Navigation autonome sans collision pour robots mobiles nonholonomes

Olivier Lefebvre

Thèse

préparée au

Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

en vue de l'obtention du

Doctorat de l'Institut National Polytechnique de Toulouse

par

Olivier Lefebvre

Navigation autonome sans collision pour robots mobiles nonholonomes

---

Soutenue le 12 juillet 2006 devant le jury composé de :

M. Florent Lamiraux	Directeur de thèse
M. Tarek Hamel	Rapporteur
M. Fawzi Nashashibi	Rapporteur
M. Pierre Rouchon	Rapporteur
M. Jean-Paul Laumond	Président
M. Javier Minguez	Examinateur

LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex 4

Remerciements

Sur cette page qui sera probablement la plus consultée de ce mémoire, je tiens tout d'abord à remercier les personnes qui ont bien voulu se laisser convaincre par ma motivation et qui m'ont permis de réaliser cette thèse, Florent Lamiraux et Raja Chatila.

Je remercie chaleureusement les rapporteurs de mon mémoire de thèse, Tarek Hamel, Fawzi Nashashibi et Pierre Rouchon, ainsi que le membre invité du jury de soutenance, Javier Minguez.

Je souhaite à nouveau remercier Florent qui fut mon directeur de thèse ; pour son organisation et la disponibilité qui en découle, pour tout ce qu'il a pu m'apprendre, pour avoir tour à tour supporté et consolidé mes modestes talents mathématiques et pour sa sympathie, "qui ne gâche rien".

Pour leur aide, leurs conseils et leur soutien, merci aux membres dits «permanents» des groupes 2I, RIA puis GEPETTO. Merci notamment à Jean-Paul pour les moments de discussion dont je ressortais toujours les idées plus claires.

Parce que nos échanges ne furent pas exclusivement scientifiques, un grand merci à tous les «apprenti-chercheurs» avec qui j'ai passé ces trois bonnes années à bronzer devant les écrans d'ordinateur ou sur les sommets pyrénéens.

Merci enfin à Ester pour sa relecture ortho-typographique minutieuse, et pour m'avoir accompagné dans cette aventure.

Table des matières

1 Introduction **1**
- 1.1 Contexte . . . 1
- 1.2 Navigation autonome des robots mobiles . . . 1
  - 1.2.1 Planification de mouvement . . . 2
  - 1.2.2 Localisation . . . 2
  - 1.2.3 Suivi de trajectoire . . . 2
  - 1.2.4 Évitement réactif d'obstacles . . . 2
  - 1.2.5 Parking . . . 3
  - 1.2.6 Intégration des différentes fonctionnalités de la navigation . . . 3
- 1.3 Spécificités de notre approche . . . 3
  - 1.3.1 Nos contributions . . . 5
  - 1.3.2 Plan de lecture . . . 6

2 Notations et définitions **9**
- 2.1 Espaces et positions . . . 9
- 2.2 Distances . . . 10
- 2.3 Système dynamique . . . 11

3 Évitement réactif d'obstacles pour robots mobiles nonholonomes **15**
- 3.1 État de l'art . . . 15
  - 3.1.1 Méthodes d'évitement d'obstacles sans trajectoire de référence . . . 16
  - 3.1.2 Méthodes d'évitement d'obstacles avec une trajectoire de référence . . . 17
- 3.2 Déformation de trajectoire pour systèmes nonholonomes . . . 18
  - 3.2.1 Déformation de trajectoire . . . 18
  - 3.2.2 S'éloigner des obstacles . . . 20
  - 3.2.3 Calcul d'une déformation admissible s'éloignant des obstacles . . . 22
  - 3.2.4 Prise en compte des conditions aux limites . . . 25
  - 3.2.5 Calcul de la trajectoire déformée . . . 27
  - 3.2.6 Respect des contraintes nonholonomes . . . 27
  - 3.2.7 Algorithme de la déformation de trajectoire . . . 29
  - 3.2.8 Prise en compte des bornes sur les entrées du système . . . 30
- 3.3 Application de la méthode de déformation de trajectoire : le robot Cycab . . . 32
  - 3.3.1 Modèle cinématique . . . 32

3.3.2 Calculs pour la déformation de trajectoire . . . 33
3.3.3 Exemple de déformation . . . 34
3.4 Calcul des interactions robot-obstacles . . . 37
3.4.1 Potentiel dans l'espace des configurations . . . 37
3.4.2 Optimisation des calculs des interactions . . . 41
3.5 Optimisation de la trajectoire suivant d'autres critères . . . 46
3.5.1 Respect d'une contrainte sur la courbure durant le processus de déformation de trajectoire . . . 47
3.5.2 Déformation d'une trajectoire non-admissible vers une trajectoire admissible . . . 49
3.6 Conclusion . . . 52

4 Parking référencé sur des amers 53
4.1 Introduction . . . 53
4.2 État de l'art . . . 54
4.2.1 Asservissement visuel . . . 55
4.2.2 Planifications successives . . . 55
4.3 Tâche de parking référencé sur des amers . . . 56
4.3.1 Définitions . . . 56
4.3.2 Formulation du problème . . . 58
4.4 Calcul de la configuration de parking . . . 60
4.4.1 Modélisation probabiliste . . . 60
4.4.2 Mise en correspondance . . . 61
4.4.3 Mise à jour de la position de parking du capteur . . . 62
4.4.4 Mise à jour de la configuration de parking . . . 64
4.4.5 Extension à plusieurs capteurs . . . 65
4.5 Déformation de trajectoire . . . 66
4.5.1 Conditions aux limites . . . 66
4.6 Application au robot Hilare2 avec remorque . . . 67
4.6.1 Tâche de parking : parking relativement à un quai de débarquement . . . 67
4.6.2 Résolution de la tâche de parking . . . 71
4.7 Conclusion . . . 75

5 Suivi de trajectoire, évitement d'obstacles et localisation 77
5.1 Introduction . . . 77
5.2 Suivi de trajectoire et évitement réactif d'obstacles . . . 78
5.2.1 Suivi de trajectoire . . . 78
5.2.2 S'arrêter sur une trajectoire . . . 80
5.2.3 Détection des obstacles . . . 83
5.2.4 Architecture pour le suivi de trajectoire et l'évitement d'obstacles . . . 84
5.2.5 Conditions garantissant l'absence de collision lors du suivi de la trajectoire 85
5.3 Intégration de la localisation globale . . . 88
5.3.1 Localisation globale . . . 88

5.3.2 Architecture pour le suivi de trajectoire avec localisation globale 89
5.3.3 Conditions garantissant l'absence de collision pour les systèmes exponentiellement stables 90
5.3.4 Cas réalistes avec un véhicule articulé 92
5.4 Architecture pour l'intégration du suivi de trajectoire, de l'évitement d'obstacles et de la localisation 94
5.4.1 Module de simulation du système 94
5.4.2 Architecture pour le suivi de trajectoire 95
5.5 Conclusion 98

6 Résultats expérimentaux 99
6.1 Introduction 99
6.2 Architecture logicielle 100
6.3 Portage sur plusieurs robots 102
6.3.1 Le rover Dala 102
6.3.2 Le robot Cycab 103
6.4 Parking référencé sur des amers 104
6.4.1 Parking référencé sur des amers avec une localisation imprécise 105
6.4.2 Parking référencé sur des amers avec un motif de parking inexact 105
6.4.3 Bilan 106
6.5 Intégration de la localisation globale, du suivi de trajectoire et de l'évitement d'obstacles 107
6.5.1 Conclusion 108

7 Conclusion 109

A Preuve de l'absence de collision pour les systèmes respectant la propriété de stabilité exponentielle 111

Chapitre 1

Introduction

1.1 Contexte

L'objet de la robotique est l'automatisation de systèmes mécaniques. En dotant le système de capacités de perception, d'action et de décision, l'objectif est de lui permettre d'interagir rationnellement avec son environnement, et de façon autonome.

Depuis les premiers automates jusqu'aux systèmes disponibles en ce début de XXI^{e} siècle, on mesure tout à la fois le chemin parcouru et celui restant à parcourir avant de réaliser les rêves qui animaient les pionniers. La robotique est un domaine de recherche qui se situe au carrefour de l'intelligence artificielle, de l'automatique, de l'informatique et de la perception par ordinateur ; cette interdisciplinarité est à l'origine d'une certaine complexité. Des applications dans des domaines aussi variés que l'industrie manufacturière, le spatial, l'automobile ou plus récemment les loisirs et le secteur médical, démontrent aujourd'hui l'intérêt économique et social de ces recherches.

La robotique mobile autonome vise plus spécifiquement à concevoir des systèmes capables de se déplacer de façon autonome. Les applications directes se situent notamment dans les domaines de l'automobile, de l'exploration planétaire ou de la robotique de service par exemple. De nombreuses applications restent à découvrir, qui ne découlent pas directement des avancées de la robotique mais qui utilisent ses méthodes et ses développements.

Notre thèse a pour cadre général la navigation autonome des robots mobiles, et elle se focalise sur un type de système et d'application spécifiques : les véhicules articulés avec des applications dans le transport automatique.

1.2 Navigation autonome des robots mobiles

Pour exécuter une tâche de navigation autonome, un robot mobile doit mettre en oeuvre un certain nombre de fonctionnalités, que nous détaillons ici.

1.2.1 Planification de mouvement

La planification de mouvement dans sa formulation classique est le problème du calcul d'un chemin sans collision pour un système mécanique, entre une configuration de départ et une configuration d'arrivée données, dans un modèle géométrique de l'environnement. Le concept d'espace des configurations introduit par [Lozano-Pérez 1983] en robotique permet de transformer le problème de la recherche d'un chemin pour un système à n degrés de liberté dans l'espace euclidien en celui du mouvement d'un point dans un espace à n dimensions.

La résolution de ce problème, dit du *déménageur de piano*, par des méthodes exactes ([Schwartz 1983a], [Schwartz 1983b], [Schwartz 1983c]), n'est pas applicable à des systèmes complexes avec de nombreux degrés de liberté. Les méthodes effectives de planification de mouvement reposent sur des algorithmes probabilistes qui explorent aléatoirement l'espace des configurations afin d'en caractériser la connexité. On trouve une synthèse de ces techniques dans [Latombe 1991], ou dans des références plus récentes [LaValle 2006], [Choset 2005].

1.2.2 Localisation

Afin d'exécuter le mouvement planifié, le robot doit se localiser dans l'environnement. La localisation est l'estimation de la position du robot par rapport à un repère fixe de l'environnement. Cette estimation de la position peut s'effectuer soit par une mesure des déplacements du robot soit par une mesure de sa position absolue dans l'environnement. Un inventaire des techniques et capteurs disponibles est recensé dans [Borenstein 1996]. L'estimation de la position absolue du robot dans l'environnement ne peut être traitée indépendamment de la construction d'une carte d'amers de l'environnement [Thrun 2002]. Du fait des incertitudes sur les mesures utiles pour la localisation, le problème de la localisation est généralement modélisé dans un cadre probabiliste.

1.2.3 Suivi de trajectoire

Le suivi de trajectoire consiste à calculer les commandes des actionneurs du système permettant de réaliser le mouvement planifié. Un robot étant considéré comme un système dynamique, on utilise des méthodes de commande par retour d'état pour asservir le système sur une trajectoire de référence.

1.2.4 Évitement réactif d'obstacles

Le suivi de la trajectoire planifiée ne permet pas de garantir l'absence de collision. En effet, des collisions peuvent se produire lors de l'exécution de la trajectoire, dues à :

- une localisation imparfaite,
- un plan imprécis,
- des obstacles qui n'étaient pas dans le modèle de l'environnement utilisé pour la planification de trajectoire.

Tout ces éléments font que le mouvement initialement planifié doit être adapté lors de son exécution, et que des stratégies d'évitement réactif d'obstacles doivent être mises en oeuvre. On

adoptera des stratégies différentes en fonction du type de système, de sa vitesse, et du champ d'application.

1.2.5 Parking

Le parking est la phase finale de la navigation autonome. Il occupe une place particulière pour deux raisons :

- la manoeuvre de parking est en général un mouvement fortement contraint et qui requiert une grande précision,
- l'objectif d'une mission de navigation est souvent d'atteindre une configuration finale spécifiée. Le succès de cette mission dépend de la réalisation de cet objectif.

Par ailleurs, le parking s'effectue généralement «relativement» à des éléments de l'environnement, et des méthodes de mouvement référencé sur des amers doivent être utilisées.

1.2.6 Intégration des différentes fonctionnalités de la navigation

L'intégration des différentes fonctionnalités de la navigation va au-delà de leur exécution en parallèle. En effet toutes ces fonctionnalités sont liées entre elles, par des variables ou des objectifs communs, et certaines doivent respecter des contraintes temps-réel. Ainsi la position du robot apparaît dans les fonctionnalités de localisation et de suivi de trajectoire par exemple, et la réalisation simultanée de ces tâches s'en trouve compliquée.

La conception d'une architecture prenant en compte ces interconnexions et assurant la réalisation de chacune des tâches est une problématique à part entière.

1.3 Spécificités de notre approche

Les spécificités de notre approche sont de deux types : les spécificités liées aux systèmes auxquels on s'intéresse et les spécificités liées aux applications.

Les spécificités liées au système tout d'abord :

- on s'intéresse dans notre étude aux robots mobiles à roues,
- le système est soumis à des contraintes cinématiques, et les trajectoires générales pour de tels systèmes n'ont pas d'expression simple comme des lignes droites ou des arcs de cercles,
- la géométrie complète du système doit être prise en compte,
- l'asservissement n'assure pas une décroissance uniforme de la distance à la consigne lors du suivi de la trajectoire.

Ainsi le type de système auquel on s'intéresse se range dans la classe des véhicules articulés, tels le robot Hilare2 et sa remorque. Les problématiques associées à de tels systèmes sont foncièrement différentes de celles associées à des robots de type «guide de musée» par exemple (figure 1.1). Ces spécificités impliquent qu'une trajectoire quelconque n'est pas nécessairement admissible pour le type de système auquel on s'intéresse. Ainsi par exemple un robot de type voiture ne peut pas effectuer de créneau latéral sans exécuter des manoeuvres. La représentation d'un

FIG. 1.1 – À gauche le robot Hilare2 avec sa remorque, un véhicule articulé. À droite un robot «guide de musée».

chemin sous la forme d'une série de points de passage est donc exclue, et on va donc travailler avec une trajectoire de référence qu'il nous faut calculer.

Pour la sous-classe des systèmes «différentiellement plats» [Fliess 1995], dont le robot Hilare2 (figure 1.1) avec sa remorque ou un robot de type voiture font partie, il existe des méthode permettant de calculer analytiquement une trajectoire faisable [Rouchon 1993], [Lamiraux 1997a], [Lamiraux 2000]. Pour d'autres systèmes on doit employer des méthodes numériques. Dans notre travail on utilise le logiciel Move3D [Siméon 2001] pour planifier une trajectoire sans collision dans un modèle de l'environnement. Dans la suite de ce document, on supposera que l'on possède déjà une telle trajectoire de référence, que le robot doit ensuite exécuter.

Les spécificités de notre approche sont aussi liées aux applications envisagées :

- on souhaite pouvoir naviguer proche des obstacles,
- le rapport entre la taille du système et la précision désirée est élevé,
- on est amené à effectuer des manoeuvres,
- les mouvements ne sont pas nécessairement réalisés à des vitesses élevées,
- les obstacles sont supposés statiques.

Les applications directes de notre travail sont par exemple l'automatisation de véhicules lourds, l'aide à la manoeuvre pour des camions, ou encore le parking automatique de véhicules articulés.

Le cadre général de notre travail est donc celui du «véhicule intelligent», mais nous mettons l'accent sur le fait que les spécificités de notre approche entraînent des problématiques singulières, et requièrent des méthodes de résolution originales. Ainsi le parking automatique d'un camion et l'automatisation d'une voiture sur autoroute ne posent par exemple pas les mêmes problèmes.

1.3.1 Nos contributions

Nos contributions portent sur la résolution de certaines fonctionnalités de la navigation autonome des robots mobiles dans le cadre défini par ces spécificités. Le fil conducteur de ce document est l'idée que ces spécificités posent des problématiques singulières, pour lesquelles les méthodes classiques développées pour des systèmes plus simples sont inadéquates.

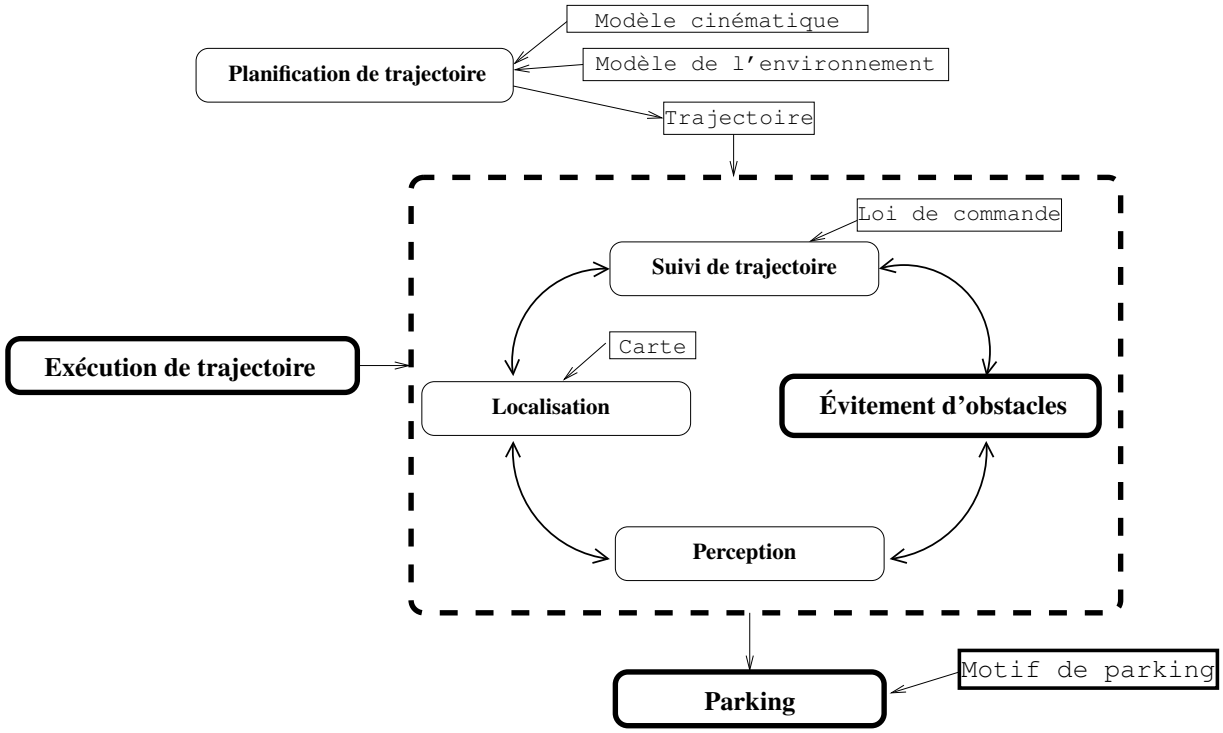

FIG. 1.2 – Fonctionnalités de la navigation autonome. Nos contributions théoriques sont représentées en gras.

Le diagramme 1.2 résume les différentes fonctionnalités nécessaires à la navigation autonome et permet de situer nos contributions, représentées en gras, qui résident dans le développement de méthodes originales de résolution de certaines fonctionnalités et dans l'intégration et l'implémentation de l'ensemble.

Notre méthodologie repose sur deux principes. Tout d'abord la volonté de développer des méthodes génériques, qui s'appliquent à une classe de systèmes plutôt qu'à un robot en particulier. Une conséquence intéressante est que des méthodes développées pour résoudre un problème dans un contexte applicatif particulier trouvent parfois des applications dans des domaines très différents, du fait de leur généricité. Ensuite le second principe est l'implémentation sur des systèmes réels, afin de se confronter à des problèmes réels. Les motivations de ce second principe sont développées dans le chapitre 6.

Nos recherches contiennent donc à la fois des contributions théoriques et expérimentales. Sur le plan théorique :

– nous avons contribué au développement d'une méthode de déformation de trajectoire pour systèmes nonholonomes, et nous avons proposé plusieurs extensions et optimisations de son algorithme,
– nous avons développé une méthode de parking référencé sur des amers pour des systèmes nonholonomes,
– nous avons proposé une architecture permettant d'intégrer les différentes fonctionnalités de la navigation autonome.

Sur le plan expérimental :

– nous avons implémenté ou adapté les différentes fonctionnalités de la navigation autonome pour trois robots évoluant dans des environnements différents : le robot Hilare2 et sa remorque (figure 1.1), le rover Dala et le Cycab de l'INRIA Rhône-Alpes (figure 1.3),
– nous avons réalisé des expérimentations avec ces robots, qui valident notre approche.

FIG. 1.3 – Les robots Dala et Cycab, supports de nos expérimentations.

À l'issue de notre travail nous proposons donc un ensemble de techniques qui permettent de résoudre le problème de la navigation autonome pour des véhicules articulés en environnement fortement contraint. Ces méthodes ont été implémentées et expérimentées sur des robots réels.

1.3.2 Plan de lecture

Le chapitre 2 présente quelques notations et définitions utilisées tout au long de ce document, qui précisent le type de systèmes auxquels on s'intéresse.

Le chapitre 3 traite de l'évitement réactif d'obstacles. Nous présentons une méthode originale d'évitement d'obstacles pour des systèmes nonholonomes. Nous proposons plusieurs extensions de cette méthodes pour des applications différentes, et présentons des optimisations de son algorithme.

Ensuite le chapitre 4 traite de la fonctionnalité de parking et propose une technique de parking référencé sur des amers pour des systèmes nonholonomes.

Le chapitre 5 présente l'intégration formelle des différentes fonctionnalités de l'exécution d'une trajectoire (voir la figure 1.2). On s'intéresse plus particulièrement à l'intégration de la localisation, du suivi de trajectoire et de l'évitement d'obstacles, en tenant compte des spécificités de notre approche.

Enfin le chapitre 6 expose des résultats expérimentaux de navigation autonome obtenus sur des robots réels.

Chapitre 2

Notations et définitions

Nous présentons dans ce chapitre les notations et définitions employées tout au long de ce document.

2.1 Espaces et positions

Espace de travail

L'espace de travail noté $\mathcal{W}$ est dans notre étude l'espace euclidien à deux dimensions, assimilé à $\mathbb{R}^2$. Un corps rigide dans cet espace occupe dans une configuration dite de référence un sous-espace $\mathcal{B} \subseteq \mathcal{W}$. On définit un repère fixe de $\mathcal{W}$ que l'on note O. La situation d'un corps dans l'espace est alors donnée par un mouvement rigide $\mathbf{x} \in SE(2)$, définissant la position et l'orientation du corps relativement à O. L'espace occupé par le corps dans la situation $\mathbf{x}$ est noté $\mathbf{x}(\mathcal{B})$.

L'ensemble des résultats que nous présentons est généralisable à $SE(3)$.

Robot

Un robot $\mathcal{R}$ est une chaîne cinématique composée de corps rigides reliés entre eux par des articulations. On considère que cette chaîne cinématique possède un corps racine, dont on note $\mathbf{x} \in SE(2)$ la position dans l'espace de travail.

Espace des configurations

L'ensemble des variables permettant de repérer la position de chacun des corps du robot dans l'espace est appelé une configuration et est noté $\mathbf{q}$. L'ensemble des configurations du robot est l'espace des configurations $\mathcal{C}$, qui a une structure de variété différentielle. On note n la dimension de $\mathcal{C}$, et on identifie localement l'espace des configurations à $\mathbb{R}^n$.

On note $\mathbf{q}_{int} \in \mathcal{C}_{int}$ les variables de configuration internes exprimées relativement au corps racine, lorsque celui-ci est dans la configuration de référence. On a alors :

$$\mathbf{q} = (\mathbf{x}, \mathbf{q}_{int}) \ \in \ CS = SE(2) \times \mathcal{C}_{int}$$

où $\mathbf{x}$ est le mouvement rigide appliqué à tous les corps du robot.

Transformation solide

À tout élément m de $SE(2)$, on associe une application de $\mathcal{C}$ dans $\mathcal{C}$ que l'on note de la même façon m :

$$\begin{array}{rccc} m: & SE(2) \times \mathcal{C}_{int} & \rightarrow & SE(2) \times \mathcal{C}_{int} \\ & (\mathbf{x}, \mathbf{q}_{int}) & \mapsto & m.\mathbf{q} = (m.\mathbf{x}, \mathbf{q}_{int}) \end{array}$$

L'application m déplace le corps racine d'un robot et garde les variables internes inchangées.

2.2 Distances

Distance entre deux configurations

La distance $d_{\mathcal{C}}$ entre deux configurations $\mathbf{q}_1$ et $\mathbf{q}_2$ est définie comme la distance maximale entre deux points du robot dans les configurations $\mathbf{q}_1$ et $\mathbf{q}_2$. Soit p un point du robot, et $\mathbf{q}(p)$ la position de ce point dans $\mathcal{W}$ lorsque le robot est dans la configuration $\mathbf{q}$. La distance entre $\mathbf{q}_1$ et $\mathbf{q}_2$ est :

$$d_{\mathcal{C}}(\mathbf{q}_1, \mathbf{q}_2) = \max_{p \in \mathcal{R}} \|\mathbf{q}_1(p) - \mathbf{q}_2(p)\| \tag{2.1}$$

Distance entre deux positions

Soit $\mathbf{x}_1$ et $\mathbf{x}_2$ deux éléments de $SE(2)$. On définit la distance entre $\mathbf{x}_1$ et $\mathbf{x}_2$:

$$d_{SE(2)}(\mathbf{x}_1, \mathbf{x}_2) = \max_{\mathbf{q}_{int} \in \mathcal{C}_{int}} d_{\mathcal{C}}((\mathbf{x}_1, \mathbf{q}_{int}), (\mathbf{x}_2, \mathbf{q}_{int})) \tag{2.2}$$

Distance entre deux corps

Soit $\mathcal{A}$ et $\mathcal{B}$ deux sous-espaces compacts de $\mathcal{W}$, représentant deux corps. On appelle distance entre ces deux corps la valeur :

$$d(\mathcal{A}, \mathcal{B}) = \min_{\mathbf{a} \in \mathcal{A}, \mathbf{b} \in \mathcal{B}} \|a - b\| \tag{2.3}$$

Cette mesure n'est pas une distance au sens mathématique. On remarque en effet par exemple que si $\mathcal{A} \subset \mathcal{B}$, alors $d(\mathcal{A}, \mathcal{B}) = 0$, sans impliquer l'égalité entre les deux sous-espaces $\mathcal{A}$ et $\mathcal{B}$. On vérifie cependant aisément que cette mesure respecte l'inégalité triangulaire.

Distance entre une configuration et un corps

Si l'on note $\mathcal{O}$ un sous-espace de $\mathcal{W}$, et $\mathbf{q}$ une configuration du robot, on appelle distance entre $\mathbf{q}$ et $\mathcal{O}$ la valeur :

$$d_O(\mathbf{q}, \mathcal{O}) = \min_{\mathbf{p} \in \mathcal{R}, \mathbf{o} \in \mathcal{O}} \|\mathbf{q}(\mathbf{p}) - \mathbf{o}\| \tag{2.4}$$

Dans le cas d'un robot composé de n_b corps $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_{n_b}$, cette distance est la valeur minimale de la distance de chacun des corps dans la configuration $\mathbf{q}$ au corps $\mathcal{O}$:

$$d_O(\mathbf{q}, \mathcal{O}) = \min_{i \in \{1,\ldots,n_b\}} d(\mathcal{B}_i(\mathbf{q}), \mathcal{O})$$

où $\mathcal{B}_i(\mathbf{q})$ est le sous-espace occupé par le corps $\mathcal{B}_i$ quand le robot est dans la configuration $\mathbf{q}$. La distance d_O est utile pour exprimer la distance entre le robot et un obstacle, mais on remarque qu'elle n'est pas une distance au sens mathématique. Elle respecte cependant la propriété suivante qui correspond à l'inégalité triangulaire :

Propriété 1. $\forall \mathbf{q}_1, \mathbf{q}_2 \in \mathcal{C}, \quad \forall \mathcal{O} \subset \mathcal{W}$

$$d_O(\mathbf{q}_1, \mathcal{O}) \geq d_O(\mathbf{q}_2, \mathcal{O}) - d_\mathcal{C}(\mathbf{q}_1, \mathbf{q}_2)$$

Démonstration. Si $d_O(\mathbf{q}_2, \mathcal{O}) \leq d_\mathcal{C}(\mathbf{q}_1, \mathbf{q}_2)$, alors $d_O(\mathbf{q}_1, \mathcal{O})$ étant positif, la conclusion est immédiate. Sinon on suppose que $d_O(\mathbf{q}_2, \mathcal{O}) \geq d_\mathcal{C}(\mathbf{q}_1, \mathbf{q}_2)$ et alors on a :
$\forall \mathbf{q}_1, \mathbf{q}_2 \in \mathcal{C}, \forall \mathbf{o} \in \mathcal{O}, \forall p \in \mathcal{R}$

$$\|\mathbf{q}_1(p) - \mathbf{o}\| \quad = \quad \|\mathbf{q}_1(p) - \mathbf{q}_2(p) + \mathbf{q}_2(p) - o\|$$

Et par inégalité triangulaire :

$$\|\mathbf{q}_1(p) - \mathbf{o}\| \quad \geq \quad \Big| \|\mathbf{q}_1(p) - \mathbf{q}_2(p)\| - \|\mathbf{q}_2(p) - o\| \Big| \tag{2.5}$$

Étant donné que $d_O(\mathbf{q}_2, \mathcal{O}) \geq d_\mathcal{C}(\mathbf{q}_1, \mathbf{q}_2)$, on a :

$$\min_{p \in \mathcal{R}, \mathbf{o} \in \mathcal{O}} \|\mathbf{q}_2(p) - \mathbf{o}\| \geq \max_{\mathbf{p} \in \mathcal{R}} \|\mathbf{q}_1(p) - \mathbf{q}_2(p)\|$$

Ce qui implique que le terme dans la valeur absolue du membre de droite de l'équation (2.5) est négatif. Ainsi $\forall \mathbf{q}_1, \mathbf{q}_2 \in \mathcal{C}, \forall \mathbf{o} \in \mathcal{O}, \forall p \in \mathcal{R}$:

$$\begin{aligned} \|\mathbf{q}_1(p) - \mathbf{o}\| \quad &\geq \quad \|\mathbf{q}_2(p) - \mathbf{o}\| - \|\mathbf{q}_1(p) - \mathbf{q}_2(p)\| \\ \|\mathbf{q}_1(p) - \mathbf{o}\| \quad &\geq \quad \min_{p \in \mathcal{R}, \mathbf{o} \in \mathcal{O}} \|\mathbf{q}_2(p) - o\| - \max_{p \in \mathcal{R}} \|\mathbf{q}_1(p) - \mathbf{q}_2(p)\| \end{aligned}$$

Et par définition :

$$\max_{p \in \mathcal{R}, \mathbf{o} \in \mathcal{O}} \|\mathbf{q}_1(p) - \mathbf{o}\| \geq \|\mathbf{q}_1(p) - \mathbf{o}\|$$

d'où le résultat. □

2.3 Système dynamique

On considère maintenant le robot comme un système dynamique. Tout système abordé dans ce document respecte les propriétés suivantes.

Trajectoire

Une trajectoire de ce système est une fonction continue de $[0, S] \subset \mathbb{R}$ dans $\mathcal{C}$, qui à toute abscisse $s \in [0, S]$ associe une configuration :

$$\begin{array}{rccc} \mathbf{q}: & [0,S] & \rightarrow & \mathcal{C} \\ & s & \mapsto & \mathbf{q}(s) \end{array}$$

La variable s n'est donc pas une abscisse curviligne.

Contrainte nonholonome

Le système est soumis à des contraintes de «roulement sans glissement». Celles-ci s'expriment comme des contraintes linéaires sur les vitesses, que l'on peut mettre sous la forme :

$$G(\mathbf{q})\mathbf{q}' = 0 \tag{2.6}$$

où G est une fonction *non-intégrable*, c'est à dire qu'il n'existe pas de fonction F telle que $\frac{\partial F(\mathbf{q})}{\partial \mathbf{q}} = G(\mathbf{q})$. On note m le rang de l'application linéaire $G(\mathbf{q})$. Ce type de contraintes qui réduit l'espace des vitesses accessibles par le système sans réduire son espace accessible est appelé contrainte nonholonome. Alors l'équation (2.6) exprime l'appartenance de $\mathbf{q}'$, la dérivée de $\mathbf{q}$, au noyau de $G(\mathbf{q})$:

$$\mathbf{q}' \in Ker(G)$$

La dimension de $Ker(G(\mathbf{q}))$ est $(n - m)$, avec $n = dim\, \mathcal{C}$.

Champs de vecteurs

Les vitesses du système soumis aux contraintes de l'équation (2.6) appartiennent donc à un espace de dimension $k = (n - m)$. On note $(\mathbf{X}_1(\mathbf{q}), \mathbf{X}_2(\mathbf{q}), \ldots, \mathbf{X}_k(\mathbf{q}))$ k champs de vecteurs formant une base de $Ker(G(\mathbf{q}))$.

Trajectoire admissible pour un système sans dérive

Le système est sans dérive et une trajectoire admissible pour le système est telle qu'il existe un vecteur $\mathbf{u} = (u_1, ..., u_k)$ de k applications continues de $[0, S]$ dans $\mathbb{R}$, tel que :

$$\forall s \in [0, S], \quad \mathbf{q}'(s) = \sum_{i=1}^{k} u_i(s)\mathbf{X}_i(\mathbf{q}(s)) \tag{2.7}$$

Symétrie par rapport à $SE(2)$

On suppose que les vitesses suivant les champs de vecteurs du système sont indépendantes du repère choisi. C'est à dire que les champs de vecteurs du système respectent la propriété suivante :

Propriété 2. $\forall m \in SE(2), \forall \mathbf{q} \in \mathcal{C}$*, en notant* $T_{\mathbf{q}}m$ *l'application linéaire tangente à* m *en* $\mathbf{q}$*, on a :*

$$\mathbf{X}(m.\mathbf{q}) = T_{\mathbf{q}}m\mathbf{X}(\mathbf{q})$$

On en déduit le corollaire suivant pour les trajectoires du système :

Corollaire 1. $\forall s \in [0, S]$*, si* $\mathbf{q}(s)$ *est la configuration à l'abscisse* s *sur la trajectoire obtenue avec les commandes* $\mathbf{u}(s)$ *depuis* $\mathbf{q}(0)$*, alors en appliquant les mêmes commandes depuis* $m.\mathbf{q}(0)$*, à l'abscisse* s*, le système est à la configuration* $m.\mathbf{q}(s)$*.*

Symétrie par changement d'échelle du temps

Le système étant sans dérive, on a la propriété suivante :

Propriété 3. *Si* $(\mathbf{q}(s), \mathbf{u}(s))$ *est une trajectoire admissible solution de (2.7) définie sur* $[0, S]$*, alors pour tout difféomorphisme* ϕ *de classe* C^1 *de* $[0, S]$ *dans* $[0, S_\phi]$*, la trajectoire :*

$$(\mathbf{q}(\phi(s)), \phi'(s)\mathbf{u}(s))$$

est une trajectoire admissible sur $[0, S_\phi]$*.*

Chapitre 3

Évitement réactif d'obstacles pour robots mobiles nonholonomes

Ce chapitre traite de la fonctionnalité d'évitement réactif d'obstacles. Tout d'abord nous présentons un état de l'art des méthodes d'évitement réactif d'obstacles, et nous montrons leur inadéquation aux spécificités de notre problématique. Ensuite nous proposons une méthode originale d'évitement réactif d'obstacles pour des robots mobiles nonholonomes. Cette méthode utilise une trajectoire de référence qui est déformée de façon à s'éloigner des obstacles et à satisfaire les contraintes cinématiques du système. Enfin nous présentons quelques optimisations de l'algorithme de cette méthode et des extensions de ses applications.

3.1 État de l'art

La plupart des méthodes d'évitement réactif d'obstacles sont des méthodes locales. En effet les méthodes globales, qui considèrent un modèle complet de l'environnement, se ramènent au problème de la planification de mouvement. Et pour des systèmes possédant de nombreux degrés de liberté (plus de 3), la complexité de la planification d'une trajectoire interdit son utilisation en cours d'exécution. Par exemple l'approche de replanification rapide D^* de [Stentz 1994] découpe l'espace de travail en cellules étiquetées «libre» ou «occupée», pour diminuer la complexité.

Pour les systèmes considérés dans notre étude, tels que nous les avons définis au chapitre 2, la planification d'une trajectoire sans collision dans un environnement fortement contraint est très coûteuse en temps de calcul. Cette étape est donc réalisée «hors-ligne» et on utilise les techniques d'exploration aléatoire de l'espace des configurations mentionnées dans le chapitre d'introduction (Probabilistic Roadmap [Kavraki 1996] et Rapidly-Exploring Random Tree [LaValle 1998]) pour calculer ces trajectoires. Le modèle de l'environnement utilisé est construit auparavant par un autre robot ou à partir d'un plan métrique. On emploie une méthode de guidage (calcul d'une trajectoire admissible entre deux configurations sans prise en compte des obstacles) pour le système considéré, si elle existe, pour connecter les configurations tirées aléatoirement. L'ensemble de ces fonctionnalités est intégré dans le logiciel de planification de mouvement Move3D [Siméon 2001] que l'on utilise dans notre étude.

On supposera donc toujours que l'on possède une trajectoire de référence calculée dans un modèle de l'environnement.

Parmi les méthodes locales d'évitement réactif d'obstacles, on peut distinguer deux catégories : les méthodes utilisant une trajectoire de référence et celles n'en utilisant pas. Le type de système et d'application orientent le choix vers une catégorie ou l'autre.

Enfin un autre type de méthodes doit être mentionné, qui tient compte explicitement de la vitesse estimée des obstacles pour réaliser un mouvement sans collision, ce qui définit une nouvelle problématique. Nous renvoyons à [Fraichard 2004] pour un cadre formel et un état de l'art de cette problématique. Dans notre étude nous considérons que les obstacles sont statiques.

3.1.1 Méthodes d'évitement réactif d'obstacles sans trajectoire de référence

Pour certains systèmes et dans certains contextes applicatifs, la définition d'une trajectoire comme une série de points de passage est possible. Des méthodes d'évitement réactif d'obstacles ont été développées qui calculent des commandes permettant de rejoindre un point de passage en évitant les collisions avec les obstacles détectés.

Champs de potentiel

Les méthodes de champs de potentiel pour la navigation en robotique, initialement proposées par [Khatib 1986] pour un bras manipulateur, consistent à construire une fonction de potentiel qui résume les objectifs de la navigation : éviter les obstacles (potentiel répulsif) et atteindre une configuration but (potentiel attractif). À chaque position du robot, une «force» résultant de l'action conjuguée des obstacles et du but est calculée, qui correspond à une direction à suivre par le robot.

De nombreuses adaptations de cette technique ont été proposées. On peut par exemple citer [Borenstein 1991], qui calcule une direction de mouvement à partir d'informations proximétriques.

Ces méthodes purement réactives sont sujettes à des minima locaux, et peuvent nécessiter une replanification globale. Par ailleurs, elles peuvent entraîner un mouvement oscillatoire du robot dans certaines situations (des passages étroits par exemple).

Steering Angle Field (SAF)

Cette approche a été proposée par [Feiten 1994] pour un robot de type unicycle. Il s'agit de calculer pour un ensemble de vitesses linéaires des domaines de l'angle de braquage (SAF) qui n'entraînent pas de collision avec les obstacles perçus. Cette méthode utilise une discrétisation en grille du plan de travail autour du robot. Elle exploite la possibilité de pré-calculer et de stocker dans des tables de recherche les SAF pour chaque cellule obstacle de la grille.

Dynamic Window

Cette technique proposée dans [Fox 1997] travaille dans l'espace des commandes du robot. La taille du domaine de recherche des vitesses accessibles (c'est à dire n'entraînant pas de collisions) est réduite par la prise en compte explicite de la dynamique (les capacités d'accélération) du système. Les commandes envoyées au robot sont le résultat de la maximisation sur ce domaine de recherche d'une fonction de coût liée à la position but. Cette méthode a été développée pour des robots se déplaçant à des vitesses élevées ($1m.s^{-1}$) dans des environnements intérieurs encombrés.

Nearness Diagram

Cette approche proposée par [Minguez 2000] repose sur un diagramme de proximité des obstacles, mis à jour au fur et à mesure du déplacement du robot. En fonction de l'allure de ce diagramme (nombre de passages sans obstacles, taille des passages, etc.), un comportement adapté (exploration, avancement vers le but, retour en arrière, etc.) est sélectionné. La direction de mouvement la plus prometteuse par rapport au but est alors choisie. Cette méthode a été appliquée à un véhicule de type unicycle grâce à une expression des obstacles dans l'espace des points accessibles en un mouvement élémentaire (un arc de cercle en l'occurence), obtenue par une transformation dans l'espace dit «Ego-cinématique» [Minguez 2002].

L'exposé de ces différentes méthodes et de leurs hypothèses fait immédiatement apparaître leur inadéquation à des systèmes à cinématique plus complexe évoluant dans des environnements fortement contraints. On ne peut en effet généralement pas présupposer de la forme des trajectoires. De même les discrétisations en grille de l'espace de travail, proposées par certaines méthodes, ne permettent pas les mouvements arbitrairement proches des obstacles.

3.1.2 Méthodes d'évitement réactif d'obstacles avec une trajectoire de référence

Dès lors que la cinématique du système est plus complexe, le calcul préalable d'une trajectoire de référence qui sera adaptée lors de l'exécution s'avère nécessaire. Nous présentons deux méthodes utilisant une trajectoire de référence pour l'évitement réactif d'obstacles.

Trajectoires d'évitement

Une méthode proposée dans [Laugier 1999] repose sur l'enchaînement de trajectoires élémentaires (Sensor Based Maneuvers SMB), dans le contexte d'une voiture automatisée roulant sur une route. Les SMB sont par exemple le suivi d'un couloir de route, le changement de file ou le parking parallèle. Les paramètres de chacune de ces modalités sont définis par les informations sensorielles. Un contrôleur, amélioré dans [Large 2000] par un réseau de neurones, permet d'enchaîner ces différentes modalités.

Bande élastique

La méthode proposée par Quinlan [Quinlan 1993] utilise une trajectoire initialement planifiée qui est représentée par une série de boules adjacentes (appelées «bulles») dans l'espace des configurations. Le rayon d'une boule centrée en une configuration est la distance de cette configuration à l'obstacle le plus proche. Ainsi une trajectoire est sans collision dès lors que les «bulles» qui la composent se recouvrent. Développée pour des systèmes sans contraintes cinématiques, cette technique considère la trajectoire comme une bande élastique, se modifiant sous l'action de forces répulsives générées par les obstacles, et de forces internes de contraction ou d'élasticité. Du fait de la représentation en bulles, la mise à jour de la trajectoire sous l'action des forces est très rapide.

Cette technique a été étendue à un robot de type voiture par [Khatib 1997]. La «forme» des «bulles» est ici donnée par la métrique des trajectoires de Reeds et Shepp (combinaisons d'arcs de cercle et de lignes droites) [Soueres 1996]. Le lissage de la courbe joignant les centres des «bulles» est réalisé en utilisant une courbe de Bézier.

Mais pour certains systèmes on ne connaît pas la plus courte distance entre une configuration et un obstacle. La méthode de la bande élastique ne peut donc s'appliquer qu'au prix d'approximations sur la forme des trajectoires de tels systèmes, ce qui rend impossible la navigation en environnement très contraint.

L'ensemble de ces considérations, concernant les limitations des méthodes utilisant une trajectoire de référence ou des techniques proposées pour les systèmes à cinématique plus simple, motive le développement d'une méthode générique d'évitement d'obstacles pour systèmes nonholonomes.

3.2 Déformation de trajectoire pour systèmes nonholonomes

Nous présentons dans cette section la méthode de déformation de trajectoire développée pour l'évitement réactif d'obstacles pour des systèmes nonholonomes, publiée dans [Lamiraux 2004].

Comme nous l'avons déjà mentionné, on suppose que l'on possède une trajectoire sans collision planifiée dans un modèle de l'environnement. Au cours de l'exécution de cette trajectoire, le robot est capable de percevoir les obstacles, et doit éventuellement adapter réactivement cette trajectoire de référence pour éviter les collisions détectées.

La méthode consiste à perturber les entrées du système de façon à obtenir une trajectoire déformée qui s'éloigne des obstacles. Tous les éléments théoriques développés ici seront repris à travers un exemple dans la section 3.3.

3.2.1 Déformation de trajectoire

Une trajectoire admissible définie sur un intervalle est déterminée par une configuration initiale $\mathbf{q}(0)$ et par des fonctions d'entrée $(u_1, ..., u_k)$ définies sur cet intervalle. Si l'on perturbe ces entrées, on obtiendra par intégration du système (2.7) une nouvelle trajectoire admissible.

Pour caractériser la perturbation des entrées, on introduit la variable $\tau \in [0, +\infty[$, et on note $\mathbf{u}(s, \tau)$ la nouvelle entrée du système. On définit alors un faisceau de trajectoire comme une famille de trajectoires indexée par le réel τ. Un faisceau de trajectoire peut être considéré comme une fonction de deux variables réelles à valeurs dans l'espace des configurations :

$$\begin{array}{rccc} \mathbf{q}: & [0,S]\times[0,+\infty[& \rightarrow & \mathcal{C} \\ & (s,\tau) & \mapsto & \mathbf{q}(s,\tau) \end{array}$$

Et pour chaque valeur de τ, on définit la fonction partielle $s \mapsto \mathbf{q}_\tau(s) = \mathbf{q}(s, \tau)$. Ainsi $\mathbf{q}_\tau(s)$ est la trajectoire d'indice τ, dont les fonctions d'entrée sont $s \mapsto \mathbf{u}(s, \tau)$. Afin d'alléger la notation, on utilise donc la même écriture pour désigner une configuration, une trajectoire et un faisceau de trajectoire. La trajectoire $\mathbf{q}_0$ est donc la *trajectoire initiale* du système et $\mathbf{q}_\tau$ est une *trajectoire déformée*.

Un faisceau de trajectoire admissible est tel que : $\forall (s, \tau) \in [0, S] \times [0, +\infty[$

$$\frac{\partial \mathbf{q}}{\partial s}(s,\tau) = \sum_{i=1}^{k} u_i(s,\tau)\mathbf{X}_i(\mathbf{q}(s,\tau)) \tag{3.1}$$

On exprime alors la variation de $\mathbf{q}(s, \tau)$ en fonction de la variation du paramètre τ, en dérivant l'équation (3.1) par rapport à τ :

$$\begin{aligned} \frac{\partial^2 \mathbf{q}}{\partial s \partial \tau}(s,\tau) &= \sum_{i=1}^{k} \left(\frac{\partial u_i}{\partial \tau}(s) X_i(\mathbf{q}(s,\tau)) + \right. \\ &\qquad \left. u_i(s,\tau) \frac{\partial X_i}{\partial \mathbf{q}}(\mathbf{q}(s,\tau)) \frac{\partial \mathbf{q}}{\partial \tau}(s,\tau) \right) \end{aligned} \tag{3.2}$$

On introduit alors les notations suivantes :

$$\begin{aligned} \textbf{perturbation d'entrée : } \mathbf{v}(s,\tau) &\triangleq \frac{\partial \mathbf{u}}{\partial \tau}(s,\tau) \\ \textbf{direction de déformation : } \eta(s,\tau) &\triangleq \frac{\partial \mathbf{q}}{\partial \tau}(s,\tau) \end{aligned}$$

Ainsi l'équation (3.2) s'écrit :

$$\eta'(s,\tau) = A(s,\tau)\eta(s,\tau) + B(s,\tau)\mathbf{v}(s,\tau) \tag{3.3}$$

où $A(s, \tau)$ est la matrice de taille $(n \times n)$:

$$\begin{aligned} A(s,\tau) &= \sum_{i=1}^{k} u_i(s,\tau) \frac{\partial X_i}{\partial \mathbf{q}}(\mathbf{q}(s,\tau)) \\ &= \sum_{i=1}^{k} u_i(s,\tau) \begin{pmatrix} \frac{\partial \mathbf{X}_i^1}{\partial \mathbf{q}^1} & \cdots & \frac{\partial \mathbf{X}_i^1}{\partial \mathbf{q}^n} \\ \vdots & & \vdots \\ \frac{\partial \mathbf{X}_i^n}{\partial \mathbf{q}^1} & & \frac{\partial \mathbf{X}_i^n}{\partial \mathbf{q}^n} \end{pmatrix} \end{aligned}$$

et $B(s,\tau)$ est la matrice de taille $(n \times k)$ dont les colonnes sont les champs de vecteurs du système évalués en $\mathbf{q}(s,\tau)$:

$$B(s,\tau) = \Big(X_1(\mathbf{q}(s,\tau)) \cdots \mathbf{X}_k(\mathbf{q}(s,\tau)) \Big)$$

Ainsi, à partir de la trajectoire $\mathbf{q}(s,\tau)$, des entrées $\mathbf{u}(s,\tau)$, des perturbations des entrées $\mathbf{v}(s,\tau)$ et de la condition initiale $\eta(0,\tau)$, on peut calculer la direction de déformation correspondante par intégration de (3.3). Et réciproquement, toute direction de déformation admissible est solution de (3.3).

Le système (3.3) est le linéarisé tangent autour de la trajectoire $\mathbf{q}_\tau$ du système dynamique d'équation (2.7). On peut vérifier qu'en intégrant le système (3.3), on obtient l'expression suivante :

$$\eta(s,\tau) = H(s,\tau) \int_0^s H^{-1}(u,\tau) B(u,\tau) \mathbf{v}(u,\tau) du$$

avec $H(s,\tau)$ la matrice de taille $n \times n$ vérifiant :

$$\begin{aligned} H(0,\tau) &= I_n \\ H'(s,\tau) &= A(s,\tau) H(s,\tau) \end{aligned}$$

Le système (3.3) est donc linéaire par rapport à la perturbation $\mathbf{v}$ des fonctions d'entrée.

3.2.2 S'éloigner des obstacles

Étant donné que l'on cherche à déformer une trajectoire initiale de façon à s'éloigner des obstacles, on suppose qu'au cours de l'éxecution de la trajectoire, le robot est capable de percevoir les obstacles et de calculer leurs interactions avec la trajectoire. Et on exprime ces interactions au moyen d'un potentiel dans l'espace des configurations.

Potentiel d'une trajectoire

Soit $U(\mathbf{q})$ le potentiel d'une configuration :

$$\begin{array}{rccc} U: & \mathcal{C} & \rightarrow & \mathbb{R} \\ & \mathbf{q} & \mapsto & U(\mathbf{q}) \end{array}$$

La fonction U est définie de façon à ce que le potentiel d'une configuration soit élevé lorsque le sous-espace occupé par le robot dans cette configuration est proche des obstacles, et qu'il diminue lorsqu'il s'en éloigne (la section 3.4 détaille l'expression de cette fonction).

On note alors $V(\tau)$ le potentiel d'une trajectoire d'indice τ :

$$V(\tau) = \int_0^S U(\mathbf{q}(s,\tau)) ds$$

La variation de ce potentiel est :

$$\begin{aligned}\frac{dV}{d\tau}(\tau) &= \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s,\tau))\frac{\partial \mathbf{q}(s,t)}{\partial \tau} \\ &= \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s,\tau))\eta(s,\tau)ds \end{aligned} \tag{3.4}$$

où $\frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s,\tau))$ est un vecteur ligne.

Si on identifie l'espace des configurations $\mathcal{C}$ à $\mathbb{R}^n$ et si on suppose que $\mathbb{R}^n$ est muni de la base canonique et du produit scalaire :

$$\forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^n,\ (\mathbf{u}|\mathbf{v}) = \mathbf{u}^T.\mathbf{v}$$

on définit alors le produit scalaire dans L^2 entre deux fonctions d'une variable réelle à valeurs dans $\mathbb{R}^n$ par :

$$(f|g)_{L^2} = \int_0^S f(s)^T.g(s)ds$$

Ainsi le membre de droite de l'équation (3.4) est le produit scalaire des fonctions : $\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau$: $s \mapsto \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s,\tau))$ et $\eta_\tau : s \mapsto \eta(s,\tau)$:

$$\frac{dV}{d\tau}(\tau) = \left(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau \middle| \eta_\tau\right)_{L^2}$$

On donne la propriété suivante du produit scalaire :

Propriété 4. *Soient f et g deux fonctions C^∞ d'une variable réelle à valeurs dans $\mathbb{R}^n$. On a :*

$$\min_{\|f\|_{L^2}=1} (f|g)_{L^2} = -\frac{g}{\|g\|_{L^2}}$$

Donc à norme L^2 constante, la direction de déformation $\eta_\tau = -\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau$ est celle qui minimise $\frac{dV}{d\tau}$. Cependant rien ne garantit que cette déformation soit admissible, c'est à dire qu'elle soit solution de (3.3).

Direction de déformation admissible

Le sous-espace vectoriel des directions de déformation admissibles d'une trajectoire $\mathbf{q}_\tau$ est noté $\mathcal{AD}(\mathbf{q}_\tau)$:

$$\begin{aligned}\mathcal{AD}(\mathbf{q}_\tau) &= \{\eta \in C^\infty([0,S],\mathbb{R}^n) \mid \exists \mathbf{v}, \eta_0, \\ &\quad \forall s \in [0,S] \quad \eta(s) = A(s,\tau)\eta(s) + B(s,\tau)\mathbf{v},\ \eta(0) = \eta_0\}\end{aligned}$$

On donne la propriété suivante :

Propriété 5. *Soient f et g deux fonctions de $C^\infty([0,S],\mathbb{R}^n)$ et soit D un sous-espace de $C^\infty([0,S],\mathbb{R}^n)$. On a :*

$$\min_{\|f\|_{L^2}=1, f\in D}(f|g)_{L^2} = -\frac{p_D(g)}{\|p_D(g)\|_{L^2}}$$

où $p_D()$ est l'opérateur de projection sur D.

Donc une solution pour obtenir une direction de déformation admissible qui fasse décroître le potentiel $\frac{dV}{d\tau}(\tau)$ à norme L^2 constante pourrait être de projeter orthogonalement $-\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau$ sur le sous-espace $\mathcal{AD}(\mathbf{q}_\tau)$. En notant $\eta_\tau^\perp$ cette projection, on a pour toute direction de déformation admissible $\eta \in \mathcal{AD}(\mathbf{q}_\tau)$ l'équation :

$$\left(-\left(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau\right) - \eta_\tau^\perp \Big| \eta\right)_{L^2} = 0$$

Cependant rien ne garantit que la projection orthogonale sur cet espace existe et nous n'avons pas pu caractériser ainsi la projection orthogonale sur l'espace des directions de déformation admissibles.

Pour résoudre ce problème nous allons approximer l'espace des directions de déformation admissibles par un sous-espace de dimension finie.

3.2.3 Calcul d'une déformation admissible s'éloignant des obstacles

On cherche à calculer une direction de déformation η_τ admissible et qui fasse décroître le potentiel de la trajectoire. Pour cela on va se donner une base d'un sous-espace vectoriel de dimension finie de l'espace des directions de déformation admissibles et on va projeter orthogonalement $-\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}$ sur l'espace engendré par cette base.

Base de fonctions de perturbations

L'espace des fonctions de perturbations des entrées $\mathbf{v}(s,\tau)$ est, à τ fixé, l'espace des fonctions continues de $[0,S]$ dans $\mathbb{R}^k$. Cet espace est de dimension infinie et pour l'approximer on se donne une base de p fonctions continues notées $\mathbf{e}_i$:

$$\begin{array}{rccc} \mathbf{e}_i : & [0,S] & \to & \mathbb{R}^k \\ & s & \mapsto & \mathbf{e}_i(s) \end{array} \tag{3.5}$$

Chaque $\mathbf{e}_i$ est dénommée fonction de **perturbation élémentaire**. En pratique, plusieurs choix sont possibles pour ces fonctions : les premiers termes d'une série de Fourier, des séries de polynômes, de sinus, de fonctions triangulaires, etc. On définit une perturbation d'entrée $\mathbf{v}_\tau$: $s \mapsto \mathbf{v}_\tau(s)$, où τ est un réel fixé, comme une combinaison linéaire des fonctions de perturbations élémentaires. Soit $\lambda \in \mathbb{R}^p$, on a alors :

$$\mathbf{v}_\tau = \sum_{i=1}^{p} \lambda_i \mathbf{e}_i$$

Base $\mathcal{E}$ de directions de déformation

On note $\mathbf{E}_i$ la solution de l'équation différentielle (3.3) avec pour condition initiale $\eta_\tau(0) = 0$ et pour fonction d'entrée $\mathbf{e}_i$. $\mathbf{E}_i$ est donc solution du système :

$$\begin{aligned} \mathbf{E}_i'(s,\tau) &= A(s,\tau)\mathbf{E}_i(s,\tau) + B(s,\tau)\mathbf{e}_i(s) \\ \mathbf{E}_i(0,\tau) &= 0 \end{aligned} \tag{3.6}$$

Chaque fonction $\mathbf{E}_i$ est dénommée **direction de déformation élémentaire**. On remarque que les $\mathbf{E}_i$ dépendent de $\mathbf{q}_\tau$ et de $\mathbf{u}_\tau$ par le biais des matrices A et B, et qu'il faut donc les recalculer pour chaque trajectoire.

Étant donné que le système (3.3) est linéaire par rapport à la perturbation $\mathbf{v}_\tau$, on a :

$$\eta_\tau = \sum_{i=1}^{p} \lambda_i \mathbf{E}_i$$

On note $\mathcal{AD}_p(\mathbf{q}_\tau)$ le sous-espace de $\mathcal{AD}(\mathbf{q}_\tau)$ engendré par les fonctions $\mathbf{E}_i$, et $\mathcal{E} = \{\mathbf{E}_1, \ldots, \mathbf{E}_p\}$ est alors une base de $\mathcal{AD}_p(\mathbf{q}_\tau)$. Le vecteur $\lambda = (\lambda_1, \ldots, \lambda_p)^T$ représente donc les coordonnées de η_τ dans la base $\mathcal{E}$.

Calcul d'une direction de déformation

Si l'on se rappelle que l'on cherche une direction de déformation admissible $\eta_\tau \in \mathcal{AD}_p(\mathbf{q}_\tau)$ et qui minimise $(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau | \eta_\tau)_{L^2}$, on comprend qu'il faut maintenant projeter orthogonalement $-(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau)$ sur $\mathcal{AD}_p(\mathbf{q}_\tau)$. Pour cela on doit d'abord se donner une base orthonormée pour le produit scalaire L^2 de ce sous-espace. On construit cette base orthonormée en utilisant le procédé d'orthonormalisation de Gram-Schmidt à partir de la base $\mathcal{E}$.

Soit $\mathcal{F} = \{\mathbf{F}_1, \ldots, \mathbf{F}_p\}$ cette base orthonormée, et soit P la matrice de passage triangulaire supérieure de taille $(p \times p)$ de $\mathcal{E}$ à $\mathcal{F}$. La i^{eme} colonne de P est formée des coordonnées de $\mathbf{F}_i$ exprimé dans la base $\mathcal{E}$. On a $\mathbf{F}_i = \sum_{j=1}^{p} P_{ji}\mathbf{E}_j$. Soit alors η un élément de $\mathcal{AD}_p(\mathbf{q}_\tau)$, et λ^F ses coordonnées dans la base $\mathcal{F}$. Les coordonnées λ^E de η dans la base $\mathcal{E}$ sont :

$$\lambda^E = P\lambda^F$$

Soit η une fonction appartenant à $C^\infty([0,S], \mathbb{R}^n)$. La projection orthogonale de η sur $\mathcal{AD}_p(\mathbf{q}_\tau)$ est donnée par :

$$p_{\mathcal{AD}_p}(\eta) = \sum_{i=1}^{p} (\mathbf{F}_i|\eta)_{L^2}\mathbf{F}_i$$

où les $(\mathbf{F}_i|\eta)_{L^2}$ sont les coordonnées de $p_{\mathcal{AD}_p}(\eta)$ dans la base $\mathcal{F}$.

Alors, si on note η_τ^0 la projection orthogonale de $-(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau)$ sur $\mathcal{AD}_p(\mathbf{q}_\tau)$, on peut écrire :

$$\eta_\tau^0 = \sum_{i=1}^{p} \lambda_i^{0F}\mathbf{F}_i \tag{3.7}$$

$$\text{avec } \lambda_i^{0F} = -\left(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau \middle| \mathbf{F}_i\right)_{L^2} \tag{3.8}$$

Pour une valeur τ donnée, on définit $\mu_i^F = \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s,\tau))\mathbf{F}_i(s,\tau)ds$. On a donc :

$$\lambda_i^{0F} = -\mu_i^F \tag{3.9}$$

et $\lambda^{0F} = (\lambda_1^{0F}, \ldots, \lambda_p^{0F})^T \in \mathbb{R}^p$ est le vecteur des coordonnées dans la base $\mathcal{F}$ de la direction de déformation admissible cherchée η_τ^0.

En reprenant l'équation (3.4), l'expression de la variation du potentiel de la trajectoire engendrée par la direction de déformation η_τ^0 s'écrit :

$$\begin{aligned}
\frac{dV}{d\tau} &= \left(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau \Big| \sum_{i=1}^{p} \lambda_i^{0F}\mathbf{F}_i\right)_{L^2} \\
&= \sum_{i=1}^{p} \lambda_i^{0F} \left(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau \Big| \mathbf{F}_i\right)_{L^2} \\
&= \sum_{i=1}^{p} \lambda_i^{0F}\mu_i^F \\
&= -\|\mu^F\|^2 \leq 0
\end{aligned}$$

La direction de déformation η_τ^0 fait donc baisser le potentiel de la trajectoire.

Remarquons enfin que l'écriture $\eta_\tau = \sum_{i=1}^p \lambda_i^F \mathbf{F}_i$ des éléments de $\mathcal{AD}_p(\mathbf{q}_\tau)$ dans la base $\mathcal{F}$ définit implicitement un isomorphisme entre $\mathcal{AD}_p(\mathbf{q}_\tau)$ et $\mathbb{R}^p$. L'image par cet isomorphisme du produit scalaire dans L^2 entre des éléments de $\mathcal{AD}_p(\mathbf{q}_\tau)$ est le produit scalaire canonique dans $\mathbb{R}^p$ entre leurs vecteurs de coordonnées.

Expression dans la base $\mathcal{E}$

L'intérêt d'exprimer les coordonnées de cette direction de déformation dans la base $\mathcal{E}$ apparaîtra plus clairement lors du résumé de l'algorithme de la déformation à la section 3.2.7. On verra en effet que l'expression de tous les éléments de la déformation dans la base $\mathcal{E}$ permet de réduire le nombre d'opérations, et donc le temps de calcul. Car ce sont les $\mathbf{E}_i$ qui sont d'abord calculés, par intégration du système (3.6), et le calcul des $\mathbf{F}_i$ n'est donc pas nécessaire.

On note :

$$\mu_j^E = \int_0^S \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s,\tau))\mathbf{E}_j(s,\tau)ds \tag{3.10}$$

En reprenant alors l'équation (3.8), on a :

$$\begin{aligned}
\lambda_i^{0F} &= -\left(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau \Big| \mathbf{F}_i\right)_{L^2} \\
&= -\sum_{j=1}^{p} P_{ji} \left(\frac{\partial U}{\partial \mathbf{q}} \circ \mathbf{q}_\tau \Big| \mathbf{E}_j\right)_{L^2} \\
&= -P^T\mu^E(\tau)
\end{aligned}$$

On en déduit :

$$\lambda_i^{0E} = P\lambda_i^{0F} = -PP^T\mu^E$$

les λ_i^{0E} étant alors les coordonnées de la direction de déformation η_τ^0 dans la base $\mathcal{E}$.

3.2.4 Prise en compte des conditions aux limites

On a donc trouvé une direction de déformation admissible η_τ^0 qui fait baisser le potentiel de la trajectoire. On doit maintenant s'assurer que cette direction de déformation respecte certaines conditions aux bornes de l'intervalle de déformation. En effet, le processus de déformation de trajectoire n'agit pas sur tout l'intervalle de définition de la trajectoire, mais seulement sur un sous-intervalle centré autour de l'abscisse de collision. Or on impose comme contrainte que la trajectoire après déformation reste inchangée aux bornes de l'intervalle de déformation.

Si l'on note $I = [0, S]$ l'intervalle de déformation, ces contraintes s'expriment par les conditions aux limites suivantes :
$\forall \tau \in \mathbb{R}$

$$\eta_\tau(0) = 0 \tag{3.11}$$
$$\eta_\tau(S) = 0 \tag{3.12}$$

Par construction des directions de déformation élémentaire (équation (3.6)), la condition (3.11) est toujours respectée. Par contre, rien ne garantit que la condition (3.12) soit respectée par la direction de déformation η_τ^0 trouvée à l'équation (3.7).

En notant L la matrice de taille $(n \times p)$ avec $p > n$, dont les colonnes sont les vecteurs $\mathbf{F}_i(S, \tau)$, la condition (3.12) s'écrit comme une contrainte linéaire sur le vecteur λ^F :

$$L\lambda^F = 0 \tag{3.13}$$

À partir de notre solution initiale notée λ^{0F}, nous cherchons donc $\bar{\lambda}^F$ solution de :

$$\min_{L\lambda^F=0} \|\lambda^F - \lambda^{0F}\|$$

La solution de problème est à nouveau donné par la projection orthogonale de η_τ^0 sur le sous-espace des directions de déformation de $\mathcal{AD}_p(\mathbf{q}_\tau)$ qui vérifient $\eta_\tau(S) = 0$. Cela revient à projeter orthogonalement λ^{0F} sur le sous-espace de $\mathbb{R}^p$ défini par l'équation (3.13). On trouve :

$$\bar{\lambda}^F = (I_p - L^+L)\lambda^{0F} \tag{3.14}$$

où L^+ est la pseudo-inverse de L, c'est a dire la matrice vérifiant $LL^+L = L$. Dans le cas où (LL^T) est inversible, on a $L^+ = L^T(LL^T)^{-1}$.

Vérification de la solution trouvée

Montrons que $\bar{\lambda}^F$ est bien la solution cherchée. On note $\bar{\eta}_\tau$ la direction de déformation dont les coordonnées dans la base $\mathcal{F}$ sont $\bar{\lambda}^F$:

$$\bar{\eta}_\tau = \sum_{i=1}^{p} \bar{\lambda}_i^F \mathbf{F}_i$$

Tout d'abord, d'après l'expression trouvée à l'équation (3.14), on remarque que $L\bar{\lambda}^F = 0$, ce qui caractérise l'appartenance de $\bar{\lambda}^F$ à l'espace des solutions de l'équation (3.13) et assure que $\bar{\eta}_\tau(S) = 0$.

On doit maintenant s'assurer que la direction de déformation $\bar{\eta}_\tau$ fait bien décroître le potentiel de la trajectoire. Pour cela vérifions que :

$$\frac{dV}{d\tau} = (\mu^F)^T \bar{\lambda}^F \leq 0$$

où μ^F est donné par l'équation (3.9). En utilisant l'expression de $\bar{\lambda}^F$ trouvée à l'équation (3.14), on a :

$$\begin{aligned} (\mu^F)^T \bar{\lambda}^F &= (\mu^F)^T (I_p - L^+ L)\lambda^{0F} \\ &= -(\mu^F)^T \mu^F + (\mu^F)^T L^+ L \mu^F \end{aligned}$$

Comme $(L^+L)(L^+L) = L^+L$, on en déduit que (L^+L) est la matrice d'un opérateur de projection, dont les seules valeurs propres sont 0 et 1. On a donc $\forall x \in \mathbb{R}^p$:

$$0 \leq x^T L^+ L x \leq x^T x$$

Et on en déduit que $(\mu^F)^T \bar{\lambda}^F \leq 0$.

Expression de cette solution dans la base $\mathcal{E}$

Exprimons à nouveau la direction de déformation $\bar{\eta}_\tau$ dans la base $\mathcal{E}$. On note L^E la matrice dont les colonnes sont les vecteurs $\mathbf{E}_i(S,\tau)$ (on rappelle que L est la matrice dont les colonnes sont les vecteurs $\mathbf{F}_i(S,\tau)$). On a alors $L = L^E P$, et comme $\lambda^E = P\lambda^F$, on peut écrire :

$$\begin{aligned} \bar{\lambda}^E &= P\bar{\lambda}^F \\ &= P(I_p - (L^E P)^+ L^E P)\lambda^{0F} \\ &= -(I_p - P(L^E P)^+ L^E) P P^T \mu^E \end{aligned} \tag{3.15}$$

cette dernière expression ne faisant intervenir que des termes exprimés dans la base $\mathcal{E}$ ou qui sont calculables directement à partir des $\mathbf{E}_i$.

3.2.5 Calcul de la trajectoire déformée

On a donc calculé un vecteur $\bar{\lambda}^E$ définissant une fonction de perturbation des entrées $\bar{\mathbf{v}}_\tau = \sum_{i=1}^{p} \bar{\lambda}_i^E \mathbf{e}_i$. Cette fonction de perturbation engendre par intégration du système (3.3) une direction de déformation $\bar{\eta}_\tau$ qui fait baisser le potentiel de la trajectoire et qui respecte les conditions aux limites. Alors en fixant une valeur de $\Delta\tau$ petite, on pourrait appliquer cette perturbation aux entrées courantes sur l'intervalle de déformation $[0, S]$:

$$\mathbf{u}(s, \tau + \Delta\tau) = \mathbf{u}(s, \tau) + \Delta\tau \bar{\mathbf{v}}(s, \tau) \tag{3.16}$$

et intégrer le système (2.7) avec ces nouvelles entrées, pour obtenir une trajectoire déformée.

Mais en raison de cette linéarisation, l'intégration des nouvelles fonctions d'entrée $\mathbf{u}$ sur l'intervalle de déformation produirait une trajectoire qui ne respecterait pas exactement la condition aux limites de l'équation (3.12). Ainsi la trajectoire déformée ne rejoindrait pas exactement la trajectoire de référence à la fin de l'intervalle de déformation.

Pour s'assurer du respect des conditions aux limites, on choisit d'appliquer la déformation calculée $\bar{\eta}_\tau = \sum_{i=1}^{p} \bar{\lambda}^E \mathbf{E}_i$ directement à la trajectoire initiale. Pour $\Delta\tau$ petit on calcule :

$$\mathbf{q}(s, \tau + \Delta\tau) = \mathbf{q}(s, \tau) + \Delta\tau \, \bar{\eta}(s, \tau) \tag{3.17}$$

Remarque

La méthode de déformation de trajectoire présentée ici a été appliquée dans un cadre où il n'y avait pas de conditions aux limites strictes, dans [Boyer 2006]. Le contexte est celui de l'évaluation de la conformité de véhicules automobiles dans des parcours de test. Étant donnée la trajectoire initiale d'un véhicule de type voiture dans le parcours, il s'agit de déformer cette trajectoire de façon à ce qu'elle effectue le parcours sans collision, à la vitesse la plus élevée possible. Il n'y a donc pas de contrainte forte sur la configuration finale. Dans ce contexte, une intégration du système (2.7) avec les nouvelles fonctions d'entrée a été possible.

3.2.6 Respect des contraintes nonholonomes

L'approximation effectuée à l'équation (3.17) nous assure que la trajectoire déformée respecte exactement les conditions aux limites. En contrepartie, en raison de cette linéarisation, cette trajectoire n'est pas exactement solution du système (2.7), et n'est donc pas à proprement parler admissible. Si l'on cherche à exécuter cette trajectoire, on se rendra compte que le système ne peut pas la suivre exactement. On appelle ce phénomène la **déviation des contraintes nonholonomes**, qui va s'amplifier au fur et à mesure des déformations successives de la trajectoire. Pour résoudre ce problème, nous considérons le système étendu de (2.7) en ajoutant $n - k$ champs de vecteurs indépendants :

$$\mathbf{q}'(s) = \sum_{i=1}^{n} u_i(s) \mathbf{X}_i(\mathbf{q}(s)) \tag{3.18}$$

tel que $\forall \mathbf{q} \in \mathcal{C}, (\mathbf{X}_1(\mathbf{q}), \ldots, \mathbf{X}_n(\mathbf{q}))$ engendre $T_\mathbf{q}\mathcal{C}$.

Ce système n'est soumis à aucune contrainte cinématique et une trajectoire de ce système est admissible pour le système (2.7) si et seulement si :

$$\forall j \in \{k+1, .., n\}, \forall s \in [0, S] \quad u_j(s) = 0$$

Correction de la déviation des contraintes nonholonomes

Étant donnée une trajectoire $\mathbf{q}(s)$, nous pouvons calculer les fonctions d'entrée $u_i(s)$ correspondantes, avec $i \in \{1, .., n\}$. Or il se peut que les composantes $u_i(s)$ ne soient pas nulles, pour $i \in \{k+1, .., n\}$. L'objectif est donc de calculer une direction de déformation qui ramène ces composantes vers 0.

Pour cela nous reprenons les calculs effectués à l'équation (3.2), en considérant cette fois le système (3.18). On note alors $\widetilde{A}(s, \tau)$ la matrice de taille $(n \times n)$:

$$\widetilde{A}(s, \tau) = \sum_{i=1}^{n} u_i(s, \tau) \frac{\partial X_i}{\partial \mathbf{q}}(\mathbf{q}(s, \tau)) \tag{3.19}$$

et $\widetilde{B}(s, \tau)$ la matrice de taille $(n \times n)$:

$$\widetilde{B}(s, \tau) = \big(B(s, \tau)\ B^{\perp}(s, \tau)\big) \tag{3.20}$$

avec $B^{\perp}(s, \tau) = \Big(\mathbf{X}_{k+1}(\mathbf{q}(s, \tau)), .., \mathbf{X}_n(\mathbf{q}(s, \tau))\Big)$ la matrice dont les colonnes sont les champs de vecteurs additionnels.

Enfin on note $\tilde{\mathbf{v}} = (\mathbf{v}, \mathbf{v}^{\perp})$, où $\mathbf{v}^{\perp}$ sont les fonctions de perturbations suivant les champs de vecteurs additionnels.

Le système (3.3) s'écrit maintenant :

$$\tilde{\eta}'(s, \tau) = \widetilde{A}(s, \tau)\tilde{\eta}(s, \tau) + B(s, \tau)\mathbf{v}(s, \tau) + B^{\perp}(s, \tau)\mathbf{v}^{\perp}(s, \tau) \tag{3.21}$$

Pour ramener $\mathbf{v}^{\perp}$ vers 0, nous effectuons une régulation proportionnelle :

$$\forall j \in \{k+1, .., n\}, \forall s \in [0, S] \quad v_i(s, \tau) = \frac{\partial u_i}{\partial \tau}(s, \tau) = -\alpha u_i(s, \tau)$$

où α est un réel positif. Ainsi, en l'absence de perturbation, les commandes sur les champs de vecteurs additionnels tendent exponentiellement vers 0 lorsque τ augmente :

$$\forall s \in [0, S],\ u_i(s, \tau) = e^{-\alpha\tau} u_i(s, 0)$$

Soit alors $\eta_{\tau}^{\perp}$ la direction de déformation engendrée par la fonction de perturbation $\mathbf{v}_{\tau}^{\perp}$, à τ fixé. $\eta_{\tau}^{\perp}$ est solution de l'équation différentielle suivante :

$$\begin{aligned} \eta'^{\perp}(s, \tau) &= \widetilde{A}(s, \tau)\eta^{\perp}(s, \tau) + B^{\perp}(s, \tau)\mathbf{v}^{\perp}(s, \tau) \\ \eta'^{\perp}(0, \tau) &= 0 \end{aligned} \tag{3.22}$$

Déformation due aux obstacles

Les fonctions de perturbations $\mathbf{v}$ suivant les champs de vecteurs $\mathbf{X}_1, .., \mathbf{X}_k$ restent identiques à celles trouvées à l'équation (3.9). On rappelle la direction de déformation associée trouvée à l'équation(3.7) :

$$\eta_\tau = \sum_{i=1}^{p} \lambda_i^{0F} \mathbf{F}_i$$

Conditions aux limites

Étant donné que le système (3.21) est linéaire par rapport à $\tilde{\mathbf{v}}$, la déformation engendrée, à τ fixé, par la somme des fonctions de perturbations $\mathbf{v}_\tau$ et $\mathbf{v}_\tau^\perp$ est :

$$\tilde{\eta}_\tau = \eta_\tau + \eta_\tau^\perp$$

Alors en appliquant le même raisonnement qu'à la section 3.2.4, on cherche à ce que cette direction de déformation respecte les conditions aux limites.

La condition initiale (3.11) est respectée par construction de $\tilde{\eta}_\tau$. La condition finale s'écrit maintenant, à τ fixé :

$$\tilde{\eta}_\tau(S) = \eta_\tau(S) + \eta_\tau^\perp(S) = 0$$

En reprenant les notations de (3.13), cette équation s'écrit :

$$L\lambda = -\eta_\tau^\perp(S) \tag{3.23}$$

On procède alors de la même façon qu'à la section 3.2.4, et on projette orthogonalement le vecteur λ^F sur l'espace des solutions de (3.23). Soit $\tilde{\lambda}^F$ cette valeur :

$$\tilde{\lambda}^F = -L^+\eta_\tau^\perp(S) + (I_p - L^+L)\lambda^F$$

Expression dans la base $\mathcal{E}$

Nous pouvons exprimer ce vecteur dans la base $\mathcal{E}$ en procédant de la même façon qu'en (3.15), et on trouve :

$$\begin{aligned} \tilde{\lambda}^E &= -P(L^EP)^+\eta_\tau^\perp(S) + P(I_p - (L^EP)^+)\bar{\lambda}^F \\ &= -P(L^EP)^+\eta_\tau^\perp(S) - (I_p - P(L^EP)^+L^E)PP^T\mu^E \end{aligned} \tag{3.24}$$

3.2.7 Algorithme de la déformation de trajectoire

Les sections précédentes ont montré comment déformer la trajectoire de façon à ce que la trajectoire déformée :

- s'éloigne des obstacles (Cf. section 3.2.3),
- respecte strictement les conditions aux limites de l'intervalle de déformation (Cf. section 3.2.4),
- soit une trajectoire admissible pour le système (2.7) (Cf. section 3.2.6).

Nous présentons ici un résumé de ces différentes étapes, qui constituent l'algorithme de la déformation de trajectoire.

Données d'entrée

- $I = [0, S]$ l'intervalle de déformation,
- une trajectoire $\mathbf{q}(s)$ définie sur I, non nécessairement solution du système (2.7),
- les n fonctions entrées $u_i(s), s \in I$ suivant les n champs de vecteurs du système (3.18),
- le gradient du potentiel que l'on souhaite minimiser $\frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}(s)), s \in I$,
- p fonctions élémentaires de perturbation $e_i(s), s \in I$,
- une valeur η_{max} de la déformation maximale autorisée de la trajectoire,
- $\tau \leftarrow 0$.

Algorithme

Calcul des matrices A et B
- Calculer $\widetilde{A}(s,\tau)$ et $\widetilde{B}(s,\tau)$ en utilisant les équations (3.19) et (3.20)

Correction de la déviation des contraintes nonholonomes
- pour $i \in \{k+1, .., n\}$ calculer la correction $v_i(s,\tau) = -\alpha u_i(s,\tau)$
- Calculer la déformation associée $\eta^{\perp}(s,\tau)$ en intégrant (3.22)

Déformation élémentaire
- Pour $i \in \{1, .., p\}$ calculer les **directions de déformation élémentaires** $\mathbf{E}_i(s,\tau)$ en intégrant (3.6)
- Pour $i \in \{1, .., p\}$ calculer μ^E en utilisant l'équation (3.10)

Base orthonormale
- Calculer la matrice P en utilisant le procédé d'orthonormalisation de Gram-Schmidt sur les vecteurs $\mathbf{E}_i(s,\tau)$

Conditions aux limites
- Calculer $\tilde{\lambda}^E$ en utilisant l'équation (3.24)

Direction de déformation
- Calculer la direction de déformation $\tilde{\eta}(s,\tau) = \sum_{i=1}^{p} \tilde{\lambda}_i^E \mathbf{E}_i(s,\tau) \ + \ \eta^{\perp}(s,\tau)$

Déformation de la trajectoire
- Calculer $\Delta\tau = \frac{\eta_{max}}{\|\tilde{\eta}(s,\tau)\|_\infty}$
- Calculer la nouvelle trajectoire : $\mathbf{q}(s, \tau + \Delta\tau) = \mathbf{q}(s,\tau) + \Delta\tau\tilde{\eta}(s,\tau)$
- Incrémenter τ : $\tau \leftarrow \tau + \Delta\tau$

Cet algorithme est appliqué itérativement jusqu'à ce que la trajectoire ne soit plus en collision.

3.2.8 Prise en compte des bornes sur les entrées du système

Un robot mobile est soumis à des contraintes sur les valeurs de ses entrées et de leurs dérivées. On suppose que ces contraintes peuvent de mettre sous la forme :
$\forall i \in \{1, ..., k\}$, et $\forall s \in [0, S]$

$$\begin{array}{rcl} -u_{iMin} \leq & u_i(s) & \leq u_{iMax} \\ -\dot{u}_{iMin} \leq & \frac{\partial u_i}{\partial s}(s) & \leq \dot{u}_{iMax} \end{array} \tag{3.25}$$

La méthode de déformation de trajectoire que nous avons présentée ne garantit pas que les fonctions d'entrée de la trajectoire déformée respectent ces contraintes. En effet, on assure qu'à chaque itération de l'algorithme la perturbation des fonctions d'entrée de la trajectoire est petite, mais au fur et à mesure du processus de déformation, les entrées peuvent augmenter et dépasser les limites définies par le système d'inégalités (3.25). Bien souvent par exemple, les entrées du système sont des vitesses et la longueur de la trajectoire déformée pour éviter un obstacle est supérieure à la longueur de la trajectoire initiale. Étant donné que la trajectoire déformée est définie sur le même intervalle d'abscisse, cela signifie que la vitesse a augmenté.

Il faut donc contrôler l'évolution des fonctions d'entrée au cours du processus de déformation et éviter les dépassements des valeurs maximales. Nous présentons ici le principe de la méthode développée par Mathieu Hillion lors de son stage [Hillion 2005], à l'encadrement duquel nous avons collaboré.

La méthode se décompose en deux parties :

– définir des fonctions de perturbations nulles sur les intervalles où les entrées dépassent leurs limites et non nulles ailleurs,
– calculer un reparamétrage de la trajectoire qui assure le respect des limites.

Fonctions de perturbations

Par exemple, une trajectoire étant définie sur l'intervalle $[0, 10]$, on remarque qu'une de ses entrées u_i ne respecte pas les contraintes de l'équation (3.25) sur l'intervalle $[2, 10]$. On va donc définir des fonctions de perturbations de cette entrée telles qu'elles soient nulles sur l'intervalle $[2, 10]$ et non nulles sur l'intervalle $[0, 2]$. La figure 3.1 représente deux fonctions de perturbations sinusoïdales $\mathbf{e}_1$ et $\mathbf{e}_2$ pour l'entrée u_i, qui respectent ces contraintes.

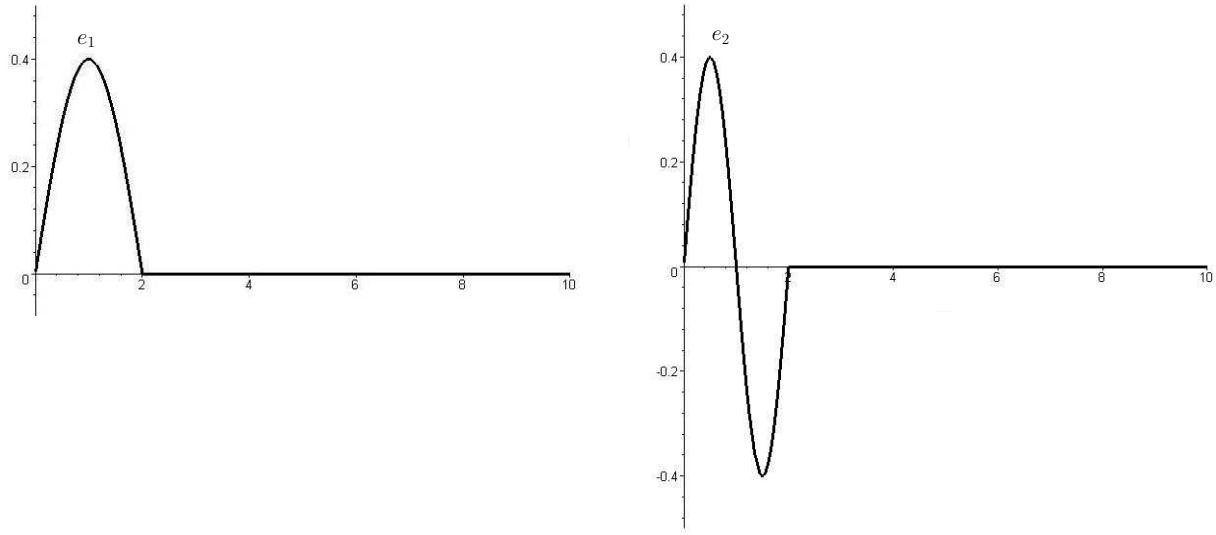

FIG. 3.1 – Fonctions de perturbations des entrées, bloquées sur l'intervalle $[2, 10]$.

Cette première amélioration permet de s'assurer que les entrées de la trajectoire déformée sont bloquées quand elles dépassent leurs valeurs limites.

Reparamétrage de la trajectoire

Il s'agit de trouver une fonction de reparamétrage $\phi, s \mapsto t = \phi(s)$, telle que les entrées de la trajectoire définie comme une fonction de l'abscisse t sur l'intervalle $[0, \phi(S)]$ respectent les contraintes de l'équation (3.25). Ce reparamétrage de la trajectoire a pour conséquence que les entrées définies comme une fonction de la nouvelle abscisse t on l'expression suivante :
$\forall i \in\in \{1, ..., k\}$

$$u_i(t) \quad = \quad \frac{1}{\phi'(s)} u_i(s)$$

L'allure de la fonction ϕ' va déterminer le rapport entre les entrées avant et après reparamétrage.

La fonction ϕ' choisie est en l'occurrence une parabole, de façon à ce que l'influence soit maximale au milieu de la trajectoire. Les paramètres de cette parabole sont des fonctions des valeurs maximales des entrées et de leurs dérivées.

Ce reparamétrage n'est bien sûr pas optimal (il n'assure pas un mouvement en temps minimal) mais il est très rapide à calculer et il pourra donc être exécuté en temps-réel, lors de l'exécution de la trajectoire par le robot.

3.3 Application de la méthode de déformation de trajectoire : le robot Cycab

Nous détaillons cet algorithme sur un exemple, développé à partir du robot Cycab représenté sur la figure (3.2).

3.3.1 Modèle cinématique

Le robot Cycab possède plusieurs modes cinématiques. Il est par exemple possible de contrôler indépendamment l'angle de braquage des roues arrière et celui des roues avant. Nous présentons ici le modèle cinématique dans lequel les roues arrière sont bloquées et seules les roues avant peuvent braquer (figure 3.3).

Ce système possède 4 variables de configuration : $\mathbf{q} = (x, y, \theta, \phi)$. x et y donnent la position du centre P de l'essieu arrière du robot, exprimés en mètre. θ est l'orientation de l'axe perpendiculaire à l'essieu arrière. On note O le centre de courbure du système, l étant la distance entre l'essieu arrière et l'essieu avant et Q le milieu de l'essieu avant. ϕ est alors l'angle qui placerait en O le centre de courbure d'une roue avant située en Q. La courbure est notée κ et on a $\kappa = \frac{\tan \phi}{l}$.

Ce système est soumis à 2 contraintes nonholonomes :

– $\vec{v_P}$ est colinéaire au vecteur $\begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$

FIG. 3.2 – Le robot Cycab, un robot de type voiture

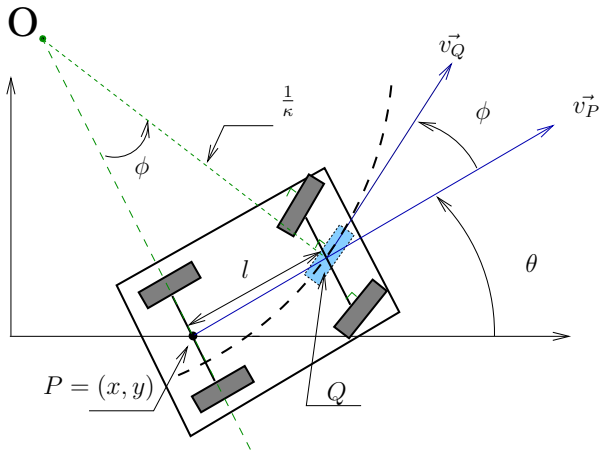

FIG. 3.3 – Modèle cinématique du robot Cycab

– $\vec{v_Q}$ est colinéaire au vecteur $\begin{pmatrix} \cos(\theta+\phi) \\ \sin(\theta+\phi) \end{pmatrix}$

Soient $\mathbf{X}_1(\mathbf{q}) = \begin{pmatrix} \cos\theta \\ \sin\theta \\ \frac{\tan\phi}{l} \\ 0 \end{pmatrix}$ et $\mathbf{X}_2(\mathbf{q}) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ deux champs de vecteurs du système. On a alors :

$$\mathbf{q}'(s) = u_1(s)\mathbf{X}_1(\mathbf{q}(s)) + u_2(s)\mathbf{X}_2(\mathbf{q}(s)) \tag{3.26}$$

où u_1 représente la vitesse linéaire du véhicule et u_2 la vitesse de braquage.

3.3.2 Calculs pour la déformation de trajectoire

Des champs de vecteurs additionnels pour obtenir le système étendu (3.18) sont par exemple :

$$\mathbf{X}_3(\mathbf{q}) = \begin{pmatrix} -\sin\theta \\ \cos\theta \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{X}_4(\mathbf{q}) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

L'expression des matrices $\widetilde{A}$ et $\widetilde{B}$ de l'équation (3.21) à partir des champs de vecteurs $\mathbf{X}_1, \ldots, \mathbf{X}_4$ est immédiate.

On trouve :

$$\widetilde{A}(s) = \begin{pmatrix} 0 & 0 & -u_1(s)\sin\theta - u_3(s)\cos\theta & 0 \\ 0 & 0 & u_1(s)\cos\theta - u_3(s)\sin\theta & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \widetilde{B}(s) = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ \frac{\tan\phi}{l} & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

avec $\theta = \theta(s)$ et $\phi = \phi(s)$.

La base de fonctions de perturbations $\mathbf{e}_i$ définie en (3.5) que nous utilisons est une série de $2p$ fonctions sinusoïdales de périodes décroissantes avec l'ordre p :

$$\mathbf{e}_1(s) = \left(\sqrt{\frac{2}{S}}\sin\left(\frac{\pi s}{S}\right), 0\right)^T \qquad \mathbf{e}_2(s) = \left(0, \sqrt{\frac{2}{S}}\sin\left(\frac{\pi s}{S}\right)\right)^T$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$\mathbf{e}_{2p-1}(s) = \left(\sqrt{\frac{2}{S}}\sin\left(\frac{p\pi s}{2S}\right), 0\right)^T \qquad \mathbf{e}_{2p}(s) = \left(0, \sqrt{\frac{2}{S}}\sin\left(\frac{(p-1)\pi s}{2S}\right)\right)^T$$

où $[0, S]$ est l'intervalle de définition de la trajectoire. Les deux composantes de $\mathbf{e}_i$ correspondent respectivement à une perturbation de u_1 et de u_2.

3.3.3 Exemple de déformation

Pour cet exemple on se donne une trajectoire initiale ($\tau = 0$) définie sur l'intervalle $[0, 10]$, avec pour fonctions d'entrée :

$$\forall s \in [0, 10] \qquad u_1(s) = 1 \quad u_2(s) = 0$$

et pour configuration initiale $\mathbf{q}(0) = (0, 0, 0, 0)^T$, ce qui correspond à une trajectoire rectiligne. Les entrées suivant les champs de vecteurs additionnels $\mathbf{X}_3(\mathbf{q})$ et $\mathbf{X}_4(\mathbf{q})$ sont nulles. L'étape dite de *correction des dérives des contraintes nonholonomes* engendre donc ici une direction de déformation $\eta^\perp$ nulle.

Le gradient du potentiel dans l'espace des configurations est constant et vaut :

$$\forall s \in [0, 10] \quad \frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}_\tau(s)) = \begin{pmatrix} 0 \\ -0.1 \\ 0 \end{pmatrix}$$

ce qui représente une répulsion de la trajectoire dans la direction des y positifs.

La figure 3.4 représente la première composante des fonctions de perturbations élémentaires $\mathbf{e}_i$ (perturbation de u_1 donc). L'ordre p est fixé à 4, et les premières composantes des fonctions $\mathbf{e}_2(s), \mathbf{e}_4(s), \mathbf{e}_6(s), \mathbf{e}_8(s)$ sont donc nulles pour $s \in [0, 10]$ et ne sont pas représentées.

La figure 3.5 représente la direction de déformation élémentaire $\mathbf{E}_2$ correspondant à l'intégration par le système (3.3) de la fonction de perturbation $\mathbf{e}_2(s) = \left(0, \sqrt{\frac{2}{S}}\sin\left(\frac{\pi s}{S}\right)\right)^T$.

La direction de déformation calculée par l'équation (3.24) est représentée sur la figure 3.6. On vérifie bien que la condition finale $\eta(S) = 0$ est bien respectée. Par ailleurs on remarque que cette direction de déformation fait apparaître une déformation vers les y positifs, ce à quoi on s'attendait intuitivement au vu du gradient du potentiel choisi.

Les coordonnées de la direction de déformation η dans la base $(\mathbf{E}_1, \ldots, \mathbf{E}_8)$ sont :

$$\lambda = (0\ 0\ 0\ -2.14\ 0\ 0\ 0\ 4.28)^T$$

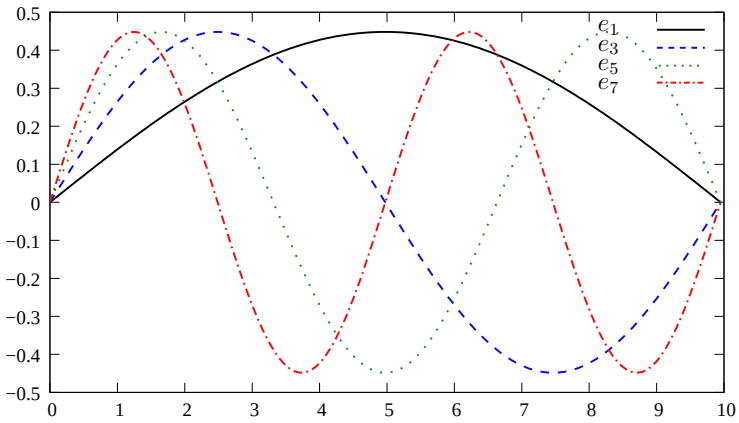

FIG. 3.4 – Premières composantes des fonctions de perturbations élémentaires

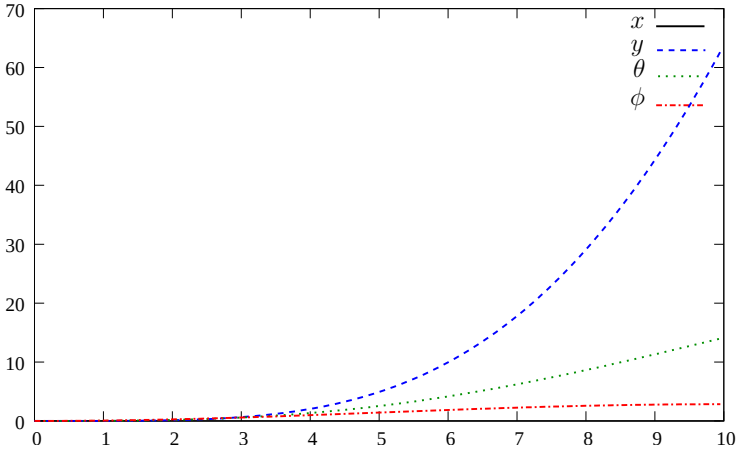

FIG. 3.5 – Direction de déformation élémentaire $\mathbf{E}_2$ engendrée par la perturbation élémentaire e_2

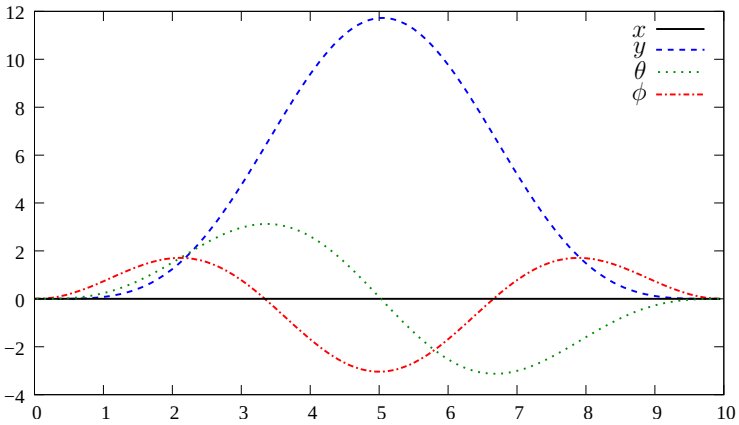

FIG. 3.6 – Direction de déformation η

c'est à dire que les perturbations suivant la première composante u_1 des fonctions d'entrée sont toutes nulles ($\lambda_1 = \lambda_3 = \lambda_5 = \lambda_7 = 0$). En effet une modification de la vitesse linéaire du robot n'aurait aucun effet sur le potentiel de la trajectoire.

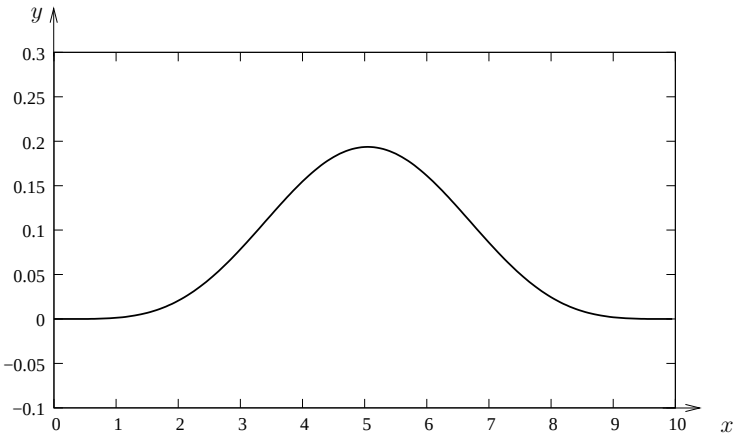

FIG. 3.7 – Trajectoire déformée

La nouvelle trajectoire déformée est calculée en appliquant :

$$\mathbf{q}(s, \Delta\tau) = \mathbf{q}(s, 0) + \Delta\tau\eta(s) \tag{3.27}$$

Elle est représentée sur la figure 3.7, sur laquelle on observe effectivement une déformation vers les y positifs. La valeur de Δ_τ est fixée par la valeur maximale de déformation autorisée : $\Delta\tau = \frac{\eta_{max}}{\|\eta(s)\|_\infty}$. En l'occurence η_{max} vaut 0.2.

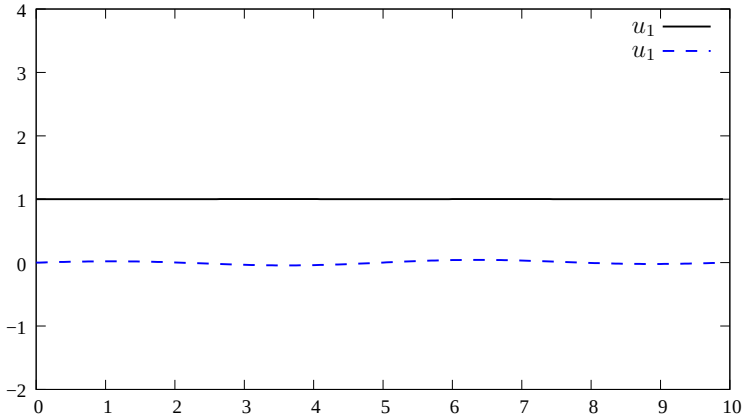

FIG. 3.8 – Fonctions d'entrée de la trajectoire déformée

La figure 3.8 représente les fonctions d'entrée u_1 et u_2 de la trajectoire déformée. On vérifie que u_1 reste inchangée (vitesse linéaire), et que u_2 est légèrement perturbée (variation de l'angle de braquage).

Enfin on conclut cet exemple en remarquant que les fonctions d'entrée suivant les champs de vecteurs additionnels $\mathbf{X}_3$ et $\mathbf{X}_4$ ne sont pas tout à fait nulles (de l'ordre de 10^{-4}, à comparer à $u_1(s) = 1$), du fait de l'approximation effectuée à l'équation (3.27). Ces entrées seront régulées à l'itération suivante grâce à la correction des dérives des contraintes nonholonomes présentée à la section 3.2.6.

3.4 Calcul des interactions robot-obstacles

La déformation de trajectoire pour systèmes nonholonomes présentée dans ce chapitre est une méthode d'évitement réactif d'obstacles. On doit donc calculer les interactions entre la trajectoire planifiée et les obstacles détectés.

Plus précisément, cette méthode réalise successivement les tâches suivantes sur la trajectoire discrétisée :

1. détection des collisions de la trajectoire avec les obstacles détectés,
2. calcul du potentiel de la trajectoire sur un intervalle centré sur la première collision,
3. déformation de la trajectoire sur cet intervalle.

La tâche 3 correspond à l'algorithme décrit à la section 3.2.7. L'intégration de ces différentes tâches au sein d'une architecture unifiée est décrite dans le chapitre 5.

On s'intéresse dans cette section aux opérations effectuées par les tâches 1 et 2. Tout d'abord nous présentons le calcul du potentiel d'une trajectoire, dans le cas où le potentiel d'une configuration est une fonction de la distance aux obstacles. Ensuite nous présentons un algorithme permettant d'optimiser les calculs réalisés dans ces deux tâches. Cet algorithme tire parti du fait que ces opérations (détections des collisions et calcul du potentiel) s'effectuent en parcourant itérativement la trajectoire discrétisée. Ces résultats ont été publiés dans [Lefebvre 2005].

3.4.1 Potentiel dans l'espace des configurations

La méthode de déformation de trajectoire que nous utilisons fait appel au potentiel d'une trajectoire, et calcule une déformation de la trajectoire qui fait baisser ce potentiel. En reprenant les notations de la section 3.2.2, on note :

$$V(\mathbf{q}) = \int_0^S U(\mathbf{q}(s))ds$$

le potentiel d'une trajectoire définie sur $[0, S]$.

Dans le cadre de l'évitement réactif d'obstacles, on définit ce potentiel comme une fonction de la distance du robot aux obstacles.

Potentiel fonction de la distance aux obstacles

On souhaite que le potentiel soit une fonction décroissante de la distance aux obstacles, jusqu'à une distance maximale d_1 au-delà de laquelle les obstacles n'ont plus d'influence. On définit

la fonction U_d suivante qui remplit ces critères :

$$\begin{array}{rccl} U_d: & \mathbb{R}^+ & \rightarrow & \mathbb{R} \\ & d & \mapsto & U_d(d) = \left\{ \begin{array}{cc} \frac{1}{d+d_0} - \frac{1}{d_1+d_0} & \text{si } 0 \leq d \leq d_1 \\ 0 & \text{si } d > d_1 \end{array} \right. \end{array} \tag{3.28}$$

où d_0 est un réel positif permettant de paramétrer la valeur maximale du potentiel.

Alors, étant donné un obstacle $\mathcal{O}$, le potentiel engendré par l'interaction entre cet obstacle et le robot dans la configuration $\mathbf{q}$ est :

$$U(\mathbf{q}) = U_d(d_O(\mathbf{q}, \mathcal{O}))$$

On étend naturellement cette définition au cas des véhicules articulés dans un environnement contenant plusieurs obstacles. Soit un robot mobile composé de n_b corps $\mathcal{B}_1$,...,$\mathcal{B}_{n_b}$, et $\mathcal{B}_i(\mathbf{q}) \subset \mathcal{W}$, le sous-espace occupé par le corps $\mathcal{B}_i$ lorsque le robot est dans la configuration $\mathbf{q}$. Soit n_o le nombre d'obstacles de l'environnement détectés par le capteur :$\mathcal{O}_1$,...,$\mathcal{O}_{n_o}$. Le potentiel d'une configuration est alors :

$$U(\mathbf{q}) = \sum_{i=1}^{n_b} \sum_{j=1}^{n_o} U_d(d_{ij}(\mathbf{q})) \tag{3.29}$$

où :

$$d_{ij}(\mathbf{q}) = d(\mathcal{B}_i(\mathbf{q}), \mathcal{O}_j)$$

La valeur qui nous intéresse dans l'algorithme de déformation (3.2.7) est le gradient du potentiel dans l'espace des configurations, qui s'écrit :

$$\frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}) = \sum_{i=1}^{n_b} \sum_{j=1}^{n_o} U'_d(d_{ij}(\mathbf{q})) \frac{\partial d_{ij}}{\partial \mathbf{q}}(\mathbf{q})$$

En notant respectivement $o(\mathbf{q})$ et $r(\mathbf{q})$ les points de l'obstacle j et du corps i les plus proches quand le robot est dans la configuration $\mathbf{q}$ (figure 3.9), on a :

$$d_{ij}(\mathbf{q}) = \|o(\mathbf{q}) - r(\mathbf{q})\|$$

Ainsi, pour une configuration $\mathbf{q}$ donnée, le calcul de $d_{ij}(\mathbf{q})$ et de $U'_d(d_{ij}(\mathbf{q})$ ne pose pas de problème particulier. Par contre, le calcul de $\frac{\partial d_{ij}}{\partial \mathbf{q}}(\mathbf{q})$ n'est *a priori* pas aisé. C'est l'objet du paragraphe suivant.

Calcul du gradient de la distance sans expression analytique des points les plus proches

Le gradient de d_{ij} est :

$$\frac{\partial d_{ij}}{\partial \mathbf{q}}(\mathbf{q}) = \frac{(o(\mathbf{q}) - r(\mathbf{q}))^T}{\|o(\mathbf{q}) - r(\mathbf{q})\|} \left(\frac{\partial o}{\partial \mathbf{q}}(\mathbf{q}) - \frac{\partial r}{\partial \mathbf{q}}(\mathbf{q}) \right) \tag{3.30}$$

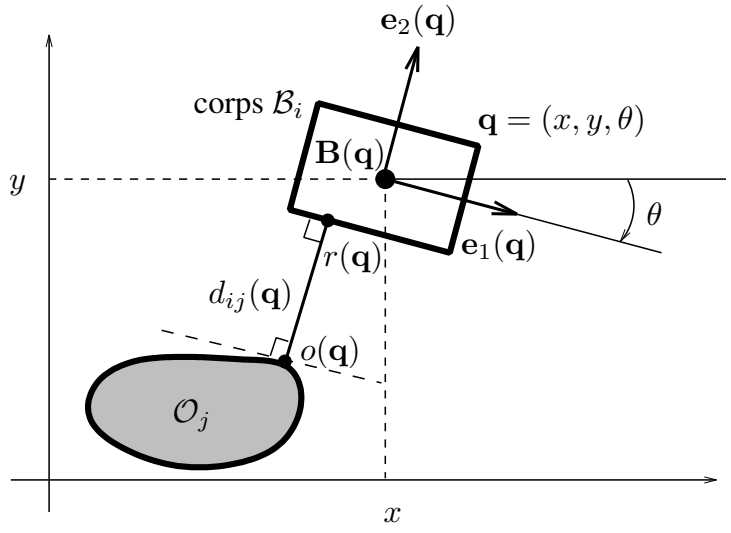

FIG. 3.9 – Le potentiel dans l'espace des configurations dépend seulement de la distance $\|o(\mathbf{q}) - r(\mathbf{q})\|$ entre le robot et l'obstacle. Le mouvement du point $r(\mathbf{q})$ sur le corps du robot (mouvement relatif) est orthogonal au vecteur $\vec{or}$.

D'après cette expression, il semblerait que l'on doive exprimer analytiquement les positions des points $o(\mathbf{q})$ et $r(\mathbf{q})$. Ces expressions peuvent être délicates à obtenir, car elles dépendent de la forme de l'obstacle et de la forme du corps du robot considéré. Les travaux pionniers de [Khatib 1986] sur l'évitement réactif d'obstacles en robotique, basé sur des champs de potentiels, donnaient ainsi l'expression analytique de la distance entre deux corps pour quelques cas particuliers (cylindre, sphère, parallélépipède, etc.). Nous allons montrer que le mouvement des points $o(\mathbf{q})$ et $r(\mathbf{q})$ sur leurs corps respectifs n'a pas d'influence pour le calcul de $\frac{\partial d_{ij}}{\partial \mathbf{q}}(\mathbf{q})$, et qu'il n'est donc pas nécessaire de posséder une expression analytique de la distance entre deux corps en fonction de leur configuration.

Soit un repère $(\mathbf{B}(\mathbf{q}), \mathbf{e}_1(\mathbf{q}), \mathbf{e}_2(\mathbf{q}))$ attaché au corps $\mathcal{B}_i$ considéré du robot[1]. On note r_1^B, r_2^B les coordonnées de $r(\mathbf{q})$ exprimées dans le repère $(\mathbf{B}(\mathbf{q}), \mathbf{e}_1(\mathbf{q}), \mathbf{e}_2(\mathbf{q}))$:

$$\overrightarrow{\mathbf{B}(\mathbf{q})r(\mathbf{q})} = \sum_{l=1}^{2} r_l^B(\mathbf{q})\mathbf{e}_l(\mathbf{q})$$

On suppose que $r(\mathbf{q})$ est un «point régulier» de la frontière de $\mathcal{B}_i$ tel que $\frac{\partial r}{\partial \mathbf{q}}(\mathbf{q})$ existe. Alors on peut écrire :

$$\frac{\partial r}{\partial \mathbf{q}}(\mathbf{q}) = \overbrace{\frac{\partial \mathbf{B}}{\partial \mathbf{q}}(\mathbf{q}) + \sum_{l=1}^{2} r_l^B(\mathbf{q})\frac{\partial \mathbf{e}_l}{\partial \mathbf{q}}(\mathbf{q})}^{\text{mouvement du point coïncidant}} + \overbrace{\sum_{l=1}^{2} \frac{\partial r_l^B}{\partial \mathbf{q}}(\mathbf{q})\mathbf{e}_l(\mathbf{q})}^{\text{mouvement relatif}} \tag{3.31}$$

[1]Ces résultats s'étendent immédiatement à un espace de travail de dimension 3

Le mouvement relatif de $r(\mathbf{q})$ sur le corps considéré est orthogonal au vecteur $(o(\mathbf{q}) - r(\mathbf{q}))$, car $r(\mathbf{q})$ se déplace sur la frontière du corps, comme illustré par la figure 3.9, et on a :

$$(o(\mathbf{q}) - r(\mathbf{q}))^T \sum_{l=1}^{2} \frac{\partial r_l^B}{\partial \mathbf{q}}(\mathbf{q}) \mathbf{e}_l(\mathbf{q}) = 0$$

Si l'on note $r_{\in\mathcal{B}}(\mathbf{q})$ le point coïncidant avec $r(\mathbf{q})$, on peut alors remplacer $\frac{\partial r}{\partial \mathbf{q}}(\mathbf{q})$ par $\frac{\partial r_{\in\mathcal{B}}}{\partial \mathbf{q}}(\mathbf{q})$ dans l'expression (3.30).

En appliquant maintenant le même raisonnement avec le point $o(\mathbf{q})$ et sous l'hypothèse que les obstacles sont statiques, on a :

$$(o(\mathbf{q}) - r(\mathbf{q}))^T \frac{\partial o}{\partial \mathbf{q}}(\mathbf{q}) = 0$$

Et l'expression (3.30) s'écrit alors simplement :

$$\frac{\partial d_{ij}}{\partial \mathbf{q}}(\mathbf{q}) = \frac{(r(\mathbf{q}) - o(\mathbf{q}))^T}{\|o(\mathbf{q}) - r(\mathbf{q})\|} \frac{\partial r_{\in\mathcal{B}}}{\partial \mathbf{q}}(\mathbf{q}) \tag{3.32}$$

Ainsi le calcul du gradient du potentiel dans l'espace des configurations ne nécessite pas l'expression analytique des points les plus proches en fonction de la configuration du robot, mais seulement leur position dans l'espace de travail pour une configuration donnée.

Remarque Si $r(\mathbf{q})$ n'est pas un «point régulier», par exemple s'il est situé à un angle de la frontière du corps considéré, on fait l'hypothèse raisonnable que le mouvement relatif de ce point est nul (voir la figure 3.10). Cela revient à faire l'hypothèse que le point le plus proche de l'obstacle est fixe dans le repère $(\mathbf{B}(\mathbf{q}), \mathbf{e}_1(\mathbf{q}), \mathbf{e}_2(\mathbf{q}))$. Et donc les $\frac{\partial r_l^B}{\partial \mathbf{q}}(\mathbf{q})$ sont nuls, c'est à dire que le mouvement relatif est nul, et le raisonnement précédent reste valable : on ne doit considérer que le mouvement du point coïncidant, $\frac{\partial r_{\in\mathcal{B}}}{\partial \mathbf{q}}(\mathbf{q})$.

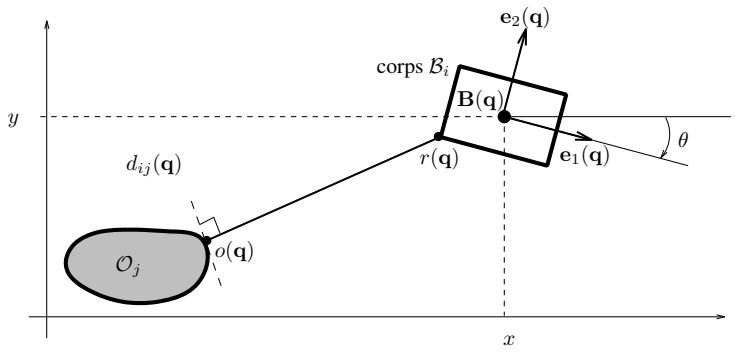

FIG. 3.10 – Dans le cas où le point le plus proche n'est pas «régulier», on fait l'hypothèse que son mouvement relatif est nul, et le raisonnement reste valable.

Exemple d'un robot dans le plan

Nous détaillons, au travers d'un exemple illustré par la figure 3.9, le calcul du gradient du potentiel généré par un obstacle $\mathcal{O}_j$ sur un robot composé d'un corps $\mathcal{B}_i$ dans un plan. La configuration du robot est notée $\mathbf{q} = (x, y, \theta)$. Le potentiel généré par l'obstacle sur le corps $\mathcal{B}_i$ est donné par l'équation (3.29) : $U(\mathbf{q}) = U_d(d(\mathcal{B}_i(\mathbf{q}), \mathcal{O}_j)$.

En notant $\mathbf{B}(\mathbf{q})$ le centre cinématique du corps $\mathcal{B}_i$, on définit le repère $(\mathbf{B}(\mathbf{q}), \mathbf{e}_1(\mathbf{q}), \mathbf{e}_2(\mathbf{q}))$ attaché à ce corps. Soit $o(\mathbf{q}) = (o_1, o_2)$ le point de $\mathcal{O}_j$ le plus proche de $\mathcal{B}_i(\mathbf{q})$. Soit $r(\mathbf{q}) = (r_1, r_2)$ le point de $\mathcal{B}_i(\mathbf{q})$ le plus proche de $\mathcal{O}_j$ et (r_1^B, r_2^B) ses coordonnées dans le repère $\mathbf{B}$. On a alors :

$$r_{\in\mathcal{B}} = \begin{pmatrix} x + r_1^B \cos\theta - r_2^B \sin\theta \\ y + r_1^B \sin\theta + r_2^B \cos\theta \end{pmatrix}$$

Et en dérivant par rapport à $\mathbf{q}$:

$$\begin{pmatrix} 1 & 0 & -r_1^B \sin\theta - r_2^B \cos\theta \\ 0 & 1 & r_1^B \cos\theta - r_2^B \sin\theta \end{pmatrix}$$

En notant $A = -r_1^B \sin\theta - r_2^B \cos\theta$ et $B = r_1^B \cos\theta - r_2^B \sin\theta$, l'équation (3.32) devient alors :

$$\begin{aligned} \frac{\partial d_{ij}}{\partial \mathbf{q}}(\mathbf{q}) &= \frac{(r(\mathbf{q}) - o(\mathbf{q}))^T}{\|o(\mathbf{q}) - r(\mathbf{q})\|} \frac{\partial r_{\in\mathcal{B}}}{\partial \mathbf{q}} \\ &= \frac{1}{d} \begin{pmatrix} r_1 - o_1 \\ r_2 - o_2 \\ (r_1 - o_1)A + (r_2 - o_2)B \end{pmatrix}^T \end{aligned}$$

où $d = \sqrt{(r_1 - o_1)^2 + (r_2 - o_2)^2}$. Et on obtient finalement le gradient du potentiel dans l'espace des configurations :

$$\frac{\partial U}{\partial \mathbf{q}}(\mathbf{q}) = \frac{U_d'(d)}{d} \begin{pmatrix} r_1 - o_1 \\ r_2 - o_2 \\ (r_1 - o_1)A + (r_2 - o_2)B \end{pmatrix}^T$$

où U_d est la fonction (3.28). On remarque que le dernier terme correspond au moment d'une force appliquée au point $r(\mathbf{q})$, qui entraîne une rotation du corps $\mathcal{B}_i$ autour de son centre $\mathbf{B}(\mathbf{q})$.

3.4.2 Optimisation des calculs des interactions

Comme nous l'avons mentionné au début de cette section, les tâches de détections des collisions (1) et de calcul du gradient du potentiel (2) parcourent un intervalle de la trajectoire discrétisée pour effectuer leurs calculs. Ainsi pour chaque configuration de la trajectoire discrétisée, on doit calculer les interactions (détection de collision ou calcul du gradient du potentiel) pour chacun des corps du robot avec chacun des obstacles perçus. Si l'on note n_s le nombre de configurations sur l'intervalle considéré, la complexité du calcul est $n_s \times n_b \times n_o$, où n_b est

le nombre de corps composant la chaîne cinématique du robot, et n_o est le nombre d'obstacles perçus.

Dans la tâche (1), n_s est proportionnel à la distance sur laquelle sont détectées les collisions sur la trajectoire planifiée, en partant de la configuration courante du robot. Dans la tâche (2), n_s est proportionnel à la taille de l'intervalle de déformation. Ainsi si le pas de discrétisation est très petit, ce qui est le cas quand on navigue proche des obstacles, la valeur de n_s peut être très grande (de l'ordre de 1000 pour une trajectoire de 5 mètres par exemple). Pour cette raison nous avons cherché à diminuer la complexité de ces calculs et nous avons développé un algorithme qui les optimise, publié dans [Lefebvre 2005]. On peut trouver des idées similaires à cet algorithme basé sur la cohérence spatiale de la trajectoire, dans des travaux d'animation graphique tels que ceux de [Baraff 1992] et [Lin 1993].

Distance d'influence

Dans le cadre de l'évitement réactif d'obstacles on définit généralement deux distances :
- une distance minimale aux obstacles en deçà de laquelle une configuration est considérée en collision, notée D^{clear},
- une distance au delà de laquelle le potentiel généré par un obstacle est nul, ce qui correspond au paramètre d_1 dans la fonction de potentiel (3.28).

Pour simplifier l'écriture, nous notons d_{infl} la distance d'influence de chacune des tâches, D^{clear} et d_1. Ainsi de nombreuses paires (corps $\mathcal{B}_i$) - (obstacle $\mathcal{O}_j$) telles que $d(\mathcal{B}_i(\mathbf{q}), \mathcal{O}_j) > d_{infl}$ ne génèrent aucune interaction et pourraient donc être éliminées des calculs réalisés par les tâches. Nous allons exploiter le principe suivant : «les obstacles très loin d'une configuration de la trajectoire discrétisée sont encore loin de la configuration suivante». Cela va nous permettre de «filtrer» les obstacles qui n'ont pas d'influence dans les calculs des interactions.

Borne inférieure de la distance à un obstacle

Nous formalisons ici le principe énoncé précédemment. Soit $\mathbf{q}_1$ et $\mathbf{q}_2$ deux configurations du robot et $\mathcal{B}_i$ le corps considéré. Soit m la transformation solide telle que :

$$\mathcal{B}_i(\mathbf{q}_2) = m(\mathcal{B}_i(\mathbf{q}_1))$$

On note Δ la borne supérieure de la distance parcourue par les points du corps $\mathcal{B}_i$ entre les configurations $\mathbf{q}_1$ et $\mathbf{q}_2$:

$$\Delta = \max_{r \in \mathcal{B}_i(\mathbf{q}_1)} \|m(r) - r\|$$

Alors on a la propriété suivante :

Propriété 6. *Pour tout sous ensemble $\mathcal{O} \subset \mathcal{W}$ et pour tout $d > 0$,*
si :

$$d(\mathcal{B}_i(\mathbf{q}_1), \mathcal{O}) \geq d$$

alors :

$$d(\mathcal{B}_i(\mathbf{q}_2), \mathcal{O}) \geq max(d - \Delta, 0)$$

Démonstration. Si $d \leq \Delta$, alors $max(d - \Delta, 0) = 0$ et la conclusion est immédiate. Si $d \geq \Delta$ on remarque que par hypothèse on a pour tout $o \in \mathcal{O}$ et pour tout $r_1 \in \mathcal{B}_i(\mathbf{q}_1)$:

$$\|r_1 - o\| \geq d$$

Et pour tout $r_2 \in \mathcal{B}_i(\mathbf{q}_2)$ on peut écrire :

$$\begin{aligned} \|r_2 - o\| &= \|r_2 - m^{-1}(r_2) + m^{-1}(r_2) - o\| \\ &\geq \big|\|r_2 - m^{-1}(r_2)\| - \|m^{-1}(r_2) - o\|\big| \end{aligned}$$

par inégalité triangulaire. De plus comme $m^{-1}(r_2) \in \mathcal{B}_i(\mathbf{q}_1)$, on a :

$$\|m^{-1}(r_2) - o\| \geq d$$

Par définition de Δ on a :

$$\|r_2 - m^{-1}(r_2)\| \leq \Delta$$

Comme $d \geq \Delta$:

$$\|r_2 - m^{-1}(r_2)\| \leq \|m^{-1}(r_2) - o\|$$

Finalement il vient :

$$\begin{aligned} \|r_2 - o\| &\geq \|m^{-1}(r_2) - o\| - \|r_2 - m^{-1}(r_2)\| \\ &\geq d - \Delta \end{aligned}$$

Cette dernière inégalité étant valable pour tout $o \in \mathcal{O}$ et pour tout $r_2 \in \mathcal{B}_i(\mathbf{q}_2)$. On obtient donc le résultat $d(\mathcal{B}_i(\mathbf{q}_2), \mathcal{O}) \geq d - \Delta$. □

Cette propriété, qui est la version pour un seul corps de la propriété (1), permet de mettre à jour de façon simple une borne inférieure de la distance d'un corps du robot à un obstacle, lors d'un mouvement entre $\mathbf{q}_1$ et $\mathbf{q}_2$. Il suffit pour cela de soustraire la borne supérieure de la distance parcourue par les points du corps, Δ.

Algorithme de filtrage

Le principe est de maintenir pour chaque corps du robot une liste triée des bornes inférieures des distances entre les obstacles et le corps. Soit l_{infl} le nombre d'éléments en tête de cette liste qui correspondent à des distances robot-obstacle inférieures à la distance d'influence d_{infl}. Il suffit donc de ne considérer dans les calculs effectués par les tâches (1) et (2) que les l_{infl} éléments en tête de liste, l_{infl} étant généralement très inférieur au nombre d'obstacles perçus n_o.

À l'initialisation, la liste est remplie par les distances exactes entre les obstacles et le corps dans la configuration initiale. Puis la trajectoire discrétisée $\mathbf{q}_s$ est parcourue itérativement sur un intervalle ($s \in \{1, ..., n_s\}$) par les tâches (1) et (2) afin de calculer les interactions du robot avec les obstacles. Lors de ce parcours de la trajectoire, pour chaque configuration $\mathbf{q}_s$ la liste est mise à jour en soustrayant la distance maximale parcourue entre $\mathbf{q}_{s-1}$ et $\mathbf{q}_s$ (notée Δ) à chacun de ses éléments. C'est à dire que les obstacles qui sont plus éloignés de $\mathbf{q}_s$ que de $\mathbf{q}_{s-1}$ sont traités comme s'ils s'étaient rapprochés. Ainsi, à chaque configuration, les l_{infl} éléments de la liste qui atteignent la valeur d_{infl} entraînent les traitements suivants :

Algorithme 1 Filtrage des interactions robot-obstacles sur un intervalle de la trajectoire

Initialisation des listes
for $i \in \{1, ..., n_b\}$ **do**
  for $j \in \{1, ..., n_o\}$ **do**
    calcul de la distance aux obstacles : $dist[j, i] \leftarrow d(\mathcal{B}_i(\mathbf{q}_s), \mathcal{O}_j)$
  end for
  TRI-RAPIDE($dist[., i]$)
end for

boucle sur l'intervalle de la trajectoire
while $(s \leq n_s)$ **do**
  for $i \in \{1, ..., n_b\}$ **do**
    $l \leftarrow 1$;
    while $(dist[l, i] \leq d_{infl})$ **do**
      $j \leftarrow$ index du l-ème obstacle dans la liste $dist[l, i]$
      CALCULER_INTERACTION($\mathcal{O}_j, \mathcal{B}_i$)
      calcul de la distance exacte à cet obstacle : $dist[l, i] \leftarrow d(\mathcal{B}_i, \mathcal{O}_j)$
      $l \leftarrow l + 1$
    end while
    $l_{infl} \leftarrow l$
    trier les l_{infl} **premiers éléments de la liste :**
    TRI-INSERTION($dist[., i]$, l_{infl})
    $\Delta[i] \leftarrow$ DIST_MAX_PARCOURUE($\mathcal{B}_i$,$\mathbf{q}_s$, $\mathbf{q}_{s+1}$)
    for $j \in \{1, ..., n_o\}$ **do**
      Soustraire la distance maximale parcourue :
      $dist[j, i] \leftarrow$ MAX$(0,\ dist[j, i] - \Delta[i])$
    end for
  end for
  $s \leftarrow s + 1$
end while

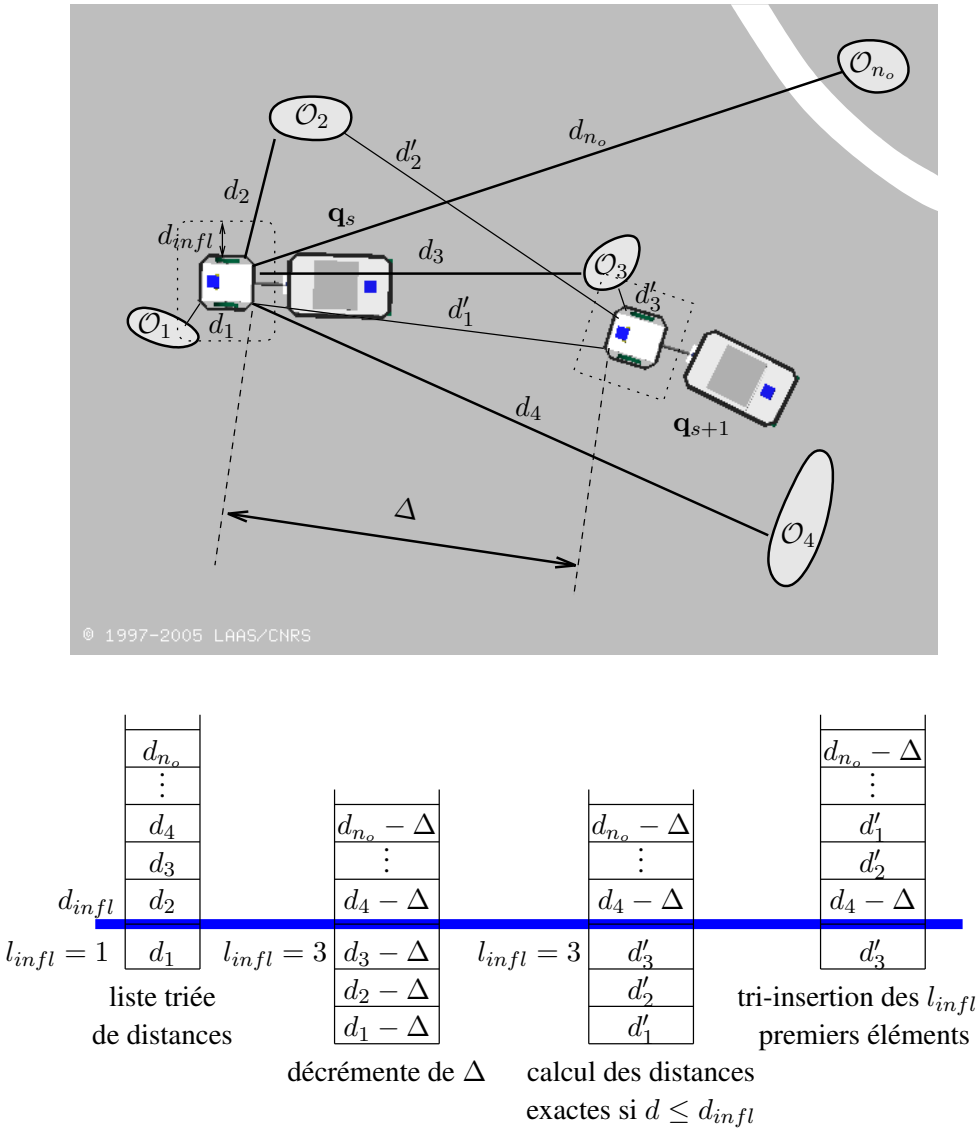

FIG. 3.11 – Illustration de l'algorithme de filtrage des interactions entre les obstacles et le robot (seule la remorque est considérée). En bas les différentes étapes de mise à jour de la liste des bornes inférieures des distances aux obstacles. Δ est la distance maximale parcourue par chacun des points de la remorque entre $\mathbf{q}_s$ et $\mathbf{q}_{s+1}$.

- ils correspondent à des obstacles qui pourraient avoir une influence sur le robot et ils sont donc pris en compte dans les calculs effectués par les tâches (1) et (2),
- leur distance exacte au corps considéré est recalculée et elle est insérée dans la liste déjà partiellement triée.

L'algorithme 1 décrit ces différentes étapes et la figure 3.11 illustre une itération de l'algorithme entre deux configurations $\mathbf{q}_s$ et $\mathbf{q}_{s+1}$, pour un corps du robot (la remorque du robot Hilare2).

Bilan

L'algorithme de filtrage présenté donne lieu à de nouveaux calculs :
- calcul de la distance de chacun des corps du robot aux obstacles,
- tri par l'algorithme de TRI-RAPIDE des listes de distances. La complexité est $O(n \log(n))$,
- tri par l'algorithme TRI-INSERTION des l_{infl} premiers éléments des listes déjà triées. La complexité est $O(n^2)$ mais cet algorithme est très efficace sur les listes déjà partiellement triées.

Il permet cependant de diminuer de façon importante les temps de calcul, car le calcul effectif des interactions n'a lieu que pour un très petit nombre d'éléments.

Ainsi avant l'implémentation de cet algorithme de filtrage, les calculs effectués par les tâches (1) et (2) constituaient le «noeud d'engorgement» de l'algorithme de déformation de trajectoire présenté à la section 3.2.7. Son utilisation a permis de diminuer les temps de calcul de ces tâches d'un facteur 6 à 8.

Il se révèle difficile d'évaluer sa complexité, mais les expériences menées sur plusieurs robots (le robot Hilare2 avec sa remorque et le robot Cycab) dans des environnements variés démontrent son efficacité.

3.5 Optimisation de la trajectoire suivant d'autres critères

La méthode de déformation de trajectoire que nous avons présentée a été développée pour l'évitement d'obstacles pour des systèmes nonholonomes. Nous montrons ici que cette méthode peut en fait agir sur la trajectoire en fonction d'autres critères et en vue d'autres applications. Les moyens de contrôle de la trajectoire sont au nombre de trois.

Potentiel d'une trajectoire Le potentiel d'une trajectoire définie sur l'intervalle $[0, S]$ est l'intégrale du potentiel d'une configuration $U(\mathbf{q})$ sur cet intervalle. L'algorithme présenté en (3.2.7) permet de faire baisser ce potentiel et donc d'optimiser la trajectoire suivant le critère exprimé par le potentiel. Jusqu'à présent ce critère était fondé sur la distance aux obstacles, mais d'autres critères pour d'autres applications sont envisageables. Avec un peu de recul, on peut donc considérer cette méthode comme une méthode d'optimisation de trajectoire suivant un critère quelconque. Il a par exemple été essayé dans [Boyer 2004] de contrôler la vitesse le long d'une trajectoire en exprimant un potentiel sur la vitesse, c'est à dire sur $\mathbf{q}'(s)$.

Nous présentons à la section 3.5.1 l'utilisation d'un potentiel sur l'angle de braquage d'un système de type voiture, publiée dans [Lefebvre 2004].

Respect des contraintes nonholonomes Certaines approximations dans la méthode de déformation de trajectoire font que le processus de déformation fournit en sortie une trajectoire qui n'est pas strictement admissible pour le système (2.7). L'idée de l'étape de correction de la déviation des contraintes nonholonomes (présentée à la section 3.2.6) est de modifier le processus de déformation afin de corriger ce phénomène indésirable. Un mécanisme est donc mis en oeuvre pour ramener les entrées suivant les champs de vecteurs additionnels du système vers 0.

On comprend donc que si la trajectoire initiale n'est pas strictement admissible, les itérations successives du processus de déformation vont tendre vers une trajectoire admissible. Nous présentons ce phénomène et ses potentielles applications à la section 3.5.2.

Respect des conditions aux limites L'étape qui s'assure que la configuration finale de l'intervalle de déformation reste inchangée au cours du processus de déformation peut-être détournée de son objectif initial. Ainsi on peut souhaiter spécifier une configuration finale différente, et utiliser la méthode pour déformer la trajectoire de façon à ce qu'elle rejoigne une configuration spécifiée. Nous avons utilisé cette technique dans le cadre d'une méthode de parking référencé sur des amers, qui est présentée au chapitre suivant.

3.5.1 Respect d'une contrainte sur la courbure durant le processus de déformation de trajectoire

Le robot Cycab présenté à la section 3.3 est soumis à des contraintes physiques sur son angle de braquage ϕ :$-\phi_{max} \leq \phi \leq \phi_{max}$.

La méthode de déformation de trajectoire pour systèmes nonholonomes assure que la trajectoire déformée respecte les contraintes cinématiques du système, mais elle ne garantit rien quant à la courbure de la trajectoire. Ainsi il se peut qu'à partir d'une trajectoire admissible mais en collision, la déformation produise un chemin sans collision mais avec une courbure non-admissible pour le système.

Pour contrer cet effet on considère l'angle ϕ_{max} comme un obstacle pour la variable ϕ, et on définit une fonction de potentiel qui est fonction de la distance à cet obstacle, de façon similaire à la fonction de potentiel de l'équation (3.28). En pratique seul le gradient de cette fonction est calculé, et on définit donc $U'_\phi(\phi)$:

$$\begin{array}{ll} 0 & \text{si} \quad 0 \leq |\phi| \leq (\phi_{max} - d_1) \\ \frac{1}{((\phi_{max}-|\phi|)+d_0)^2} - \frac{1}{(d_1+d_0)^2} & \text{si} \quad (\phi_{max} - d_1) \leq |\phi| \leq \phi_{max} \\ \frac{1}{(d_0)^2} - \frac{1}{(d_1+d_0)^2} & \text{si} \quad |\phi| \geq \phi_{max} \end{array}$$

Le paramètre d_0 permet de régler la valeur maximale $U_\phi^{'max}$ prise par la fonction U'_ϕ en $\phi = \phi_{max}$, et le paramètre d_1 définit l'intervalle sur lequel cette fonction est nulle, comme illustré sur la figure 3.12.

Application à un robot de type rover

Nous avons appliqué ce principe au robot Dala, un rover représenté sur la figure 3.13. Son modèle cinématique est celui d'un unicycle, mais nous ajoutons une roue directrice virtuelle située à un mètre à l'avant du centre cinématique, afin de contrôler la courbure des trajectoires du système (figure 3.14). Le système est ainsi équivalent à un robot de type voiture, dont le modèle cinématique est donné par l'équation (3.26). Le contrôle de la courbure lors du processus de déformation permet en pratique de limiter les glissements et les mouvements peu naturels du robot.

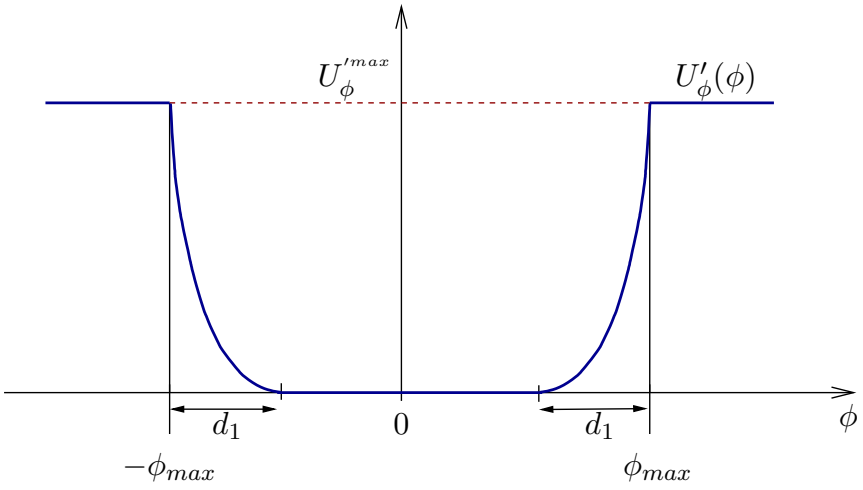

FIG. 3.12 – Gradient du potentiel sur l'angle de braquage ϕ

FIG. 3.13 – Le robot Dala, un robot de type unicycle

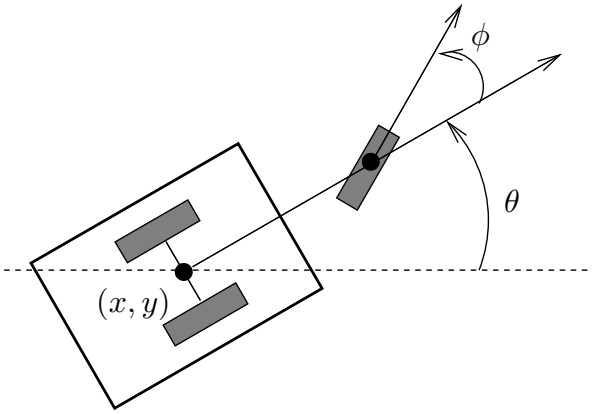

FIG. 3.14 – Modèle cinématique d'un unicycle, avec une roue avant virtuelle située à un mètre du centre cinématique

La figure 3.15 représente les déformations successives d'une trajectoire. La valeur de ϕ_{max} est de 1 radian, et la trajectoire initiale ne respecte pas cette contrainte. La méthode de déformation de trajectoire va générer des perturbations des entrées qui font baisser le potentiel sur l'angle de braquage U_ϕ. On observe que les déformations successives entraînent naturellement des manoeuvres, qui diminuent la courbure maximale de la trajectoire.

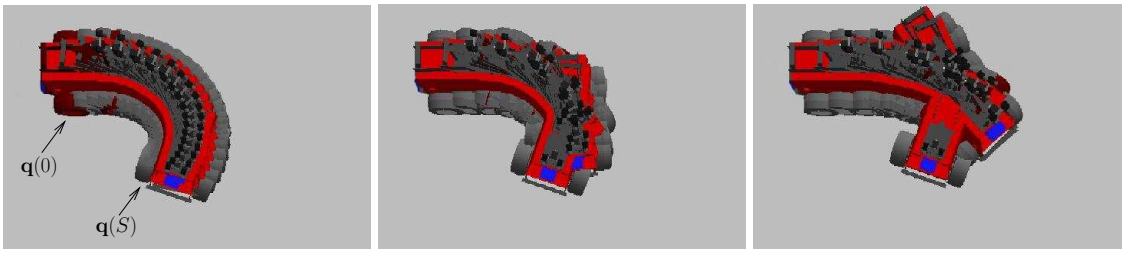

FIG. 3.15 – Déformation d'une trajectoire initiale sans collision, mais avec une courbure trop élevée

3.5.2 Déformation d'une trajectoire non-admissible vers une trajectoire admissible

Comme nous l'avons mentionné en introduction de cette section, la méthode de déformation de trajectoire implémente un mécanisme qui assure que la trajectoire déformée respecte les contraintes cinématiques du système, dont le détail est donné à la section 3.2.6. Nous en rappelons brièvement le principe.

Étant donné le système dynamique de l'équation (2.7), dont les vitesses sont une combinaison linéaire de k champs de vecteurs $(\mathbf{X}_1, \ldots, \mathbf{X}_k)$, on considère le système étendu en ajoutant $n-k$ champs de vecteurs indépendants :

$$\mathbf{q}'(s) = \sum_{i=1}^{n} u_i(s)\mathbf{X}_i(\mathbf{q}(s)) \tag{3.33}$$

tel que $\forall \mathbf{q} \in \mathcal{C}, (\mathbf{X}_1(\mathbf{q}), \ldots, \mathbf{X}_n(\mathbf{q}))$ engendre $T_\mathbf{q}\mathcal{C}$.

Une trajectoire du système (3.33) est admissible pour le système (2.7) si et seulement si :

$$\forall j \in \{k+1, .., n\}, \forall s \in [0, S] \quad u_j(s) = 0$$

Mais les approximations effectuées au cours du processus de déformation font que les entrées suivant les champs de vecteurs additionnels ne sont pas exactement nulles, et que la trajectoire déformée n'est pas exactement admissible ; nous avions alors appelé ce phénomène la «déviation des contraintes nonholonomes».

Le principe de la correction de la déviation des contraintes nonholonomes est de réguler les commandes $u_j, j \in \{k+1, \ldots, n\}$ suivant ces champs de vecteurs afin de les faire tendre vers 0, par le biais des fonctions de perturbations v_j.

Dans l'algorithme de la méthode de déformation de trajectoire (section 3.2.7), on calcule une direction de déformation $\eta^{\perp}$ engendrée par ces fonctions de perturbations, à laquelle s'ajoute éventuellement une direction de déformation η due aux obstacles. La trajectoire déformée est calculée en ajoutant une fraction de cette direction de déformation $\tilde{\eta} = \eta + \eta^{\perp}$ à la trajectoire initiale :

$$\forall s \in [0, S], \quad \mathbf{q}(s, \tau + \Delta\tau) = \mathbf{q}(s, \tau) + \Delta\tau\tilde{\eta}(s, \tau)$$

Ainsi, en donnant en entrée du processus de déformation une trajectoire qui n'est pas admissible, les itérations successives du processus vont tendre vers une trajectoire admissible. Nous en présentons un exemple à la section suivante.

Exemple de déformation d'une trajectoire non-admissible

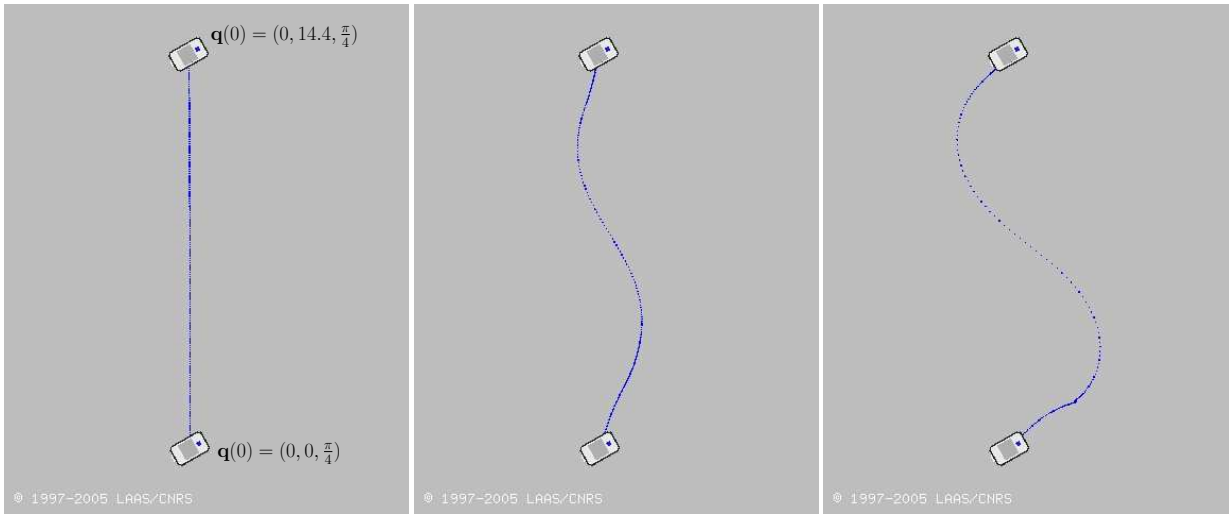

FIG. 3.16 – Déformations successives d'une trajectoire non-admissible

Soit un système de type unicycle et $\mathbf{q} = (x, y, \theta)$ une configuration de ce système. Les deux champs de vecteurs pour ce système sont par exemple :

$$\mathbf{X}_1(\mathbf{q}) = \begin{pmatrix} \cos\theta \\ \sin\theta \\ 0 \end{pmatrix} \quad \mathbf{X}_2(\mathbf{q}) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Un champ de vecteurs additionnel est par exemple $\mathbf{X}_3(\mathbf{q}) = \begin{pmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{pmatrix}$.

Alors toute trajectoire admissible pour le système sur l'intervalle $[0, 10]$ est telle que :

$$\forall s \in [0, 10], \quad \mathbf{q}'(s) = u_1(s)\mathbf{X}_1(\mathbf{q}(s)) + u_2(s)\mathbf{X}_2(\mathbf{q}(s)) + u_3(s)\mathbf{X}_3(\mathbf{q}(s))$$

où u_3 est nulle.

Dans cet exemple, la configuration initiale du système est $\mathbf{q}(0) = (0, 0, \pi/4)$. Les fonctions d'entrée de la trajectoire initiale sont :

$$u_1 = 1 \quad u_2 = 0 \quad u_3 = 1$$

Cette trajectoire, pour laquelle u_3 n'est pas nulle, n'est donc pas admissible pour l'unicycle. Mais les déformations successives vont ramener cette entrée vers 0 et générer des entrées suivant les champs de vecteurs $\mathbf{X}_1$ et $\mathbf{X}_2$.

Dans cet exemple on suppose qu'il n'y a pas d'obstacles, et on fixe $\Delta\tau$ à 0.2. La figure 3.16 illustre les déformations successives de la trajectoire initiale, après respectivement 0, 3 et 15 itérations. La figure 3.17 présente les fonctions d'entrée de ces différentes trajectoires. On observe la régulation vers 0 de l'entrée u_3 suivant le champ de vecteurs additionnel $\mathbf{X}_3$ au fur et à mesure des itérations. C'est à dire que la trajectoire après 15 itérations de déformation est admissible pour le système d'équation (2.7) de l'unicycle.

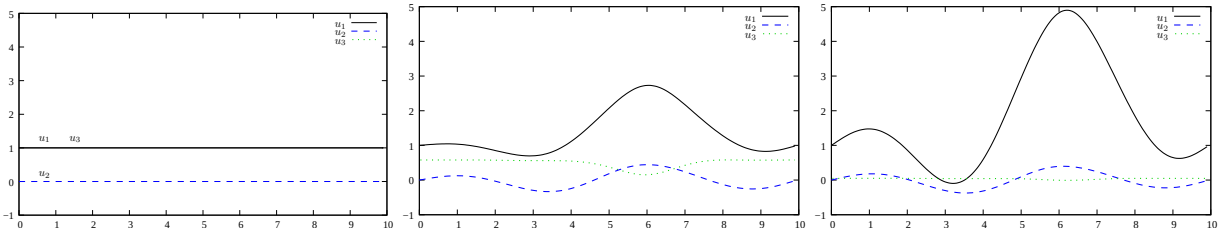

FIG. 3.17 – Fonctions d'entrée de la trajectoire, respectivement aux itérations 0, 3 et 15

Singularités du système de la déformation Il faut noter que ce phénomène de déformation d'une trajectoire non-admissible est sujet à de nombreuses singularités qui apparaissent dans l'expression du système de l'équation (3.3). Si les matrices A ou B sont singulières, ce qui dépend de la trajectoire initiale et des champs de vecteurs du système, il n'existera pas de direction de déformation admissible qui diminue les entrées selon les champs de vecteurs non-admissibles.

Dans cet exemple, on a choisi de fixer : $\mathbf{q}(0) \neq \mathbf{0}$ et $u_1 \neq 0$, pour éviter ce type de singularité.

Vers une méthode de planification de trajectoire pour systèmes nonholonomes ?

On pourrait envisager de coupler ce mécanisme de déformation d'une trajectoire non-admissible avec la méthode permettant de contrôler la configuration finale de la trajectoire lors du processus de déformation, qui est présentée plus en détail au chapitre suivant.

Ces deux «contrôles» de la trajectoire laissent envisager le développement d'une méthode numérique de planification de trajectoire pour systèmes nonholonomes. Cela pourrait être particulièrement intéressant pour des systèmes pour lesquels on ne possède pas de méthode analytique (les systèmes qui ne sont pas «différentiellement plats», par exemple), mais dont on possède une expression des champs de vecteurs.

Nous n'avons pas exploré toutes les potentialités de cette méthode de déformation en tant que méthode numérique de planification de trajectoire. Nous faisons cependant remarquer que

cette idée est assez proche de la méthode proposée par [Divelbiss 1997]. Les auteurs écrivent le problème de la recherche d'une commande amenant le robot à une configuration finale sous la forme d'un système d'équations non-linéaires. Ce système est résolu en utilisant l'algorithme de Newton-Raphson, pour trouver les coordonnées de la commande dans l'espace des séries de Fourier. Les obstacles sont pris en compte, comme des contraintes sur l'espace de recherche. Leur contribution réside donc dans une méthode numérique de planification de trajectoire pour systèmes nonholonomes, à laquelle notre méthode pourrait se comparer.

Enfin, il serait intéressant de vérifier que cette méthode numérique de planification de trajectoire que nous suggérons respecte la propriété de commandabilité en temps petit :

Propriété 7. *Soit* $\mathbf{q} \in \mathcal{C}$ *une configuration et* $V(\mathbf{q})$ *un voisinage de cette configuration. Une méthode de guidage rend compte de la commandabilité en temps petit si l'ensemble des configurations accessibles à partir de* $\mathbf{q}$ *et sans sortir de* $V(\mathbf{q})$ *est lui même un voisinage de* $\mathbf{q}$.

Cette propriété est nécessaire en pratique pour qu'une méthode de guidage puisse être utilisée dans un planificateur de trajectoire global qui tient compte des obstacles. Si par exemple on cherche à connecter deux configurations proches dans une zone fortement contrainte, il faut que le chemin qui les relie reste dans un voisinage de ces configurations.

3.6 Conclusion

Ce chapitre nous a permis d'évaluer dans quelle mesure les spécificités de notre approche (spécificités du système et spécificités des applications telles que présentées au chapitre 1), ne sont pas sans conséquences sur la réalisation d'une fonctionnalité particulière de la navigation autonome, à savoir l'évitement réactif d'obstacles. Ainsi la présentation des techniques existantes et de leur inadéquation à nos spécificités a motivé le développement d'une méthode originale de déformation de trajectoire pour systèmes nonholonomes.

Nous avons ensuite présenté des techniques qui optimisent les calculs des interactions robot-obstacles nécessaires à cette méthode de déformation de trajectoire. Un des inconvénients de cette méthode réside en effet dans son coût calculatoire. L'autre inconvénient est inhérent à l'utilisation des champs de potentiel basés sur la distance aux obstacles, et à leurs minima locaux. Ces différents aspects sont discutés dans le chapitre 6, en relation avec les résultats expérimentaux.

Enfin, nous avons montré que cette méthode générique, vue comme une méthode d'optimisation de trajectoire, peut être employée à d'autres fins que l'évitement d'obstacles.

Chapitre 4

Parking référencé sur des amers

4.1 Introduction

Dans le cadre de notre étude sur la navigation autonome de véhicules articulés à roues, nous appelons parking la phase finale du mouvement, quand celle-ci à lieu à proximité d'obstacles.

Au cours de l'exécution d'un mouvement, la phase de parking revêt un intérêt particulier car elle constitue généralement le but de la tâche de navigation. Il s'agit en effet souvent pour le robot de rejoindre une configuration finale spécifiée. La précision souhaitée varie d'une application à l'autre, mais on peut citer quelques exemples qui requièrent une très grande précision dans la phase de parking :

- le créneau parallèle pour une voiture,
- l'amarrage d'un véhicule à remorque à un quai de débarquement (figure 4.1 à gauche),
- le positionnement d'un instrument de mesure porté par un robot mobile (figure 4.1 à droite).

Dans ces différents exemples, la configuration de parking n'est pas définie comme une configuration à atteindre dans un modèle de l'environnement, mais comme une configuration définie relativement à des éléments de l'environnement. Ainsi un créneau consiste à se garer **entre 2 voitures**. Pour l'amarrage d'un camion il s'agit de se garer **entre 2 camions** et **contre le quai**. Enfin un robot mobile doté d'un instrument de mesure doit se positionner **relativement à l'objet mesuré**.

Au contraire, si la configuration de parking est définie de façon absolue dans un modèle de l'environnement, on devra assurer l'exactitude de ce modèle et de la localisation du robot pour garantir un parking précis.

Pour réaliser une tâche de parking dans laquelle la configuration de parking est définie relativement à des éléments de l'environnement dénommés «amers»[1] on doit se doter d'un capteur qui identifie et fournit des informations sur la position de ces éléments. Nous employons l'expression de *«parking référencé sur des amers»* ou plus généralement de *«mouvement référencé sur des amers»* pour désigner cette approche (parfois appelée *«mouvement référencé capteur»*).

[1] Terme de navigation maritime désignant un repère fixe et identifiable sans ambiguïté, utilisé pour la localisation par triangulation.

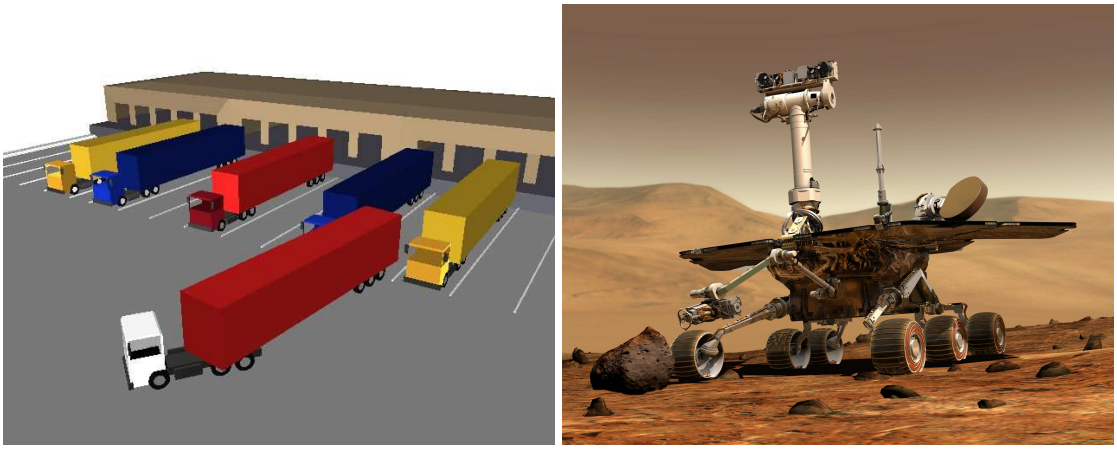

FIG. 4.1 – À gauche l'amarrage d'un camion à un quai de débarquement de marchandises. À droite le positionnement d'un instrument de mesure par le robot Spirit (image NASA/JPL-Caltech).

Alors l'intérêt de définir une configuration de parking relativement à des éléments de l'environnement est double. Tout d'abord cela permet de traiter les cas réalistes d'un modèle de l'environnement imprécis et d'une localisation inexacte. Ensuite cela permet une adaptation de la tâche de parking à la variabilité du monde réel. En résumé le parking référencé sur des amers permet de casser la corrélation entre précision du modèle de l'environnement et précision de la tâche de parking.

La difficulté d'une tâche de parking référencé sur des amers dépend de plusieurs éléments tels que l'encombrement de l'environnement, la précision souhaitée, la qualité du modèle de l'environnement dans lequel on planifie le mouvement, ou encore la commandabilité du système. Ainsi garer une semi-remorque avec une marge de quelques centimètres est une tâche plus difficile que garer un robot rond qui ne serait soumis à aucune contrainte cinématique dans un environnement dégagé.

Notre contribution porte sur le parking de systèmes articulés dans des environnements fortement encombrés. Dans la section 4.2 nous présentons différentes approches traitant de ce problème. Ensuite nous introduisons notre approche de tâche de parking référencé sur des amers dans la section 4.3. Les sections 4.4 et 4.5 présentent la résolution d'une tâche de parking. Enfin la section 4.6 développe au travers d'un exemple une tâche de parking référencé sur des amers pour un robot avec remorque.

4.2 État de l'art

L'idée de définir une configuration relativement à des éléments de l'environnement et de générer une trajectoire permettant de rejoindre cette configuration est la base de la «commande référencée capteur». Nous introduisons tout d'abord cette approche dans le cadre de l'asservissement visuel, qui a été utilisé pour résoudre le problème du parking référencé sur des amers

pour des véhicules nonholonomes. Nous présentons ensuite d'autres approches qui permettent de résoudre ce problème.

4.2.1 Asservissement visuel

Étant donné un robot dont la configuration est notée $\mathbf{q}$ et qui prend en entrée les commandes $\mathbf{u}$, on souhaite générer une trajectoire qui amène ce robot à la configuration $\mathbf{q}^{park}$. On dote le robot d'un capteur qui fournit un vecteur d'observation $\mathbf{o}$ et on se donne alors une observation désirée $\mathbf{o}^{park}$ qui définit la configuration finale désirée $\mathbf{q}^{park}$. On peut séparer les techniques d'asservissement visuel en deux classes : *l'asservissement visuel basé position* dans lequel la configuration $\mathbf{q}^{park}$ est exprimée relativement à $\mathbf{q}$ dans l'espace cartésien, et *l'asservissement visuel basé image* qui consiste à exprimer la variation de l'observation comme une fonction des commandes : $\dot{\mathbf{o}} = f(\mathbf{q}, \mathbf{u})$. Les commandes du système qui amènent le robot en $\mathbf{q}^{park}$ sont alors, lorsque J_f est inversible, $\mathbf{u} = kJ_f^{-1}(\mathbf{o}^{park} - \mathbf{o})$, où J_f est la jacobienne de f évaluée en $\mathbf{q}$, appelée *matrice d'interaction*, et k le gain de la loi de commande.

Les travaux de [Espiau 1992] formalisent cette approche, développée initialement pour la commande de bras manipulateurs. Ils ont été étendus aux systèmes nonholonomes par [Tsakiris 1997] et appliqués à un robot unicycle, en introduisant un degré de liberté supplémentaire par une rotation de la caméra.

Dans [Wei 2003] et [Wei 2005], les auteurs proposent une méthode d'asservissement visuel d'un unicycle avec une caméra panoramique et sans information de profondeur. Ces travaux sont basés sur [Hamel 2002] et ils supposent que le système est capable de tourner sur lui même. Leur extension à des véhicules articulés n'est donc pas directement envisageable.

Dans [Usher 2003], l'auteur présente l'asservissement visuel avec une caméra panoramique d'un véhicule de type voiture. Il repose sur l'enchaînement de deux lois de commandes, l'une amenant le véhicule sur une droite orientée passant par la position finale et l'autre guidant le robot le long de cette droite. Cette technique ne peut pas être généralisée à des systèmes plus complexes tels que des véhicules à remorque.

4.2.2 Planifications successives

Mobile Camera Space Manipulation

Une autre approche de parking référencé capteur repose aussi sur la perception visuelle : MCMS (*Mobile Camera Space Manipulation*) [Seelinger 2003]. Comme dans l'asservissement visuel, la configuration finale est spécifiée par une observation désirée. En fonction de l'observation courante, la configuration finale est calculée relativement à la configuration courante du robot. Une trajectoire est planifiée vers cette configuration et exécutée par la suite. Quand une certaine portion de la trajectoire a été exécutée, le robot s'arrête et répète l'étape précédente. On réitère le processus jusqu'à atteindre une précision souhaitée.

L'inconvénient de cette approche est qu'elle oblige le robot à s'arrêter et qu'elle fait de nombreuses fois appel à un planificateur de trajectoire.

Localization-Planning-Execution Cycles

L'approche décrite dans [Paromtchik 1996] et [Laugier 1999] repose sur l'exécution successive de cycles de localisation, planification et exécution de trajectoire. Cette méthode a été appliquée au parking parallèle d'un robot de type voiture. La place de parking est identifiée par des capteurs et les commandes du système sont des fonctions sinusoïdales paramétrées par les caractéristiques de la place (largeur, profondeur, etc.). Puis le mouvement est exécuté et ce cycle est répété jusqu'à atteindre une précision désirée.

Cette approche est spécifique à certaines «géométries» de place de parking (les articles ne traitent que du cas du parking parallèle) et n'est pas aisément généralisable à des véhicules soumis à des contraintes cinématiques plus complexes, comme les véhicules à remorque.

4.3 Tâche de parking référencé sur des amers

Nous proposons de définir une tâche de parking qui s'applique à tout système nonholonome dont les trajectoires sont solutions de (2.7), sans condition sur la géométrie de la place de parking, et qui s'exécute sans que le robot ne doive s'arrêter. Comme dans le chapitre 3 on suppose que l'on possède une trajectoire initialement planifiée. Ces résultats ont été publiés dans [Lefebvre 2006].

D'une manière similaire à l'approche par asservissement visuel, nous définissons un **motif de parking** noté $\mathbf{M}$ associé à un capteur. Ce motif de parking provient d'un modèle de l'environnement et représente la perception du capteur lorsque le robot est en configuration de parking. Alors à partir des observations $\mathbf{P}$ de ce capteur, le but est de repérer ce motif dans l'environnement réel puis de déformer la trajectoire initiale de façon à ce qu'elle se termine en une configuration notée $\mathbf{q}^{park}$. La configuration de parking $\mathbf{q}^{park}$ est telle que depuis cette configuration l'observation $\mathbf{P}$ est égale au motif de parking $\mathbf{M}$.

4.3.1 Définitions

Position du capteur

Étant donné une configuration $\mathbf{q}$ du robot, on note $\mathbf{c}(\mathbf{q}) \in SE(2)$ la position (position et orientation) du capteur dans le plan lorsque le robot est dans cette configuration.

La position $\mathbf{c}(\mathbf{q})$ définit un repère dans lequel on peut exprimer une position. Ainsi soit $\mathbf{r} \in SE(2)$ un repère, la position de $\mathbf{r}$ par rapport à $\mathbf{c}(\mathbf{q})$ est notée $\mathbf{r}/\mathbf{c}(\mathbf{q})$.

Amer

Un amer est un sous-espace de $\mathcal{W}$ repérable par le capteur, et dont la position est définie par un ensemble de k paramètres. On note $\mathbf{l} \in \mathbb{R}^k$ le vecteur constitué des paramètres de l'amer. On suppose que l'environnement contient l amers repérables par le capteur, que l'on identifie par un indice : $\forall i \in \{1, .., l\}, \mathbf{l}^i$ est un amer.

Par exemple, un amer peut être un point de l'espace de travail, une droite, la position d'un objet quelconque, etc. Pour un point, le nombre de paramètres nécessaires pour le repérer dans le plan est 2, et l'on a $\mathbf{l} \in \mathbb{R}^2$.

Perception d'un amer

Étant donné une configuration $\mathbf{q}$ du robot et un amer $\mathbf{l}$ visible, la perception de cet amer par le capteur est notée $\mathbf{p}_{/\mathbf{c}(\mathbf{q})}$. Ce sont les paramètres définissant la position de l'amer $\mathbf{l}$ dans le repère $\mathbf{c}(\mathbf{q})$, et $\mathbf{p}_{/\mathbf{c}(\mathbf{q})} \in \mathbb{R}^k$.

Pour chaque type d'amer, on peut définir une fonction f qui donne les paramètres définissant la position d'un amer $\mathbf{l}$ dans le plan, à partir des paramètres z donnant la position de cet amer relativement à la position $\mathbf{c}$ du capteur :

$$\begin{array}{rccc} f: & \mathbb{R}^k \times SE(2) & \rightarrow & \mathbb{R}^k \\ & (z, \mathbf{c}) & \mapsto & \mathbf{l} = f(z, \mathbf{c}) \end{array} \tag{4.1}$$

On note $\mathbf{p}$ les paramètres de l'amer $\mathbf{l}$ calculés à partir de la perception $\mathbf{p}_{/\mathbf{c}}$ de cet amer réalisée depuis la position $\mathbf{c}$ du capteur :

$$\mathbf{p} = f(\mathbf{p}_{/\mathbf{c}}, \mathbf{c}) \tag{4.2}$$

Alors en l'absence d'erreur sur $\mathbf{p}_{/\mathbf{c}}$ et sur $\mathbf{c}$ on a :

$$\mathbf{l} = \mathbf{p} \tag{4.3}$$

On note $\mathbf{P}$ le vecteur constitué des paramètres définissant les positions dans le plan des p amers perçus :

$$\mathbf{P} = (\mathbf{p}^1, .., \mathbf{p}^p)^T$$

où $p \leq l$.

Position de parking du capteur

On définit la position de parking du capteur comme la position du capteur lorsque le robot est en configuration de parking.

$$\mathbf{c}^{park} = \mathbf{c}(\mathbf{q}^{park})$$

Motif de parking

Un motif de parking pour un capteur est un vecteur $\mathbf{M} = (\mathbf{m}^1, .., \mathbf{m}^m)^T$ constitué de m amers dont les paramètres sont exprimés relativement au capteur en position de parking $\mathbf{c}^{park}$. Soit $\mathbf{m} \in \mathbb{R}^k$ un de ces amers et $\mathbf{l}$ les paramètres définissant la position de cet amer dans le plan. En utilisant l'équation (4.2) on trouve :

$$\mathbf{l} = f(\mathbf{m}, \mathbf{c}^{park}) \tag{4.4}$$

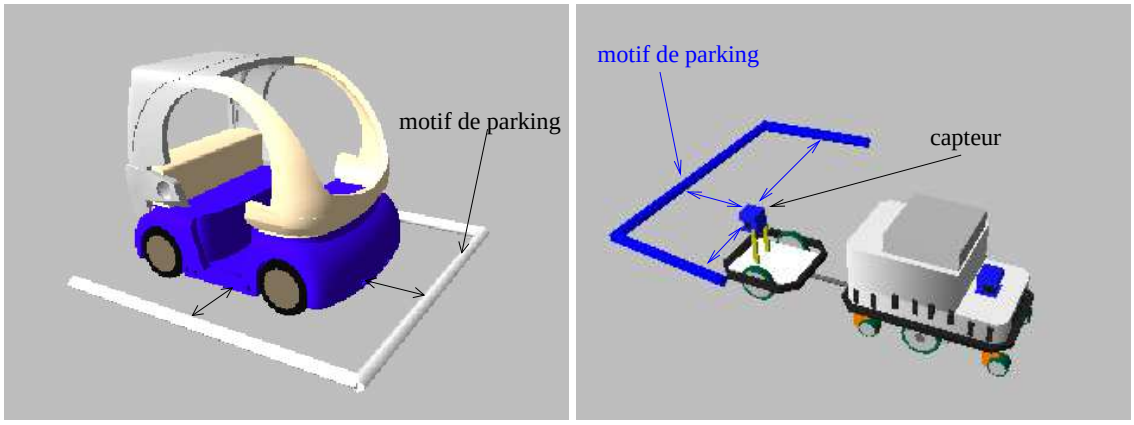

FIG. 4.2 – *Motifs de parking.* Un motif de parking est un vecteur $\mathbf{M}$ d'amers définis relativement au capteur qui fournit une observation de ces amers. Ici les amers sont des segments.

Soit maintenant une perception $\mathbf{p}$ du capteur telle que $\mathbf{p} = \mathbf{l}$. Alors on a $\mathbf{p}_{/\mathbf{c}^{park}} = \mathbf{m}$, c'est à dire que les paramètres d'un amer du motif de parking sont égaux aux paramètres de la perception de cet amer par le capteur depuis la position de parking $\mathbf{c}^{park}$.

La figure 4.2 présente des exemples de motifs de parking constitués de segments pour deux robots différents. En pratique un motif de parking $\mathbf{M}$ est calculé en positionnant le capteur dans la configuration désirée par rapport aux amers, puis en relevant ou en simulant la perception fournie par le capteur.

4.3.2 Formulation du problème

À partir de ces définitions on spécifie une tâche de parking référencé sur des amers comme un processus itératif, qui consiste successivement à :

- repérer le motif de parking $\mathbf{M}$ dans l'environnement à partir d'un vecteur d'observation noté $\mathbf{P}$,
- calculer la configuration de parking $\mathbf{q}^{park}$,
- déformer la trajectoire courante de façon à ce que la configuration finale se rapproche de $\mathbf{q}^{park}$.

Les deux premiers points sont traités dans la section 4.4 et le dernier point est présenté dans la section 4.5. La figure 4.3 illustre les différentes étapes de ce processus. Une trajectoire est planifiée dans une carte de l'environnement, et un motif de parking est défini relativement à un capteur situé sur la remorque du robot (voir la figure 4.2). Puis la trajectoire est itérativement déformée de façon à ce que la configuration finale rejoigne $\mathbf{q}^{park}$.

Une tâche de parking référencé sur des amers prend donc en entrée un motif de parking $\mathbf{M}$ et une estimation de l'incertitude sur $\mathbf{q}^{park}$ notée $\mathbf{V}_{\mathbf{q}^{park}}$. À l'initialisation, la configuration de parking est égale à la configuration finale de la trajectoire planifiée $\mathbf{q}(S)$. La configuration de parking est donc recherchée dans la zone définie par $\mathbf{V}_{\mathbf{q}^{park}}$ et centrée en $\mathbf{q}(S)$.

Ensuite à chaque itération du processus, en fonction d'un vecteur d'observation $\mathbf{P}_{/\mathbf{c}(\mathbf{q})}$, de la configuration courante $\mathbf{q}$ du robot et de la trajectoire planifiée $\mathbf{q}(s), s \in [0, S]$, la tâche de

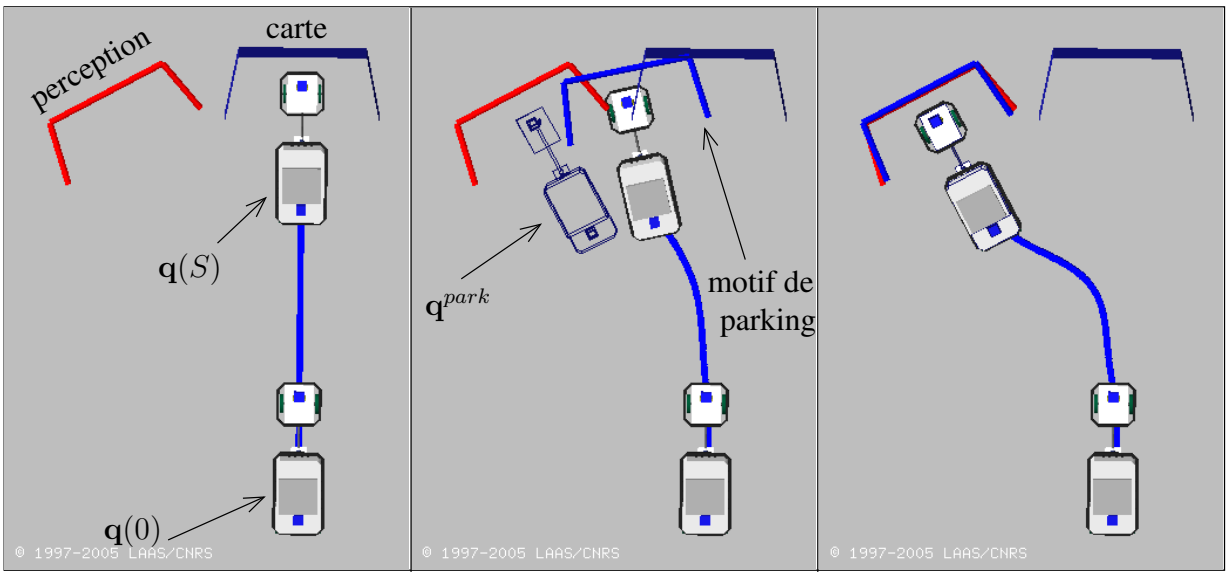

FIG. 4.3 – Tâche de parking référencé sur des amers. À chaque itération du processus, la trajectoire est déformée de façon à ce que la configuration finale se rapproche de $\mathbf{q}^{park}$.

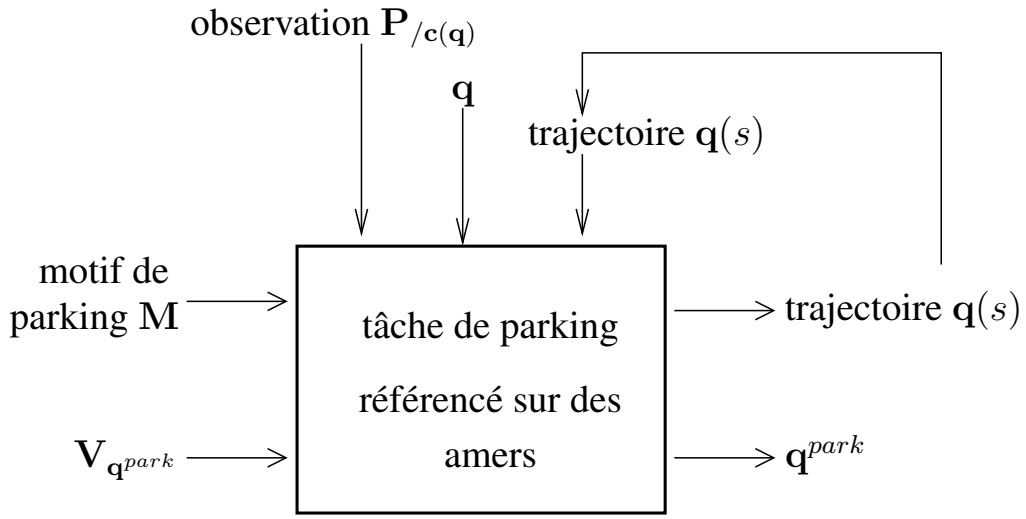

FIG. 4.4 – Diagramme d'une tâche de parking référencé sur des amers

parking produit en sortie une nouvelle valeur de $\mathbf{q}^{park}$ et une nouvelle trajectoire admissible telle que celle-ci se rapproche de $\mathbf{q}^{park}$. La figure 4.4 illustre les entrées et sorties d'une tâche de parking référencé sur des amers.

4.4 Calcul de la configuration de parking

En l'absence d'information sur le motif de parking, $\mathbf{q}^{park}$ est définie comme la configuration finale de la trajectoire planifiée : $\mathbf{q}^{park} = \mathbf{q}(S)$. Ensuite la perception $\mathbf{P}$ des amers correspondant au motif de parking fournit des informations pour mettre à jour $\mathbf{q}^{park}$.

4.4.1 Modélisation probabiliste

En pratique on doit tenir compte de plusieurs sources d'erreur :
- la position du capteur $\mathbf{c}(\mathbf{q})$ n'est pas connue exactement,
- le capteur ne fournit pas une observation exacte de la position relative $\mathbf{p}^i_{/\mathbf{c}}$ d'un amer,
- les amers $\mathbf{m}$ du motif de parking ne sont pas définis avec exactitude.

Pour tenir compte de ces imprécisions, on modélise les variables précédentes par des variables aléatoire réelles. On suppose que les bruits de mesure sont distribués suivant une loi normale de moyenne nulle. On note alors pour tout vecteur réel Y de taille n :

$$\begin{array}{rll} \text{la vraie valeur} & y & \\ \text{la valeur mesurée} & \hat{y} = & y + \epsilon_y \\ \text{le bruit de mesure} & \epsilon_y \sim & \mathcal{N}(0, \mathbf{V}_y) \\ \text{la matrice de covariance} & \mathbf{V}_y = & E(\epsilon_y \epsilon_y^T) \end{array}$$

Soit alors Z un vecteur de variables aléatoires réelles de taille p et g une fonction de $\mathbb{R}^n$ dans $\mathbb{R}^p$ telle que : $Z = g(Y)$. On a $z = g(y)$ et on approxime $\hat{z}$ par :

$$\hat{z} = g(\hat{y})$$

En linéarisant la fonction g autour de $\hat{y}$ on trouve la variance de Z notée $\mathbf{V}_z$:

$$\mathbf{V}_z = J_g \mathbf{V}_y J_g^T \tag{4.5}$$

où $J_g = \frac{dg}{dy}\Big|_{\hat{y}}$ est la matrice jacobienne de taille $p \times n$ de g évaluée en $\hat{y}$.

Configuration et position

La configuration courante du robot est modélisée par $\mathbf{q} \sim \mathcal{N}(\hat{\mathbf{q}}, \mathbf{V}_\mathbf{q})$. La position courante du capteur est donc :

$$\hat{\mathbf{c}} = \mathbf{c}(\hat{\mathbf{q}}) \tag{4.6}$$

$$\mathbf{V}_\mathbf{c} = J_\mathbf{c} \mathbf{V}_\mathbf{q} J_\mathbf{c}^T \tag{4.7}$$

L'estimation initiale de $\mathbf{q}^{park}$ est $\mathbf{q}(S)$. Sa variance est une entrée du problème :

$$\begin{aligned} \hat{\mathbf{q}}^{park} &= \mathbf{q}(S) \\ \mathbf{V}_{\mathbf{q}^{park}} & \end{aligned}$$

L'estimation de $\mathbf{c}^{park}$ est alors :

$$\begin{aligned} \hat{\mathbf{c}}^{park} &= \mathbf{c}(\hat{\mathbf{q}}^{park}) \\ \mathbf{V}_{\mathbf{c}^{park}} &= J_{\mathbf{c}} \mathbf{V}_{\mathbf{q}^{park}} J_{\mathbf{c}}^T \end{aligned}$$

Perception

La perception de l'amer $\mathbf{l}^i$ depuis la configuration courante est modélisée par $\mathbf{p}^i_{/\mathbf{c}(\mathbf{q})} \sim \mathcal{N}(\hat{\mathbf{p}}^i_{/\mathbf{c}(\mathbf{q})}, \mathbf{V}_{\mathbf{p}^i_{/\mathbf{c}(\mathbf{q})}})$. Cette valeur est fournie par le capteur, et on remarque que les $\mathbf{p}^i_{/\mathbf{c}(\mathbf{q})}$ sont indépendante deux à deux. En utilisant l'équation (4.2), on trouve les paramètres donnant l'estimation de la position dans le plan de cet amer :

$$\begin{aligned} \hat{\mathbf{p}}^i &= f(\hat{\mathbf{p}}^i_{/\mathbf{c}}, \hat{\mathbf{c}}) \\ \mathbf{V}_{\mathbf{p}^i} &= J_{f/\mathbf{p}} \mathbf{V}_{\mathbf{p}^i_{/\mathbf{c}(\mathbf{q})}} J_{f/\mathbf{p}}^T + J_{f/\mathbf{c}} \mathbf{V}_{\mathbf{c}} J_{f/\mathbf{c}}^T \end{aligned} \tag{4.8}$$

où $J_{f/\mathbf{p}}$ est la jacobienne de f par rapport à $\mathbf{p}$ et $J_{f/\mathbf{c}}$ est la jacobienne de f par rapport à $\mathbf{c}$.

Motif de parking

Le motif de parking est le vecteur de variables aléatoires réelles noté $\mathbf{M} \sim \mathcal{N}(\hat{\mathbf{M}}, \mathbf{V}_{\mathbf{M}})$. C'est une entrée du problème. On suppose que les $\mathbf{m}_i$ sont indépendants deux à deux et donc que la matrice $\mathbf{V}_{\mathbf{M}}$ est bloc-diagonale.

4.4.2 Mise en correspondance

Le robot arrivant à proximité de la fin de la trajectoire $\hat{\mathbf{q}}^{park}$, le capteur fournit une perception $\hat{\mathbf{P}}$ de p amers, calculée à partir de l'équation (4.8). Il s'agit alors de déterminer quels sont les amers perçus $\mathbf{p}^i$ qui correspondent à des amers du motif de parking $\hat{\mathbf{M}}$.

Soit $\hat{\mathbf{m}}^j$ un amer du motif de parking. En supposant que l'amer correspondant existe dans l'environnement, les paramètres permettant de repérer sa position dans le plan sont notés $\hat{\mathbf{l}}^j$ et sont donnés par l'équation (4.4) :

$$\begin{aligned} \hat{\mathbf{l}}^j &= f(\hat{\mathbf{m}}^j, \hat{\mathbf{c}}^{park}) \\ \mathbf{V}_{\mathbf{l}^j} &= J_{f/\mathbf{m}} \mathbf{V}_{\mathbf{m}^j} J_{f/\mathbf{m}}^T + J_{f/\mathbf{c}} \mathbf{V}_{\mathbf{c}^{park}} J_{f/\mathbf{c}}^T \end{aligned} \tag{4.9}$$

$\hat{\mathbf{l}}^j$ est donc la prédiction de la perception de l'amer $\hat{\mathbf{m}}^j$, qui est à comparer avec les perceptions du capteur $\hat{\mathbf{p}}^i$.

On note $\mathbf{z}^{ij}$ la différence entre $\mathbf{l}^j$ et $\mathbf{p}^i$:

$$\mathbf{z}^{ij} = (\mathbf{p}^i - \mathbf{l}^j)$$

Si l'observation $\mathbf{p}^i$ correspond au motif $\mathbf{m}^j$, on a $\mathbf{z}^{ij} = 0$. Cependant ces valeurs vraies sont inaccessibles et l'on ne peut traiter que des estimations de ces valeurs. Et en raison des erreurs de mesures, on n'a jamais exactement $\hat{\mathbf{p}}^i = \hat{\mathbf{l}}^j$, et donc $\hat{\mathbf{z}}^{ij} \neq 0$. Alors en remarquant que $\mathbf{p}^i$ et $\mathbf{l}^j$ sont des variables indépendantes, on a :

$$\begin{aligned} \hat{\mathbf{z}}^{ij} &= (\hat{\mathbf{p}}^i - \hat{\mathbf{l}}^j) \\ \mathbf{V}_{\mathbf{z}^{ij}} &= \mathbf{V}_{\mathbf{p}^i} + \mathbf{V}_{\mathbf{l}^j} \end{aligned} \tag{4.10}$$

On utilise la distance de Mahalanobis pour mesurer la vraisemblance de la mise en correspondance de $\mathbf{p}^i$ avec $\mathbf{m}^j$. On note D^{ij} la valeur telle que :

$$(D^{ij})^2 = \hat{\mathbf{z}}^{ijT} \mathbf{V}_{\mathbf{z}^{ij}} \hat{\mathbf{z}}^{ij} \tag{4.11}$$

Étant donné que $\mathbf{z}$ suit une distribution de loi normale, $(D^{ij})^2$ suit une distribution du χ^2_k où k est la dimension du vecteur $\mathbf{z}$. Si l'on souhaite que la probabilité de rejeter une mise en correspondance alors que celle-ci est vraie soit inférieure à $\alpha \in [0, 1]$, on va fixer le seuil λ_α tel que $p(X < \lambda_\alpha) = 1 - \alpha$, où X est distribué suivant χ^2_k. En pratique, on prend par exemple $\alpha = 0.05$, ce qui donne pour $k = 2 : \lambda_\alpha = 5.99$.

Si $(D^{ij})^2 < \lambda_\alpha$, alors on note $\mathbf{u}^{ij} = (\hat{\mathbf{p}}^i, \hat{\mathbf{m}}^j)$ le couple mis en correspondance et on note $\mathbf{U}$ l'ensemble des u couples qui vérifient $(D^{ij})^2 < \lambda_\alpha$, parmi les $(m.p)$ couples possibles. L'algorithme 2 décrit la mise en correspondance de p observations $\mathbf{P}$ avec m amers du motif de parking $\mathbf{M}$.

4.4.3 Mise à jour de la position de parking du capteur

L'information apportée par la mise en correspondance de tout ou partie du motif de parking avec les perceptions permet de mettre à jour la position de parking du capteur $\hat{\mathbf{c}}^{park}$.

En adoptant le formalisme du filtre de Kalman étendu, on utilise l'équation (4.9) comme *fonction d'observation*, que l'on va linéariser. Alors *l'innovation* z est donnée par l'équation (4.10). En notant $\hat{\mathbf{c}}_0^{park}$ l'estimation initiale de la position de parking du capteur et $\mathbf{V}_{\mathbf{c}_0^{park}}$ sa matrice de covariance, on calcule alors récursivement pour chaque couple $\mathbf{u}_k = (\hat{\mathbf{p}}^i, \hat{\mathbf{m}}^j)$ de $\mathbf{U}$, une nouvelle estimation $\hat{\mathbf{c}}_k^{park}$:

$$\begin{aligned} \hat{\mathbf{c}}_{k+1}^{park} &= \hat{\mathbf{c}}_k^{park} + K_k^{\mathbf{c}} \hat{\mathbf{z}}_k \\ \mathbf{V}_{\mathbf{c}_{k+1}^{park}} &= (I - K_k^{\mathbf{c}} J_{\mathbf{c}}) \mathbf{V}_{\mathbf{c}_k^{park}} \end{aligned} \tag{4.12}$$

avec le gain de Kalman :

$$K_k^{\mathbf{c}} = \mathbf{V}_{\mathbf{c}_k^{park}} J_{\mathbf{c}}^T . (J_{\mathbf{c}} \mathbf{V}_{\mathbf{c}_k^{park}} J_{\mathbf{c}}^T + J_{\mathbf{m}} \mathbf{V}_{\mathbf{m}^j} J_{\mathbf{m}}^T + \mathbf{V}_{\mathbf{p}^i})^{-1}$$

Algorithme 2 *Mise en correspondance des éléments perçus et du motif de parking*

Données d'entrée :
- valeur α de la probabilité de rejeter une mise en correspondance alors que celle ci est valide.
- configuration courante du robot $\mathbf{q}$
- p observations $\mathbf{p}^i_{/\mathbf{c(q)}}, i \in \{1, ...p\}$
- m amers $\mathbf{m}^j, j \in \{1, ..., m\}$ du motif de parking $\mathbf{M}$
- $\hat{\mathbf{q}}^{park}$ et $\mathbf{V}_{\mathbf{q}^{park}}$

Données de sortie :
- ensemble $\mathbf{U}$ des couples $\mathbf{u}^{ij}$ mis en correspondance

Algorithme

$\lambda_\alpha \leftarrow p(X < \lambda) = 1 - \alpha$
$\hat{\mathbf{q}}^{park} \leftarrow \mathbf{q}(S)$
$\hat{\mathbf{c}}^{park} \leftarrow \mathbf{c}(\hat{\mathbf{q}}^{park})$
$\mathbf{U} \leftarrow \{\emptyset\}$
for chaque perception $\mathbf{p}^i_{/\mathbf{c(q)}}$ **do**
  calculer les $\mathbf{p}^i$ (équation 4.8) de $obsvect$.
end for
for chaque perception $\mathbf{p}^i$ de $\mathbf{P}$ **do**
  $D^2 \leftarrow \infty$
  $\mathbf{m}^\star \leftarrow \varnothing$
  for chaque amer $\mathbf{m}^j$ du motif de parking $\mathbf{M}$ **do**
    $\hat{\mathbf{l}}^j \leftarrow$ équation 4.9, $\mathbf{m}^j$ et $\hat{\mathbf{c}}^{park}$
    $\hat{\mathbf{z}}^{ij} \leftarrow$ équation 4.10, $\hat{\mathbf{l}}^j$, $\hat{\mathbf{p}}^i$
    $(D^{ij})^2 \leftarrow$ équation 4.11, $\hat{\mathbf{z}}^{ij}$
    if $((D^{ij})^2 < \lambda_\alpha \wedge (D^{ij})^2 < D^2)$ **then**
      $\mathbf{m}^\star \leftarrow \mathbf{m}^j$
      $D^2 \leftarrow (D^{jk})^2$
    end if
  end for
  if $\mathbf{m}^\star \neq \varnothing$ **then**
    insérer $\{\mathbf{p}^i, \mathbf{m}^\star\}$ dans $\mathbf{U}$
  end if
end for

et *l'innovation* :

$$\hat{\mathbf{z}}_k = (\hat{\mathbf{p}}^i - f(\hat{\mathbf{m}}^j, \hat{\mathbf{c}}_0^{park}))$$

avec $J_{\mathbf{c}} = \frac{df}{d\mathbf{c}}\big|_{\hat{\mathbf{c}}_0^{park}}$ la jacobienne de f par rapport à $\mathbf{c}$ évaluée en $\hat{\mathbf{c}}_0^{park}$, et $J_{\mathbf{m}} = \frac{df}{d\mathbf{m}}\big|_{\hat{\mathbf{m}}^j}$ la jacobienne de f par rapport à $\mathbf{m}$ évaluée en $\hat{\mathbf{m}}^j$.

L'estimation de la position de parking calculée après les u mises à jour successives est notée $\hat{\mathbf{c}}_u^{park}$. Elle correspond à l'estimation de variance minimale (c'est à dire minimisant : trace$(\mathbf{V}_{\mathbf{c}_u^{park}})$).

Remarque sur l'indépendance des variables

Cette écriture de l'équation (4.12) n'est possible que sous l'hypothèse que les différentes perceptions $\mathbf{p}^i$ sont indépendantes. Or l'équation (4.8) montre que ces différentes perceptions sont liées par la configuration courante du robot.

On note $\mathbf{P} = (\mathbf{p}^1, ..., \mathbf{p}^u)$ le vecteur composé des u perceptions mises en correspondance. Ce vecteur est obtenu en définissant la fonction F qui correspond à la fonction (4.8) pour u variables :

$$\begin{array}{ccc} \mathbf{P} & = & F(\mathbf{P}_{/\mathbf{c}}, \mathbf{c}) \\ \begin{pmatrix} \mathbf{p}^1 \\ \vdots \\ \mathbf{p}^u \end{pmatrix} & = & \begin{pmatrix} f(\mathbf{p}^1_{/\mathbf{c}}, \mathbf{c}) \\ \vdots \\ f(\mathbf{p}^u_{/\mathbf{c}}, \mathbf{c}) \end{pmatrix} \end{array}$$

Soit $\mathbf{V_P}$ la matrice de covariance de $\mathbf{P}$, calculée de façon similaire à $\mathbf{V}_{\mathbf{p}^i}$ dans l'équation (4.8), en remplaçant f par F. La dépendance des différentes perceptions s'exprime par le fait que la matrice $\mathbf{V_P}$ n'est pas bloc-diagonale, et l'approche de mise à jour récursive n'est donc pas immédiatement utilisable.

Pour résoudre ce problème, deux solutions sont possibles :

- diagonaliser la matrice de covariance $\mathbf{V_p}$. Nous renvoyons par exemple à [Bar-Shalom 1993] pour une présentation de la décorrélation des observations.
- faire une mise à jour par l'équation (4.12) en une seule étape plutôt que de façon séquentielle. Pour cela on définit le vecteur $\mathbf{Z} = \mathbf{P} - \mathbf{L}$ où $\mathbf{L} = (\mathrm{l}^1, ..., \mathrm{l}^u)$ est le vecteur constitué des u prédictions d'observations calculé par la «version à u variables» de l'équation (4.9), de façon similaire à la fonction F définie ci-dessus. L'équation (4.12) reste identique, seuls la taille du vecteur d'innovation et le calcul du gain changent. En pratique c'est la solution que nous adoptons, car la dimension du vecteur $\mathbf{P}$ restant petite, cela ne pénalise pas les temps de calcul.

4.4.4 Mise à jour de la configuration de parking

À partir de la nouvelle estimée $\hat{\mathbf{c}}_u^{park}$ de la position de parking du capteur, on peut mettre à jour l'estimation de la configuration de parking $\hat{\mathbf{q}}^{park}$. La *fonction d'observation* est ici donnée

par l'équation 4.6. L'*innovation* vaut $(\hat{\mathbf{c}}_u^{park} - \hat{\mathbf{c}}_0^{park})$. En notant $\hat{\mathbf{q}}_0^{park} = \mathbf{q}(S)$ l'estimation initiale de la configuration de parking, on a :

$$\begin{aligned} \hat{\mathbf{q}}_1^{park} &= \hat{\mathbf{c}}_0^{park} + K_0^{\mathbf{q}}(\hat{\mathbf{c}}_u^{park} - \hat{\mathbf{c}}_0^{park}) \\ \mathbf{V}_{\mathbf{q}_1^{park}} &= (I - K_0^{\mathbf{q}} J_{\mathbf{q}}) \mathbf{V}_{\mathbf{q}_0^{park}} \end{aligned} \tag{4.13}$$

avec le gain de Kalman :

$$K_0^{\mathbf{q}} = \mathbf{V}_{\mathbf{q}_0^{park}} J_{\mathbf{q}}^T (J_{\mathbf{q}} \mathbf{V}_{\mathbf{q}_0^{park}} J_{\mathbf{q}}^T + \mathbf{V}_{\mathbf{c}_u^{park}})^{-1}$$

où $J_{\mathbf{q}} = \left.\frac{d\mathbf{c}(\mathbf{q})}{d\mathbf{q}}\right|_{\hat{\mathbf{q}}_0^{park}}$ est la jacobienne de la fonction $\mathbf{c} = \mathbf{c}(\mathbf{q})$ évaluée en $\hat{\mathbf{q}}_0^{park}$.

4.4.5 Extension à plusieurs capteurs

On peut naturellement étendre ce formalisme au cas où le robot possède plusieurs capteurs pour la tâche de parking référencé sur des amers. Soit $n^{\mathbf{c}}$ le nombre de capteurs utilisés. Pour chaque capteur $\mathbf{c}_i$:

- on définit un motif de parking associé $\mathbf{M}_i$, avec $i \in \{1, .., n^{\mathbf{c}}\}$. On note $\hat{\mathbf{M}}_i$ l'estimation de ce motif de parking,
- on note $\mathbf{P}_i$ la perception des amers par le capteur $\mathbf{c}_i$,
- on note $\mathbf{c}_i(\mathbf{q})$ la fonction qui donne la position du capteur $\mathbf{c}_i$ à partir de la configuration du robot.

Pour calculer les estimées $\hat{\mathbf{c}}_i^{park}$ de chaque capteur à partir des perceptions, on utilise les mêmes étapes de mise en correspondance (section 4.4.2) et de mise à jour de la position de parking du capteur (section 4.4.3).

Seule l'étape de mise à jour de la configuration de parking doit être reformulée, de façon à tenir compte des $n^{\mathbf{c}}$ estimées des positions de parking $\hat{\mathbf{c}}_i^{park}$. On pourrait appliquer récursivement l'étape définie à la section 4.4.4, mais on retrouve le même problème que lors de la mise à jour de la position de parking à partir de plusieurs couples amer-perception quand les différentes perceptions ne sont pas indépendantes entre elles. En appliquant récursivement l'étape définie à la section 4.4.4, on négligerait la dépendance entre les $\hat{\mathbf{c}}_i^{park}, i \in \{1, ..., n^{\mathbf{c}}\}$, qui sont toutes des fonctions de la configuration courante du robot $\hat{\mathbf{q}}$ et de l'estimation de la configuration finale $\hat{\mathbf{q}}_0^{park}$.

Pour résoudre ce problème, on fait le calcul en une seule étape. On note $\mathbf{C}(\mathbf{q})$ le vecteur colonne constitué de l'ensemble des positions des différents capteurs et $\mathbf{V}_{\mathbf{C}}$ sa matrice de covariance :

$$\mathbf{C}(\mathbf{q}) = (\mathbf{c}_1(\mathbf{q}), \mathbf{c}_2(\mathbf{q}), .., \mathbf{c}_{n^{\mathbf{c}}}(\mathbf{q}))^T$$

L'estimée initiale $\hat{\mathbf{C}}_0^{park}$ de ce vecteur colonne est calculée à partir de $\hat{\mathbf{q}}_0^{park}$. La valeur mise à jour $\hat{\mathbf{C}}_u^{park}$ est donnée par l'étape de mise à jour des positions de parking de chaque capteur, comme présentée à la section 4.4.3.

Alors on calcule l'estimée de la configuration de parking en écrivant :

$$\hat{\mathbf{q}}_{n^{\mathbf{c}}}^{park} = \hat{\mathbf{q}}_0^{park} + K^{\mathbf{q}}(\hat{\mathbf{C}}_u^{park} - \hat{\mathbf{C}}_0^{park})$$

et le gain de Kalman vaut :

$$K^{\mathbf{q}} = \mathbf{V}_{\mathbf{q}_0^{park}} J_{\mathbf{q}}^T (J_{\mathbf{q}} \mathbf{V}_{\mathbf{q}_0^{park}} J_{\mathbf{q}}^T + \mathbf{V}_{\mathbf{C}_u^{park}})^{-1}$$

avec $J_{\mathbf{q}} = \frac{d\mathbf{C}(\mathbf{q})}{d\mathbf{q}}\Big|_{\hat{\mathbf{q}}^{park}}$ la jacobienne de $\mathbf{C}(\mathbf{q})$ par rapport à $\mathbf{q}$ évaluée en $\hat{\mathbf{q}}^{park}$.

4.5 Déformation de trajectoire

La section précédente a montré comment calculer la configuration de parking à partir des informations fournies par les capteurs à propos des motifs de parking. Il s'agit maintenant de modifier la trajectoire planifiée de façon à ce que celle-ci se termine en $\hat{\mathbf{q}}_{n^{\mathbf{c}}}^{park}$.

Nous utilisons pour cela la méthode de déformation de trajectoire nonholonome présentée au chapitre 3, en ajoutant une contrainte sur les conditions aux limites énoncées à la section 3.2.4.

Nous avions pensé déformer la trajectoire en utilisant un potentiel attractif vers la configuration de parking, en tirant parti des remarques de la section 3.5 sur l'optimisation de la trajectoire sur d'autres critères que la distance aux obstacles. L'utilisation des conditions aux limites a été retenue afin de s'assurer un meilleur contrôle de la déformation sous les effets conjugués des obstacles et de la tâche de parking.

4.5.1 Conditions aux limites

Nous rappelons la condition sur la fin de l'intervalle de déformation quand il s'agit de déformer la trajectoire afin que celle-ci s'éloigne des obstacles :

$$\tilde{\eta}(S,\tau) = \eta(S,\tau) + \eta^{\perp}(S,\tau) = 0$$

qui peut se mettre sous la forme de l'équation (3.23) :

$$L\lambda = -\eta^{\perp}(S,\tau)$$

Ces conditions aux limites sont valables même quand la fin de l'intervalle de déformation est la dernière configuration de la trajectoire, et elles imposent que $\mathbf{q}(S)$ reste inchangée après déformation. Dans une tâche de parking référencé sur des amers on désire au contraire que $\mathbf{q}(S)$ se rapproche de $\hat{\mathbf{q}}_{n^{\mathbf{c}}}^{park}$.

Si l'on note $\delta\mathbf{q}$ la différence entre $\hat{\mathbf{q}}_{n^{\mathbf{c}}}^{park}$ et $\hat{\mathbf{q}}_0^{park}$:

$$\delta\mathbf{q} = (\hat{\mathbf{q}}_{n^{\mathbf{c}}}^{park} - \hat{\mathbf{q}}_0^{park})$$

la condition aux limites imposant que $\mathbf{q}(S)$ après déformation se rapproche de $\hat{\mathbf{q}}_{n^{\mathbf{c}}}^{park}$ est :

$$\tilde{\eta}(S,\tau) = \eta(S,\tau) + \eta^{\perp}(S,\tau) = \delta\mathbf{q}$$

Ce qui peut se mettre sous la forme :

$$L\lambda = -\eta^{\perp}(S,\tau) + \delta\mathbf{q} \tag{4.14}$$

On calcule alors les coordonnées dans la base $\mathcal{E}$ de la direction de déformation associée en utilisant l'équation (3.24) :

$$\tilde{\lambda}^E = -P(L^E P)^+(\eta^\perp(S,\tau) - \delta\mathbf{q}) - (I_p - P(L^E P)^+ L^E)PP^T\mu^E \tag{4.15}$$

Cette nouvelle condition aux limites est la seule modification apportée à l'algorithme de la déformation présenté à la section 3.2.7.

4.6 Application au robot Hilare2 avec remorque

Nous illustrons cette méthode par un exemple d'application avec le robot Hilare2 avec remorque, représenté sur la figure 4.5.

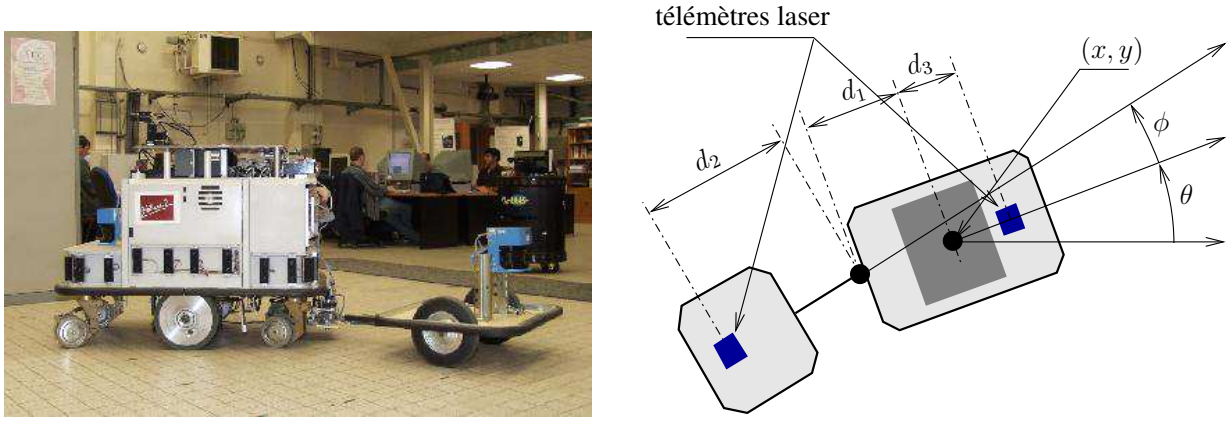

FIG. 4.5 – Le robot Hilare2 avec sa remorque

Le robot possède 4 variables de configuration : $\mathbf{q} = (x, y, \theta, \phi)$. x et y donnent la position du centre de l'essieu central du robot, exprimés en mètre. θ est l'orientation de l'axe perpendiculaire à l'essieu, et ϕ est l'orientation de la remorque par rapport au robot, exprimés en radians. d_1 est la distance entre le centre du robot et le point d'attache de la remorque, d_2 est la distance entre ce point d'attache et le capteur situé sur la remorque et d_3 est la distance entre le centre du robot et le capteur situé à l'avant.

4.6.1 Tâche de parking : parking relativement à un quai de débarquement

La tâche de parking consiste à positionner la remorque dans un emplacement représentant un quai de débarquement de marchandise pour des camions, visant à reproduire la situation représentée sur la figure 4.1. On définit alors un motif de parking qui spécifie cet emplacement par rapport au capteur situé sur la remorque, que l'on note $\mathbf{c}$. Lors de nos expériences, nous utiliserons un télémètre laser SICK, qui fournit après traitement les segments perçus dans un plan horizontal. Aussi choisissons nous dans cet exemple d'utiliser comme amers des segments, que l'on représentera par leurs droites porteuses afin d'être robuste aux occlusions.

Position du capteur

Le capteur situé sur la remorque est noté $\mathbf{c}$. Soit $(x_\mathbf{c}, y_\mathbf{c}, \theta_\mathbf{c})^T = \mathbf{c}(\mathbf{q})$ la situation du capteur lorsque le robot est dans la configuration $\mathbf{q}$:

$$\begin{pmatrix} x_\mathbf{c} \\ y_\mathbf{c} \\ \theta_\mathbf{c} \end{pmatrix} = \begin{pmatrix} x - d_1 \cos\theta - d_2 \cos(\theta + \phi) \\ y - d_1 \sin\theta - d_2 \sin(\theta + \phi) \\ \theta + \phi + \pi \end{pmatrix} \tag{4.16}$$

Motif de parking

Le motif de parking est constitué de trois segments $[A_iB_i], i \in \{1, 2, 3\}$, représentant un quai de débarquement. Les paramètres d'une droite porteuse (A_iB_i) sont notés $(o_i, \rho_i) \in ([-\pi, \pi], \mathbb{R})$. Les droites sont orientées et $\vec{n}_i$ est le vecteur unitaire normal à la droite et pointant vers l'espace

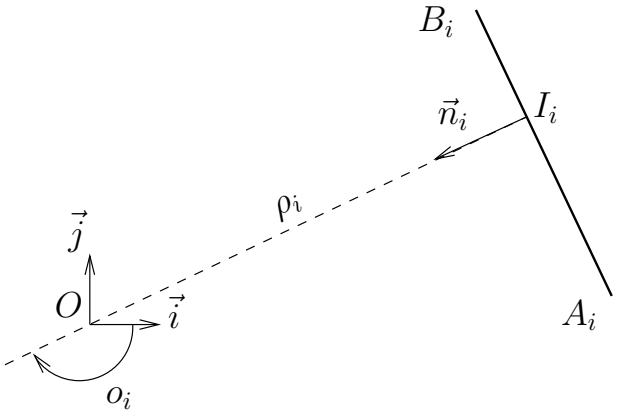

FIG. 4.6 – Les paramètres (o_i, ρ_i) de la droite porteuse du segment $[A_iB_i]$. $o_i = (\vec{i}, \vec{n}_i)$ et $\rho_i =< I_i\vec{O}, \vec{n}_i >$.

libre. Alors o_i est l'angle entre ce vecteur et l'axe des abscisses et on a :

$$\vec{n}_i = (\cos o_i, \sin o_i)^T$$

Soit I le point de concours de la droite (A_iB_i) et de la droite de vecteur directeur $\vec{n}_i$ passant par O. Alors ρ_i est :

$$\rho_i =< \overrightarrow{I_iO}, \vec{n}_i >$$

L'équation de la droite (A_iB_i) est alors :

$$x \cos o_i + y \sin o_i + \rho_i = 0$$

La figure 4.6 illustre le calcul des paramètres d'une droite porteuse orientée et la figure 4.7 donne les détails de construction du motif de parking utilisé dans cet exemple.

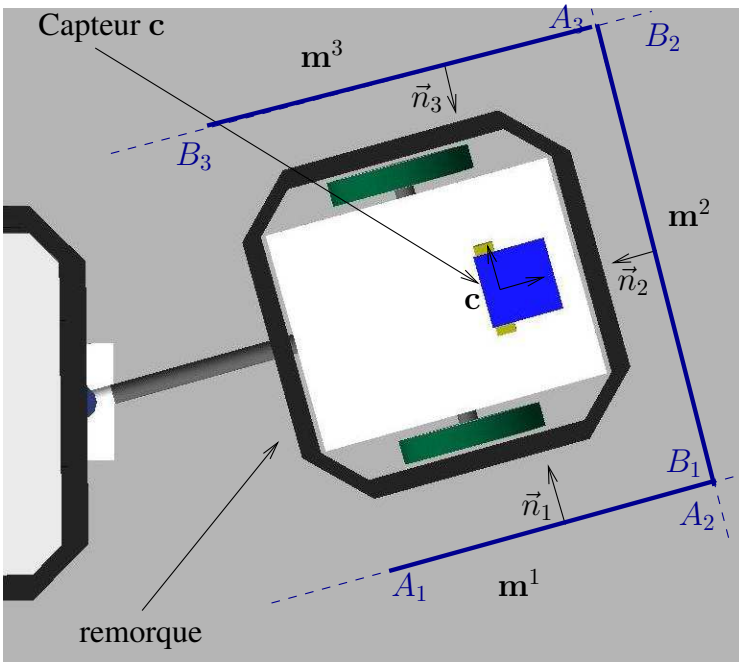

FIG. 4.7 – Les paramètres des droites porteuses constituant le motif de parking, exprimés relativement au capteur. Les valeurs des paramètres des amers exprimés dans le repère $\mathbf{c}$ sont : $\mathbf{m}^1 = (\frac{\pi}{2}, 0.5)^T$, $\mathbf{m}^2 = (\pi, 0.27)^T$ et $\mathbf{m}^3 = (-\frac{\pi}{2}, 0.5)^T$.

Changement de repère des paramètres d'un amer

Le capteur étant dans la situation $\mathbf{c} = (x_\mathbf{c}, y_\mathbf{c}, \theta_\mathbf{c})^T$, les paramètres d'un amer exprimés relativement au capteur sont notés $\mathbf{p}_{/\mathbf{c}} = (o_\mathbf{c}, \rho_\mathbf{c})^T$. Alors les paramètres de cet amer exprimés dans le repère d'origine O sont notés $\mathbf{p} = (o_O, \rho_O)^T$ et ont pour valeur :

$$\begin{aligned} o_O &= o_\mathbf{c} + \theta_\mathbf{c} \\ \rho_O &= \rho_\mathbf{c} - x_\mathbf{c}\cos(\theta_\mathbf{c} + o_\mathbf{c}) - y_\mathbf{c}\sin(\theta_\mathbf{c} + o_\mathbf{c}) \end{aligned} \tag{4.17}$$

Cette équation définit la fonction f utilisée dans les équations (4.8) et (4.9) :

$$\begin{aligned} \mathbf{p} &= f(\mathbf{p}_{/\mathbf{c}}, \mathbf{c}) \\ \mathbf{l} &= f(\mathbf{m}, \mathbf{c}) \end{aligned}$$

La figure 4.8 illustre ce changement de repère.

Valeurs initiales des incertitudes

On formule les hypothèses suivantes :

- les amers du motif de parking sont indépendants deux à deux et la matrice $\mathbf{V}_\mathbf{M}$ est donc bloc-diagonale. Cette matrice est une donnée d'entrée de la tâche de parking.
- les perceptions $\mathbf{p}_{/\mathbf{c}}$ des amers par le capteur dans la position $\mathbf{c}$ sont indépendantes deux à deux et la matrice $\mathbf{V}_{\mathbf{p}_{/\mathbf{c}}}$ est donc bloc-diagonale.
- la configuration finale $\mathbf{q}(S)$ de la trajectoire planifiée est l'estimée initiale de la configuration de parking $\mathbf{q}_0^{park}$. La matrice de covariance $\mathbf{V}_{\mathbf{q}_0^{park}}$ est diagonale.

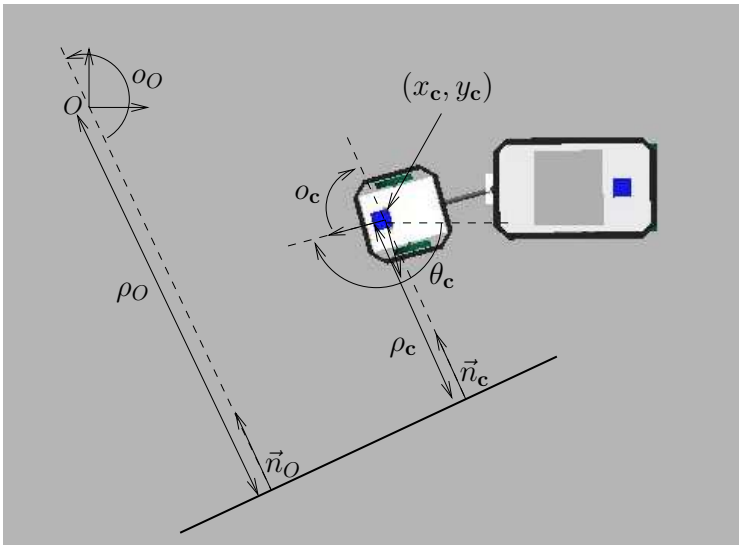

FIG. 4.8 – Changement de repère des paramètres d'une droite. La droite est perçue depuis la situation $\mathbf{c} = (x_\mathbf{c}, y_\mathbf{c}, \theta_\mathbf{c})^T$, et ses paramètres dans ce repère sont notés $(o_\mathbf{c}, \rho_\mathbf{c})^T$. Les paramètres de cette droite dans le repère d'origine O sont notés $(o_O, \rho_O)^T$, et sont calculés par l'équation (4.17).

Dans cet exemple, on ajoute l'hypothèse que l'incertitude sur la configuration courante du robot est nulle, afin de simplifier les calculs. Cette hypothèse est bien entendu levée lors des expériences.

Incertitude sur les paramètres du motif de parking Pour chaque élément $\mathbf{m} = (o, \rho)^T$ du motif de parking, la matrice de covariance de $\hat{\mathbf{m}}$ est :

$$\mathbf{V_m} = \begin{pmatrix} (\frac{0.1}{3})^2 & 0 \\ 0 & (\frac{1}{3})^2 \end{pmatrix}$$

Incertitude sur les amers perçus Dans cet exemple, on considère que l'incertitude d'un amer perçu est constante, quelle que soit sa position par rapport au capteur. Et comme l'incertitude sur la configuration courante du robot est nulle, on a $\mathbf{V_{p/c}} = \mathbf{V_p}$. C'est à dire que l'incertitude des paramètres d'un amer perçu exprimé relativement au capteur est égale à l'incertitude des paramètres de cet amer exprimé dans le plan. Soit $\mathbf{p} = (o_\mathbf{p}, \rho_\mathbf{p})^T$ les paramètres d'un amer perçu. On a :

$$\mathbf{V_p} = \begin{pmatrix} (\frac{0.1}{3})^2 & 0 \\ 0 & (\frac{1}{3})^2 \end{pmatrix}$$

Incertitude sur l'estimée initiale de la configuration de parking L'estimée initiale de la configuration de parking est la configuration finale de la trajectoire planifiée dans un modèle de

l'environnement : $\hat{\mathbf{q}}_0^{park} = \mathbf{q}(S)$. La matrice de covariance de $\mathbf{q}_0^{park}$ est notée $\mathbf{V}_{\mathbf{q}_0^{park}}$, et vaut :

$$\mathbf{V}_{\mathbf{q}_0^{park}} = \begin{pmatrix} (\frac{1}{3})^2 & 0 & 0 & 0 \\ 0 & (\frac{1}{3})^2 & 0 & 0 \\ 0 & 0 & (\frac{0.4}{3})^2 & 0 \\ 0 & 0 & 0 & (\frac{0.4}{3})^2 \end{pmatrix}$$

Les valeurs de la matrice $\mathbf{V}_{\mathbf{q}_0^{park}}$ définissent un intervalle de confiance pour chacune des variables de configuration, centré sur $\hat{\mathbf{q}}_0^{park}$. On a choisi ces valeurs de telle façon que l'intervalle de confiance à 99% pour les variables x et y soit de plus ou moins 1 autour de la valeur moyenne ($x^{park} \in [\hat{x}^{park} - 1, \hat{x}^{park} + 1]$), et de plus ou moins 0.4 rad pour les variables θ et ϕ.

En simplifiant, on peut dire que si la configuration de parking calculée à l'issue de la tâche de parking n'est pas dans cet intervalle, elle est rejetée car jugée trop peu vraisemblable. En réalité on utilise la distance de Mahalanobis pour mesurer la vraisemblance de la configuration de parking calculée par rapport à son estimée initiale.

4.6.2 Résolution de la tâche de parking

Modèle de l'environnement

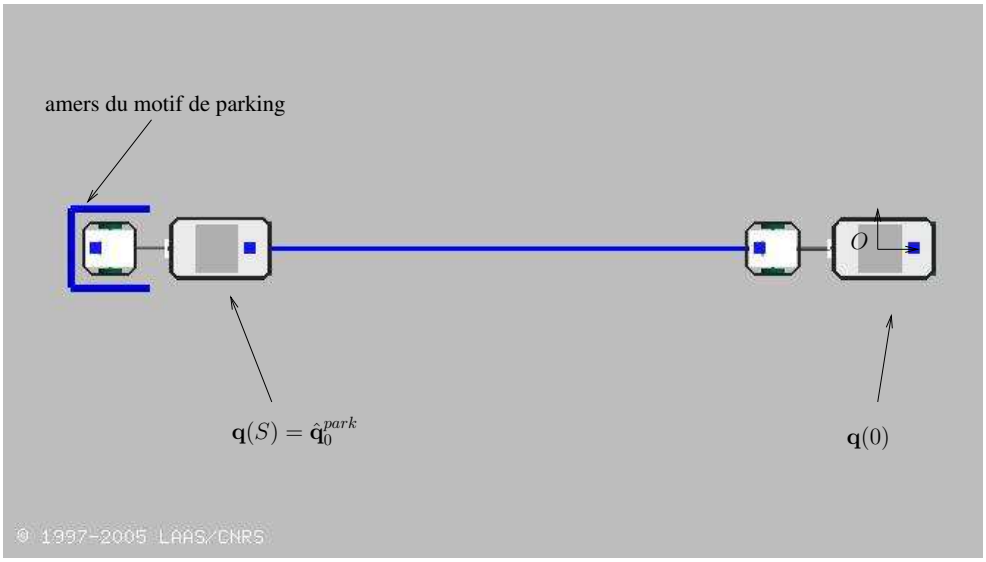

FIG. 4.9 – Modèle de l'environnement. Le robot est en configuration initiale $\mathbf{q}(0)$ et une trajectoire est planifiée vers la position de parking calculée dans ce modèle.

Dans cet exemple, le modèle de l'environnement dans lequel est planifiée la trajectoire contient trois amers $\mathbf{l}^1, \mathbf{l}^2, \mathbf{l}^3$, qui correspondent au motif de parking défini plus haut. Les paramètres de ces amers exprimés dans le repère d'origine O sont :

$$\mathbf{l}^1 = (\frac{\pi}{2}, 0.5) \quad \mathbf{l}^2 = (\pi, 10.27) \quad \mathbf{l}^3 = (-\frac{\pi}{2}, 0.5)$$

Dans ce modèle, la situation du capteur en position de parking est donc : $\mathbf{c}_0^{park} = (-10, 0, 0)^T$. Et une configuration finale est par exemple $\mathbf{q}(S) = (-8.383, 0, 0, 0)^T$, car elle vérifie $\mathbf{c}(\mathbf{q}(S)) = \mathbf{c}_0^{park}$. On note $\hat{\mathbf{q}}_0^{park}$ l'estimée initiale de la configuration de parking, qui vaut donc $\mathbf{q}(S)$.

La configuration initiale de robot étant $\mathbf{q}(0) = (0, 0, 0, 0)^T$, une trajectoire en ligne droite est planifiée de $\mathbf{q}(0)$ à $\mathbf{q}(S)$.

La figure 4.9 représente le modèle de l'environnement, dans lequel les amers du motif de parking sont positionnés. Une trajectoire rectiligne en marche arrière est planifiée, depuis $\mathbf{q}(0)$ jusqu'à la configuration de parking $\mathbf{q}(S)$ calculée dans ce modèle.

Environnement perçu

Dans notre exemple la perception simulée du capteur ne correspond pas au modèle de l'environnement. En effet on suppose qu'un seul amer est détecté, et ses paramètres exprimés dans le repère O ne sont strictement égaux à aucun des paramètres des amers du modèle de l'environnement. Les paramètres de l'amer perçu exprimés dans le repère O sont : $\hat{\mathbf{p}}^1 = (\pi + \frac{\pi}{8}, 10.27 \cos \frac{\pi}{8})$. C'est à dire que cette perception correspond à l'amer $\mathbf{l}^2$ ayant subi une rotation d'angle $\frac{\pi}{8}$ et de centre $(-10.27, 0)$.

La figure 4.10 représente l'environnement perçu par le robot qui est visiblement différent du modèle de l'environnement.

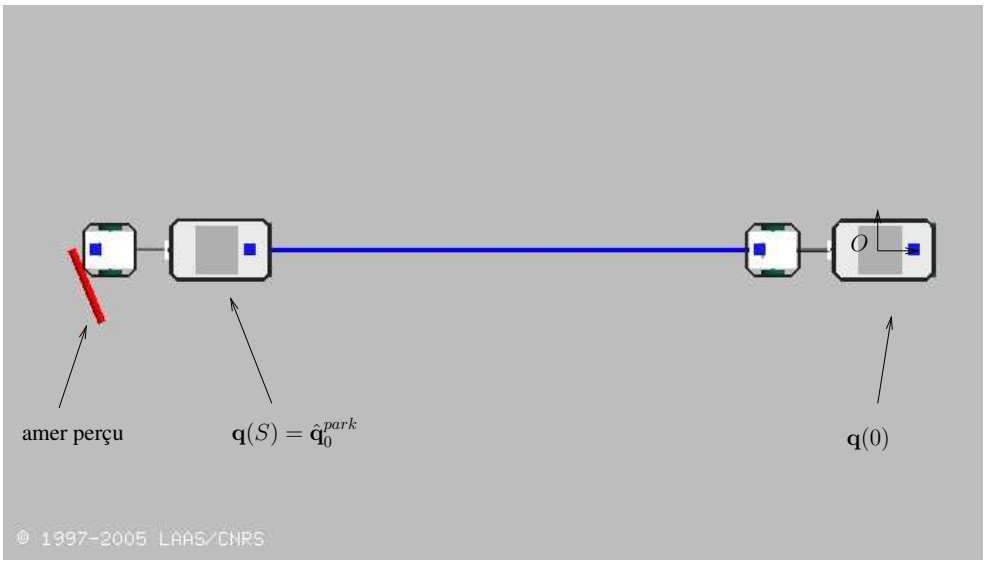

FIG. 4.10 – Environnement perçu par le capteur. Un seul des trois amers du motif de parking est présent, et ses paramètres sont différents.

Mise en correspondance

On calcule la différence entre la perception et les paramètres estimés d'un amer du motif de parking et on applique l'algorithme 2 de mise en correspondance, présenté à la section 4.4.2.

L'*innovation* est $\hat{\mathbf{z}}^{ij} = \hat{\mathbf{p}}^i - \hat{\mathbf{l}}^j$, avec $i = 1$ et$j \in \{1, 2, 3\}$.

L'unique couple amer-perception retourné par l'algorithme est bien le couple $\mathbf{u}^{12} = (\mathbf{p}^1, \mathbf{l}^2)$. La valeur de la distance de Mahalanobis calculée par l'équation (4.11) pour ce couple est de 5.94, qui est la plus faible valeur parmi les trois couples possibles.

Mise à jour de la position de parking du capteur

Avec cette information de mise en correspondance, on peut mettre à jour la position de parking du capteur en procédant comme indiqué à la section 4.4.3. L'équation (4.12) de mise à jour s'écrit :

$$\begin{aligned} \hat{\mathbf{c}}_1^{park} &= \mathbf{c}(\hat{\mathbf{q}}_0^{park}) + K^{\mathbf{c}}(\mathbf{p}_1 - \mathbf{l}^2) \\ \mathbf{V}_{\mathbf{c}_1^{park}} &= (I - K_0^{\mathbf{c}} J_{\mathbf{c}}) \mathbf{V}_{\mathbf{c}_0^{park}} \end{aligned}$$

avec $\mathbf{V}_{\mathbf{c}_0^{park}}$ calculée par l'équation (4.6) et la donnée de $\mathbf{V}_{\mathbf{q}_0^{park}}$. Et le gain $K_0^{\mathbf{c}}$ est :

$$K^{\mathbf{c}} = \mathbf{V}_{\mathbf{c}_0^{park}} J_{\mathbf{c}}^T . (J_{\mathbf{c}} \mathbf{V}_{\mathbf{c}_0^{park}} J_{\mathbf{c}}^T + J_{\mathbf{m}} \mathbf{V}_{\mathbf{m}^2} J_{\mathbf{m}}^T + \mathbf{V}_{\mathbf{p}^1})^{-1}$$

où $J_{\mathbf{c}}$ et $J_{\mathbf{m}}$ sont les jacobiennes de la fonction f définie dans cet exemple par l'équation (4.17), respectivement par rapport à $\mathbf{c}$ et évaluée en $\mathbf{c}_0^{park}$, et à $\mathbf{m}$ et évaluée en $\hat{\mathbf{m}}^2$.

La position de parking du capteur mise à jour est $\hat{\mathbf{c}}_1^{park} = (-9.739, -0.477, 3.511)^T$.

Mise à jour de la configuration de parking

On peut maintenant mettre à jour la configuration de parking du robot en procédant comme indiqué à la section 4.4.4. L'équation (4.13) de mise à jour de la configuration s'écrit :

$$\begin{aligned} \hat{\mathbf{q}}_1^{park} &= \hat{\mathbf{q}}_0^{park} + K^{\mathbf{q}}(\hat{\mathbf{c}}_1^{park} - \mathbf{c}(\hat{\mathbf{q}}_0^{park})) \\ \mathbf{V}_{\mathbf{q}_1^{park}} &= (I - K^{\mathbf{q}} J_{\mathbf{q}}) \mathbf{V}_{\mathbf{q}_0^{park}} \end{aligned}$$

avec $J_{\mathbf{q}}$ la jacobienne de la fonction $\mathbf{q} \mapsto \mathbf{c}(\mathbf{q})$, évaluée en $\hat{\mathbf{q}}_0^{park}$.

La configuration de parking mise à jour est $\hat{\mathbf{q}}_1^{park} = (-8.227, 0, 0.174, 0.174)^T$, à comparer avec l'estimation initiale de la configuration de parking :$\hat{\mathbf{q}}_0^{park} = (-8.383, 0, 0, 0)$. La matrice de covariance $\mathbf{V}_{\mathbf{q}_1^{park}}$ ainsi calculée est :

$$\mathbf{V}_{\mathbf{q}_1^{park}} = \begin{pmatrix} 0.044 & 0 & 0 & 0 \\ 0 & 0.057 & 0.003 & -0.003 \\ 0 & 0.003 & 0.009 & -0.008 \\ 0 & -0.003 & -0.008 & 0.009 \end{pmatrix}$$

On observe que l'incertitude sur la configuration de parking a diminué par rapport à sa valeur initiale.

Déformation de la trajectoire

La configuration de parking calculée $\hat{\mathbf{q}}_1^{park}$ n'est pas égale à la configuration finale de la trajectoire. On va donc déformer la trajectoire de façon à ce que la configuration finale se rapproche de la configuration de parking, en utilisant les résultats présentés à la section 4.5. Dans cet exemple on ne tient pas compte des collisions avec les obstacles.

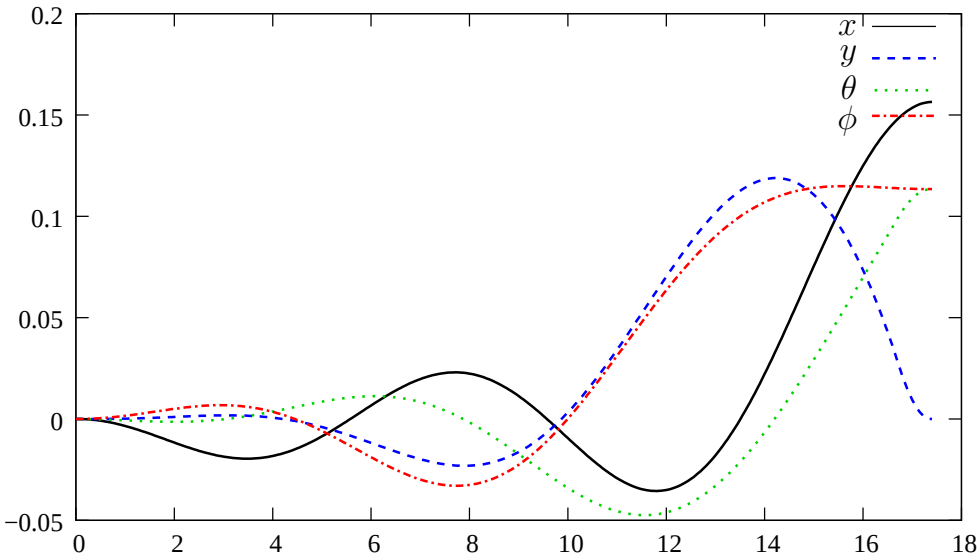

FIG. 4.11 – Direction de déformation $\tilde{\eta}$ de la tâche de parking

On note $\delta\mathbf{q} = \hat{\mathbf{q}}_1^{park} - \hat{\mathbf{q}}_0^{park}$. Étant donné que la trajectoire initiale est parfaitement admissible pour le système, la direction de déformation $\eta^{\perp}$ de correction de la dérive des contraintes nonholonomes est nulle sur tout l'intervalle de déformation (voir la section 3.2.6). La contrainte sur la fin de l'intervalle de déformation exprimée par l'équation (4.14) s'écrit donc :

$$L\lambda = \delta\mathbf{q}$$

où $\lambda = \mathbf{0}$ car les obstacles n'ont pas d'influence.

On calcule alors le vecteur λ des coordonnées de la direction de déformation $\tilde{\eta}$ en utilisant l'équation (4.15).

La figure 4.11 présente la direction de déformation $\tilde{\eta}$ calculée. On vérifie que cette direction de déformation est telle que $\tilde{\eta}(S) = \delta\mathbf{q}$.

En appliquant maintenant une partie de cette direction de déformation à la trajectoire initiale, on trouve une trajectoire déformée dont la configuration finale se rapproche de $\hat{\mathbf{q}}_1^{park}$:

$$\forall s \in [0, S], \quad \mathbf{q}(s) \leftarrow \mathbf{q}(s) + \Delta\tau\tilde{\eta}(s)$$

Dans cet exemple on fixe $\Delta\tau$ à 0.2.

Après dix itérations, la distance entre la configuration finale et la configuration de parking est inférieure à 10^{-2}. La figure 4.12 représente cette trajectoire déformée.

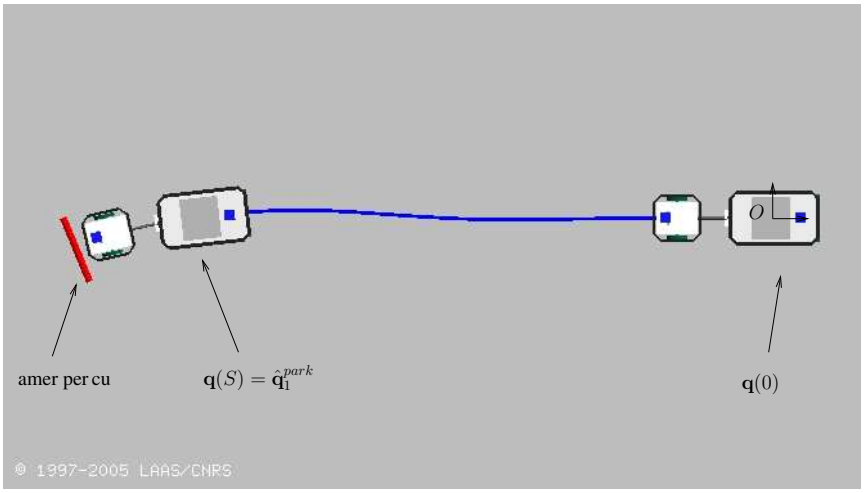

FIG. 4.12 – Trajectoire déformée par la tâche de parking référencé sur des amers

4.7 Conclusion

La section précédente nous a permis de définir tous les éléments nécessaires à une tâche de parking dans un exemple réaliste. Les résultats expérimentaux obtenus lors de la réalisation de cet exemple avec le robot réel Hilare2 sont présentés dans le chapitre 6.

Pour conclure ce chapitre nous proposons de résumer les spécificités de notre approche du parking référencé sur des amers. Tout d'abord on peut noter sa généricité, qui découle de la formulation très générale d'une tâche de parking référencé sur des amers, et de la généricité de la méthode de déformation de trajectoire employée. Les contreparties de cette généricité résident dans la nécessité de posséder une trajectoire de référence.

Par ailleurs cette approche peut être employée dans une grande variété de situations. Elle peut, par exemple, être directement utilisée avec plusieurs capteurs, ou dans des applications où le motif de parking ne définit pas complètement la position de parking. On traite ainsi les problèmes sous déterminés tels que se garer «le long d'un mur», ou bien «entre deux voitures», sans que ces éléments ne spécifient complètement la position de parking. De même, cette méthode s'accommode très bien de l'absence de tout ou partie du motif de parking lors de l'exécution. Par contre, notre formalisme impose que le motif de parking soit constitué d'éléments dont on peut déterminer la position. Ainsi, les amers de type point d'intérêt dans une image sont exclus, et doivent être spécifiés sous la forme de points dans le plan ou dans l'espace.

De plus, le problème de l'évitement d'obstacles est directement intégré dans cette approche. Il peut cependant y avoir des cas d'échec que nous n'avons pas traités : par exemple si la configuration de parking calculée $\mathbf{q}^{park}$ est en collision. Ceci peut se produire car la tâche de parking ne considère pas les éléments de l'environnement comme des obstacles, mais uniquement comme des amers. Un compromis entre tâche de parking et évitement d'obstacles doit alors être trouvé, mais cette problématique n'a pas été développée.

Chapitre 5

Intégration du suivi de trajectoire, de l'évitement d'obstacles et de la localisation

5.1 Introduction

Certaines fonctionnalités nécessaires à la navigation autonome sont réalisées «hors-ligne». C'est le cas de la modélisation de l'environnement, de la planification d'une trajectoire dans ce modèle et de la définition d'un motif de parking par exemple. D'autres fonctionnalités au contraire doivent être réalisées au cours de la navigation. C'est le cas du suivi de trajectoire, de l'évitement réactif d'obstacles, de la localisation et du parking référencé capteur. L'objet de ce chapitre est de montrer que la réalisation simultanée de ces différentes fonctionnalités ne se résume pas à la réalisation isolée de chacune d'entre elles. Nous proposons donc une architecture permettant d'intégrer ces différentes fonctionnalités.

Dans la première section nous présentons une architecture permettant la réalisation simultanée du suivi de trajectoire et de l'évitement réactif d'obstacles. Nous mettons en évidence la nécessité de pouvoir ralentir le long de la trajectoire planifiée, et nous montrons que les spécificités des systèmes abordés dans nos travaux posent des problèmes particuliers.

Dans la section 5.3 nous ajoutons à cette architecture la fonctionnalité de localisation globale. En effet pour pouvoir effectuer de longs mouvements, le robot ne peut pas compter sur la seule intégration de son déplacement pour estimer sa position, et il doit pouvoir se localiser dans une carte de son environnement. Après avoir remarqué que cette localisation globale produit une estimation discontinue de la position du robot, nous mettons en évidence les problèmes que cela pose pour l'intégration dans une architecture de suivi de trajectoire, compte tenu des spécificités de nos systèmes. En effet les systèmes que nous étudions, les véhicules articulés, et les environnements dans lesquels nous souhaitons les faire évoluer, les environnements fortement contraints, font que le suivi de trajectoire ne pourra pas «absorber» les perturbations de la localisation globale. Et nous illustrons sur un exemple le fait que le rattrapage immédiat de ces discontinuités par un processus de suivi de trajectoire qui n'assure pas une décroissance uniforme de la distance à la consigne, peut entraîner des collisions.

Enfin, à la section 5.4, nous présentons une architecture adaptée aux spécificités de notre

approche, qui intègre ces différentes fonctionnalités et qui garantit l'absence de collision lors du suivi de trajectoire avec une localisation globale.

5.2 Suivi de trajectoire et évitement réactif d'obstacles

Le suivi de trajectoire consiste à asservir la configuration du robot sur une trajectoire de référence. L'évitement réactif d'obstacles, présenté au chapitre 3 doit assurer que la trajectoire est sans collision et la déformer sur un intervalle quand une collision est détectée. On rappelle que les obstacles sont supposés statiques.

Ainsi l'évitement d'obstacles peut amener le suivi de trajectoire à ralentir, voire même à s'arrêter sur la trajectoire si la collision ne peut être éliminée.

5.2.1 Suivi de trajectoire

Le processus de suivi de trajectoire en boucle fermée consiste à calculer une commande à appliquer au robot étant données sa configuration courante et une configuration de référence. On note respectivement $\mathbf{q}^{ref}$ et $\mathbf{u}^{ref}$ la configuration et la commande de référence le long d'une trajectoire admissible, c'est à dire solution de l'équation (2.7).

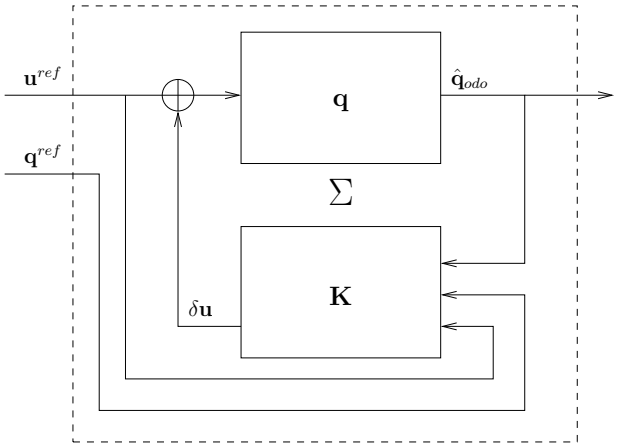

FIG. 5.1 – Système (Σ) de suivi de trajectoire en boucle fermée

Système de commande en boucle fermée

On note (Σ) le système de suivi de trajectoire en boucle fermée, représenté dans la figure 5.1. L'évolution de la configuration du robot représentée par le bloc $\mathbf{q}$ est donnée par l'équation (2.7), à un bruit de mesure près. L'état $\mathbf{q}$ du système est inconnu, et on note $\hat{\mathbf{q}}_{odo}$ l'estimation de la configuration du robot.

Cette estimation de l'état du système est fournie par l'odométrie ou par une centrale inertielle par exemple. Parfois qualifiée de proprioceptive[1], ce type de localisation à l'estime («dead-reckoning» pour «deduced-reckoning» en anglais) ne repose pas sur une carte d'amers et son erreur s'accroît au fur et à mesure de l'exécution de la trajectoire. On suppose pour l'instant que c'est la seule estimation de la configuration du robot que l'on possède. On admet cependant qu'à l'initialisation le robot est localisé de telle sorte que l'estimation de la configuration est égale à la configuration initiale de la trajectoire :

$$\hat{\mathbf{q}}_{odo}(0) = \mathbf{q}^{ref}(0)$$

La loi de commande notée $\mathbf{K}$ calcule la correction $\delta\mathbf{u}$ à appliquer à la commande de référence $\mathbf{u}$:

$$\delta\mathbf{u} = \mathbf{K}(\hat{\mathbf{q}}_{odo}, \mathbf{q}^{ref}, \mathbf{u}^{ref})$$

Nous ne donnons pas plus de détail sur la loi de commande adoptée. Une littérature très riche est en effet disponible sur le sujet, et nous renvoyons à [Luca 1998] pour une synthèse.

Stabilité

Le concept de stabilité d'un système autour d'une trajectoire est utile à notre étude, car le suivi de trajectoire a en charge de stabiliser le système autour de sa trajectoire de référence. Les définitions de la stabilité que nous allons donner ne sont pas celles habituellement utilisées en théorie de la commande. En effet, dans le cadre de notre étude, c'est l'évolution de la distance du système à sa trajectoire de référence qui nous intéresse.

Propriété 8. *Stabilité exponentielle :*
si $(\mathbf{q}^{ref}, \mathbf{u}^{ref})$ *est une trajectoire admissible sur* $[0, S]$ *exécutée par le système* (Σ)*, alors* $\exists\lambda > 0$ $\exists\mu > 0$ *tels que :*

$$\textit{si } \exists s_1 \in [0, S] \quad \textit{tel que} \quad d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(s_1), \mathbf{q}^{ref}(s_1)) < \mu \quad \textit{alors}$$
$$\forall s \in [s_1, S] \quad d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(s), \mathbf{q}^{ref}(s)) \leq e^{-\lambda(s-s_1)} d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(s_1), \mathbf{q}^{ref}(s_1))$$

Les systèmes complètement actionnés sont généralement exponentiellement stables.

Les véhicules articulés et les véhicules nonholonomes en général n'ont par contre pas la propriété de stabilité exponentielle. La distance à la consigne ne décroît pas uniformément, mais peut au contraire augmenter avant de diminuer, comme illustré sur la figure 5.2. Pour ces systèmes on définit la propriété suivante.

Propriété 9. *Stabilité effective :*
il existe une constante réelle positive D^{suivi} *telle que si* $(\mathbf{q}^{ref}, \mathbf{u}^{ref})$ *est une trajectoire admissible sur* $[0, S]$ *exécutée par le système* (Σ)*, et si*

$$\hat{\mathbf{q}}_{odo}(0) = \mathbf{q}^{ref}(0)$$

Alors pour tout $s \in [0, S]$*,*

$$d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(s), \mathbf{q}^{ref}(s)) < D^{suivi}$$

[1] c'est à dire qui mesure l'état interne du système, par opposition à extéroceptive

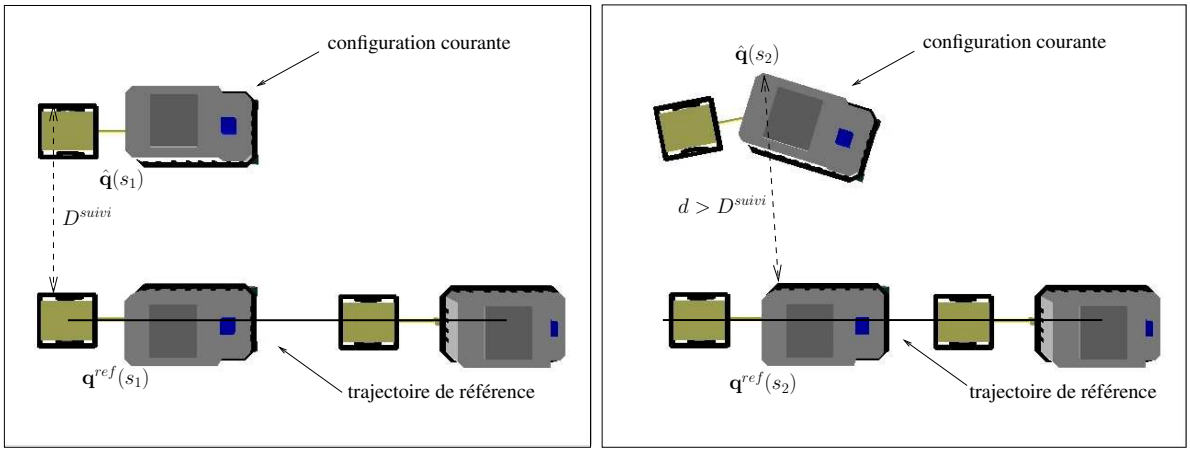

FIG. 5.2 – Pour les véhicules articulés, la distance à la consigne ne décroît pas uniformément

Dans les deux propriétés précédentes, la loi de commande en boucle fermée assure que l'estimation de la configuration $\hat{\mathbf{q}}_{odo}$ reste dans un voisinage de la configuration de référence $\mathbf{q}^{ref}$ au cours du suivi de trajectoire.

5.2.2 S'arrêter sur une trajectoire

Une propriété essentielle du suivi de trajectoire dans le contexte de l'évitement réactif d'obstacles est de pouvoir s'arrêter sur la trajectoire quand une collision est détectée et que la trajectoire ne peut pas être déformée suffisamment rapidement.

Pour un système soumis à des contraintes d'accélération et suivant une trajectoire quelconque, cette propriété n'est pas immédiate. Nous rapportons ici des résultats de [Bonnafous 2003b] et [Bonnafous 2003a].

Bornes cinématiques

Un robot mobile est soumis à des contraintes sur ses vitesses et accélérations, qu'on suppose de la forme : $\forall i \in \{1, ..., k\}$

$$-v_{iM} \leq \quad u_i(t) \quad \leq v_{iM} \tag{5.1}$$

$$-\dot{v}_{iM} \leq \quad \dot{u}_i(t) \quad \leq \dot{v}_{iM} \tag{5.2}$$

où v_{iM} et $\dot{v}_{iM}$ sont les bornes sur les commandes du système (vitesses) et leurs dérivées (accélérations).

Paramétrage d'une trajectoire

Le paramétrage d'une trajectoire qui respecte ces contraintes a été étudié dans le cadre du mouvement en temps minimal de robots manipulateurs (contraintes d'accélération concurrentes) dans [Bobrow 1985, Shin 1986, Slotine 1989, Renaud 1992]. Ces résultats ont été étendus aux robots

mobiles en prenant en compte les contraintes sur les vitesses dans [Lamiraux 1997b]. Le résultat d'un tel paramétrage est donc une trajectoire admissible définie sur un intervalle $[0, S]$ et qui satisfait :

$$-v_{iM} \leq \quad u_i(s) \quad \leq v_{iM} \tag{5.3}$$
$$-\dot{v}_{iM} \leq \quad \frac{du_i}{ds}(s) \quad \leq \dot{v}_{iM} \tag{5.4}$$

Ce paramétrage est tel que si $s = t$, le système respecte les bornes cinématiques.

Suivi de trajectoire

Le processus de suivi de trajectoire est un processus temps-réel qui est en charge de l'évolution de l'abscisse s au cours de l'exécution de la trajectoire, de façon éventuellement à ralentir et à s'arrêter sur cette trajectoire.

En considérant s comme une fonction du temps, on exprime les commandes réellement envoyées au système en fonction des commandes de la trajectoire :

$$\mathbf{v}(t) = \dot{s}(t)\mathbf{u}(s(t)) \tag{5.5}$$

Et en dérivant cette expression :

$$\dot{\mathbf{v}}(t) = \dot{s}(t)^2 \frac{d\mathbf{u}}{ds}(s(t)) + \ddot{s}(t)\mathbf{u}(s(t)) \tag{5.6}$$

L'expression des contraintes (5.1-5.2) devient alors :
$\forall i \in \{1, ..., k\}$

$$-v_{iM} \leq \quad \dot{s}(t)u_i(s(t)) \quad \leq v_{iM} \tag{5.7}$$
$$-\dot{v}_{iM} \leq \quad \dot{s}(t)^2 \frac{du_i}{ds}(s(t)) + \ddot{s}(t)u_i(s(t)) \quad \leq \dot{v}_{iM} \tag{5.8}$$

Le processus de suivi de trajectoire est donc en charge de l'évolution de l'état $(s, \dot{s})$, sous les contraintes précédentes.

Si l'on suppose que :

$$0 \leq \dot{s}(t) \leq 1 \tag{5.9}$$

alors (5.3) assure que la contrainte (5.7) est toujours respectée.

On peut mettre la contrainte (5.8) sous la forme :

$$\ddot{s}_{min}(s, \dot{s}) \leq \ddot{s} \leq \ddot{s}_{max}(s, \dot{s})$$

Pour s'arrêter le long de la trajectoire, le processus de suivi de trajectoire doit donc calculer en temps réel :

$$\ddot{s} = \ddot{s}_{min}(s, \dot{s}) \tag{5.10}$$

jusqu'à $\dot{s} = 0$.

Pour des trajectoires simples telles des combinaisons d'arcs de cercle et de lignes droites, le calcul de la distance de freinage est immédiat, mais pour des trajectoires quelconques il faut

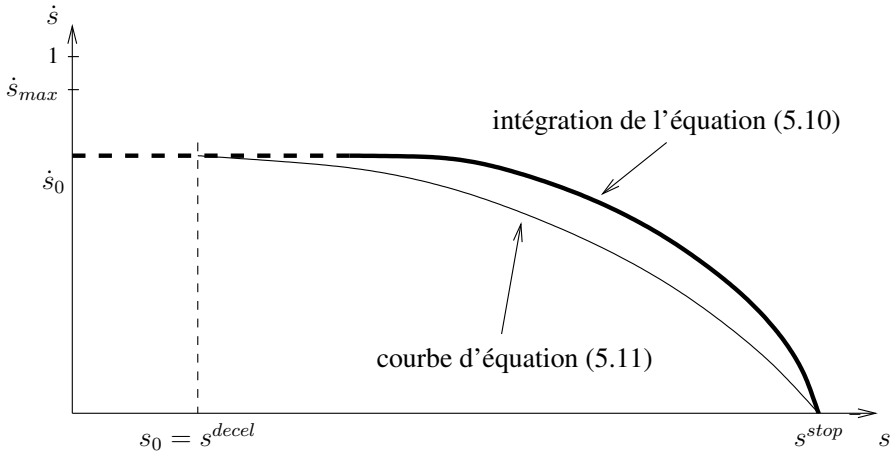

FIG. 5.3 – Courbe de décélération optimale calculée par intégration (en gras), et courbe de décélération minorante donnée par l'équation (5.11).

intégrer l'équation (5.10) depuis l'état courant $(s, \dot{s})$ jusqu'à $\dot{s} = 0$. Et cette opération peut être incompatible avec les contraintes temps-réel de la tâche de suivi de trajectoire.

La méthode donnée dans [Bonnafous 2003b] permet de trouver directement une expression analytique d'une courbe de décélération qui respecte la contrainte (5.8), sous l'hypothèse que :

$$0 \leq \dot{s} \leq \dot{s}_{max} < 1$$

L'intérêt pour notre application de suivi de trajectoire est que le calcul de l'évolution de l'abscisse s (et de $\dot{s}$) lors d'une décélération sur la trajectoire est immédiat, et qu'il peut donc être réalisé en temps-réel.

L'expression de la courbe de décélération dans le plan de phase $(\dot{s}, s)$ est :

$$\dot{s} = \sqrt{1 - (1 - \dot{s}_0^2) e^{2m(s - s_0)}} \tag{5.11}$$

où $(s_0, \dot{s}_0)$ est l'état initial de la décélération et $m = \min_{i \in [0,k]} \frac{\dot{v}_{iM}}{v_{iM}}$. La figure 5.3 illustre ce résultat. Cette courbe est bien un minorant de la courbe de décélération optimale correspondant à l'intégration de l'équation (5.10).

Fonctionnalités du suivi de trajectoire

On utilise ce résultat pour définir deux fonctions utiles au suivi de trajectoire dans notre étude : DECEL et RALENTIR. La fonction DECEL$(\dot{s}, s^{stop})$ calcule l'abscisse s^{decel} à partir de laquelle le robot doit décélérer de façon à s'arrêter à un s^{stop} donné. En utilisant l'équation (5.11), on a :

$$s^{decel} = \text{DECEL}(\dot{s}, s^{stop}) = s^{stop} + \frac{1}{2m} \log(1 - \dot{s}^2) \tag{5.12}$$

La fonction RALENTIR(s^{decel}, s) donne l'évolution de l'état $(s, \dot{s})$ le long de la courbe de décélération :

$$\dot{s} = \text{RALENTIR}(s^{decel}, s) = \sqrt{1 - (1 - \dot{s}_0^2) e^{2m(s - s^{decel})}}$$

où $\dot{s}_0$ est la valeur de $\dot{s}$ en $s = s^{decel}$. Cette fonction garantit que les bornes cinématiques sont respectées et elle assure un arrêt du robot à l'abscisse s^{stop} désirée.

5.2.3 Détection des obstacles

Soient $\mathbf{q} = (\mathbf{x}, \mathbf{q}_{int})$ et $\hat{\mathbf{q}}_{odo} = (\hat{\mathbf{x}}_{odo}, \mathbf{q}_{int})$ respectivement la configuration réelle et la configuration estimée du robot quand un obstacle O est détecté par les capteurs. On note $\mathcal{O}$ le sous-espace de l'espace de travail occupé par O. Seule la partie visible de l'obstacle, notée $\mathcal{O}^{visible}$, est perçue. L'obstacle détecté exprimé dans le repère du robot est donc $\mathbf{x}^{-1}(\mathcal{O}^{visible})$, au bruit de mesure près. On note alors $\hat{\mathcal{O}}_{loc}$ l'union de la partie visible augmentée de l'erreur de perception $D^{capteur}$, et la partie cachée par cette partie, comme illustré sur la figure 5.4. On suppose que $\hat{\mathcal{O}}_{loc}$ est une donnée fournie par le capteur, et donc exprimée relativement au robot. L'estimation du sous-espace occupé par l'obstacle est notée $\hat{\mathcal{O}}$, et correspond à la transformation de $\hat{\mathcal{O}}_{loc}$ par l'estimation de la position du robot $\hat{\mathbf{x}}_{odo}$:

$$\hat{\mathcal{O}} = \hat{\mathbf{x}}_{odo}(\hat{\mathcal{O}}_{loc})$$

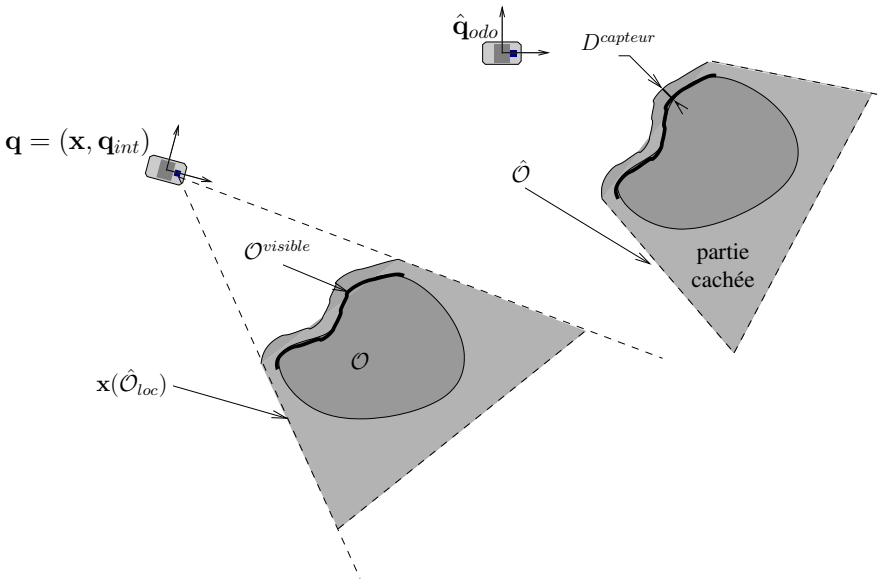

FIG. 5.4 – Détection des obstacles *(voir texte)*

Et du fait que $\hat{\mathcal{O}}_{loc}$ inclut le bruit de mesure, on admet que :

$$\mathcal{O} \subset \mathbf{x}(\hat{\mathcal{O}}_{loc}) = \mathbf{x}.\hat{\mathbf{x}}_{odo}^{-1}(\hat{\mathcal{O}}) \tag{5.13}$$

En notant $D^{clear} > 0$ le réel exprimant la distance minimale désirée aux obstacles, une configuration sur la trajectoire est considérée en collision si et seulement si :

$$d_O(\mathbf{q}^{ref}, \hat{\mathcal{O}}) > D^{clear} \tag{5.14}$$

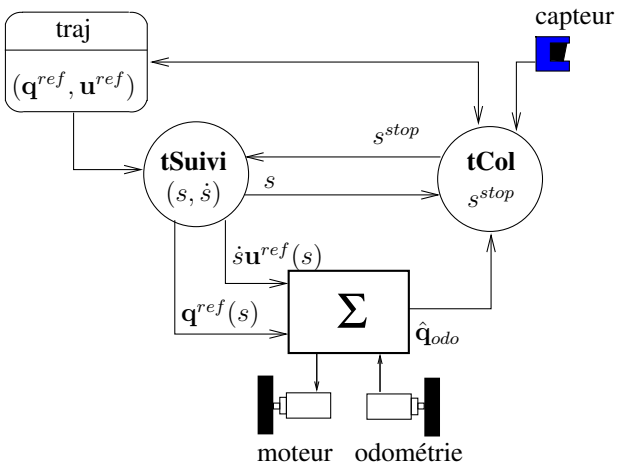

FIG. 5.5 – Architecture de contrôle de suivi de trajectoire et de la l'évitement d'obstacles.

5.2.4 Architecture pour le suivi de trajectoire et l'évitement d'obstacles

Le suivi de trajectoire et l'évitement d'obstacles sont naturellement séparés en deux tâches périodiques différentes, notés respectivement **tCol** et **tSuivi**. La figure 5.5 présente le diagramme d'architecture de ces deux tâches.

tCol (Algorithme 3)

Cette tâche réalise l'évitement réactif d'obstacle par la méthode de déformation de trajectoire présentée au chapitre 3. Elle implémente la fonction CHECK qui détecte les collisions sur l'intervalle $[s, s^{stop}]$ et la fonction DEFORME qui déforme la trajectoire sur l'intervalle $[s^{stop} - \Delta s^{col}, s^{stop} + \Delta s^{col}]$. Sa période est grande du fait de la complexité des calculs des fonctions CHECK et DEFORME évoquée à la section 3.4.2.

Algorithme 3 tâche **tCol**

Entrée : $s, \mathbf{q}^{ref}, \hat{\mathbf{q}}_{odo}, \hat{\mathcal{O}}_{loc}$
Sortie : s^{stop}
loop
  $s^{stop} \leftarrow$ CHECK ($\mathbf{q}^{ref}(s), \hat{\mathbf{x}}_{odo}(\hat{\mathcal{O}}_{loc})$) sur $[s, s + \Delta s^{check}]$
  if $s^{stop} \leq (s + \Delta s^{check})$ **then**
    $\mathbf{q}^{ref}(s) \leftarrow$ DEFORME($\mathbf{q}^{ref}(s)$) sur $[s^{stop} - \Delta s^{col}, s^{stop} + \Delta s^{col}]$
  end if
end loop

tSuivi (Algorithme 4)

Cette tâche assure le suivi de la trajectoire à une fréquence élevée. Elle implémente la fonction DECEL qui calcule l'abscisse s^{decel} à partir de laquelle le robot doit ralentir pour s'arrêter

en s^{stop} et la fonction RALENTIR qui calcule une décélération faisable. Elle envoie les consignes au système (Σ). Sa période est petite, et égale à celle du système (Σ).

Algorithme 4 tâche **tSuivi**

Entrée : $s^{stop}, \mathbf{q}^{ref}, \mathbf{u}^{ref}$
Sortie : $s, \mathbf{q}^{ref}(s), \dot{s}.\mathbf{u}^{ref}(s)$
loop
  $s^{decel} \leftarrow$ DECEL($\dot{s}, s^{stop} - \Delta s^{col}$)
  if $s < s^{decel}$ **then**
    augmenter $\dot{s}$ vers $\dot{s}_{max}$
  else
    $\dot{s} \leftarrow$ RALENTIR($s^{stop} - \Delta s^{col}, s$)
  end if
  $s + \dot{s}\Delta t \rightarrow s$
end loop

Accès concurrents à la trajectoire traj

Il est important de remarquer que la trajectoire **traj** est une donnée accédée en concurrence par les tâches **tCol** et **tSuivi**. L'intégrité de cette donnée est assurée par le fait que l'accès en écriture par la tâche **tCol** s'effectue uniquement sur l'intervalle $[s^{stop} - \Delta s^{col}, s^{stop} + \Delta s^{col}]$ et que l'accès en lecture par la tâche **tSuivi** ne se fait que sur l'intervalle $[s, s^{stop} - \Delta s^{col}]$.

5.2.5 Conditions garantissant l'absence de collision lors du suivi de la trajectoire

On montre ici que sous certaines hypothèses, l'architecture présentée permet de garantir l'absence de collision lors de l'exécution de la trajectoire de référence.

Définissons tout d'abord l'accroissement d'erreur odométrique entre deux abscisses sur une trajectoire.

Définition 1. *On note* $\hat{\mathbf{q}}_{odo}(s) = (\hat{\mathbf{x}}_{odo}(s), \hat{\mathbf{q}}_{int})$ *et* $\mathbf{q}(s) = (\mathbf{x}(s), \mathbf{q}_{int})$ *respectivement l'estimation de la configuration et la configuration réelle du robot à l'abscisse* s*. L'accroissement d'erreur odométrique entre l'abscisse* s_1 *et l'abscisse* s_2 *est :*

$$d_{SE(2)}(\hat{\mathbf{x}}_{odo}^{-1}(s_1).\hat{\mathbf{x}}_{odo}(s_2), \mathbf{x}(s_1)^{-1}.\mathbf{x}(s_2))$$

on rappelle la propriété de la distance $d_{SE(2)}$: $\forall m, \mathbf{x}_1, \mathbf{x}_2 \in SE(2)$

$$d_{SE(2)}(m.\mathbf{x}_1, m.\mathbf{x}_2) = d_{SE(2)}(\mathbf{x}_1, \mathbf{x}_2)$$

La figure 5.6 illustre cette définition.

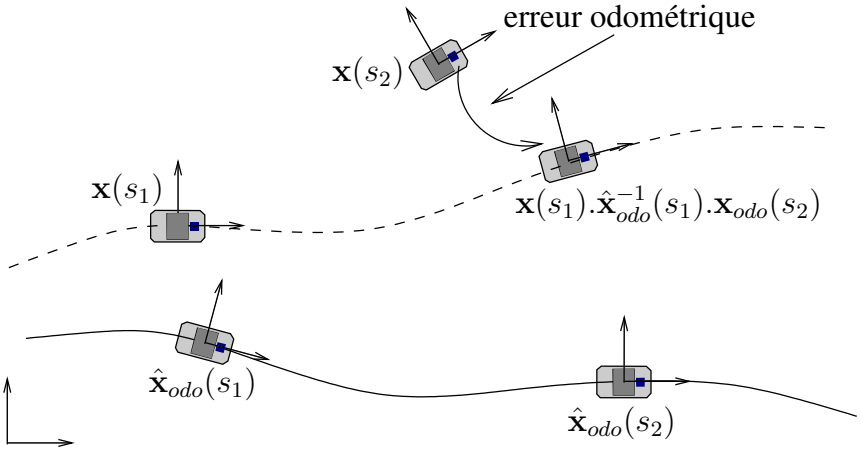

FIG. 5.6 – Accroissement de l'erreur odométrique entre s_1 et s_2

Propriété 10. *Soit $(\mathbf{q}^{ref}, \mathbf{u}^{ref})$ une trajectoire définie sur l'intervalle $[0, S]$. À l'abscisse $s = 0$ le robot effectue une perception de l'obstacle O. Si :*

1. *la trajectoire est vérifiée sans collision : pour tout $s \in [0, S]$, $d_O(\mathbf{q}^{ref}(s), \hat{\mathcal{O}}) > D^{clear}$,*
2. *l'équation (5.13) est respectée : $\mathcal{O} \subset \mathbf{x}(0).\hat{\mathbf{x}}_{odo}^{-1}(0)(\hat{\mathcal{O}})$,*
3. *l'erreur odométrique entre 0 et s est bornée par D^{odo}, pour tout $s \in [0, S]$,*
4. *$\hat{\mathbf{q}}_{odo}(0) = \mathbf{q}^{ref}(0)$ et (Σ) satisfait la propriété 9 de stabilité effective,*
5. *les variables de configuration internes sont parfaitement mesurées : $\hat{\mathbf{q}}_{odo}(s) = (\hat{\mathbf{x}}_{odo}(s), \mathbf{q}_{int}(s))$,*
6. *$D^{odo} + D^{suivi} < D^{clear}$,*

alors le robot n'est pas en collision le long de la trajectoire de référence :

$$\forall s \in [0, S], \quad d_O(\mathbf{q}(s), \mathcal{O}) > 0$$

Démonstration. On note m_0 la transformation solide $\mathbf{x}(0).\hat{\mathbf{x}}_{odo}^{-1}(0)$. En appliquant m_0 à $\mathbf{q}^{ref}$ et $\hat{\mathcal{O}}$ dans l'hypothèse 1, on obtient :

$$d_O(\mathbf{x}(0).\hat{\mathbf{x}}_{odo}^{-1}(0).\mathbf{q}^{ref}(s), \mathbf{x}(0).\hat{\mathbf{x}}_{odo}^{-1}(0)(\hat{\mathcal{O}})) > D^{clear}$$

En utilisant l'hypothèse 2, il vient :

$$d_O(\mathbf{x}(0).\hat{\mathbf{x}}_{odo}^{-1}(0).\mathbf{q}^{ref}(s), \mathcal{O}) > D^{clear} \tag{5.15}$$

En appliquant maintenant m_0 à $\mathbf{q}^{ref}$ et $\hat{\mathbf{q}}_{odo}$ dans la propriété 9 on a :

$$d_{\mathcal{C}}(\mathbf{x}(0).\hat{\mathbf{x}}_{odo}^{-1}(0).\mathbf{q}^{ref}(s), \mathbf{x}(0).\hat{\mathbf{x}}_{odo}^{-1}(0).\hat{\mathbf{q}}_{odo}(s)) \leq D^{suivi} \tag{5.16}$$

En utilisant la définition de $d_{SE(2)}$ et l'hypothèse 5, on peut écrire :

$$\begin{aligned} d_{\mathcal{C}}(\mathbf{x}(0).\hat{\mathbf{x}}_{odo}^{-1}(0).\hat{\mathbf{q}}_{odo}(s), \mathbf{q}(s)) &= d_{SE(2)}(\mathbf{x}(0).\hat{\mathbf{x}}_{odo}^{-1}(0).\hat{\mathbf{x}}_{odo}(s), \mathbf{x}(s)) \\ &= d_{SE(2)}(\hat{\mathbf{x}}_{odo}^{-1}(0).\hat{\mathbf{x}}_{odo}(s), \mathbf{x}^{-1}(0).\mathbf{x}(s)) \end{aligned}$$

On reconnaît là l'expression de l'erreur odométrique entre 0 et s, donnée dans la définition 1 illustrée par la figure 5.6. En utilisant l'hypothèse 3 on peut alors écrire :

$$d_{\mathcal{C}}(\mathbf{x}(0).\hat{\mathbf{x}}_{odo}^{-1}(0).\hat{\mathbf{q}}_{odo}(s), \mathbf{q}(s)) \leq D^{odo} \tag{5.17}$$

Alors en utilisant l'inégalité triangulaire avec l'expression à gauche de la somme de (5.16) et (5.17), il vient :

$$d_{\mathcal{C}}(\mathbf{x}(0).\hat{\mathbf{x}}_{odo}^{-1}(0).\mathbf{q}^{ref}(s), \mathbf{q}(s)) \leq D^{suivi} + D^{odo} \tag{5.18}$$

On applique alors l'inégalité triangulaire de la propriété 1 à la différence entre (5.15) et (5.18) :

$$d_O(\mathbf{q}(s), \mathcal{O}) > D^{clear} - D^{suivi} - D^{odo} > 0$$

□

La propriété 10 nous assure donc qu'en utilisant seulement l'odométrie comme estimation de la configuration, le robot n'entre pas en collision avec les obstacles en suivant la trajectoire planifiée. Pour ce faire les obstacles sont grossis de la somme de l'erreur de suivi de trajectoire D^{suivi} et de l'erreur odométrique D^{odo}, comme illustré dans la figure 5.7.

Remarquons que nous avons utilisé la propriété de stabilité 9, que respectent les systèmes abordés dans notre étude. Il est immédiat que les systèmes respectant la propriété 8 respectent aussi cette propriété, et que la démonstration reste valable.

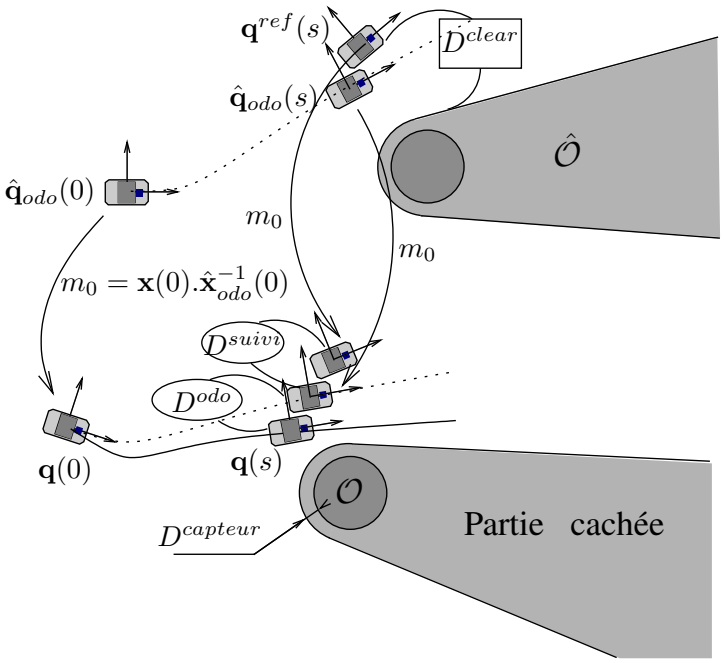

FIG. 5.7 – Illustration de la preuve de la propriété 10. La borne inférieure de la distance aux obstacles D^{clear} est supérieure à la somme des bornes supérieures de l'erreur d'odométrie D^{odo} et de l'erreur de suivi de trajectoire D^{suivi}.

5.3 Intégration de la localisation globale

Dans la section précédente nous avons montré que l'on pouvait garantir l'absence de collision lors du suivi d'une trajectoire avec l'odométrie (ou tout autre donnée de localisation à l'estime) comme seule estimation de la configuration du robot. Cependant pour réaliser de longs mouvements, il est nécessaire de localiser le robot dans une carte d'amers, de façon à réduire la dérive de l'odométrie (ou de la localisation «proprioceptive»). Nous appelons ce type de localisation basée sur des amers la «localisation globale». Nous présentons tout d'abord l'intégration de la localisation globale dans notre architecture de contrôle. Nous montrons ensuite que cette architecture ne permet pas de garantir l'absence de collision pour les systèmes respectant la propriété de stabilité 9, c'est à dire les systèmes auxquels on s'intéresse dans notre étude.

Finalement nous montrons que sous certaines hypothèses très restrictives quant aux discontinuités de la localisation et à la distance minimale aux obstacles, il est possible de garantir l'absence de collision pour les systèmes exponentiellement stables avec cette architecture. Cependant ces hypothèses ne sont pas réalistes, et ce résultat ne peut donc pas s'étendre aux cas définis par les spécificités de notre étude. Cela motivera le développement dans la section suivante d'une architecture adéquate pour le suivi de trajectoire sécurisé pour les véhicules articulés naviguant proches des obstacles.

5.3.1 Localisation globale

Nous appelons «localisation globale» la localisation du robot dans une carte. La localisation par GPS ou par mise en correspondance d'amers perçus avec une carte d'amers rentrent dans cette catégorie. En fusionnant ces données de localisation avec la localisation proprioceptive, dans un filtre de Kalman par exemple, on obtient une estimation de la position du robot de variance minimale.

Soit $(l_n)_{n\in\{1,\cdots,n^{loc}\}}$ la suite d'abscisses auxquelles a lieu la localisation globale. À chacune de ces abscisses, une estimation de la position du robot $\hat{\mathbf{x}}^n(l_n)$ est calculée. Pour la suite de notre étude il est utile d'exprimer la transformation solide m_n qui transforme $\hat{\mathbf{x}}_{odo}(l_n)$ en $\hat{\mathbf{x}}(l_n)$. On a :

$$m_n = \hat{\mathbf{x}}(l_n).\hat{\mathbf{x}}_{odo}^{-1}(l_n) \tag{5.19}$$

Ainsi entre deux localisations globales, l'estimation de la position du robot est obtenue en appliquant la dernière transformation m_n à l'estimation donnée par l'odométrie : $\forall s \in [l_n, l_{n+1}]$

$$\hat{\mathbf{x}}(s) = \hat{\mathbf{x}}^n(s) = m_n.\hat{\mathbf{x}}_{odo}(s) \tag{5.20}$$

Discontinuités de l'estimation de la position

Du fait des mesures bruitées du capteur et des imprécisions ou incohérences du plan, l'estimation de la position du robot peut être discontinue. Et ce malgré la fusion entre la localisation globale et l'odométrie.

On peut borner ces discontinuités, en testant la vraisemblance de la nouvelle estimée par rapport à l'estimation courante, et en n'acceptant cette nouvelle estimée que si la vraisemblance est

assez grande. On peut par exemple utiliser la distance de Mahalanobis entre ces deux estimations (présentée dans la section 4.4) pour mesurer cette vraisemblance.

Nous notons D^{loc} la borne maximale autorisée des discontinuités de localisation, dont nous préciserons plus loin la définition. Ce qu'il est important de remarquer, c'est que ces sauts dans l'estimation de la position du robot entraînent des discontinuités dans la transformation m_n calculée à chaque abscisse de localisation l_n. Nous allons voir les conséquences de ces discontinuités.

5.3.2 Architecture pour le suivi de trajectoire avec localisation globale

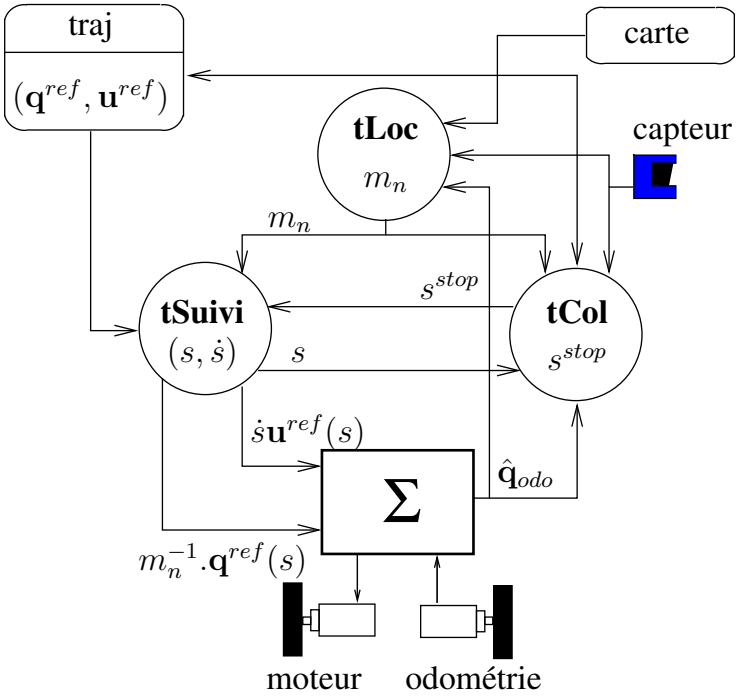

FIG. 5.8 – Intégration du processus de localisation globale **tLoc** dans l'architecture de suivi de trajectoire avec évitement d'obstacles.

Le processus de localisation globale décrit ci-dessus s'intègre aisément dans l'architecture proposée dans la section 5.2.4. La figure 5.8 présente le diagramme de cette architecture. La localisation globale est assurée par une nouvelle tâche notée **tLoc**, qui est en charge du calcul de m_n. On suppose qu'initialement le robot est localisé de telle sorte que $\hat{\mathbf{q}}(0) = \mathbf{q}^{ref}(0)$.

Cependant le système (Σ) de suivi de trajectoire présenté à la section 5.2 assure la stabilité de $\hat{\mathbf{q}}_{odo}$ autour d'une trajectoire de référence, mais ne dit rien à propos de $\hat{\mathbf{q}}$. Pour assurer la stabilité de $\hat{\mathbf{q}}$ autour de $\mathbf{q}^{ref}$, on définit $\mathbf{q}^{ref}_{odo}$, la trajectoire obtenue en appliquant la transformation solide m_n^{-1} à $\mathbf{q}^{ref}$:

$$\mathbf{q}^{ref}_{odo}(s) = m_n^{-1}.\mathbf{q}^{ref}(s) \tag{5.21}$$

C'est sur cette trajectoire que va s'asservir le système (Σ). En utilisant la propriété 1 de symétrie par rapport à $SE(2)$ définie au chapitre 2, on vérifie aisément que si $\hat{\mathbf{q}}_{odo}$ est stable

autour de $\mathbf{q}_{odo}^{ref}$, alors $\hat{\mathbf{q}}$ est stable autour de $\mathbf{q}^{ref}$.

On remarque néanmoins que la transformation m_n^{-1} n'est pas constante, et donc que $\mathbf{q}_{odo}^{ref}$ n'est pas continue. Il nous faut alors trouver un ensemble d'hypothèses pour garantir l'absence de collision lors du suivi de trajectoire dans ces conditions. C'est l'objet de la section suivante.

5.3.3 Conditions garantissant l'absence de collision pour les systèmes exponentiellement stables

Dans le cas des systèmes exponentiellement stables au sens de la propriété 8, nous allons montrer que les discontinuités de localisation peuvent être englobées dans la distance minimale souhaitée aux obstacles D^{clear}.

Détection des collisions

Soit $c_i (i \geq 1)$, l'abscisse à laquelle le robot perçoit l'environnement avec son capteur afin de détecter les obstacles, et soit j le plus grand entier tel que $l_j \leq c_i$: l_j est l'abscisse de la dernière localisation, et m_j est la transformation correspondante définie à l'équation (5.19). En reprenant les notations du paragraphe 5.2.3, on a :

$$\hat{\mathcal{O}} = \hat{\mathbf{x}}(c_i)(\hat{\mathcal{O}}_{loc}) = m_j \hat{\mathbf{x}}_{odo}(c_i)(\hat{\mathcal{O}}_{loc}) \tag{5.22}$$

Ainsi, alors que l'algorithme (3) de la tâche **tCol** détectait les collisions en appliquant la fonction :

$$\text{CHECK}(\mathbf{q}^{ref}(s), \hat{\mathbf{x}}_{odo}(\hat{\mathcal{O}}_{loc}))$$

on doit maintenant appliquer :

$$\text{CHECK}(\mathbf{q}^{ref}(s), m_j \hat{\mathbf{x}}_{odo}(c_i)(\hat{\mathcal{O}}_{loc}))$$

Localisation

On suppose qu'entre les abscisses c_i et $c_i + \Delta s^{check}$ la tâche **tLoc** effectue k localisations globales :

$$l_j \leq c_i < l_{j+1} < \cdots < l_{j+k} < c_i + \Delta s^{check}$$

Et entre 2 localisations globales l_p et l_{p+1}, la trajectoire de référence envoyée au système (Σ) est $(m_p^{-1}\mathbf{q}^{ref}, \mathbf{u}^{ref})$.

Alors on a la propriété suivante :

Propriété 11. *Si :*

1. **à l'abscisse c_i, l'estimation de la configuration du robot est dans un voisinage de la configuration de référence :**

$$d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(c_i), m_j^{-1}\mathbf{q}^{ref}(c_i)) \leq D^{loc}$$

2. **le système (Σ) est exponentiellement stable** *avec $\mu \geq 2D^{loc}$, (propriété 8),*

3. **la trajectoire de référence est détectée sans collision** *:*

$$\forall s \in [c_i, c_i + \Delta s^{check}] \quad d_O(\mathbf{q}^{ref}(s), \hat{\mathcal{O}}) > D^{clear}$$

4. **les obstacles détectés englobent les obstacles réels** *(équations (5.13) et (5.22)) :*

$$\mathcal{O} \subset \mathbf{x}(c_i)\hat{\mathbf{x}}^{-1}(c_i)(\hat{\mathcal{O}}) = \mathbf{x}(c_i)\hat{\mathbf{x}}_{odo}^{-1}(c_i)m_j^{-1}(\hat{\mathcal{O}})$$

5. **l'accroissement d'erreur odométrique entre** c_i **et** s **est bornée par** D^{odo} *pour tout* $s \in [c_i, c_i + \Delta s^{check}]$,

6. **les variables de configurations internes sont parfaitement mesurées** *:*
$\hat{\mathbf{q}}_{odo}(s) = (\hat{\mathbf{x}}_{odo}(s), \mathbf{q}_{int}(s))$

7. **les discontinuités de localisation sont bornées** *:*
$\forall p, q,\ j \leq p, q \leq j + k$ *et* $\forall s \in [c_i, c_i + \Delta s^{check}]$,

$$d_{\mathcal{C}}(m_p^{-1}\mathbf{q}^{ref}(s), m_q^{-1}\mathbf{q}^{ref}(s)) \leq D^{loc}$$

8. **les abscisses de localisations sont suffisamment éloignées pour que le système ait le temps de converger vers la trajectoire de référence** *:*
$\forall p, j \leq p \leq j + k - 1$

$$d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(l_{p+1}), m_p^{-1}\mathbf{q}^{ref}(l_{p+1})) \leq \frac{1}{2} d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(l_p), m_p^{-1}\mathbf{q}^{ref}(l_p))$$

9. $D^{odo} + 3D^{loc} < D^{clear}$

alors le long de la trajectoire de référence le robot n'est pas en collision :

$$\forall s \in [c_i, c_i + \Delta s^{check}], \qquad d_O(\mathbf{q}(s), \mathcal{O}) > 0$$

La preuve de cette propriété est donnée dans l'annexe A. L'intérêt de la propriété 11 est d'apporter une preuve formelle à une connaissance validée par l'expérience, à savoir que pour les systèmes exponentiellement stables, en prenant une marge raisonnable par rapport aux obstacles (hypothèse 9), le suivi de trajectoire s'effectue sans collision. Ainsi pour un robot rond évoluant dans un environnement peu contraint, les perturbations induites par les discontinuités de la localisation peuvent être englobées dans la distance minimale aux obstacles.

Mais pour assurer la même propriété pour les systèmes ne respectant pas la propriété 8 il faudrait augmenter encore la distance minimale aux obstacles D^{clear}. En effet pour de tels systèmes la distance à la consigne peut augmenter avant de décroître, comme illustré sur la figure 5.2. Dans le cadre de notre étude où les mouvements doivent parfois s'effectuer à proximité des obstacles, cette option n'est pas envisageable.

5.3.4 Cas réalistes avec un véhicule articulé

Dans des cas réalistes, la distance minimale aux obstacles D^{clear} est inférieure à la discontinuité de la localisation D^{loc}. De plus les systèmes auxquels on s'intéresse ne respectent pas la propriété de stabilité exponentielle 8. On ne peut donc pas utiliser la propriété 11 et on n'a donc pas de garantie de l'absence de collision lors du suivi de la trajectoire. La figure 5.9, dans laquelle chaque image représente un instant différent, donne l'exemple d'une collision suite à un saut de localisation dû à une imprécision de la carte, dans un cas réaliste. La trajectoire planifiée est une ligne droite sans collision avec les obstacles de la carte ou avec les autres obstacles perçus.

Figure du haut

Sur la figure du haut, l'abscisse c_i du début de la détection des collisions sur l'intervalle $[c_i, c_i + \Delta s^{check}]$ est représentée, ainsi que l'abscisse de la dernière localisation globale l_j. On suppose alors que le robot est parfaitement localisé et qu'il suit parfaitement sa trajectoire, c'est à dire que :

$$m_j.\hat{\mathbf{q}}_{odo}(s) = \mathbf{q}(s) = \mathbf{q}^{ref}(s)$$

La tâche **tCol** vérifie à cet instant la configuration $\mathbf{q}^{ref}(s_1^{check})$, qui n'est pas en collision.

Figure du milieu

Sur la figure du milieu, le robot a avancé, et il se trouve maintenant à l'abscisse l_{j+1}, à laquelle une nouvelle localisation globale a lieu. Du fait de l'imprécision du plan, la localisation globale produit une estimation de la configuration qui apparaît comme erronée, mais qui est cohérente avec la carte : le segment perçu en haut est bien mis en correspondance avec le segment de la carte. La vraie configuration du robot ne change pas et reste donc $m_j.\hat{\mathbf{q}}_{odo}(l_{j+1})$. Du fait de cette discontinuité dans la localisation du robot ($m_j \neq m_{j+1}$), la trajectoire de référence envoyée au système (Σ) est discontinue :

$$d_{\mathcal{C}}(m_j^{-1}.\mathbf{q}^{ref}(l_{j+1}),\ m_{j+1}^{-1}.\mathbf{q}^{ref}(l_{j+1})) = D^{loc}$$

Le système va donc effectuer une trajectoire de rattrapage pour ramener $\hat{\mathbf{q}}$ vers la trajectoire de référence $\mathbf{q}^{ref}$. On peut remarquer qu'à cet instant l'obstacle carré perçu (qui n'est pas dans la carte, mais cela est sans conséquence) est en collision avec la trajectoire de référence, mais la tâche **tCol** est alors en train de vérifier la configuration $\mathbf{q}^{ref}(s_2^{check})$. Cet obstacle s'est déplacé entre la figure du haut et la figure du milieu, car la transformation permettant d'exprimer la position des obstacles dans le plan à partir de leur position exprimée relativement au robot ($\hat{\mathcal{O}}_{loc}$) est passée de m_j à m_{j+1}.

Figure du bas

Sur la figure du bas, la tâche **tCol** vérifie actuellement la configuration $\mathbf{q}^{ref}(s_3^{check})$, mais il est déjà trop tard, le robot est entré en collision avec l'obstacle carré en cherchant à rattraper la trajectoire de référence.

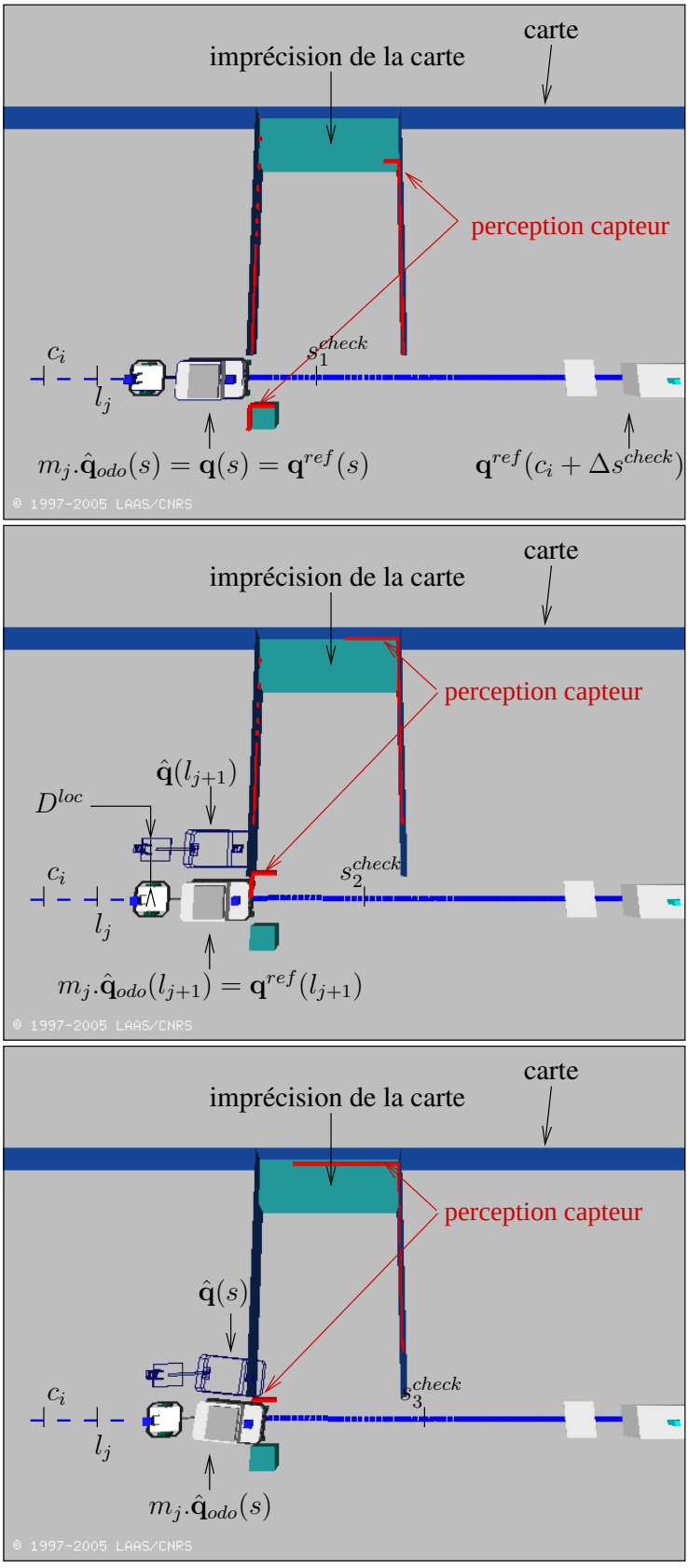

FIG. 5.9 – Collision suite à un saut de localisation dû à une imprécision de la carte. Voir le texte.

Bilan Deux hypothèses de la propriété 11 ont donc été violées ici. Tout d'abord l'hypothèse 2, car le système de commande du chariot à remorque n'est pas exponentiellement stable. Ensuite l'hypothèse 9, car le saut de localisation D^{loc} est supérieur à la distance minimale autorisée aux obstacles D^{clear}. C'est d'ailleurs pour cette dernière raison que sur la figure du milieu, après localisation, la trajectoire de référence apparaît en collision avec les obstacles perçus.

En résumé, dans cet exemple la collision provient du fait que la trajectoire suivie par le robot après la localisation, la trajectoire de rattrapage, n'a pas été testée. C'est ce défaut que nous allons corriger dans la section suivante afin de garantir un suivi de trajectoire sans collision.

5.4 Architecture pour l'intégration du suivi de trajectoire, de l'évitement d'obstacles et de la localisation

Dans cette section nous proposons une architecture de suivi de trajectoire robuste aux discontinuités de la localisation, pour les systèmes ne satisfaisant pas la propriété de stabilité exponentielle. C'est à dire que nous proposons une solution au suivi de trajectoire compte tenu des spécificités de notre approche.

Le principe de cette architecture est de simuler le rattrapage de la trajectoire de référence suite à une discontinuité de localisation et de repousser son application plus loin sur la trajectoire, de façon à ce que la trajectoire suivie par le système soit toujours vérifiée avant d'être exécutée.

5.4.1 Module de simulation du système

Pour prédire les effets des discontinuités de la trajectoire de référence envoyée au système (Σ), on se dote d'un module de simulation noté $(\bar{\Sigma})$. Ce système prend en entrée une trajectoire de référence $(\mathbf{q}^{ref}, \mathbf{u}^{ref})$. L'état et la sortie du système $(\bar{\Sigma})$ sont identiques et sont notés $\bar{\mathbf{q}}$. De plus le système $(\bar{\Sigma})$ produit la commande associée à la trajectoire $\bar{\mathbf{q}}$, que l'on note $\bar{\mathbf{u}}$. La figure 5.10 donne le diagramme de ce système de simulation.

Le système $(\bar{\Sigma})$ satisfait la propriété de convergence suivante :

Propriété 12. $\forall \epsilon > 0, \forall \mu > 0,\ \exists T > 0$ *tel que*

$$\textit{si :}\ d_{\mathcal{C}}(\bar{\mathbf{q}}(0), \mathbf{q}^{ref}(0)) < \mu$$

$$\textit{alors :}\ d_{\mathcal{C}}(\bar{\mathbf{q}}(T), \mathbf{q}^{ref}(T)) < \epsilon$$

et la trajectoire $(\bar{\mathbf{q}}, \bar{\mathbf{u}})$ *est admissible.*

Trajectoire de rattrapage

Partant de l'état initial $\bar{\mathbf{q}}(0)$, le système $(\bar{\Sigma})$ produit une trajectoire de rattrapage admissible $(\bar{\mathbf{q}}, \bar{\mathbf{u}})$ vers la trajectoire de référence $(\mathbf{q}^{ref}, \mathbf{u}^{ref})$ lorsque son entrée est cette trajectoire de référence. C'est cette propriété que nous utilisons pour calculer une trajectoire de rattrapage.

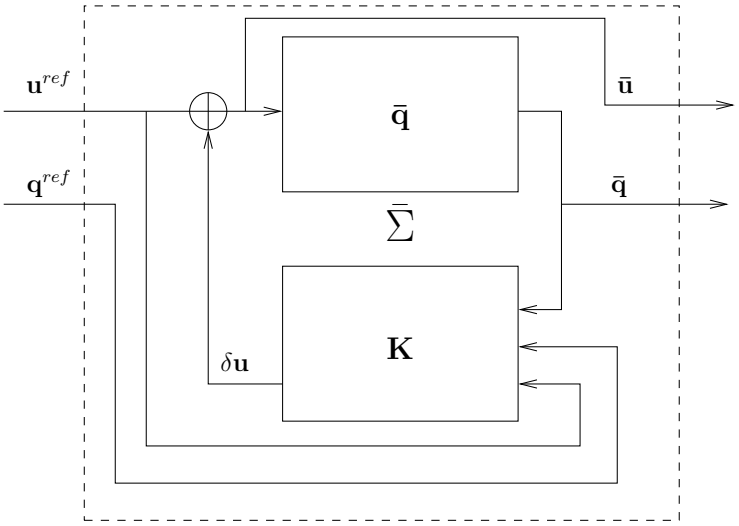

FIG. 5.10 – Module de simulation $(\bar{\Sigma})$ du système (Σ). L'entrée est une trajectoire de référence $(\mathbf{q}^{ref}, \mathbf{u}^{ref})$ non nécessairement admissible. La sortie est une trajectoire admissible $(\bar{\mathbf{q}}, \bar{\mathbf{u}})$.

Étant donné $\epsilon > 0$ fixé et $(\mathbf{q}^{ref}, \mathbf{u}^{ref})$ une trajectoire définie sur $[0, S]$, on note alors RATTRAP $(\bar{\mathbf{q}}(0), \mathbf{q}^{ref}, \mathbf{u}^{ref})$ la fonction qui renvoie la trajectoire $(\bar{\mathbf{q}}, \bar{\mathbf{u}})$ de rattrapage entre $\bar{\mathbf{q}}(0)$ et $\mathbf{q}^{ref}$ sur l'intervalle $[0, \bar{S}]$, tel que $d_{\mathcal{C}}(\bar{\mathbf{q}}(\bar{S}), \mathbf{q}^{ref}(\bar{S})) < \epsilon$.

$$(\bar{\mathbf{q}}, \bar{\mathbf{u}}, \bar{S}) = \text{RATTRAP}(\bar{\mathbf{q}}(0), \mathbf{q}^{ref}, \mathbf{u}^{ref}) \tag{5.23}$$

5.4.2 Architecture pour le suivi de trajectoire

Dans l'architecture de suivi de trajectoire présentée dans cette section, la tâche **tCol** est comme auparavant en charge de l'évitement des collisions, et elle a en plus pour fonction de gérer les effets de la localisation globale. À cette fin on ajoute dans la définition d'une trajectoire $(\mathbf{q}^{ref}, \mathbf{u}^{ref})$ définie sur l'intervalle $[0, S]$ la transformation solide $m(s)$ telle que : $\forall s \in [0, S]$

$$\mathbf{q}^{ref}_{odo}(s) = m(s)^{-1}\mathbf{q}^{ref}(s)$$

où $\mathbf{q}^{ref}_{odo}(s)$ est la trajectoire de référence envoyée au système (Σ), de manière similaire à l'équation (5.21). La figure 5.11 présente le diagramme de cette architecture. On peut remarquer que la donnée $\mathbf{traj}$, qui contient maintenant $m(s)$, est partagée par les tâches **tSuivi** et **tCol**, mais encore une fois l'accès en écriture par **tCol** ne s'effectue que sur l'intervalle $[s^{stop} - \Delta s^{check}, S]$, alors que l'accès en lecture par **tSuivi** s'effectue sur l'intervalle $[s, s^{stop} - \Delta s^{check}]$.

L'algorithme 5 décrit la tâche **tCol** qui est maintenant en charge de mettre à jour la transformation solide $m(s)$ en fonction des localisations globales réalisées par la tâche **tLoc**. La tâche **tCol** doit aussi calculer une trajectoire de rattrapage avec la fonction RATTRAP. Une première localisation globale a lieu à l'initialisation, telle que $\hat{\mathbf{q}}(0) = \mathbf{q}^{ref}(0)$. La transformation m_0, telle que $\hat{\mathbf{q}} = m_0.\hat{\mathbf{q}}_{odo}(0)$, est utilisée pour initialiser $m(s)$ sur l'intervalle $[0, S]$.

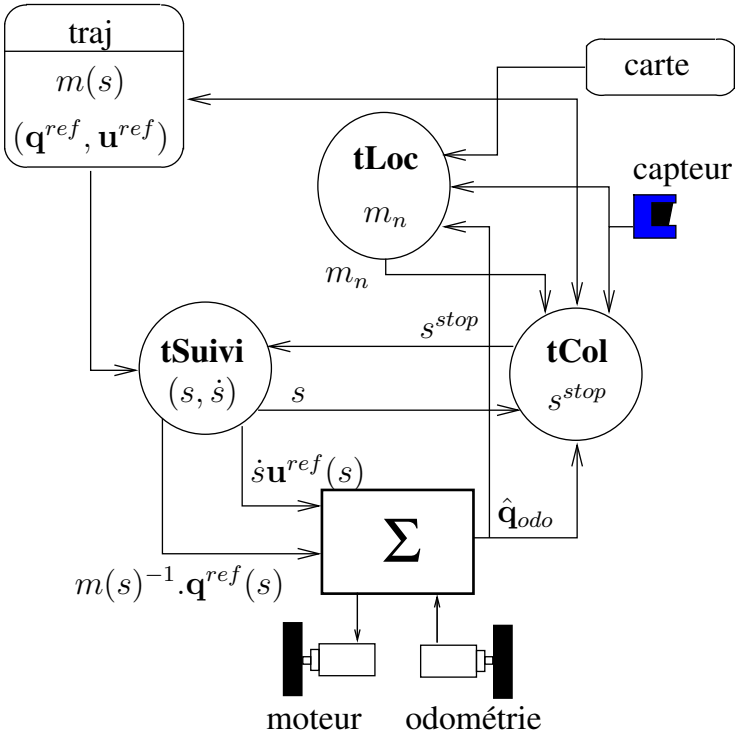

FIG. 5.11 – Architecture pour le suivi de trajectoire sécurisée.

Algorithme 5 tCol avec gestion de la localisation globale

Entrée : $s, \mathbf{traj}, \hat{\mathbf{q}}_{odo}, \hat{\mathcal{O}}_{loc}, m_n$
Sortie : $\mathbf{traj}, s^{stop}$
Initialisation : $\forall s \in [0, S], \quad m(s) \leftarrow m_0,$
loop
  Détection des collisions sur $[s, s + \Delta s^{check}]$
  $s^{stop} \leftarrow$ CHECK $(m_n.m(s)^{-1}.\mathbf{q}^{ref}(s), m_n.\hat{\mathbf{x}}_{odo}(\hat{\mathcal{O}}_{loc}))$
  Calcul de la trajectoire de rattrapage
  $\bar{\mathbf{q}}_{s^{stop}} \leftarrow m_n.m(s^{stop})^{-1}.\mathbf{q}^{ref}(s^{stop})$
  $(\bar{\mathbf{q}}, \bar{\mathbf{u}}, \bar{S}) \leftarrow$ RATTRAP$\big(\bar{\mathbf{q}}_{s^{stop}},\ \mathbf{q}^{ref}(s^{stop}),\ \mathbf{u}^{ref}(s^{stop})\big)$
  if $\bar{S} \leq S$ **then**
    $\forall s \in [s^{stop}, \bar{S}], \quad \mathbf{q}^{ref}(s) \leftarrow \bar{\mathbf{q}}(s)$
    $\forall s \in [s^{stop}, S], \quad m(s) \leftarrow m_n$
  end if
  Déformation de la trajectoire sur $[s^{stop} - \Delta s^{col}, s^{stop} + \Delta s^{col}]$
  if $s^{stop} \leq (s + \Delta s^{check})$ **then**
    $\mathbf{q}^{ref}(s) \leftarrow$ DEFORME$(m_n.m(s)^{-1}.\mathbf{q}^{ref}(s),\ m_n.\hat{\mathbf{x}}_{odo}(\hat{\mathcal{O}}_{loc}))$
  end if
end loop

Cet algorithme ramène la problématique du suivi de trajectoire avec localisation globale dans le cadre des hypothèses de la propriété 10, qui garantit l'absence de collisions lors du suivi de trajectoire sans localisation. Il assure en effet que :

– la trajectoire $\mathbf{q}_{odo}^{ref}(s) = m(s)^{-1}.\mathbf{q}^{ref}(s)$ envoyée au système (Σ) est continue et admissible,

– la trajectoire $\mathbf{q}^{ref}(s)$ exécutée par le robot est toujours vérifiée avant d'être exécutée,

À la fin de l'exécution le robot atteint la configuration finale désirée ($\hat{\mathbf{q}}(S) = \mathbf{q}^{ref}(S)$), à l'erreur d'odométrie D^{odo} et de suivi de trajectoire D^{suivi} près.

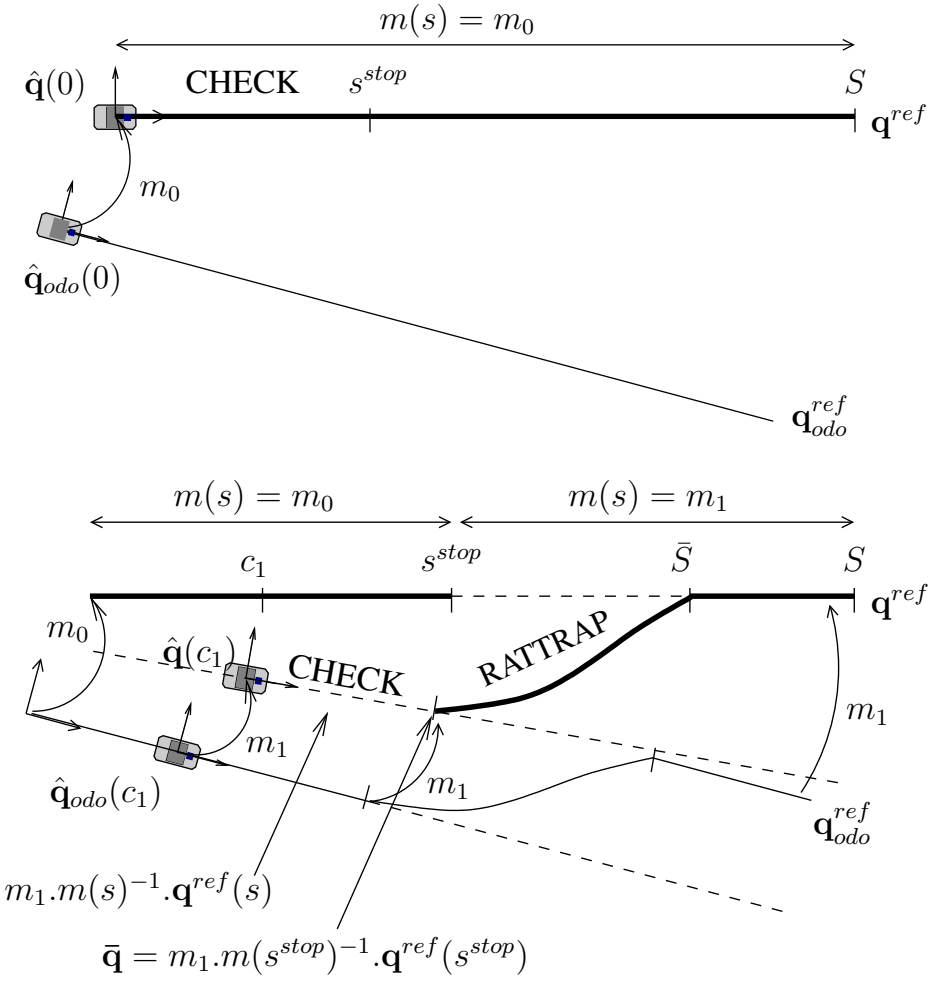

FIG. 5.12 – Les calculs de la tâche **tCol** pour garantir l'absence de collision. La ligne en gras est la trajectoire de référence $\mathbf{q}^{ref}(s)$. La ligne d'épaisseur standard est $\mathbf{q}_{odo}^{ref}(s)$, la trajectoire envoyée au système (Σ). La détection des collisions s'effectue sur la trajectoire que l'on considère comme réellement exécutée par le robot à partir des dernières informations de localisation disponibles. Le rattrapage calculé par la fonction RATTRAP assure que $\mathbf{q}_{odo}^{ref}(s)$ est continue.

La figure 5.12 illustre le comportement de l'algorithme 5 lors de l'exécution d'une trajectoire. À chaque abscisse s le long de la trajectoire, la configuration de référence envoyée au système (Σ) est $\mathbf{q}_{odo}^{ref}(s) = m(s)^{-1}\mathbf{q}^{ref}(s)$. Sur la figure du haut le robot est à l'abscisse $s = 0$ sur la trajectoire

planifiée et la transformation solide initiale vaut m_0. La tâche **tCol** détecte alors les éventuelles collisions sur la trajectoire jusqu'à l'abscisse s^{stop}.

Sur la figure du bas, le robot est à l'abscisse c_1 et une localisation globale a eu lieu à une abscisse non précisée située entre 0 et c_1. La localisation globale produit une discontinuité : $m_1 \neq m_0$. La tâche **tCol** vérifie alors jusqu'à une nouvelle abscisse s^{stop} la trajectoire considérée à partir de ces nouvelles informations de localisation comme celle réellement exécutée par le robot : $m_1.m(s)^{-1}.\mathbf{q}^{ref}(s)$. Ensuite elle calcule une trajectoire de rattrapage entre $\bar{\mathbf{q}}$ et $\mathbf{q}^{ref}$, à l'aide de la fonction RATTRAP. La transformation solide $m(s)$, qui transforme $\mathbf{q}^{ref}_{odo}$ en $\mathbf{q}^{ref}$, est mise à jour avec la nouvelle transformation m_1, sur l'intervalle $[s^{stop}, S]$.

5.5 Conclusion

Dans ce chapitre, nous avons abordé le problème de l'intégration de plusieurs fonctionnalités de la navigation autonome au sein d'une architecture unifiée. Plus précisément on s'est intéressé à la réalisation simultanée du suivi de trajectoire, de l'évitement réactif d'obstacles et de la localisation globale.

Nous avons montré que les spécificités de notre approche, c'est à dire la navigation de véhicules articulés à proximité des obstacles, nécessitaient le développement d'une architecture spécifique. Les éléments clés sont :

- l'absence de décroissance uniforme de la distance à la trajectoire de référence lors du suivi de trajectoire,
- la forme quelconque des trajectoires qui exclue des approximations simples pour le calcul des distances de freinage,
- la navigation à proximité des obstacles.

Pour ces raisons, certaines hypothèses habituelles pour garantir l'absence de collision (marge de sécurité, erreur de localisation faible par rapport à ces marges, etc.) ne sont pas applicables à notre étude.

Nous avons donc proposé une architecture qui intègre ces différentes fonctionnalités et qui tienne compte de ces spécificités. L'implémentation de cette architecture doit permettre de réaliser des mouvements longs avec une localisation globale. Ainsi même en naviguant à proximité des obstacles, les discontinuités de la localisation n'entraînent pas de collision, car leur effet est reporté afin d'être pris en compte dans l'évitement réactif d'obstacles.

Chapitre 6

Résultats expérimentaux

6.1 Introduction

Dans ce chapitre nous présentons quelques résultats expérimentaux de navigation autonome obtenus sur des robots réels. Mais avant tout, il nous paraît important de motiver l'expérimentation sur des systèmes réels, car elle constitue une part importante de notre travail.

Motivation de l'expérimentation L'expérimentation en robotique mobile autonome consiste tout d'abord à implémenter sur un système informatique, lui même embarqué sur un système mécanique, les méthodes théoriques résolvant les différentes fonctionnalités de la navigation. Il s'agit ensuite de définir une mission de navigation que le robot doit exécuter, et d'évaluer dans quelle mesure le robot remplit sa mission. L'intégration des différentes fonctionnalités et leurs interconnexions, une problématique que nous avons abordée au chapitre 5, rendent parfois délicate l'évaluation indépendante de chaque fonctionnalité.

L'expérimentation permet de «valider» une méthode censée résoudre une fonctionnalité de la navigation, ou une architecture qui intègre plusieurs méthodes. Mais une expérience, même répétée et couronnée de succès, ne laisse rien présager de la «véracité» d'une théorie ou d'un modèle. Par «valider» nous entendons plutôt que l'expérimentation démontre deux propriétés d'une méthode : la faisabilité de son implémentation, et sa capacité à résoudre des problèmes réels.

En effet l'implémentation d'une méthode qui a été développée à partir du modèle mathématique d'un phénomène physique (évitement d'obstacle, localisation, etc.) se heurte bien souvent à des problèmes de complexité algorithmique (et de temps de calcul prohibitifs associés) et d'intégration dans une architecture temps-réel.

Quant à la capacité d'une méthode à résoudre des problèmes réels, c'est un point fondamental que l'expérimentation permet d'étudier. C'est en effet lors d'expériences réelles que vont apparaître les «points durs», d'un point de vue pratique, d'un problème, qui ne correspondent pas toujours aux problèmes théoriques résolus par la méthode. À titre d'exemple, on peut citer les méthodes de planification de trajectoire basées sur une exploration aléatoire de l'espace des configurations : les cas difficiles pour ces techniques sont les passages étroits, et ce sont

précisément les cas que l'on rencontre le plus souvent dans la pratique.

L'expérimentation est donc un élément logique de la boucle : observation d'un problème réel, modélisation mathématique, implémentation, expérimentation, observation du résultat. Ainsi, de nouveaux problèmes sont généralement soulevés par des expériences. Mais l'expérimentation sur des systèmes réels demande beaucoup de temps, et il est donc nécessaire de se doter d'environnements de simulation réalistes pour préparer les expériences.

Enfin, mais on quitte là le domaine de la recherche pour entrer dans celui de la valorisation de la recherche, l'expérimentation est un formidable outil de communication qui permet de valoriser une méthode ou une théorie, en démontrant sa capacité à résoudre des problèmes réels en vue d'applications pratiques. Mais tous les domaines ne sont pas sur un même pied d'égalité de ce point de vue : certains domaines donnent lieu à des développements théoriques complexes mais à des applications peu spectaculaires, quand d'autres trouvent des applications qui reflètent bien la complexité du problème abordé.

Présentation des expérimentations Tout d'abord nous présentons l'architecture logicielle qui implémente les différentes fonctionnalités de la navigation autonome. Ensuite nous donnons quelques résultats d'évitement réactif d'obstacles, obtenus avec différents robots. Nous présentons ensuite la méthode de parking référencé sur des amers dans un cas réaliste avec un robot de type chariot à remorque. Enfin nous présentons une expérience qui traite de l'intégration de la localisation globale dans le suivi de trajectoire, une problématique que nous avons exposée au chapitre 5.

6.2 Architecture logicielle

Nous présentons ici l'architecture logicielle qui implémente les différentes fonctionnalités de la navigation autonome. Nous avons contribué au développement de certains modules, représentés en gras sur la figure 6.1.

On remarque que cette architecture logicielle s'abstrait de la couche matérielle. Cela est rendu possible par l'utilisation de l'environnement de programmation GenoM (Générateur de Modules [Fleury 1997]). Chaque module (représenté par un rectangle sur la figure 6.1), est écrit en langage C et implémente une fonctionnalité. GenoM gère alors les éléments suivants :

- la compilation spécifique pour le système informatique embarqué sur le robot,
- les communications et le partage de ressources entre les différents modules,
- la synchronisation temporelle des différents modules,

SICK Dans nos applications, les obstacles sont détectés par un télémètre laser SICK, qui fournit la distance aux points les plus proches dans un plan horizontal. On peut extraire des segments à partir de ces données. Le module **SICK** implémente ces fonctionnalités.

SEGLOC Le module **SEGLOC** calcule la position «globale» du robot en mettant en correspondance les segments du module **SICK** avec une carte de segments de l'environnement.

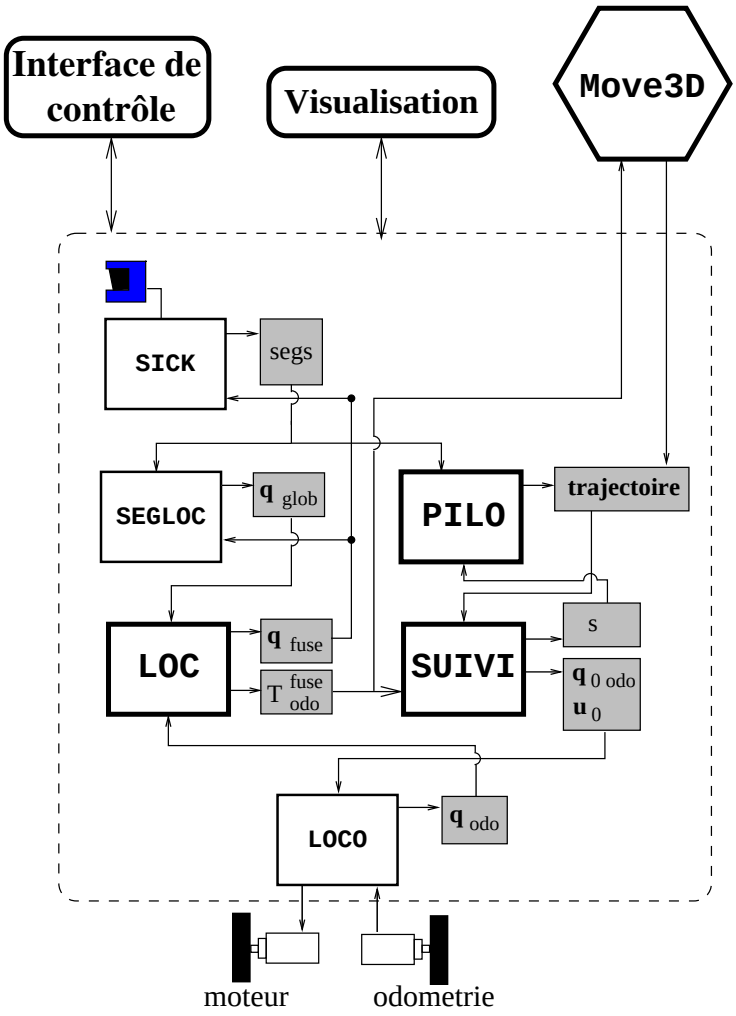

FIG. 6.1 – Architecture logicielle pour la navigation autonome

LOC Le module LOC fusionne les positions odométriques et la position calculée par SEGLOC.

LOCO Le module LOCO implémente la loi de commande en boucle fermée de suivi de trajectoire et envoie les commandes aux moteurs.

SUIVI Le module SUIVI gère la progression (abscisse s) du robot le long de la trajectoire, lui permettant de ralentir pour éviter un obstacle si la déformation de trajectoire n'est pas assez rapide par exemple. Il correspond à la tâche **tSuivi** décrite à la section 5.2.4.

PILO Le module PILO «pilote» le robot au cours de l'exécution de sa trajectoire. Il gère l'évitement réactif d'obstacles par la méthode décrite au chapitre 3 et implémente donc la tâche **tCol** présentée à la section 5.2.4. Mais il implémente en plus la tâche de parking référencé sur des amers décrite au chapitre 4.

Move3D Une couche supérieure dite décisionnelle permet de définir des tâches de plus haut niveau pour le robot. Dans notre étude, cette couche est principalement constituée par le logiciel de planification de trajectoire Move3D [Siméon 2001], qui calcule une trajectoire sans collision dans un modèle de l'environnement entre une configuration initiale et une configuration finale spécifiée.

Autres logiciels Nous avons développé une interface conviviale permettant de contrôler l'exécution des modules (démarrage, arrêt, réglage des paramètres) en langage Tcl/Tk.

Pour la visualisation du déroulement de la navigation, nous utilisons le logiciel GDHE développé au LAAS-CNRS. Nous avons réalisé quelques développements pour ce logiciel, permettant notamment d'afficher un modèle VRML du robot Cycab, ou encore de suivre en temps-réel le déroulement d'une tâche de parking référencé sur des amers.

6.3 Portage sur plusieurs robots

Cette architecture, initialement développée pour le robot Hilare2 avec sa remorque (voir la figure 4.5, page 67), a été «portée» sur le robot Dala et le Cycab de l'INRIA Rhône-Alpes (voir la figure 1.3, page 6).

Ce portage a nécessité deux types de développements : des modifications des programmes des modules GenoM et des adaptations des algorithmes. Nous précisons ici le travail effectué et les résultats obtenus.

6.3.1 Le rover Dala

Le rover Dala est équipé d'un processeur pentium à 2 GHz (au moment des expérimentations), et utilise le système d'exploitation Linux. Le portage sur le rover Dala de l'architecture

proposée pour la navigation autonome n'a pas posé de difficulté. En effet, seul le module LOCO, lié à la partie matérielle, était différent, et il avait déjà était développé pour ce robot.

Par contre il a fallu adapter la méthode d'évitement d'obstacles afin de limiter la courbure de la trajectoire lors du processus de déformation. Cette extension a été présentée à la section 3.5. Nous rappelons un résultat intéressant présenté à cette occasion : étant donné une trajectoire de référence dont la courbure dépasse une valeur maximale spécifiée notée κ_{max}, les itérations successives de la méthode de déformation de trajectoire génèrent des manoeuvres qui diminuent la courbure de la trajectoire (voir la figure 3.15, page 49).

6.3.2 Le robot Cycab

Le portage sur le robot Cycab a nécessité un travail différent, car ce robot n'était pas sur le site du LAAS-CNRS à Toulouse où nous avons effectué notre thèse. En effet le robot Cycab sur lequel nous avons travaillé se trouve à l'INRIA Rhône-Alpes, et le portage à été réalisé dans le cadre du projet ROBEA ParkNav (*Interprétation de scène dynamique complexe et planification réactive de mouvement*). Ce robot possédait déjà une architecture pour la navigation autonome (utilisant le formalisme de la programmation Bayésienne orientée objets [Pradalier 2004]), sous le système d'exploitation RT-Linux, et seuls les modules PILO, SUIVI et le logiciel Move3D ont été portés, car la contribution du LAAS-CNRS résidait dans l'apport d'une méthode originale d'évitement d'obstacles.

Le développement et l'utilisation d'un environnement de simulation a été un élément fondamental dans la réussite de cette partie du projet. Cédric Pradalier, avec qui nous avons collaboré, a tout d'abord effectué un séjour de trois jours au LAAS pour présenter son environnement de simulation du Cycab. Puis nous avons réalisé une interface de communication entre les modules GenoM et le Cycab. Enfin deux séjours de trois jours à l'INRIA Rhône-Alpes nous ont permis de réaliser toutes nos expérimentations sur le robot.

Ce travail a mis en valeur la généricité de l'architecture proposée pour la navigation et de la méthode de déformation de trajectoire pour systèmes nonholonomes. La seule donnée du modèle cinématique du système permet d'appliquer la méthode. Encore une fois nous avons utilisé l'extension de la méthode de déformation de trajectoire de façon à respecter l'angle de braquage maximal des roues avant (de 0.35 radians).

Évitement d'obstacle sur un parking Nous avons réalisé une expérience qui valide l'intégration de la méthode d'évitement réactif d'obstacles sur le robot Cycab, dans un contexte de navigation dans un parking (le contexte du projet ParkNav). Le cadre expérimental est le suivant : une trajectoire a été planifiée afin d'amener le robot d'une place de parking à une autre, et un obstacle absent du modèle utilisé pour la planification est en collision avec cette trajectoire. La figure 6.2 présente les résultats obtenus. L'obstacle est détecté par le capteur SICK, et la trajectoire est déformée de façon à l'éviter tout en gardant la courbure de la trajectoire en deçà de sa valeur maximale autorisée.

Lors de cette expérience, la méthode d'optimisation des calculs des interactions entre le robot et les obstacles présentée à la section 3.4.2, n'avait pas encore été implémentée. Aussi le

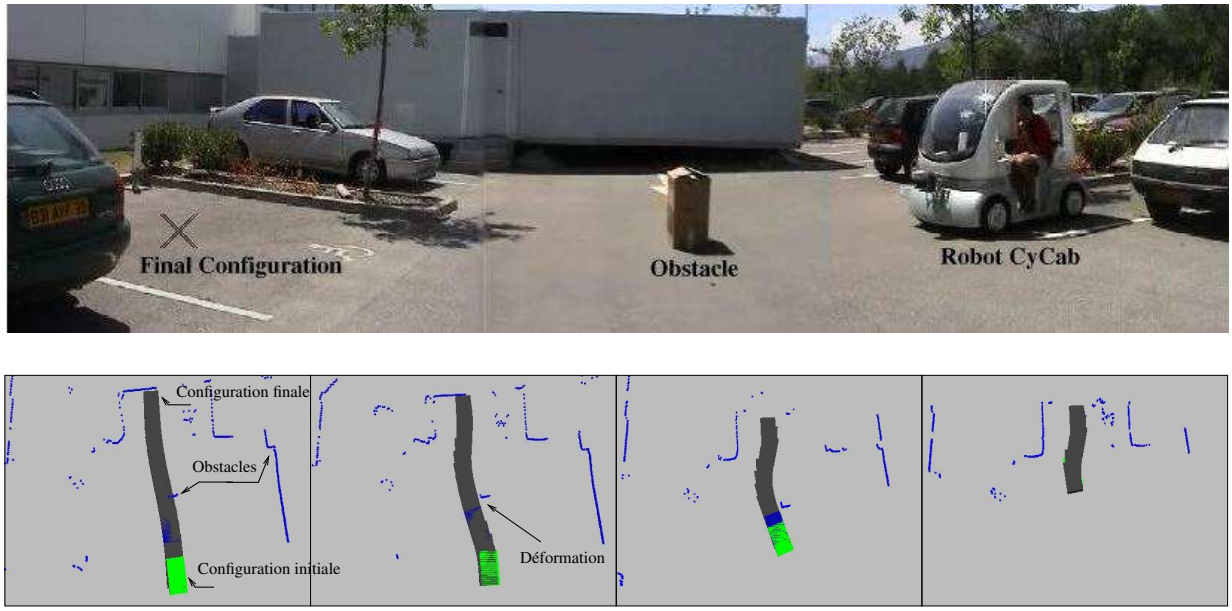

FIG. 6.2 – Expérimentation avec le robot Cycab. Sur la figure du haut, le cadre expérimental : une trajectoire est planifiée, depuis une place de parking vers une autre, mais un obstacle non modélisé se trouve sur cette trajectoire. Sur la figure du bas, on observe les déformations de la trajectoire au cours de son exécution.

processus de déformation de trajectoire était très coûteux en temps de calcul, et le robot devait s'arrêter le long de sa trajectoire de référence pour avoir le temps de déformer la trajectoire lorsqu'une collision était détectée. C'est l'observation de ce phénomène indésirable qui a motivé le développement de cet algorithme d'optimisation.

6.4 Parking référencé sur des amers

Reprenant l'exemple développé dans la section 4.6, nous présentons ici les résultats obtenus lors d'expérimentations avec le robot Hilare2 et sa remorque.

La tâche de parking consiste ici à positionner la remorque dans un emplacement représentant un quai de débarquement de marchandise pour des camions, matérialisé ici par un mur et des cubes (voir la figure 6.3). Le motif de parking est constitué de trois segments perçus par un télémètre laser SICK situé sur la remorque. Le robot Hilare2 possède 1 carte PPC et 4 cartes M68K reliées par un bus de communication VME. Il utilise le système d'exploitation temps-réel VxWorks. Du fait de la faible puissance de calcul de ces cartes, un ordinateur portable pentium à 2 GHz est utilisé pour le module PILO. La position de la remorque est mesurée par un codeur optique situé sur le point d'attache entre la remorque et le robot.

6.4.1 Parking référencé sur des amers avec une localisation imprécise

Étant donné un modèle de l'environnement, une trajectoire est planifiée de façon à amener la remorque au centre du quai de débarquement. Dans cette expérience, le quai de débarquement matérialisé par deux boîtes accolées à un mur, est positionné dans la scène réelle aussi fidèlement que possible par rapport au modèle de l'environnement.

Cependant l'estimation de la configuration courante du robot est imprécise, comme le montre la figure 6.3 : la perception de l'environnement est décalée par rapport au modèle. Si le robot exécute sa trajectoire en boucle ouverte, il est manifeste qu'il n'atteindra pas la position désirée par rapport au quai de débarquement, et que des collisions vont se produire. Le robot doit donc adapter sa trajectoire afin de rejoindre la configuration définie par le motif de parking tout en évitant les obstacles. Les images du bas de la figure 6.3 représentent les déformations successives de la trajectoire au cours de son exécution.

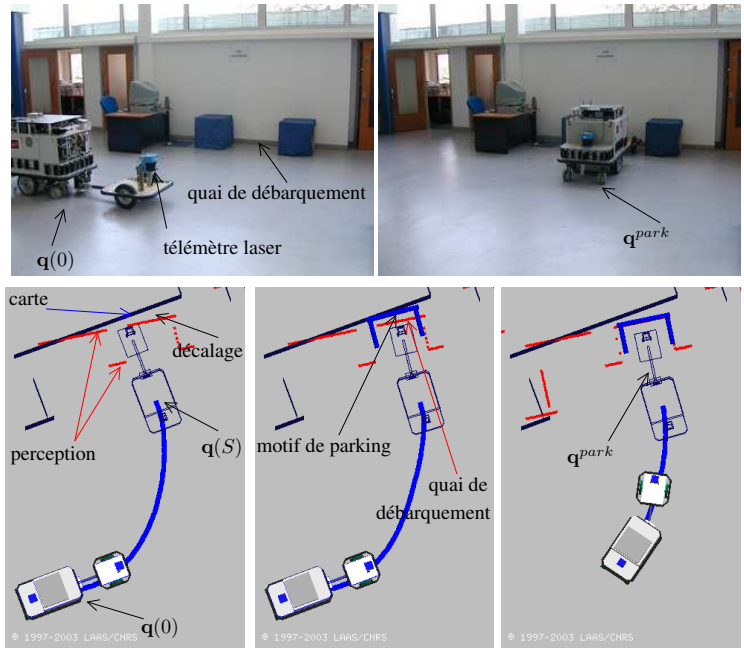

FIG. 6.3 – Tâche de parking dans un quai de débarquement avec une localisation imprécise

6.4.2 Parking référencé sur des amers avec un motif de parking inexact

Dans cette expérience la localisation du robot serait assez précise pour réaliser la tâche de parking en boucle ouverte. Cependant les boîtes matérialisant le motif de parking ne sont pas exactement dans la même position que dans le modèle de l'environnement. Elles ont été déplacées vers la droite et l'écart entre elles est supérieur à la valeur du modèle : le motif de parking est inexact par rapport à la réalité. Ainsi si la trajectoire planifiée était exécutée sans tenir compte de ce changement, la configuration finale serait en collision avec le quai de débarquement, comme le montre la figure 6.4.

Le robot doit donc déformer la trajectoire planifiée afin que la configuration finale soit positionnée «au mieux» par rapport aux amers perçus. Dans notre approche cela signifie que la position de parking $\mathbf{q}^{park}$ est la position de variance minimale, compte tenu des incertitudes de la localisation, du modèle et de la perception et étant donné la différence entre le modèle et la perception. Les images du bas de la figure 6.4 représentent les déformations successives de la trajectoire au cours de son exécution.

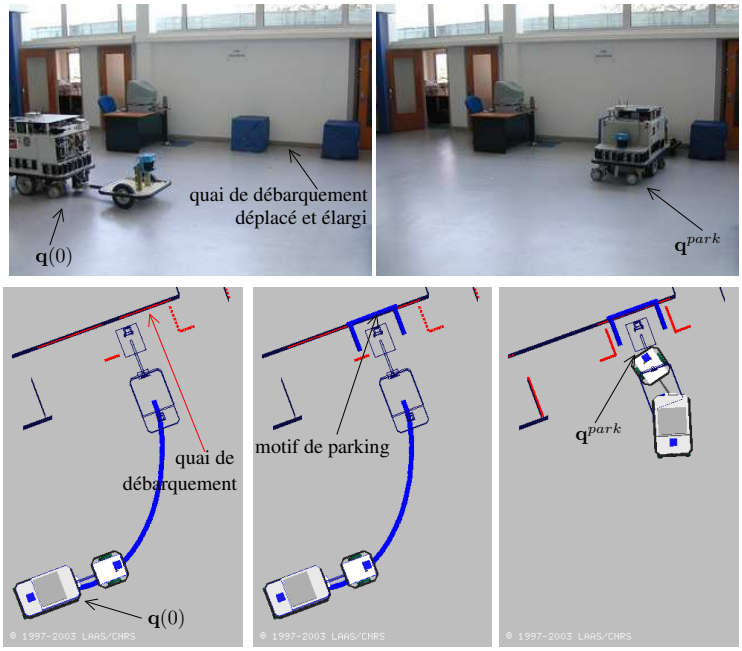

FIG. 6.4 – Tâche de parking dans un quai de débarquement à un emplacement différent et avec une forme modifiée par rapport au modèle

6.4.3 Bilan

Dans ces deux expériences, nous avons obtenu de très bons résultats quantitatifs. La remorque se positionne à la distance désirée des amers, avec une erreur de l'ordre de 5 cm. Cette erreur est plus marquée dans la direction transversale au mouvement, du fait de la convergence lente de la loi de commande dans cette direction.

Nous avons cependant relevé certains problèmes lors des expériences réelles. On a en effet remarqué que le robot devait parfois s'arrêter au cours de l'exécution de la trajectoire. Nous avons recensé deux causes à ce phénomène.

Tout d'abord le robot doit parfois s'arrêter pour déformer la trajectoire, au moment où un amer qui était caché devient visible. Par exemple, sur les deux images en bas à gauche de la figure 6.4, une partie du quai de débarquement n'est pas perçue, car elle est occultée par un côté de la boîte. La tâche de parking va donc considérer qu'un des amers du motif de parking est absent de l'environnement réel, et va aligner la remorque avec l'amer de droite, qui est perçu

entièrement (cette étape n'est pas représentée). Ce n'est que lorsque le robot sera assez proche de la configuration de parking (image en bas à droite de la figure 6.4) que cet amer occulté sera enfin révélé. Le robot devra alors recalculer une configuration de parking qui tienne compte de tous les amers, et sera contraint de s'arrêter car le temps de calcul de la déformation de la trajectoire n'est pas connu *a priori*.

Ensuite, nous avons conclu que la datation des données de perception est primordiale. Il faut en effet connaître avec précision à quel moment une perception a été réalisée pour la mettre en relation avec la position du robot lors de cette perception. Ceci devient crucial quand les sources de perception sont multiples. Dans nos expériences, le manque de précision sur cette information entraînait un phénomène de «déplacement» de la perception du capteur SICK avec le mouvement du robot : la perception n'était pas associée précisément à la position du robot lors de cette perception. En l'occurence, on peut imputer une grande part de ce décalage à la fréquence peu élevée des perceptions (5 Hz), le système informatique du robot Hilare2 ne permettant pas de traiter les données à une fréquence plus élevée. Ce type de problème illustre l'intérêt d'un système d'acquisition temps-réel tel que RTMAPS [Nashashibi 2000], utilisé par exemple pour la fusion de données de localisation à grande fréquence dans un contexte automobile [Abuhadrous 2003].

6.5 Intégration de la localisation globale, du suivi de trajectoire et de l'évitement d'obstacles

Nous présentons dans cette section les résultats de l'implémentation de l'architecture proposée à la section 5.4 pour l'intégration de la localisation globale, du suivi de trajectoire et de l'évitement d'obstacles, de façon à garantir l'absence de collision.

Comme nous l'avons montré dans le chapitre 5, l'intégration de la localisation globale dans l'architecture de suivi de trajectoire peut entraîner des collisions dès lors que le robot navigue proche des obstacles, en raison des discontinuités de la localisation et des spécificités de notre approche. Les expériences rapportées précédemment ont pourtant été réalisées avec cette architecture et ces défauts, et parfois même lors de longs mouvements utilisant la fonctionnalité de localisation globale. Mais les discontinuités de localisation n'étaient généralement pas suffisantes pour entraîner des collisions.

Suivi de trajectoire sans collision L'expérimentation que nous avons simulée reproduit le cas présenté à la section 5.3.4 et à la figure 5.9, qui conduisait à une collision. Les expériences que nous rapportons ici ont été réalisées en simulant la perception (le module SICK), la commande (le module LOCO) et l'environnement. Tous les autres modules sont les mêmes que ceux employés pour une expérimentation sur le système réel, c'est à dire le robot Hilare2 avec sa remorque. L'intérêt de présenter des résultats expérimentaux obtenus en simulation est d'isoler le problème qui nous intéresse, à savoir le rattrapage d'une trajectoire suite à une discontinuité de l'estimation de la position du robot.

La figure 6.5 illustre ainsi le rattrapage d'une trajectoire suite à une discontinuité de la localisation, dans un environnement de simulation. L'image de gauche représente la trajectoire

rectiligne planifiée dans l'environnement de simulation et la perception de cet environnement par le robot. Sur cette image, la localisation basée sur des amers (des segments de droite dans cette expérience) n'a pas encore pris en compte la perception de l'obstacle qui constitue une imprécision de la carte. En ce sens on peut considérer que le robot est mal localisé car sa perception n'est pas correctement apparié avec la carte.

L'image de droite présente la vision du robot quelques instants plus tard. La localisation globale a identifié la perception de l'imprécision de la carte à un amer de la carte et l'estimation de la position du robot représentée par son modèle filaire a subi un saut. Cette nouvelle estimation de la position n'est pas prise en compte immédiatement, et la perception relativement à la trajectoire planifiée reste. Le rattrapage de la trajectoire prenant en compte la discontinuité de la localisation a été calculé en «aval» sur la trajectoire, permettant à la tâche de détection des collisions de vérifier cette portion de trajectoire. Au final le robot suit une trajectoire continue et sans collision.

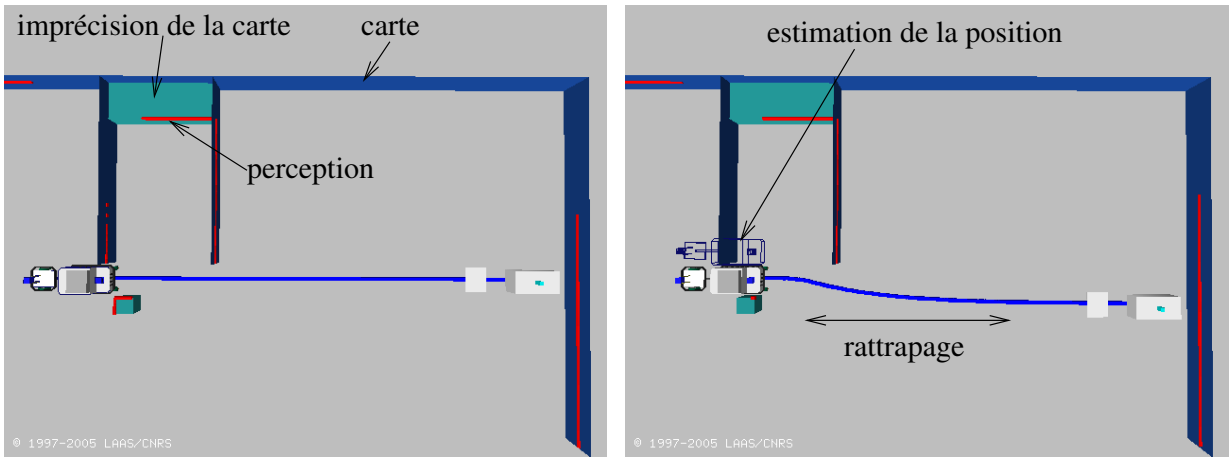

FIG. 6.5 – Suivi de trajectoire sans collision. Le rattrapage de la trajectoire n'est pas réalisé dès la discontinuité de la localisation, mais est reporté en «aval» sur la trajectoire.

6.5.1 Conclusion

Dans ce chapitre nous avons tout d'abord motivé l'expérimentation en robotique et rappelé les difficultés de sa mise en oeuvre. Nous avons présenté plusieurs expériences sur des robots différents, qui rendent compte de la généricité des méthodes développées.

Après avoir présenté l'architecture logicielle utilisée pour la navigation autonome, nous avons rapporté une expérience d'évitement réactif d'obstacle sur le robot Cycab. Nous avons ensuite illustré la méthode de parking référencé sur des amers par une expérience reproduisant l'amarrage d'un véhicule articulé à un quai de débarquement. Enfin nous avons présenté une expérience réalisée dans un environnement de simulation qui montre que l'architecture proposée pour l'intégration des fonctionnalités de suivi de trajectoire, d'évitement d'obstacles et de localisation permet de garantir l'absence de collision.

Chapitre 7

Conclusion

Notre travail traite du problème de la navigation autonome des robots mobiles dans un cadre particulier. Ce cadre est défini par les spécificités des systèmes considérés et des applications envisagées, que nous pouvons résumer par la navigation pour robot mobiles nonholonomes à proximité des obstacles.

Le fil conducteur de ce document est l'idée que ces spécificités posent des problématiques singulières pour la résolution des fonctionnalités de la navigation. Notre contribution porte sur le développement de méthodes permettant de résoudre certaines de ces fonctionnalités.

Évitement réactif d'obstacles pour systèmes nonholonomes Nous avons présenté une méthode d'évitement réactif d'obstacles pour systèmes nonholonomes, et avons proposé plusieurs extensions de cette méthode et des optimisations de son algorithme. Le principe de cette méthode est de perturber les entrées d'une trajectoire de référence de façon à ce que la trajectoire déformée s'éloigne des obstacles et respecte les contraintes cinématiques du système. Nous avons proposé un algorithme permettant de réduire la complexité du calcul des interactions entre la trajectoire et les obstacles. Nous avons aussi montré comment cette méthode de déformation de trajectoire peut être considérée comme une méthode d'optimisation de trajectoire, suivant d'autres critères que la distance aux obstacles.

Parking référencé sur des amers Nous avons proposé une méthode originale de parking référencé sur des amers pour systèmes nonholonomes. Le principe est de définir un motif de parking relativement à un capteur du robot, de repérer ce motif dans l'environnement et de déformer la trajectoire de façon à ce que la configuration finale rejoigne la configuration de parking. Cette méthode permet par exemple de réaliser une tâche de parking pour un robot avec remorque, en positionnant la remorque relativement à des amers de l'environnement. Le formalisme utilisé est indépendant du type de capteur, du type d'amer ou des contraintes cinématiques du système.

Intégration du suivi de trajectoire, de l'évitement réactif d'obstacles et de la localisation Nous avons montré que l'intégration des fonctionnalités de suivi de trajectoire, d'évitement réactif d'obstacles et de localisation globale constituait une problématique à part entière de la

navigation autonome. Nous avons identifié les difficultés de cette intégration compte tenu des spécificités de notre approche et avons proposé une architecture qui intègre ces fonctionnalités.

Résultats expérimentaux Enfin nous avons présenté des résultats expérimentaux obtenus avec différents robots évoluant dans des environnements variés : le robot Hilare2 avec une remorque, le rover Dala et le robot Cycab de l'INRIA Rhône-Alpes. Ces résultats montrent la maturité des méthodes proposées et leur capacité à résoudre des problèmes réels.

Perspectives

Plusieurs perspectives sont envisageables suite à nos travaux. Tout d'abord, nous avons mentionné la possibilité d'utiliser la méthode de déformation de trajectoire en tant que méthode numérique de planification de trajectoire pour des systèmes nonholonomes. Il faudrait pour cela étudier notamment la stabilité et la vitesse de convergence de cette technique. Il faudrait par ailleurs vérifier que cette méthode respecte la propriété de commandabilité et temps petit.

Ensuite, il serait intéressant d'introduire la «dynamique» dans notre étude de la navigation autonome. D'une part la dynamique de l'environnement en prenant en compte les obstacles mobiles, ce qui soulève des problèmes particulièrement délicats du fait des spécificités de notre étude (calcul d'une courbe de décélération, forme quelconque des trajectoires, difficulté de rattraper une trajectoire de référence, etc.). D'autre part la dynamique du système, en tenant compte de la masse du système et des forces physiques auxquelles il est soumis. Nous pourrions alors réaliser des mouvements à grande vitesse. Par ailleurs on pourrait envisager d'utiliser d'autres capteurs pour détecter les obstacles, tels que des caméras. On pourrait ainsi détecter une plus grande quantité et variété d'obstacles.

Enfin un lien pourrait être établi avec des travaux récents portant sur l'exécution de trajectoires référencées sur des amers [Malti 2005]. Le principe est de définir une trajectoire sensorimotrice comme un chemin géométrique et un ensemble d'amers perçus lors de l'exécution de ce chemin. Ainsi la trajectoire est exprimée relativement à l'environnement : passer au milieu d'un couloir, le long d'un mur, etc. Il serait intéressant d'intégrer cette technique à l'architecture pour la navigation autonome que nous avons proposée.

Annexe A

Preuve de l'absence de collision pour les systèmes respectant la propriété de stabilité exponentielle

Nous donnons ici la preuve de la propriété 11 page 90, qui assure que sous certaines hypothèses, le suivi de trajectoire avec localisation globale pour les systèmes exponentiellement stables au sens de la propriété 8 s'effectue sans collision. Nous énonçons de nouveau ces hypothèses.

Tout d'abord on rappelle qu'on note $c_i (i \geq 1)$, l'abscisse à laquelle le robot perçoit l'environnement avec son capteur afin de détecter les obstacles, et j le plus grand entier tel que $l_j \leq c_i$: l_j est l'abscisse de la dernière localisation, et m_j est la transformation correspondante, telle que définie à l'équation (5.19).

On suppose qu'entre les abscisses c_i et $c_i + \Delta s^{check}$, c'est à dire sur l'intervalle sur lequel sont détectées les collisions, la tâche **tLoc** effectue k localisations globales :

$$l_j \leq c_i < l_{j+1} < \cdots < l_{j+k} < c_i + \Delta s^{check}$$

Et entre 2 localisations globales d'abscisses l_p et l_{p+1}, la trajectoire de référence fournie en entrée du système (Σ) est $(m_p^{-1}\mathbf{q}^{ref}, \mathbf{u}^{ref})$.

Les hypothèses sont alors les suivantes :

1. **à l'abscisse c_i, l'estimation de la configuration du robot est dans un voisinage de la configuration de référence :**
$$d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(c_i), m_j^{-1}\mathbf{q}^{ref}(c_i)) \leq D^{loc}$$
2. **le système (Σ) est exponentiellement stable** avec $\mu \geq 2D^{loc}$, (propriété 8),
3. **la trajectoire de référence est détectée sans collision** :
$$\forall s \in [c_i, c_i + \Delta s^{check}] \quad d_O(\mathbf{q}^{ref}(s), \hat{\mathcal{O}}) > D^{clear}$$
4. **les obstacles détectés englobent les obstacles réels** (équations (5.13) et (5.22)) :
$$\mathcal{O} \subset \mathbf{x}(c_i)\hat{\mathbf{x}}^{-1}(c_i)(\hat{\mathcal{O}}) = \mathbf{x}(c_i)\hat{\mathbf{x}}_{odo}^{-1}(c_i)m_j^{-1}(\hat{\mathcal{O}})$$

5. **l'accroissement d'erreur odométrique entre c_i et s est bornée par** D^{odo} pour tout $s \in [c_i, c_i + \Delta s^{check}]$,
6. **les variables de configurations internes sont parfaitement mesurées** :
   $\hat{\mathbf{q}}_{odo}(s) = (\hat{\mathbf{x}}_{odo}(s), \mathbf{q}_{int}(s))$
7. **les discontinuités de localisation sont bornées** :
   $\forall p, q, j \leq p, q \leq j + k$ et $\forall s \in [c_i, c_i + \Delta s^{check}]$,
   $$d_{\mathcal{C}}(m_p^{-1}\mathbf{q}^{ref}(s), m_q^{-1}\mathbf{q}^{ref}(s)) \leq D^{loc}$$
8. **les abscisses de localisations sont suffisamment éloignées pour que le système ait le temps de converger vers la trajectoire de référence** :
   $\forall p, j \leq p \leq j + k - 1$
   $$d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(l_{p+1}), m_p^{-1}\mathbf{q}^{ref}(l_{p+1})) \leq \frac{1}{2} d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(l_p), m_p^{-1}\mathbf{q}^{ref}(l_p))$$
9. $D^{odo} + 3D^{loc} < D^{clear}$

Alors sous ces hypothèses, la propriété 11 assure que le robot n'est pas en collision lors du suivi de la trajectoire de référence :

$$\forall s \in [c_i, c_i + \Delta s^{check}], \qquad d_O(\mathbf{q}(s), \mathcal{O}) > 0$$

Démonstration. La preuve se décompose en deux parties, la première étant très similaire à la preuve de la propriété 10, page 86.

Première partie En appliquant la transformation solide $\mathbf{x}(c_i)\hat{\mathbf{x}}_{odo}^{-1}(c_i)$ à l'inégalité de l'hypothèse 3, on obtient $\forall s \in [c_i, c_i + \Delta s^{check}]$:

$$d_O(\mathbf{x}(c_i)\hat{\mathbf{x}}^{-1}(c_i)\mathbf{q}^{ref}(s), \mathbf{x}(c_i)\hat{\mathbf{x}}^{-1}(c_i)(\hat{\mathcal{O}})) > D^{clear}$$

Ce qui implique d'après l'hypothèse 4 que pour tout $s \in [c_i, c_i + \Delta s^{check}]$

$$d_O(\mathbf{x}(c_i)\hat{\mathbf{x}}^{-1}(c_i)\mathbf{q}^{ref}(s), \mathcal{O}) > D^{clear} \tag{A.1}$$

De façon similaire à la preuve de la propriété 10, la définition de $d_{SE(2)}$ et les hypothèses 5 et 6 entraînent :

$$\begin{aligned} d_{\mathcal{C}}(\mathbf{x}(c_i).\hat{\mathbf{x}}_{odo}^{-1}(c_i).\hat{\mathbf{q}}_{odo}(s), \mathbf{q}(s)) &= d_{SE(2)}(\mathbf{x}(c_i).\hat{\mathbf{x}}_{odo}^{-1}(c_i).\hat{\mathbf{x}}_{odo}(s), \mathbf{x}(s)) \\ &= d_{SE(2)}(\hat{\mathbf{x}}_{odo}^{-1}(c_i).\hat{\mathbf{x}}_{odo}(s), \mathbf{x}^{-1}(c_i).\mathbf{x}(s)) \leq D^{odo} \end{aligned} \tag{A.2}$$

Seconde partie La seconde partie de la preuve exploite la stabilité exponentielle du système (Σ). La trajectoire de référence $(m_p^{-1}\mathbf{q}^{ref}, \mathbf{u}^{ref})$ envoyée au système est admissible par morceau sur chaque intervalle $[l_p, l_{p+1}]$ avec p compris entre j et $j+k-1$. L'hypothèse 1 et la stabilité exponentielle entraînent que pour tout $s \in [c_i, l_{j+1}]$,

$$d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(s), m_j^{-1}\mathbf{q}^{ref}(s)) \leq D^{loc} \tag{A.3}$$

À l'abscisse l_{j+1}, une nouvelle localisation globale a lieu, et la trajectoire de référence envoyée au système (Σ) devient $(m_{j+1}^{-1}\mathbf{q}^{ref}(l_{j+1}), \mathbf{u}^{ref})$. L'hypothèse 7 assure que :

$$d_{\mathcal{C}}(m_j^{-1}\mathbf{q}^{ref}(l_{j+1}), m_{j+1}^{-1}\mathbf{q}^{ref}(l_{j+1})) \leq D^{loc} \tag{A.4}$$

En appliquant l'inégalité triangulaire à (A.3) avec $s = l_{j+1}$ et (A.4), on obtient :

$$d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(l_{j+1}), m_{j+1}^{-1}\mathbf{q}^{ref}(l_{j+1})) \leq 2D^{loc} \tag{A.5}$$

À nouveau, la stabilité exponentielle du système implique que cette inégalité reste valable sur l'intervalle $[l_{j+1}, l_{j+2}]$, et pour tout $s \in [l_{j+1}, l_{j+2}]$:

$$d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(s), m_{j+1}^{-1}\mathbf{q}^{ref}(s)) \leq 2D^{loc} \tag{A.6}$$

En utilisant l'hypothèse 8 de convergence entre deux abscisses de localisation, il vient même la majoration plus fine :

$$d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(l_{j+2}), m_{j+1}^{-1}\mathbf{q}^{ref}(l_{j+2})) \leq D^{loc} \tag{A.7}$$

Récurrence sur tout l'intervalle La séquence des inégalités (A.3)-(A.7) peut être appliquée récursivement pour tout $p, j+1 \leq p \leq j+k-1$. Et on peut écrire : $\forall s \in [l_p, l_{p+1}]$

$$d_{\mathcal{C}}(\hat{\mathbf{q}}_{odo}(s), m_p^{-1}\mathbf{q}^{ref}(s)) \leq 2D^{loc} \tag{A.8}$$

On écrit alors l'hypothèse 7 de la façon suivante : pour tout $s \in [c_i, c_i + \Delta s^{check}]$

$$d_{\mathcal{C}}(m_j^{-1}\mathbf{q}^{ref}(s), m_p^{-1}\mathbf{q}^{ref}(s)) \leq D^{loc} \tag{A.9}$$

L'inégalité triangulaire entre (A.9) et (A.8) donne directement pour tout $s \in [c_i, c_i + \Delta s^{check}]$:

$$d_{\mathcal{C}}(m_j^{-1}\mathbf{q}^{ref}(s), \hat{\mathbf{q}}_{odo}(s)) \leq 3D^{loc} \tag{A.10}$$

Conclusion En appliquant maintenant la transformation solide $\mathbf{x}(c_i)\hat{\mathbf{x}}_{odo}^{-1}(c_i)$ à (A.10), on obtient :

$$d_{\mathcal{C}}(\mathbf{x}(c_i)\hat{\mathbf{x}}_{odo}^{-1}(c_i)m_j^{-1}\mathbf{q}^{ref}(s), \mathbf{x}(c_i)\hat{\mathbf{x}}_{odo}^{-1}(c_i)\hat{\mathbf{q}}_{odo}(s)) \leq 3D^{loc}$$

Étant donné que $\hat{\mathbf{x}}(c_i) = m_j\hat{\mathbf{x}}_{odo}(c_i)$, cette dernière inégalité s'écrit :

$$d_{\mathcal{C}}(\mathbf{x}(c_i)\hat{\mathbf{x}}^{-1}(c_i)\mathbf{q}^{ref}(s), \mathbf{x}(c_i)\hat{\mathbf{x}}_{odo}^{-1}(c_i)\hat{\mathbf{q}}_{odo}(s)) \leq 3D^{loc} \tag{A.11}$$

En appliquant alors l'inégalité triangulaire entre (A.2) et (A.11), on obtient :

$$d_{\mathcal{C}}(\mathbf{x}(c_i)\hat{\mathbf{x}}^{-1}(c_i)\mathbf{q}^{ref}(s), \mathbf{q}(s)) \leq 3D^{loc} + D^{odo} \tag{A.12}$$

En utilisant maintenant l'inégalité triangulaire définie par la propriété 1 avec la différence entre (A.1) et (A.12), on obtient le résultat désiré :

$$d_O(\mathbf{q}(s), \mathcal{O}) > D^{clear} - (3D^{loc} + D^{odo}) \geq 0$$

□

Bibliographie

[Abuhadrous 2003] I. Abuhadrous, F. Nashashibi & C. Laurgeau. *3-D Land Vehicle Localization : a Real-time Multi-Sensor Data Fusion Approach Using RTMAPS*. International Conference on Advanced Robotics. Coimbra, Portugal, July 2003.

[Bar-Shalom 1993] Y. Bar-Shalom & X.R. Li. Estimation and tracking : Principles, techniques, and software. Artech House, Incorporated, 1993.

[Baraff 1992] D. Baraff. *Dynamic Simulation of Non-Penetrating Rigid Bodies*. Thèse de Doctorat, CUCS, Ithaca, 1992. Cornell Computer Science Technical Report 92-1275.

[Bobrow 1985] J.E. Bobrow, S. Dubowsky & J.S. Gibson. *Time-Optimal Control of Robotic Manipulators along Specified Paths*. International Journal on Robotics Research, vol. 4, pages 3–17, 1985.

[Bonnafous 2003a] D. Bonnafous. *Exécution réactive de trajectoire pour robots mobiles nonholonomes*. Thèse de Doctorat, INP Toulouse, 2003.

[Bonnafous 2003b] D. Bonnafous & F. Lamiraux. *Sensor Based Trajectory Following for Nonholonomic Systems in Highly Cluttered Environment*. International Conference on Intelligent Robots and Systems [IEE 2003], pages 892–897.

[Borenstein 1991] J. Borenstein & Koren Y. *The Vector Field Histogram-Fast Obstacle avoidance for Mobile Robots*. IEEE Transactions on Robotics and Automation, vol. 7, n° 3, pages 278–288, June 1991.

[Borenstein 1996] J. Borenstein, B. Everett & L. Feng. Navigating mobile robots : Systems and techniques. A. K. Peters, Ltd., Wellesley, 1996.

[Boyer 2004] F. Boyer. *Optimisation de trajectoires pour systèmes nonholonomes*. Rapport technique, LAAS-CNRS, Toulouse, 2004.

[Boyer 2006] F. Boyer & F. Lamiraux. *Trajectory deformation applied to kinodynamic motion planning for a realistic car model*. International Conference on Robotics and Automation. Orlando, USA, May 2006.

[Choset 2005] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki & S. Thrun. Principles of robot motion : Theory, algorithms, and implementations. MIT Press, Cambridge, MA, 2005.

[Divelbiss 1997] A. Divelbiss & J. Wen. *A Path Space Approach to Nonholonomic Motion Planning in the Presence of Obstacles*. IEEE Transactions on Robotics and Automation, vol. 13, n° 3, pages 443–451, June 1997.

[Espiau 1992] B. Espiau, F. Chaumette & P. Rives. *A New Approach to Visual Servoing in Robotics*. IEEE Trans. on Robotics and Automation, vol. 8, n° 3, pages 313–326, June 1992.

[Feiten 1994] W. Feiten, R. Bauer & G. Lawitzky. *Robust obstacle avoidance in unknown and cramped environments*. International Conference on Robotics and Automation [IEE 1994].

[Fleury 1997] S. Fleury, M. Herrb & R. Chatila. *GenoM : a tool for the specification and the implementation of operating modules in a distributed robot architecture*. IROS, Grenoble, France, vol. 2, pages 842–848, septembre 1997.

[Fliess 1995] M. Fliess, J. Lévine, Ph. Martin & P. Rouchon. *Flatness and defect of nonlinear systems : introductory theory and examples*. International Journal of Control, vol. 61, n° 6, pages 1327–1361, 1995.

[Fox 1997] D. Fox, W. Burgard & S. Thrun. *The Dynamic Window Approach to Collision Avoidance*. IEEE Robotics and Automation Magazine, vol. 4, n° 1, pages 23–33, March 1997.

[Fraichard 2004] Th. Fraichard & H. Asama. *Inevitable collision states - a step towards safer robots ?* Advanced Robotics, vol. 18, n° 10, pages 1001–1024, 2004.

[Hamel 2002] T. Hamel & R. Mahony. *Visual servoing of an under-actuated dynamic rigid-body system : An image based approach*. IEEE Transactions on Robotics and Automation, vol. 18, n° 2, pages 187–198, Apr 2002.

[Hillion 2005] M. Hillion. *Optimisation de trajectoires pour systèmes nonholonomes*. Rapport technique, LAAS-CNRS, Toulouse, 2005.

[IEE 1994] IEEE. International conference on robotics and automation, San Diego, CA, USA, mai 1994.

[IEE 2003] IEEE/RSJ. International conference on intelligent robots and systems, Las Vegas, NE, USA, octobre 2003.

[Kavraki 1996] L. E. Kavraki, Svestka. P., J.-C. Latombe & M.H. Overmars. *Probabilistic Roadmaps for Path Planning in High-Dimensioinal Configuration Spaces*. IEEE Transactions on Robotics and Automation, vol. 12, n° 4, pages 566–580, 1996.

[Khatib 1986] O. Khatib. *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*. International Journal on Robotics Research, vol. 5, n° 1, pages 90–98, Spring 1986.

[Khatib 1997] M. Khatib, H. Jaouni, R. Chatila & J.P. Laumond. *Dynamic path modification for car-like nonholonomic mobile robots*. International Conference on Robotics and Automation. Albuquerque, NM, USA, avril 1997. IEEE.

[Lamiraux 1997a] F. Lamiraux & J.-P. Laumond. *Flatness and small-time controllability of multi-body mobile robots : applications to motion planning*. European Control Conference. Brussels, Belgium, 1997.

[Lamiraux 1997b] F. Lamiraux & J.-P. Laumond. *From paths to trajectories for multi-body mobile robots*. International Symposium on Experimental Robotics, pages 237–245. Springer Verlag, 1997.

[Lamiraux 2000] F. Lamiraux & J.-P. Laumond. *Flatness and small-time controllability of multi-body mobile robots : application to motion planning*. IEEE Transactions on Automatic Control, vol. 45, n° 10, pages 1878–1881, octobre 2000.

[Lamiraux 2004] F. Lamiraux, D. Bonnafous & O. Lefebvre. *Reactive Path Deformation for Non-holonomic Mobile Robots*. IEEE Transactions on Robotics, vol. 20, n° 6, pages 967–977, Dec 2004.

[Large 2000] F. Large, S. Sekhavat, C. Laugier & E. Gauthier. *Towards robust sensor-based maneuvers for a car-like vehicle*. International Conference on Robotics and Automation. San Francisco, CA, USA, avril 2000. IEEE.

[Latombe 1991] J.-C. Latombe. Robot motion planning. Kluwer, Boston, MA, 1991.

[Laugier 1999] C. Laugier, Th. Fraichard, Ph. Garnier, I. E. Paromtchik & A. Scheuer. *Sensor-Based Control Architecture for a Car-Like Vehicle*. Autonomous Robots, vol. 6, n° 2, 1999.

[LaValle 1998] S.M. LaValle. *Rapidly-Exploring Random Trees : A New Tool for Path Planning*. Rapport technique n° TR 98-11, Computer Science Dept., Iowa State University, 1998.

[LaValle 2006] S. M. LaValle. Planning algorithms. Cambridge University Press, Cambridge, U.K., 2006.

[Lefebvre 2004] O. Lefebvre, F. Lamiraux, C. Pradalier & Th. Fraichard. *Obstacles Avoidance for Car-Like Robots : Integration And Experimentation on Two Robots*. International Conference on Robotics and Automation. New Orleans, LA, USA, Apr 2004. IEEE.

[Lefebvre 2005] O. Lefebvre, F. Lamiraux & D. Bonnafous. *Fast Computation of Robot-Obstacle Interactions in Nonholonomic Trajectory Deformation*. International Conference on Robotics and Automation. Barcelona, Spain, Apr 2005.

[Lefebvre 2006] O. Lefebvre & F. Lamiraux. *Docking Task for Nonholonomic Mobile Robots*. International Conference on Robotics and Automation. Orlando, USA, May 2006.

[Lin 1993] M.C Lin. *Efficient collision detection for animation and robotics*. Thèse de Doctorat, University of California, Berkley, 1993.

[Lozano-Pérez 1983] T. Lozano-Pérez. *Spatial Planning : A Configuration Space Approach*. IEEE Transactions on Computing, vol. C-32, n° 2, pages 108–120, 1983.

[Luca 1998] A. De Luca, G. Oriolo & C. Samson. Robot motion planning and control, chapitre Feedback Control of a Nonholonomic Car-like Robot, pages 171–253. Springer, NY, 1998.

[Malti 2005] A. Malti. *Planification et exécution de mouvements référencés sur des amers*. Thèse de Doctorat, UPS Toulouse, 2005.

[Minguez 2000] J. Minguez & L. Montano. *Nearness diagram navigation (ND) : a new real time collision avoidance approach*. International Conference on Intelligent Robots and Systems. Takamatsu, Japan, novembre 2000. IEEE/RSJ.

[Minguez 2002] J. Minguez, L. Montano & J. Santos-Victor. *Reactive navigation for non-holonomic robots using the ego-kinematic space*. International Conference on Robotics and Automation. Washingtom, DC, USA, mai 2002. IEEE.

[Nashashibi 2000] F. Nashashibi, B. Steux, P. Coulombeau & C. Laurgeau. *RTMPAS a framework for prototyping automotive multisensors applications*. IEEE Intelligent Vehicles Symposium. Deaborn, MI, USA, 2000.

[Paromtchik 1996] I.E. Paromtchik & C. Laugier. *Motion generation and control for parking an autonomous vehicle*. Proceedings of the IEEE International Conference on Robotics and Automation, 1996.

[Pradalier 2004] C. Pradalier. *Navigation intentionnelle d'un robot mobile*. Thèse de Doctorat, INP Grenoble, 2004.

[Quinlan 1993] S. Quinlan & O. Khatib. *Elastic Bands : Connecting Path Planning and Control*. International Conference on Robotics and Automation. Atlanta, GA, USA, mai 1993. IEEE.

[Renaud 1992] M. Renaud & J.-Y. Fourquet. *Time-optimal motions of robot manipulators including dynamics*. O. Khatib, J.J. Craig & T. Lozano-Pérez, editeurs, The robotics Review 2. MIT Press, 1992.

[Rouchon 1993] P. Rouchon, M. Fliess, J. Lévine & Ph. Martin. *Flatness and motion planning : the car with n-trailers*. Procedings of ECC'93. Groningen, 1993.

[Schwartz 1983a] J. T. Schwartz & M. Sharir. *On the Piano Movers' Problem : I. The Case of a Two-Dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers*. Communications on Pure and Applied Mathematics, vol. 36, pages 345–398, 1983.

[Schwartz 1983b] J. T. Schwartz & M. Sharir. *On the Piano Movers' Problem : II. General Techniques for Computing Topological Properties of Algebraic Manifolds*. Advanced applied Mathematics, vol. 4, pages 298–351, 1983.

[Schwartz 1983c] J. T. Schwartz & M. Sharir. *On the Piano Movers' Problem : III. Coordinating the Motion of Several Independent Bodies*. International Journal of Robotics Research, vol. 2, n° 3, pages 97–140, 1983.

[Seelinger 2003] M. Seelinger, J.-D. Yoder, E.T. Baumgartner & S.B. Skaar. *High Precision Visual Control of Mobile Manipulators*. IEEE Transactions on Robotics and Automation, vol. 18, n° 6, pages 957–965, December 2003.

[Shin 1986] K.G. Shin & N.D. McKay. *Minimum-Time Control of Robotic Manipulators with Geometric Path Constraints*. IEEE Transactions on Automatic Control, vol. 30, pages 531–541, 1986.

[Siméon 2001] T Siméon, J.-P. Laumond & F. Lamiraux. *Move3D : a generic platform for path planning*. 4th International Symposium on Assembly and Task Planning, 2001.

[Slotine 1989] J.J.E. Slotine & H.S. Yang. *Improving the efficiency of time-optimal path-following algorithms*. IEEE Transactions on Robotics and Automation, vol. 5, n° 1, pages 118–124, 1989.

[Soueres 1996] P. Soueres & J.-P. Laumond. *Shortest paths synthesis for a car-like robot*. IEEE Transactions on Robotics and Automation, vol. 41, n° 5, pages 672–688, May 1996.

[Stentz 1994] A. Stentz. *Optimal and Efficient Path Planning for Partially-Known Environments*. International Conference on Robotics and Automation [IEE 1994].

[Thrun 2002] S. Thrun. *Robotic Mapping : A Survey*. G. Lakemeyer & B. Nebel, editeurs, Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann, 2002.

[Tsakiris 1997] D. Tsakiris, P. Rives & C. Samson. *Applying Visual Servoing Techniques to Control Nonholonomic Mobile Robots*. Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS). Grenoble, France, September 1997.

[Usher 2003] K. Usher, P Ridley & P. Corke. *Visual servoing of a car-like vehicle - an application of omnidirectional vision*. International Conference on Robotics and Automation, pages 4288–4293. Taipei, Taiwan, septembre 2003. IEEE.

[Wei 2003] R. Wei, R. Mahony & D. Austin. *A bearing-only control law for stable docking of unicycles*. International Conference on Intelligent Robots and Systems [IEE 2003], pages 3793–3798.

[Wei 2005] R. Wei, D. Austin & R. Mahony. *Biomimetic applicaton of desert ant visual navigation for mobile robot docking with weighted landmarks*. International Journal on Intelligent Systems Technologies and Applications, vol. 1, pages 174–190, 2005.

Navigation autonome sans collision pour robots mobiles nonholonomes

Cette thèse traite de la navigation autonome en environnement encombré pour des véhicules à roues soumis à des contraintes cinématiques de type nonholonome. Les applications de ces travaux sont par exemple l'automatisation de véhicules ou l'assistance au parking. Notre contribution porte sur le développement de méthodes qui réalisent certaines des fonctionnalités de la navigation autonome et sur l'intégration de ces différentes fonctionnalités au sein d'une architecture générique, en tenant compte des spécificités des systèmes considérés. Nous présentons une méthode d'évitement réactif d'obstacles pour systèmes nonholonomes et nous proposons une méthode de parking référencé sur des amers pour de tels systèmes. Ensuite nous présentons une architecture générique pour l'intégration des fonctionnalités de localisation, d'évitement d'obstacles et de suivi de trajectoire. Enfin nous illustrons l'ensemble de ces travaux par des résultats expérimentaux obtenus avec plusieurs robots.

Collision-Free Autonomous Navigation for Nonholonomic Mobile Robots

This work deals with autonomous navigation in cluttered environments for wheeled mobile robots subject to nonholonomic kinematic constraints. The potential applications of this work are for instance the development of autonomous cars and of parking assistance systems. Our contribution lies in the development of original methods to solve some of the functionalities of autonomous navigation and in their integration into a generic software architecture, while taking into account the specificities of the systems we deal with. We present an obstacle avoidance method for nonholonomic systems and we propose a landmark-based parking method for such systems. Then, we present a generic architecture for the integration of the functionalities of localisation, obstacle avoidance and trajectory following. Eventually, we illustrate this work with some experimental results obtained with several robots.