 décrit la démarche à suivre pour ajouter les amers améliorant le conditionnement de W au mouvement asservi sur amers. Il admet en entrée l'ensemble des amers de l'environnement $\mathscr{L}_{env}$, l'ensemble des capteurs du robot $\mathscr{S}$ de cardinal n, le mouvement asservi sur amers *maa* qui contient le chemin planifié $\gamma(u)$, $u \in [0,U]$ et n séquences d'enchaînement maa_S_i où chacune d'elles contient une succession de p_i séquences locales d'amers. Le dernier argument d'entrée de l'algorithme est le conditionnement maximal $K_{d_{max}}$ qu'on tolère pour la matrice de localisation pondérée W. Rappelons que la valeur du conditionnement d'une matrice est représentée par le rapport entre sa plus grande valeur singulière et sa plus petite valeur sin-

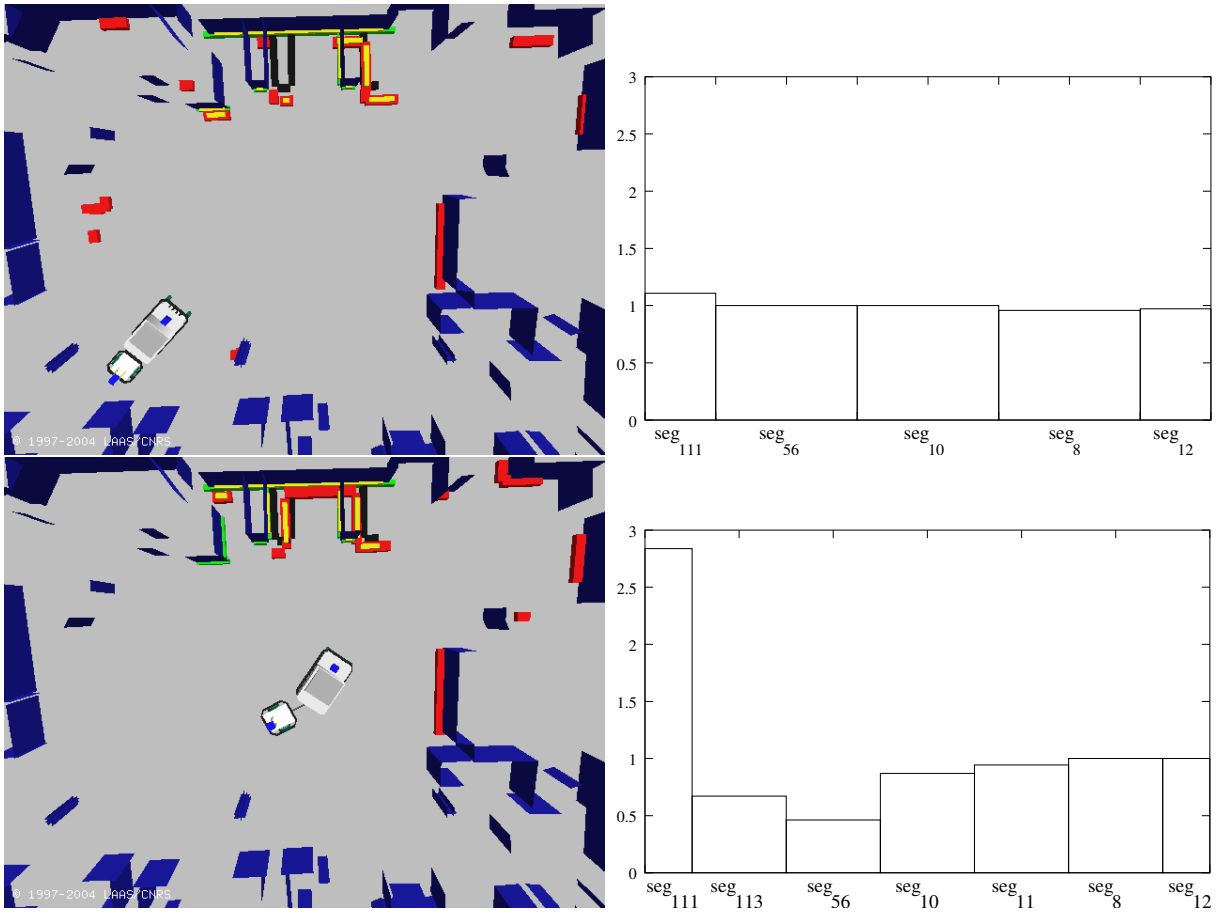

FIG. V.17 – Exécution du mouvement asservi sur amers pour effectuer une manoeuvre de parking : phase de départ. Les segments verts représentent les segments sélectionnés pour l'asservissement. Les segments rouges sont les segments perçus par le robot dans l'environnement d'exécution. Les segments avec une pastille jaune sont les segments que la procédure d'appariement a réussi à faire correspondre.

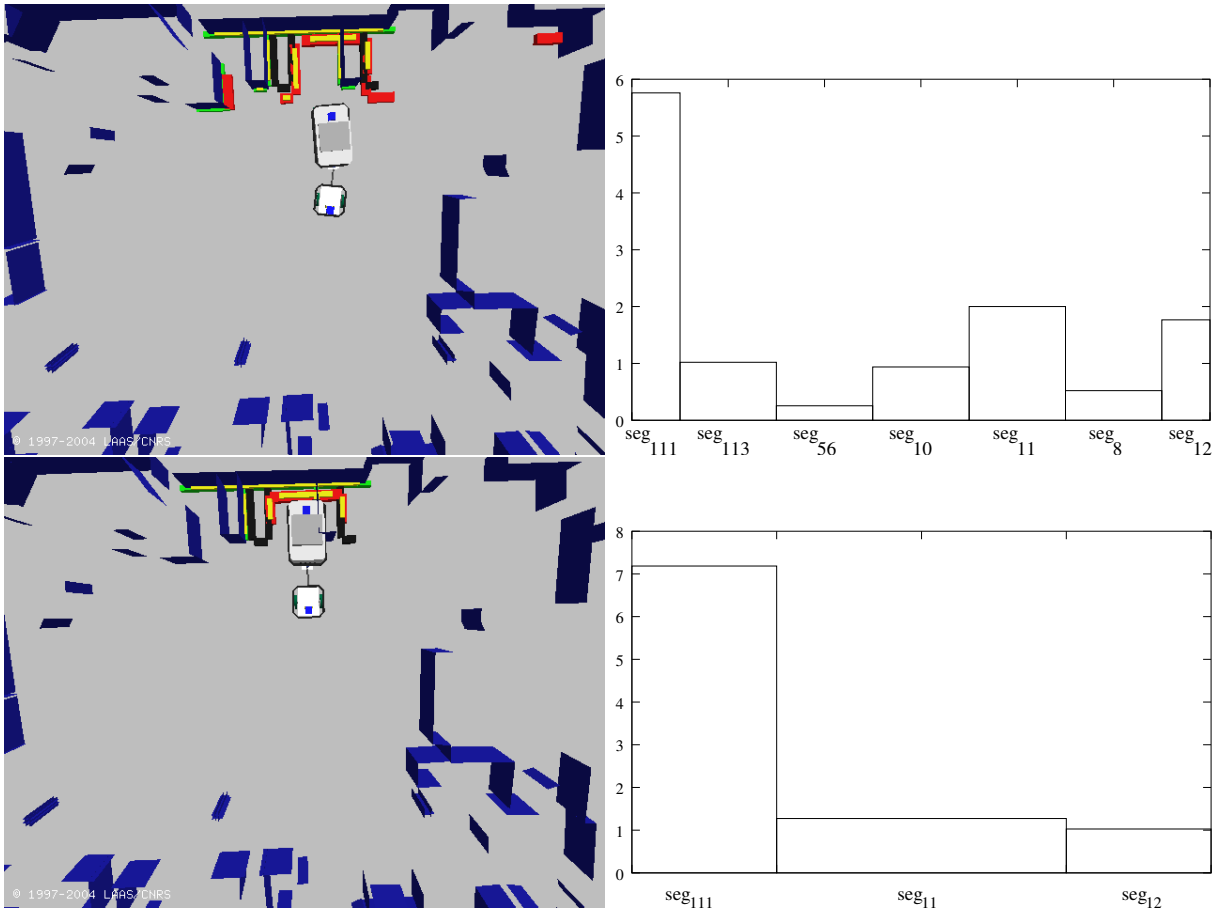

FIG. V.18 – Exécution du mouvement asservi sur amers pour effectuer une manoeuvre de parking : phase finale. Les segments verts représentent les segments sélectionnés pour l'asservissement. Les segments rouges sont les segments perçus par le robot dans l'environnement d'exécution. Les segments avec une pastille jaune sont les segments que la procédure d'appariement a réussi à faire correspondre.

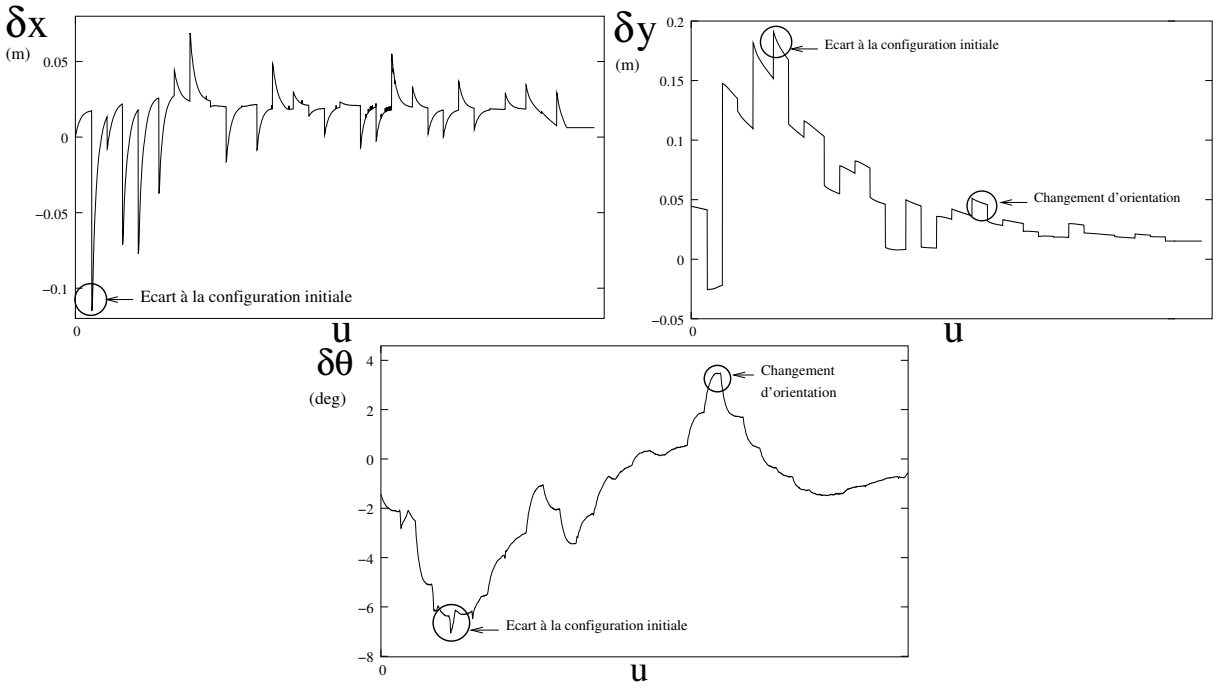

FIG. V.19 – Manoeuvre de garage : Les écarts de localisation entre configurations de référence et configurations de localisation (voir figure V.9) tout au long de la trajectoire.

gulière. Une matrice bien conditionnée est une matrice dont le nombre de conditionnement est proche de 1. Elle est mal conditionnée si ce nombre est très grand par rapport à 1 et singulière s'il est infini.

Pour chaque capteur $S_i \in \mathscr{S}$, $i = \overline{1,n}$ l'algorithme parcourt les p_i séquences locales d'amers de la séquence d'enchaînement d'amers maa_S_i. Pour une séquence d'amers $\mathscr{L}^i_{select}$ définie sur un intervalle $[u_1,u_2] \subset [0,U]$, la fonction Nombre_Conditionnement() retourne la valeur du conditionnement qui lui correspond. Si ce nombre est supérieur au $K_{d_{max}}$ toléré cela veut dire que le système est mal conditionné. Dans ce cas la fonction Améliorer_Conditionnement() tente de trouver parmi les amers de $\mathscr{L}_{env}$ visible depuis S_i durant l'intervalle $[u_1,u_2]$ ceux qui rendent le conditionnement de la matrice de localisation inférieur à $K_{d_{max}}$. Si de tels amers existent alors ils sont ajoutés à $\mathscr{L}_{K_d}$, ensemble des amers améliorant le conditionnement du système de localisation pour la séquence d'enchaînement maa_S_i. Une fois l'ensemble $\mathscr{L}_{K_d}$ construit durant l'intervalle $[0,U]$, la fonction Ajouter_Amer_Conditionnement() insère dans la séquence d'enchaînement d'amers maa_S_i. Cette fonction permet aussi de diminuer les valeurs des fonctions de pondération des amers ajoutés afin de limiter leurs effets sur le résultat de la localisation.

Données : $\mathscr{L}_{env}$, $\mathscr{S}$, $K_{d_{max}}$, maa
Résultat : maa
début
  pour $i \leftarrow 1$ ***à*** n **faire**
    $\mathscr{L}_{K_d} \leftarrow Null$;
    pour $j \leftarrow 1$ ***à*** p_i **faire**
      $\mathscr{L}_{select} \leftarrow$Séquence_Locale_Amers(maa_S_i, j);
      $K_d \leftarrow$Nombre_Conditionnement($\mathscr{L}_{select}$);
      si $K_d \geq K_{d_{max}}$ **alors**
        $\mathscr{L} \leftarrow$Améliorer_Conditionnement($\mathscr{L}_{env}$,$\mathscr{L}_{select}$,$K_{d_{max}}$);
        si $\mathscr{L} \neq Null$ **alors** $\mathscr{L}_{K_d} \leftarrow \mathscr{L}_{K_d} \oplus \mathscr{L}$;
    si $\mathscr{L}_{K_d} \neq Null$ **alors** maa_$S_i \leftarrow$Ajouter_Amer_Conditionnement($\mathscr{L}_{K_d}$,maa_S_i);
fin

Algorithme 4 – Amélioration du conditionnement de la matrice de localisation

On peut avoir une idée de l'utilité de cette procédure en considérant un cas de figure assez classique où le robot mobile Hilare2 avec sa remorque navigue dans un couloir assez long d'une configuration initiale $\mathbf{q}_i$ à une configuration finale $\mathbf{q}_f$. Soit $\mathscr{L}_{env} = \{L_1, L_2, L_3\}$ les amers du plan de référence où L_1 et L_2 représentent les deux côtés du couloir. En ne considérant que le capteur avant, on suppose que le mouvement asservi sur amers obtenu par l'algorithme de sélection 3 (c.f.§ IV.5.3) contient la séquence $\mathscr{L}_{select} = \{L_1, L_2\}$ qui contient les plus proches amers au long de la trajectoire planifié. D'un point de vu succés de la tâche de navigation, ce résultat est tout à fait acceptable étant donné que selon le chemin planifié les deux murs du couloir sont les amers sur lesquels il faut s'asservir. Pour la localisation on considère uniquement la composante (x,y) du vecteur de configuration du robot et on suppose que le vecteur d'image ne contient que la distance du robot au mur. On obtient ainsi un système de localisation à deux inconnues et deux lignes.

Dans le cas où les murs L_1 et L_2 sont parfaitement parallèles, voir plan 1 de la figure V.20, il est évident que nous obtenons, tout au long de la trajectoire, une matrice du système de localisation de rang un. Quand les murs du couloir sont légérement en entonnoir, voir plan 2 de la même figure, bien que les système soit de rang 2, il est mal conditionné car il présente une très grande sensibilité sur le résultat de la composante x du vecteur de configuration (x,y). L'application de la procédure décrite ci-dessus permet d'ajouter l'amer L_3 à la séquence $\mathscr{L}_{select}$ sous réserve qu'il soit perceptible par le capteur du robot. Elle permet aussi de rendre négligeable l'effet d'une erreur de perception sur L_3 en diminuant son poids. Ainsi nous aurons amélioré le résultat de la localisation en le rendant moins sensible aux erreurs de mesures tout en gardant le caractère pertinent des amers déjà sélectionnés.

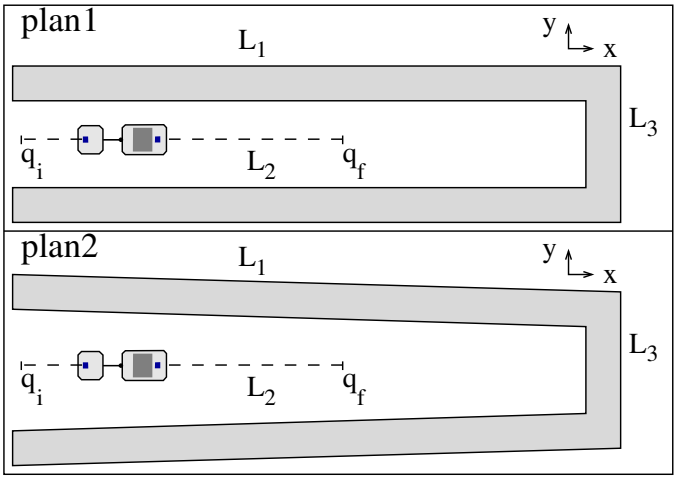

FIG. V.20 – Hilare2 avec sa remorque traversant un long couloir.

V.5.2 Sélection tenant compte de l'appariement des amers

Un autre aspect important pour le succès de l'exécution d'un mouvement asservi sur amers en environnement réel est de pouvoir retrouver dans l'environnement réel, et de manière continue, les amers sélectionnés lors de la phase de planification du mouvement.

Retrouver ces amers est primordial pour principalement deux raisons :
– pour éviter de grands écarts dans le calcul des erreurs de localisation successives,
– étant considérés pertinents par le critère de sélection ils doivent être pris en compte pour exécuter avec succès le chemin planifié.

S'affranchir de certains amers non pertinents mais visibles dans le processus d'asservissement de la trajectoire est judicieux pour donner plus d'attention aux amers importants. Toutefois ceci réduit la probabilité de les retrouver d'autant plus si l'environnement est riche en amer et si l'ensemble sélectionné ne constitue pas une empreinte assez discriminante permettant de retrouver ces éléments lors du processus d'appariement.

On peut s'apercevoir de cette problématique en observant le schéma de la figure V.21 représentant un plan de référence constitué des amers $\mathscr{L}_{env} = \{L_1, \ldots, L_6\}$. Dans ce plan Hilare2 avec sa remorque suit une ligne droite d'une configuration initiale $\mathbf{q}_i$ à une configuration finale $\mathbf{q}_f$. On suppose que le mouvement asservi sur amers construit à partir de cette trajectoire par l'algorithme des amers les plus proches présenté au paragraphe IV.5.3 produit pour le capteur monté à l'avant du robot une séquence locale d'amers $\mathscr{L}_{select} = \{L_3, L_4, L_5\}$ pour l'exécution de l'ensemble du chemin planifié. Or il se trouve que d'après la disposition des amers $\{L_2, \ldots, L_6\}$, les éléments de l'ensemble $\mathscr{L}_{select}$ peuvent probablement ne pas être reconnus d'autant plus si dans l'environnement d'exécution ces amers sont légèrement décalés (le pire des cas étant quand ils ont subis une translation suivant l'axe x du repère du plan). Un choix judicieux permettant

de rendre plus robuste la phase d'appariement consiste à prendre l'amer L_1 avec les séquences locales d'amers afin que l'ensemble ainsi construit présente une empreinte assez reconnaissable. Les amers utilisés pour améliorer l'appariement n'interviennent pas dans le processus de localisation. Le poids associé à de tels amers est nul tout au long de la trajectoire.

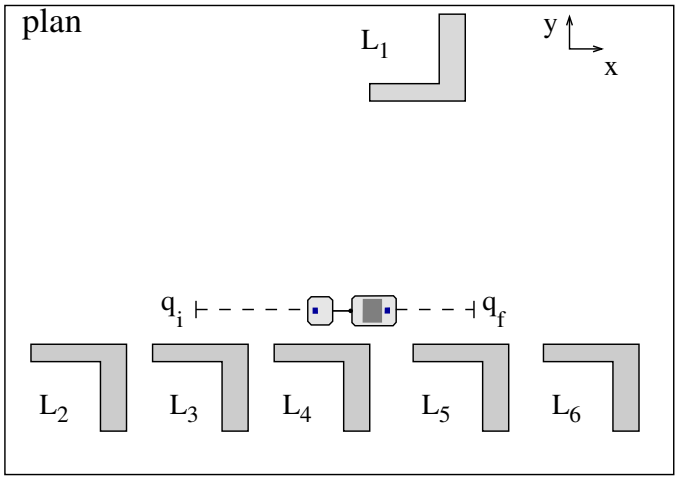

FIG. V.21 – Problème d'appariement des amers sélectionnés du plan de référence avec les amers de l'environnement réel.

L'algorithme qu'on propose pour améliorer l'appariement des amers lors de la phase d'exécution d'un mouvement asservi sur amers prend en entrée l'ensemble des amers de l'environnement $\mathscr{L}_{env}$, et un mouvement asservi sur amers. Dans un premier temps l'algorithme introduit des perturbations dans les configurations des amers des séquences locales afin de simuler des perceptions réelles. Ensuite il tente de trouver une correspondance entre les amers perturbés $\mathscr{L}_{reel}$ et les amers du plan de référence par l'intermédiaire de la fonction Appariement(). Si la totalité des amers de $\mathscr{L}_{reel}$ sont appariés avec les bons amers de $\mathscr{L}_{env}$ alors la séquence locale d'amers contient *a priori* une empreinte assez discriminante permettant de la retrouver lors d'une phase d'exécution. En résumé on fait rejouer l'exécution du chemin asservi pour des amers perturbés pour vérifier sa robustesse. On considère que si un pourcentage d'amers $\mathscr{L}_{reel}$ n'est pas retrouvé alors la procédure d'appariement échoue. Dans ce cas l'algorithme tente de compléter la séquence locale d'amers par d'autre amers de $\mathscr{L}_{env}$ pour pouvoir ainsi faire des appariements avec succès. Si l'algorithme arrive à trouver de tels amers il les ajoute aux amers de la séquence locale par la fonction Ajouter_Amer_Appariement. Cette fonction met à zéro les fonctions de pondération des amers ajoutés pour qu'ils n'interviennent pas dans le processus de localisation car ils ne servent qu'à donner plus de robustesse à l'appariement des amers.

Données : $\mathscr{L}_{env}$, maa
Résultat : maa
début
  $L \leftarrow Null$;
  $\mathscr{L}_{reel} \leftarrow Null$;
  pour $i \leftarrow 1$ ***à*** n **faire**
    pour $j \leftarrow 1$ ***à*** p_i **faire**
      $\mathscr{L}_{app} \leftarrow Null$;
      $\mathscr{L}_{select} \leftarrow$Séquence_Locale_Amers(maa_S_i, j);
      $\mathscr{L}_{reel} \leftarrow$Perturabation($\mathscr{L}_{select}$);
      tant que *Appariement*($\mathscr{L}_{reel}$, $\mathscr{L}_{env}$) *sans succès* **faire**
        $L \leftarrow$Amer_visible($\mathscr{L}_{env}$, $\mathscr{L}_{select}$);
        $\mathscr{L}_{app} \leftarrow \mathscr{L}_{app} \oplus \{L\}$;
        $\mathscr{L}_{reel} \leftarrow$Perturabation($\mathscr{L}_{select} \oplus \mathscr{L}_{app}$);
        si $\mathscr{L}_{app} \neq Null$ **alors** maa_$S_i \leftarrow$Ajouter_Amer_Appariement($\mathscr{L}_{app}$,maa_S_i);
fin

Algorithme 5 – Amélioration de l'appariement des amers d'un mouvement asservi sur amers.

D'autres problèmes ont émergé lors de la phase de validation expérimentale de notre approche. La plupart de ces problèmes produisent des effets indésirables de discontinuité de la localisation. On peut les résumer ainsi :

– Le problème d'incohérence des changements de l'environnement : ce problème se traduit par le fait que dans le système linéaire (III.5) le second membre représenté par le vecteur des écarts de perception n'appartient pas à l'espace image de la matrice de localisation W. Même si la variation lente des poids dans ce cas tente de lisser le résultat de localisation, ils ne peuvent pas prendre en charge de grands écart de perception entre l'environnement réel et le plan de référence.

– Parce que les amers ne sont pas exactement aux configurations désirées, ils peuvent apparaître ou disparaître plus tôt ou plus tard durant l'exécution du mouvement. Cet effet peut engendrer des sauts dans l'estimation de la localisation car on risque de considérer comme nul le poids d'un amer qui doit continuer d'être pris en compte (e.g. dans le cas où il disparaît plus tard que prévu). Aussi de manière analogue le robot peut commencer à considérer un amer alors que dans l'environnement réel il n'est pas encore entré dans le champ de perception du capteur. Une situation similaire se produit lorsque un des amers prévu pour l'asservissement se trouve caché par un autre.

– Au départ de l'exécution du mouvement les valeurs des fonctions de pondération ne sont pas nécessairement nulles car souvent des amers visibles sont présents dans le champ de perception des capteurs du robot. Cette situation est similaire à l'intervention brutale (avec un poids assez conséquent) d'un amer durant l'exécution de trajectoire.

Dans le paragraphe suivant nous allons montrer sur une expérimentation réalisée dans les couloirs de notre laboratoire l'implantation des algorithmes de sélection des amers qui améliorent le conditionnement de la matrice de localisation et la robustesse de l'appariement des amers planifiés avec les amers perçus.

V.6 Réalisation d'une tâche de navigation en environnement structuré

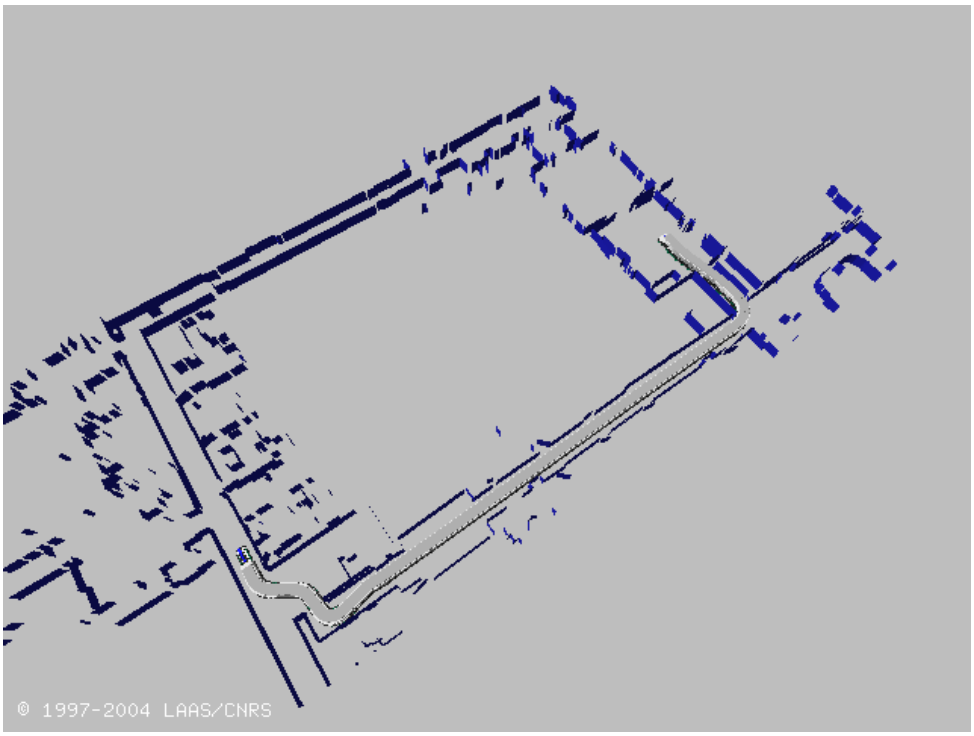

FIG. V.22 – Trajectoire planifiée.

Dans les couloirs du bâtiment du *LAAS/CNRS* nous planifions une trajectoire géométrique sans collisions d'une configuration initiale à une configuration finale à l'aide de **Move3D** (voir figure (V.22)). Ensuite, cette trajectoire ainsi que la carte sont passées en arguments d'entrée à notre planificateur de mouvement asservi sur amers afin de sélectionner et construire les fonctions de pondération des différents amers qui seront utilisés lors de l'exécution du mouvement. Notons que cette sélection sera faite uniquement pour les amers visibles par le scanner laser avant. Dans cette expérimentation, la production du mouvement asservi sur amers résulte de la réunion de trois étapes de sélection :

1. une première étape où sont sélectionnés les amers de plus grande valeur de poids tout au long du chemin planifié. Pour cela, nous utilisons l'algorithme (2, c.f § IV.5.2) avec

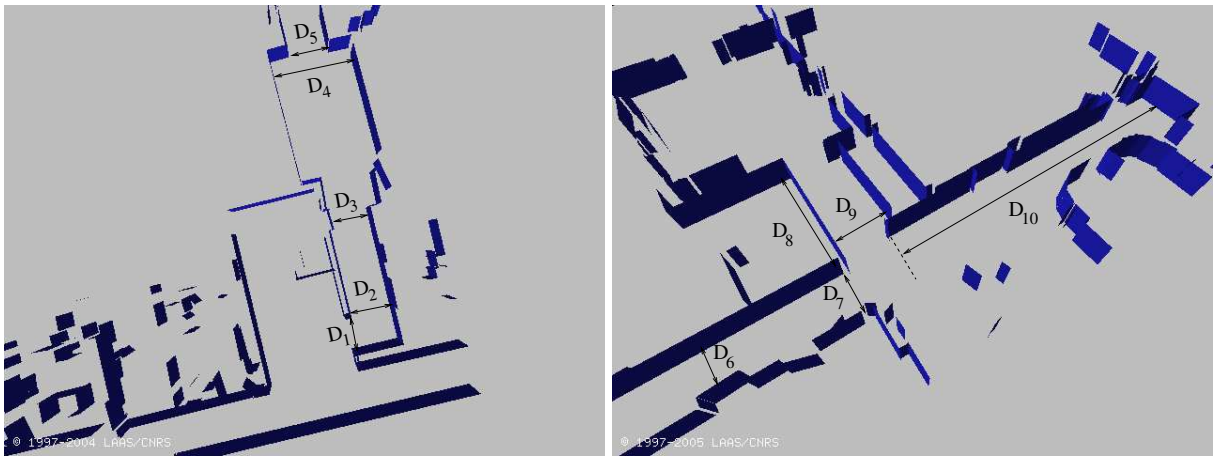

FIG. V.23 – Représentation des distances dans le plan pour les passages critiques de la tâche de navigation.

un nombre d'amers $NB_L = 2$. Ce choix pour un petit nombre d'amers nous permettra de mettre en evidence les amers sélectionnés pour le conditionnement et l'appariement.

2. la seconde étape consiste à sélectionner les amers qui améliorent le conditionnement de la matrice de localisation pondérée. Ces amers sont nécessaires surtout lorsqu'il s'agit de naviguer dans des longs couloirs comme c'est le cas pour cette expérimentation.
3. la troisième étape consiste à sélectionner les amers qui rendent plus robuste le processus d'appariement. Ces amers sont indispensables lorsque le robot se trouve dans un environnement très riche en amers car ils permettront à la procédure d'appariement de retrouver, et de manière plus robuste, les amers sur lesquels le robot a prévu de s'asservir.

Le plan dont nous disposons n'est pas exact et le tableau V.1 illustre certaines incohérences entre ce plan et l'environnement d'exécution. Les distances qui y sont mentionnées sont représentées dans la figure (V.23). En moyenne l'écart entre notre modèle et la réalité est d'une dizaine de centimètre. De plus les murs des couloirs ne sont pas parfaitement parallèles et peuvent donc induire une erreur en orientation et/ou un mauvais conditionnement du système de localisation (dans le cas où la localisation est calculée uniquement par rapport aux côtés du couloir). L'erreur issue de la localisation globale à la configuration initiale de la trajectoire et l' accumulation de l'erreur odométrique ne permet pas d'exécuter correctement cette tâche de navigation. Le problème est d'autant plus critique que la taille du robot par rapport à l'environnement de navigation contraint fortement les mouvements de ce dernier. Nous allons à présent discuter de certains passages critiques lors de l'exécution du mouvement asservi sur amers obtenu après les trois étapes décrites plus haut dans cette section (voir figures (V.24) et (V.25). Les figures de la colonne de droite représentent des situations du robot au cours de la navigation. Les figures de la colonne de gauche représentent les poids associés aux amers illustrés dans la colonne de droite).

Distance (m)	Plan	Exécution
D_1	1.296	1.210
D_2	1.570	1.545
D_3	1.320	1.260
D_4	3.080	3.040
D_5	1.520	1.475

Distance (m)	Plan	Exécution
D_6	1.400	1.260
D_7	1.600	1.340
D_8	4.023	3.945
D_9	1.886	1.735
D_{10}	11.373	11.200

TAB. V.1 – Tableau représentant les distances entre certains amers dans le plan et les distances correspondantes dans l'environnement réel (c.f figure (V.23)).

La figure (V.24) du haut, montre le premier passage délicat, il faut noter que même si l'erreur due à l'odométrie n'est pas importante, les incohérences entre le plan et l'environnement peuvent faire échouer la tâche de navigation. Dans ce passage, les amers sélectionnés dans la première étape de construction du mouvement asservi sur amers suffisent à bien conditionner la matrice de localisation. Les segments seg_{197}, seg_{232}, seg_{235} et seg_{236} avec des poids nuls sont des amers d'appariement qui aident à retrouver les amers de localisation. Le passage réel étant plus étroit que le passage dans le plan (voir valeur de la distance D_1 (c.f tableau V.1), les segments seg_{165} et seg_{244} ont des poids très importants pour pouvoir traverser ce passage délicat.

La figure (V.24) du milieu, illustre un passage où il est nécessaire de sélectionner des amers pour améliorer le conditionnement de la matrice de localisation, dans ce cas ces amers n'ont pas des poids pertinents par rapport aux amers sélectionnés dans la première étape de construction du mouvement asservi sur amers. L'algorithme n'a sélectionné au départ que des amers parallèles ou quasiment parallèles (e.g seg_{560} et seg_{258}), la procédure d'amélioration du conditionnement ajoute alors les amers seg_{259} et seg_{260}.

La figure (V.24) du bas, représente le cas typique d'un long couloir où le robot ne dispose que des murs des deux côtés qui ne permettent pas de déterminer la position longitudinale du véhicule. Même si le mouvement asservi sur amers prévoit d'utiliser le segment seg_{587}, référencé dans la figure (V.25), le robot ne le prend pas en compte car il n'a pas pu être apparié. Dans ce cas le robot continue à avancer dans le couloir en corrigeant sa configuration suivant le sens transversal et sans pouvoir le faire dans la direction longitudinale.

La figure (V.25) du haut, montre que le robot a enfin réussi à apparier le segment seg_{587} qui va lui permettre de se recaler dans le sens longitudinal et préparer le virage à gauche qu'il va effectuer ensuite. Dans cette situation, le robot traverse un passage où le couloir devient plus étroit et où, de plus, dans la réalité il est plus étroit que dans le plan (c.f valeur de D_6 dans le tableau V.1). On remarque que les segments du coin seg_{575} et seg_{576} avec des poids nuls servent à améliorer l'appariement.

La figure (V.25) du bas, illustre le dernier passage délicat de la tâche de navigation du robot. Ce passage se caractérise par une double incohérence entre le plan et l'environnement réel :

– la première par rapport à la distance D_{10} (tableau V.1) qui rend plus proche le segment seg_{587} dans la réalité que dans le plan.

– la seconde concerne le fait que le passage suivant le virage est plus étroit dans la réalité que dans le plan.

Aussi, la taille du robot par rapport à la taille de l'espace libre de l'environnement rend cette manoeuvre très difficile.

Au début de la réalisation de cette expérimentation, le succès de cette manoeuvre n'était pas systématique. Ceci est principalement dû au fait qu'au départ le segment seg_{580} était un segment de localisation mais la présence de certains amers non modélisés dans le plan (principalement des chaises et une table) amenaient la procédure de localisation à le confondre avec ces obstacles non modélisés. Pour pouvoir systématiser la réussite de cette manoeuvre nous avons manuellement mis à zéro le poids de seg_{580} afin qu'il n'intervienne pas dans la localisation. Ceci montre à quel point les amers de localisation sont important pour ce type de manoeuvre et à quel point il est aussi important de pouvoir les retrouver pendant l'exécution. Cette étape utilise beaucoup de segments aidant à l'appariement car l'environnement y est plus riche en amers non sélectionnés par rapport au cas précédents.

V.7 Conclusion

Dans ce chapitre nous avons présenté l'architecture logicielle orientée objet que nous avons développée afin de mettre en oeuvre notre formalisme de façon générique. Nous avons montré comment nous avons instancié les différents objets et leurs méthodes pour construire des mouvements asservis sur amers pour le robot mobile Hilare2 tractant une remorque. Nous avons présenté l'architecure des modules de contrôle de mouvement embarquée sur ce robot. A cette plate-forme nous avons intégré un nouveau module qui prend en entrée un mouvement asservi sur amers retourné par notre logiciel et l'utilise pour asservir l'exécution de la trajectoire planifiée sur les primitives planifiées.

Nous avons présenté deux manoeuvres que nous avons réalisées avec le robot réel afin de valider la pertinence de notre approche. Ces expérimentations nous ont permis d'avoir un retour d'expérience et d'envisager ainsi des améliorations à nos algorithmes pour pouvoir faire face à des cas plus complexes. Nous avons alors implanté ces améliorations dans notre architecture logicielle et réalisé une expérimentation de navigation permettant de les valider. Les différentes expérimentations ont permis de mettre en évidence le problème d'incohérence des changements de l'environnement par rapport au modèle utilisé en phase de planification.

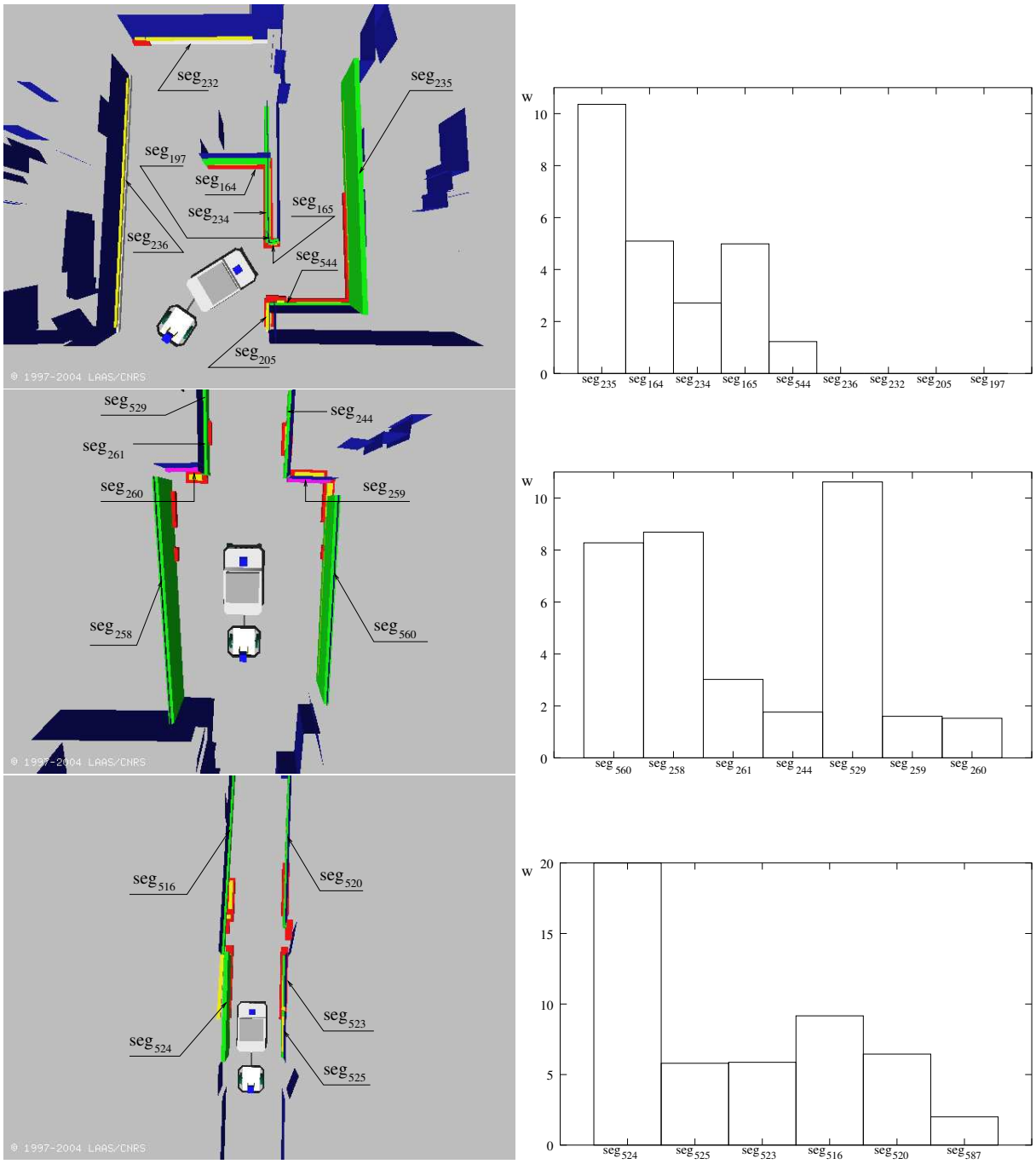

FIG. V.24 – Premiers passages critiques de la tâche de navigation. Les segments verts représentent les segments sélectionnés lors de la première étape de planification. Les segments violets représentent les segments ajoutés lors de l'étape de l'amélioration du conditionnement. Les segments blancs représentent les segments ajoutés lors de l'étape de l'amélioration de l'appariement. Les segments rouges sont les segments perçus par le robot dans l'environnement d'exécution. Les segments avec une pastille jaune sont les segments que la procédure d'appariement a réussi à faire correspondre. Les figures de la colonne de droite représentent les poids instantanés associés aux amers du mouvement (représentés à droite).

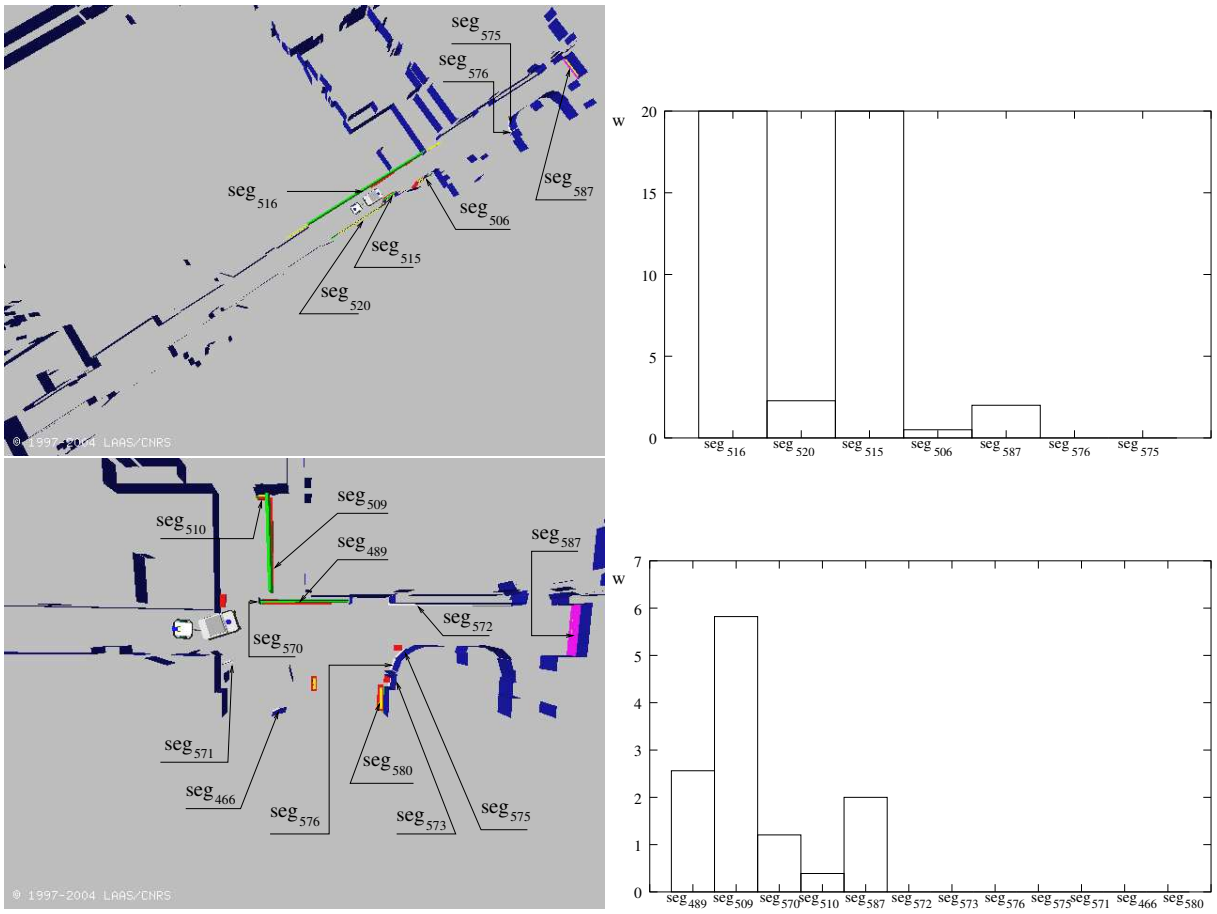

FIG. V.25 – Derniers passages difficiles de la tâche de navigation. Les segments verts représentent les segments sélectionnés lors de la première étape de planification. Les segments violets représentent les segments ajoutés lors de l'étape de l'amélioration du conditionnement. Les segments blancs représentent les segments ajoutés lors de l'étape de l'amélioration de l'appariement. Les segments rouges sont les segments perçus par le robot dans l'environnement d'exécution. Les segments avec une pastille jaune sont les segments que la procédure d'appariement a réussi à faire correspondre. Les figures de la colonne de droite représentent les poids instantanés associés aux amers du mouvement (représentés à droite).

Chapitre VI

Conclusion

Le travail présenté dans ce mémoire de thèse initie une nouvelle approche dans le domaine de la planification de mouvement en robotique. La contribution principale de l'approche présentée réside dans la définition générique du mouvement asservi sur amers. Cette définition présente un tel mouvement pour un robot évoluant dans un environnement, dont le plan est supposé connu, comme un chemin géométrique sans collision reliant la configuration de départ à la configuration d'arrivée et une liste de primitives capteurs-amers pondérées sur lesquelles s'asservit le robot lors de l'exécution de son mouvement.

Cet asservissement est présenté comme une régulation de la configuration $\mathbf{q}$ du robot dans l'environnement d'exécution autour de la configuration de référence $\mathbf{q}_0$ appartenant au chemin géométrique planifié. Ceci nous a permis dans un premier temps de considérer que la configuration de localisation $\mathbf{q}_{loc}$ se trouve dans un voisinage de la configuration de référence $\mathbf{q}_0$. Dans un second temps de démontrer que notre asservissement atteint une configuration *asymptotiquement localement stable* dans un voisinage de cette configuration de référence.

L'appartenance de $\mathbf{q}_{loc}$ au voisinage de $\mathbf{q}_0$ nous a permis de linéariser l'équation de localisation autour de la configuration de référence afin de pouvoir formuler l'écart de localisation comme une transformation linéaire de l'écart de perception entre le plan de référence et l'environnement d'exécution. Nous avons ainsi formulé le problème de localisation comme un problème d'estimation par des moindres carrés que nous avons pondérés afin d'attribuer à chaque erreur de perception un poids proportionnel à la pertinence de chaque paire capteur-amer.

Nous avons démontré, sous certaines conditions liées à la matrice de transformation linéaire de localisation, qu'il *existe* une *unique* configuration, appelée *configuration corrigée* $\mathbf{q}_c$, vers

laquelle le robot converge dans son environnement réel. Cette configuration vérifie certaines propriétés notamment le fait que pour une localisation donnée, l'écart de configuration pour atteindre la configuration corrigée est l'*opposé* de l'écart entre la configuration de localisation et la configuration de référence. Une seconde propriété stipule que le résidu de perception en $\mathbf{q}_c$ est égal au résidu de perception en $\mathbf{q}_{loc}$.

Au cours de ce travail nous avons aussi abordé la question critique de la construction des fonctions de pondération qui constituent une spécification intrinsèque au mouvement asservi sur amers. Cette construction présente de telles fonctions comme le produit de sous fonctions continues et positives définies sur une partie de l'espace issu du produit cartésien de l'espace des configurations du robot (ou espace des configurations du capteur) et de l'espace des configurations de l'amer pour lequel on calcule cette fonction. Chacune de ces sous fonctions représente un aspect particulier de la pertinence de cet amer. Nous avons proposé dans cette formalisation deux sous fonctions :

– la première sous fonction prend en charge la visibilité de l'amer par le capteur qui lui est associé dans la primitive capteur-amer. Cette fonction attribue un poids décroissant aux amers qui tendent à s'éloigner du champs de perception du capteur.

– la seconde sous fonction prend en charge le danger de collision que peut représenter un amer par rapport à la configuration du robot. Cette sous fonction se présente comme la fonction de répulsion assurant l'évitement d'obstacle dans les méthodes de mouvement réactif.

La combinaison de ces deux sous fonctions donne un plus grand poids aux amers les plus visibles et qui constituent un danger de collision pour le robot.

Ces fonctions sont définies sur un espace issu du produit cartésien de deux espaces de configuration. Le calcul des différentes fonctions de pondération pour une trajectoire donnée est alors nécessaire pour chaque paire capteur-amer. À cet effet nous avons proposé trois algorithmes différents. Le premier algorithme calcule essentiellement ces fonctions pour l'ensemble des amers perçus le long de la trajectoire planifiée. Le second et le troisième algorithme définissent des critères de sélection des amers perçus basés sur l'importance de ces derniers. Par ailleurs, ces algorithmes définissent un critère d'enchaînement des amers sélectionnés qui consiste à dire que «*si un amer est pertinent selon un certain critère de sélection alors il faut le prendre en compte dès qu'il entre dans le champs de perception du capteur (i.e. dès que son poids est différent de zéro)*».

Ces algorithmes produisent un mouvement asservi sur amers comme un ensemble de séquences d'enchaînement d'amers associés aux différents capteurs du robot. Chaque séquence d'enchaînement d'amers est composée d'une succession de séquences locales d'amers. Une séquence locale d'amer contient les informations nécessaires pour la reconstitution de la fonction de pondération durant l'exécution du mouvement.

Les différentes spécifications formelles présentées ci dessus ont été intégrées dans une architecture logicielle orientée objet afin de tirer profit de la généricité de notre approche. Nous avons ainsi implémenté quatre classes abstraites : robot, capteur, amer et paire-capteur-amer. Nous

avons défini la gestion de l'interaction entre ces différentes classes et les méthodes génériques permettant de mettre en oeuvre notre approche. L'intégration des algorithmes a permis d'aboutir à une plate-forme qui admet en entrée un chemin géométrique sans collision et un plan de l'environnement et qui retourne en sortie un mouvement asservi sur amers. Cette plate-forme nous a permis dans un premier temps de valider notre approche en simulation et dans un second temps de faire une validation expérimentale réelle.

Les deux validations sont effectuées sur le robot mobile non-holonome Hilare2 tractant une remorque. Ce robot dispose de deux scanners laser embarqués ; le premier monté à l'avant sur la partie mobile et le second est monté à l'arrière sur la remorque. Les tâches de navigation de ce robot sont gérées par des modules construit par le logiciel de prototypage temps-réel GenoM développé au sein de notre laboratoire. L'architecture des modules embarqués sur le robot ne prend en charge que l'exécution de la trajectoire planifiée. La phase de validation expérimentale a nécessité la génération d'un nouveau module prenant en charge les spécifications de notre approche. Notre nouveau module a été intégré au sein de l'architecture existante afin d'asservir le robot sur les primitives de l'environnement produites par notre logiciel de planification de mouvement asservi sur amers.

Nous avons présenté dans ce manuscrit trois expérimentations permettant de valider la pertinence de notre approche. Une première expérience montre comment le robot adapte son mouvement quand il passe dans un couloir qui est plus étroit que le couloir représenté dans son plan de référence. La seconde expérience montre comment le robot modifie son mouvement pour mener à bien une manoeuvre de garage où ce dernier n'est pas à sa position désirée. La troisième qui est une tâche de navigation en environnement structuré a permis de tester notre approche à une plus grande échelle.

Les résultats que nous avons obtenus sont encourageants mais néanmoins présentent quelques inconvénient dont le principal est le fait de se baser sur un chemin géométrique dont la planification n'a tenu compte que des obstacles et des contraintes cinématique. En effet il serait plus intéressant de prendre en compte des informations de perception comme une spécification d'entrée dans notre formalisme. Ces quelques inconvénients nous permettent de proposer les prospectives suivantes.

Prospective de notre travail

Le travail présenté ici, comme tout travail de recherche, est loin d'être complet ou fini. Ainsi, plusieurs axes de recherche sont envisageables. Certains ont pour objectif d'améliorer les fonctionnalités déjà proposées au cours de ce travail, d'autres explorent de nouvelles pistes que nous mentionneront plus loin dans ce paragraphe.

Un des points de ce travail qu'il faudra certainement améliorer concerne les algorithmes proposés qui ont pour objectif principal la construction du mouvement asservi sur amers. Il est nécessaire d'optimiser le temps de calcul de ces algorithmes afin de pouvoir leur ajouter d'autres

fonctionnalités tout en gardant un temps de calcul raisonnable.

Il serait intéressant de gérer les problèmes liés à la perte d'un amer lors de l'exécution du mouvement. Pour cela, nous proposons de définir des fonctions calculées en temps réel qui diminueraient le poids des amers qui risquent de disparaître du champs de perception du capteur. Nous résoudrons ainsi, par exemple, le problème des amers qui disparaissent plus tôt que prévu ou des amers qui sont occultés par d'autres amers que le plan ne prend pas en compte.

Pour pouvoir introduire convenablement les contraintes cinématiques incombant aux systèmes robotiques il est nécessaire de définir de manière plus rigoureuse la dépendance entre la dérivée temporelle de la fonction de pondération $\frac{\partial w}{\partial t}$ et la dérivée des variables de configuration $\frac{\partial \mathbf{q}}{\partial t}$.

Par la suite, il faudrait faire des tests de validation expérimentale sur la généricité de l'approche en l'appliquant sur d'autres types de robots, d'autres types de capteurs et d'autres types d'amers.

Parmi les nouvelles pistes que nous souhaitons explorer par la suite, définir les configurations initiales et finales par rapport à des primitives pertinentes. Ceci nous permettra alors d'aborder d'un autre point de vue le problème de la planification de trajectoire. En effet, au lieu de s'appuyer sur un chemin géométrique déjà construit pour planifier un mouvement asservi sur amers il serait plus judicieux de construire un chemin géométrique qui tient compte de la pertinence des amers. Cette planification s'effectuerait alors dans l'espace issu du produit cartésien de l'espace des configurations du robot (ou espace des configurations du capteur) et de l'espace des configurations des amers avec une heuristique permettant de maximiser les poids des primitives capteurs-amers. Une telle formalisation pourra alors facilement être intégrée à une architecture bas niveau de commande référencée capteur.

Nous envisagerons alors de porter ce formalisme sur des applications plus larges telles que la navigation autonome de voiture en milieu urbain pour aider les conducteurs à se garer plus facilement dans des situations très contraintes, rendre plus souple la tâche d'amarrage d'un camion de livraison sur le portail d'un hangar ou encore suivre des itinéraires en plein centre ville. Ce formalisme peut aussi apporter des solutions aux tâches qui nécessitent beaucoup de dextérité comme les applications robotiques dans le domaine médical.

Références bibliographiques

[1] K.O. Arras, J. Persson, N. Tomatis, and R. Siegwart. Real-time obstacle avoidance for polygonal robots with a reduced dynamic window. In *IEEE International Conference on Robotics and Automation*, pages 3050–3054, Washington, DC, May 2002.

[2] I. Ashokaraj, A. Tsourdos, P. Silson, and B. A. White. Sensor based robot localisation and navigation : Using interval analysis and unscented kalman filter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7–12, Sendai, Japan, October 2004.

[3] N. Ayache and O. D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, 5(6) :819–804, December 1989.

[4] J. Barraquand and J.C. Latombe. Robot motion planning : a distributed representation approach. *IEEE International Journal of Robotics Research*, 10(6) :628–649, 1991.

[5] M. Betke and L. Gurvits. Mobile robot localization using landmarks. *IEEE Transactions on Robotics and Automation*, 13(2) :251–263, April 1997.

[6] V. Boor, M. Overmars, and A.F. van der Stapen. The gaussian sampling strategy for probabilistic roadmap planners. In *IEEE International Conference on Robotics and Automation*, pages 1018–1023, 1999.

[7] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe. Mobile robot positioning : Sensors and techniques. *Invited paper for the Journal of Robotic Systems, Special Issue on Mobile Robots*, 14(4) :231–249, April 1997.

[8] B. Bouilly. *Planification de Stratégies de Déplacement Robustes pour un Robot Mobile*. PhD thesis, INPT, Toulouse, France, 1997.

[9] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *IEEE International Conference on Robotics and Automation*, pages 341–346, Detroit, Michigan, May 1999.

[10] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *International Conference on Robotics and Automation*, San Francisco, CA, April 2000. IEEE.

[11] O. Brock and O. Khatib. Real-time replanning in high dimensional configuration spaces using sets of homotopic paths. In *International Conference on Robotics and Automation*, pages 550–555, San Francisco, CA, April 2000. IEEE.

[12] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1) :14–23, Mars 1986.

[13] H. Bulata and M. Devy. Incremental construction of a landmark-based and topological model of indoor environments by a mobile robot. In *IEEE International Conference on Robotics and Automation*, pages 1054–1060, Mineapolis, Minnesota, April 1996.

[14] Wolfram Burgard, Dieter Fox, Daniel Hennig, and Timo Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *The Thirteenth National Conference on Artificial Intelligence*, volume 2, pages 896–901, Portland, USA, 1996.

[15] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien. Acting under uncertainty : discrete bayesian models for mobile-robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 963–972, 1996.

[16] R. Chatila and J. Laumond. Position referencing and consistent world modeling for mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 138–145, Mars 1985.

[17] W. Choi and J.C. Latombe. A reactive architecture for planning and executing robot motions with incomplete knowledge. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 24–29, Osaka, Japan, 1991.

[18] Juan Cortés. *Algorithmes pour la Planification de Mouvement de Mécanismes Articulés avec Chaînes Cinématiques Fermées*. PhD thesis, INP, Toulouse, France, 2003.

[19] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 1322–1328, Detroit, Michigan, May 1999.

[20] X. Deng, E. Milios, and A. Mirzaian. Landmark selection for path execution. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1339– 1346, Yokohama, Japan, July 1993.

[21] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3) :313–326, 1992.

[22] B. Faverjon. Obstacle avoidance using an octree in the configuration space of a manipulator. In *IEEE International Conference on Robotics and Automation*, pages 504–512, Mars 1984.

[23] S. Fleury, M. Herrb, and R. Chatila. Genom : a tool for the specification and the implementation of operating modules in a distributed robot architecture. In *International Conference on Intelligent Robots and Systems*, pages 842–848, Grenoble, France, September 1997. IEEE/RSJ.

[24] N. Le Fort-Piat, I. Collin, and D. Meizel. Planning robust displacement missions by means of robot-tasks and local maps. *Robotics and Autonomous Systems*, 20 :99–114, 1997.

[25] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3/4) :195–207, 1998.

[26] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11 :391–427, 1999.

[27] Fabien Gravot. *ASYMOV : Fondation d'un Planificateur Robotique Intégrant le Symbolique et le Géométrique*. PhD thesis, Université Paul Sabatier, Toulouse, France, 2004.

[28] R. Greiner and R. Isukapalli. Learning to select useful landmark. *IEEE Transactions on Systems, Man and Cybernetics–Part B : Cybernetics*, 26(3) :437–449, 1996.

[29] G. Z. Grudic and P. D. Lawrance. A nonparametric learning approach to vision based mobile robot localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 724–729, 1998.

[30] J. S. Gutmann. Markov-kalman localization for mobile robots. In *International Conference on Pattern Recognition*, pages 601–604, Quebec, Canada, August 2002.

[31] J. S. Gutmann and D. Fox. An experimental comparison of localization methods continued. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 454–459, EPFL, Lausanne, Switzerland, October 2002.

[32] J. S. Gutmann and C. Schlegel. Amos : comparison of scan matching approaches for self-localization in indoor environments. In *1st Euromicro Workshop on Advanced Mobile robots, IEEE computer Society Press*, 1996.

[33] L. Jetto, S. Longhi, and G. Venturini. Development and experimental kalman filter for the localization of mobile robots. *IEEE Transactions on Robotics and Automation*, 15(2) :219–229, April 1999.

[34] S. Julier and J. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, Orlando, FL, 1997.

[35] T. Kanbara, J. Miura, and Y. Shirai. Selection of efficient landmarks for an autonomous vehicle. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1332– 1338, Yokohama, Japan, July 1993.

[36] L. Kavraki, P. Svestka, J.C. Latombe, and M. Overmars. Probabilistic roadmaps for path planningin in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4) :566–580, 1996.

[37] M. Khatib, H. Jaouni, R. Chatila, and J.-P. Laumond. Dynamic path modification for car-like nonholonomic mobile robots. In *International Conference on Robotics and Automation*, pages 2920–2925, Albuquerque, NM, April 1997. IEEE.

[38] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *IEEE International Journal of Robotic Research*, 5(1) :90–98, 1986.

[39] K. Komoyira, E. Oyama, and K. Tani. Planning of landmark measurement for the navigation of a mobile robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1476–1481, Raleigh, California, July 1992.

[40] D. J. Kriegman, E. Triendl, and T. O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Transactions on Robotics and Automation*, 5(6) :792–803, December 1989.

[41] J.J. Kuffner and S.M. LaValle. RRT-connected : An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, San Francisco, 2000.

[42] Cody Kwok, D. Fox, and Marina Meila. Adaptive real-time particle filters for robot localization. In *IEEE International Conference on Robotics and Automation*, pages 2836–2841, Taipei, Taiwan, Mars 2003.

[43] F. Lamiraux. *Robots mobiles a‘ remorque : de la planification de chemins à l'exécution de mouvements*. PhD thesis, INPT, LAAS-CNRS, Toulouse, France, 1997.

[44] F. Lamiraux, D. Bonnafous, and O. Lefebvre. Reactive path deformation for nonholonomic mobile robots. *IEEE Transactions on Robotics*, 20(6) :967–977, Dec 2004.

[45] F. Lamiraux and J.P. Laumond. Smooth motion planning for car-like vehicles. *IEEE Transactions on Robotics and Automation*, 17(2) :498–501, 2001.

[46] F. Lamiraux, S. Sekhavat, and J.-P. Laumond. Motion planning and control for hilare pulling a trailer. *IEEE Transactions on Robotics and Automation*, 15(4) :640–652, August 1999.

[47] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[48] J.-P. Laumond, editor. *Robot Motion Planning and Control*. Number ISBN 3-540-76219-1 in Lectures Notes in Control and Information Sciences 229. Springer, 1998.

[49] J.P. Laumond. *La Robotique Mobile*. Hermes science, Paris, France, 2001.

[50] J.P. Laumond, P. Jacobs, M. Taïx, and R. Murray. A motion planner for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 10(5) :577–584, 1994.

[51] J.P. Laumond, S. Sekhavat, and F. Lamiraux. *Guidelines in Nonholonomic Motion Planning for Mobile Robots*. J.P. Laumond, Robot Motion Planning and Control, pages 1-53. Springer-Verlag, 1998.

[52] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. In *IEEE International Conference on Robotics and Automation*, 1999.

[53] J.Y. Lee and H. Choset. Sensor-based exploration for convex bodies : A new roadmap for a convex-shaped robot. *IEEE Transactions on Robotics*, 21(2) :240–247, April 2005.

[54] J. J. Little, J. Lu, and D. R. Murray. Selecting stable image features for robot localization using stereo. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1072–1077, 1998.

[55] T. Lozano-Pérez. Spatial planning : A configuration space approach. *IEEE Transaction on computer*, 32(2) :108–120, 1983.

[56] A. De Luca, G. Oriolo, and C. Samson. *Robot Motion Planning and Control*, chapter Feedback Control of a Nonholonomic Car-like Robot, pages 171–253. Lecture Notes in Control and Information Sciences. Springer, NY, 1998.

[57] A.C. Malti, F. Lamiraux, and M. Taix. Sensor landmark motion planning in mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004.

[58] A.C. Malti, F. Lamiraux, and M. Taix. Sensor landmark succession for motion planning along a planned trajectory. In *IEEE Conf. on Mechatronics and Robotics*, Aachen, Germany, 2004.

[59] N. Mansard and F. Chaumette. Tasks sequencing for visual servoing. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 992–997, Sendai, Japan, October 2004.

[60] N. Mansard and F. Chaumette. Visual servoing sequencing able to avoid obstacles. In *IEEE International Conference on Robotics and Automation*, pages 3154–3159, Barcelona, Spain, April 2005.

[61] Y. Mezouar and F. Chaumette. Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, 18(4) :534–549, 2002.

[62] J. Minguez and L. Montano. Nearness diagram (nd) navigation : collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20(1) :45–59, Feb 2004.

[63] J. Minguez, L. Montano, T. Siméon, and R. Alami. Global nearness diagram navigation (gnd). In *International Conference on Robotics and Automation*, pages 33–39, Seoul KR, May 2001. IEEE.

[64] P. Moutarlier. *Modélisation autonome de l'environnement par un robot mobile*. PhD thesis, Université Paul Sabatier, Toulouse, France, 1991.

[65] R. R. Murphy, D. Hershberger, and G. R. Blauvelt. Learning landmark triples by experimentation. *Robotics and Autonomous Systems*, 22(3/4) :377–392, 1997.

[66] J. Neira, J. D. Tardos, J. Horn, and Gunther Schmidt. Fusing range and intensity images for mobile robot localization. *IEEE Transactions on Robotics and Automation*, 15(1) :76–84, February 1999.

[67] C. Olson. Landmark selection for terrain matching. In *IEEE International Conference on Robotics and Automation*, pages 1447– 1452, San Francisco, California, May 1999.

[68] C. F. Olson and L. H. Matthies. Maximum likelihood rover localization by matching range maps. In *IEEE International Conference on Robotics and Automation*, pages 272–277, Leuven, Belgium, 1998.

[69] Clark F. Olson. Probabilistic self-localization for mobile robots. *IEEE Transactions on Robotics and Automation*, 16(1) :55–66, February 2000.

[70] Julien Pettré. *Planification de Mouvement de Marche pour Acteurs Digitaux*. PhD thesis, Université Paul Sabatier, Toulouse, France, 2003.

[71] R. Pissard-Gibollet. *Conception et commande par asservissement visuel d'un robot mobile*. PhD thesis, Ecole des mines, Paris, France, 1993.

[72] S. Quinlan and O. Khatib. Elastic bands : Connecting path planning and control. In *IEEE International Conference on Robotics and Automation*, pages 802–807, Atlanta,, USA, 1993.

[73] S. Quinlan and O. Khatib. Elastic bands : Connecting path planning and control. In *International Conference on Robotics and Automation*, pages 802–807, Atlanta, GA, May 1993. IEEE.

[74] S. I. Roumeliotis and G. A. Bekey. Bayesian estimation and kalman filtering : A unified framework for mobile robot localization. In *IEEE International Conference on Robotics and Automation*, pages 2985–2992, San Francisco, California, 2000.

[75] Anis Sahbani. *Planification de Tâches de Manipulation en Robotique par des Approches Probabiliste*. PhD thesis, Université Paul Sabatier, Toulouse, France, 2003.

[76] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control The Task Function Approach*. Oxford science publications, 1991.

[77] B. Schiele and J. L. Crowley. A comparison of position estimation techniques using occupancy grids. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1628–1634, 1994.

[78] J.T. Schwartz and M. Sharir. On the piano mover's problem : I. the case of a two-dimensional rigid polygonal body moving admist polygonal barriers. *Communications on Pure and Applied Mathematics*, 36 :345–398, 1983.

[79] R. Sim and G. Dudek. Mobile robot localization from learned landmarks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1060–1065, 1998.

[80] S. Simhon and G. Dudek. Selecting targets for local reference frames. In *IEEE International Conference on Robotics and Automation*, pages 2840–2845, 1998.

[81] Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

[82] T. Siméon, J-P. Laumond, and F. Lamiraux. Move3d : a generic platform for path planning. In *IEEE Int. Symp. on Assembly and Task Planning*, pages 25–30, 2001.

[83] T. Siméon, J.P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics Journal*, 14(6) :477–494, 2000.

[84] K. T. Sutherland and W. B. Thompson. Localizing in unstructured environments. *IEEE Transactions on Robotics and Automation*, 10(6) :740–754, 1994.

[85] K. T. Sutherland and W. B. Thompson. Pursuing projections : Keeping a robot on path. In *IEEE International Conference on Robotics and Automation*, pages 3355–3361, May 1994.

[86] R. Swain-Oropeza, Michel Devy, and S. Hutchinson. Sensor-based navigation in cluttered environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1662–1669, Maui, Hawaï, October 2001.

[87] H. Takeda, C. Facchinetti, and J.C. Latombe. Planning the motions of a mobile robot in a sensory uncertainty field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10) :1002–1017, 1994.

[88] S. Thrun. Bayesian landmark learning for mobile robot localizzation. *Machine Learning*, 33(1) :41–76, 1998.

[89] Ching-Chih Tsai. A localization system of a mobile robot by fusing dead-reckoning and ultrasonic measurements. *IEEE Transactions On Instrumentation and Measurement*, 47(5) :1399–1404, October 1998.

[90] I. Ulrich and J. Borenstein. Vfh+ : Reliable obstacle avoidance with look-ahead verification. In *IEEE International Conference on Robotics and Automation*, pages 1572–1577, 1998.

[91] N. Vandapel and R. Chatila. Affine trackability for landmark selection in natural environnement. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 37–42, Lausanne, Switzerland, Octobre 2002.

[92] A.C. Victorino and Patrick Rives. A relative motion estimation by combining laser measurement and sensor based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3924–3929, Washington, DC, May 2002.

[93] Dejun Wang, Jiali Zhao, and Seok Cheol Kee. A novel heat kernel based monte carlo localization algorithm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2494–2499, Sendai, Japan, 2004.

[94] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization by combining an image-retrieval system with monte carlo localization. *IEEE Transactions on Robotics*, 21(2) :208–216, April 2005.

[95] E. Yeh and D. J. Kriegman. Toward selecting and recognizing natural landmarks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 47–53, 1995.

[96] Y. Yu and K. Gupta. Sensor-Based probabilistic roadmaps : experiments with an eye-in-hand system. *Advanced Robotics*, 41(6) :515–536, 2000.

RÉSUMÉ :

Planifier un chemin géométrique pour un robot d'une configuration initiale à une configuration finale est aujourd'hui un problème quasiment résolu moyennant une représentation géométrique de l'environnement statique du robot, une modélisation de la chaîne cinématique du robot et de ses contraintes cinématiques. L'exécution de tels chemins en environnement réel est en revanche un problème qui est loin d'être résolu malgré une littérature fournie sur le sujet. De nombreuses raisons expliquent cette difficulté parmi lesquelles l'inexactitude des modèles d'environnement utilisés et des moyens de localisation. L'objectif de notre travail est de proposer une approche générique de planification de mouvements référencés sur des amers. Le principe de notre approche consiste à associer à une trajectoire géométrique sans collision des couples amers-capteurs qui pendant l'exécution sont utilisés pour asservir localement le mouvement du robot. Cette approche nous permet de produire des mouvements sûrs en donnant plus d'importance aux amers qui ont de bonnes propriétés de localisation ou bien à ceux qui représentent un danger de collision. Des résultats expérimentaux réalisés sur un robot mobile non holonome de type Hilare avec remorque valident notre approche.

SUMMARY :

Robot motion execution is a difficult task for mainly three reasons. The first one comes from the high complexity of the path planning problem. The second one resides in the inaccuracy of the map of the environment used to plan the trajectory and the third reason is that the motion task in cluttered environments requires precise localization. The first of these three issues has raised a lot of interest for the past fifteen years and solutions have been proposed to solve the path planning problem for simple or complex kinematic systems. Our work deals with the two last issues. The generic approach we propose aims at producing motion features composed of a reference trajectory and a set of sensor-landmark pairs. These motion features define along the planned trajectory closed-loop motion strategies for the robot. We develop a generic strategy planning algorithms within a software platform to select the most relevent landmarks. These strategies have to take into account obstacles that represent a danger of collision and landmarks that yield a good localization. Experimental results on-board mobile robot Hilare 2 towing a trailer validate our approach as well for maneuvers like parking as for navigation task in cluttered environment.