Ecole Normale Supérieure de Lyon /

Laboratoire d'Informatique Fondamentale
et d'Intelligence Artificielle

D.E.A. d'Informatique Fondamentale

Planification de trajectoire non-holonome en espace dynamique.

Alexis Scheuer

Directeur de projet : Ch. Laugier

Jury :
E. Mazer
J. Mazoyer
J. M. Muller
Y. Robert

Remerciements

Je tiens à remercier les personnes suivantes :

- Ch. Laugier qui m'a accueilli dans son équipe ;
- Th. Fraichard pour ses conseils et son aide de tout moment ;
- E. Mazer d'avoir bien voulu être membre de mon jury ;
- Ch. Bard d'avoir répondu à mes incessantes questions ;
- le reste de l'équipe SHARP, l'équipe MOVI, et bien d'autres pour leur soutient amical ;
- Elton, Paul, Phil, John et Bob pour leur indispensable assistance musicale.

Contents

List of Figures

Notations

Dans tout ce rapport, on va utiliser les notations générales suivantes :

1. si E est un espace topologique, et F un de ses sous-espaces, on note respectivement $\overset{\circ}{F}$, $\overline{F}$ et $\partial(F)$ l'intérieur, la fermeture et la frontière de F ; de plus, le complément de F dans E est noté $E \backslash F$;

2. les ensembles des entiers, des entiers relatifs et des réels sont notés respectivement $\mathbb{N}$, $\mathbb{Z}$ et $\mathbb{R}$; si E est un de ces ensembles, on pose $E^* = E \backslash \{0\}$;

3. lors de la formalisation des problèmes de planification, on utilise $\mathcal{W}$ pour désigner l'espace de travail (souvent isomorphe à $\mathbb{R}^n$ $(n \in \mathbb{N}^*)$; on note aussi $\mathcal{A}$ le robot, et $\{\mathcal{B}_i, i \in \{0, \ldots, n_B\}\}$ l'ensemble des obstacles dans $\mathcal{W}$;

4. l'espace des configurations de $\mathcal{A}$ est noté $\mathcal{C}$; il est de dimension m ; un élément de $\mathcal{C}$ est désigné par q, et la région de $\mathcal{W}$ occupée par $\mathcal{A}$ à la configuration q est appelée $\mathcal{A}(q)$.

Introduction

Robotique et action

Les principales capacités d'*un système robotique* sont la perception, la décision et l'action. La perception lui permet de fournir une base (une représentation de l'espace qui l'entoure) au module de décision, pour que celui-ci puisse déterminer les actions à accomplir afin d'exécuter les tâches qu'on lui a confiées (par exemple une surveillance, une manipulation ou un déplacement).

L'objectif actuel de la robotique est de developper des systèmes *autonomes*, c'est-à-dire capables d'exécuter des tâches à partir de descritions de haut niveau de celles-ci. De telles descriptions indiquent **ce** que l'on veut obtenir plutôt que **comment** on peut l'obtenir, laissant le système libre de trouver la manière d'exécuter cette tâche. La réalisation d'un tel objectif demande la résolution d'un certain nombre de problèmes plus spécifiques.

C'est dans ce cadre que, depuis une dizaine d'années, l'équipe SHARP (l'équipe robotique du LIFIA[1]) se consacre à des problèmes tels que la saisie automatique, le contrôle d'un ou plusieurs bras manipulateur(s), le contrôle d'un ou plusieurs robot(s) mobile(s), l'étude des mouvements 'compliants' (asservis sous capteurs), la planification automatique de tâche d'assemblage, etc...

Le problème abordé

Bien souvent, l'exécution d'une tâche nécessite un déplacement, que ce soit du robot ou d'un autre objet. Le module de décision doit donc être capable de planifier un mouvement au sein de son environnement. Ce problème a été étudié avec attention ces vingt dernieres années par de nombreux laboratoires de recherches, tant en Europe qu'aux Etats-Unis.

Dans le cas général, le problème de la planification de mouvement est le suivant : on veut savoir si, partant d'un point donné dans l'environnement avec une certaine vitesse, une partie de notre robot pourra rejoindre un autre point de l'environnement et l'atteindre avec une autre vitesse. Dans l'affirmative, nous voudrions trouver la

[1] Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle.

solution qui minimise un certain critère (distance, temps de parcours, etc...).

Nous nous sommes intéressé à ce problème dans un cadre assez particulier. On considère en effet un environnement plan ($I\!R^2$) caractérisé par des obstacles fixes et des obstacles mobiles, et qui est connu (avec une précision 'suffisante') sur une certaine durée. On place ensuite dans cet environnement notre robot qui est du type 'voiture', i.e. un unique objet rigide de forme simple (rectangulaire), et qui peut se déplacer en restant soumis à certaines contraintes :

- des contraintes physiques liées à sa structure (par exemple, celle sur sa manœuvrabilité) ;
- des contraintes liées à sa nature (par exemple, limitant les forces qu'il peut subir) ;
- des contraintes données par ses capacités (par exemple, ses limites en vitesse et accélération).

Une application de ce travail est la création d'un co-pilote pour une voiture se déplaçant dans un réseau routier[2].

Contenu du rapport

Afin de formaliser plus précisément ce problème et comment nous avons essayé de le résoudre, nous devons présenter un rapide historique de la planification de mouvement, dont une première forme a été la planification de chemin. Ensuite, nous pourrons revenir à notre problème plus particulier, et donner plus de détails sur l'approche que nous avons utilisée. Pour finir, nous présenterons concrètement l'apport effectué durant la période de ce stage, et quelques résultats fournis par une implantation de notre solution.

[2]dans le cadre du projet européen Eurêka Prometheus Pro-Art.

Chapter 1

La planification de mouvement

Comme nous l'avons vu dans l'introduction, un système robotique doit être capable de gérer un déplacement sans collision dans son environnement, alors que celui-ci peut être encombré d'obstacles aussi bien statiques que mobiles. Le problème de cette gestion se décompose en deux parties : *la planification de mouvement* (en anglais "motion planning") et son exécution. Nous allons nous intéresser au premier de ces deux points.

1.1 Présentation générale

Le problème de la planification de mouvement peut être posé, de manière générale et intuitive, sous la forme :

> Etant donné un robot $\mathcal{A}$ placé dans un environnement $\mathcal{W}$, étant données p_i la position[1] initiale de $\mathcal{A}$ et p_f la position où l'on désire mener $\mathcal{A}$, on cherche un mouvement qui mène de p_i à p_f, qui soit *exécutable* par $\mathcal{A}$ et *sans collision* avec les éventuels obstacles de $\mathcal{W}$.

Cette planification ne se limite pas au seul aspect géométrique de l'évitement des obstacles de $\mathcal{W}$. Elle peut comprendre de nombreux autres points, comme par exemple :

- la prise en compte des caractéristiques physiques de $\mathcal{A}$ et $\mathcal{W}$:
- l'optimisation d'un critère donné à priori (distance parcourue, temps de déplacement, etc...) ;
- la traitement de la dimension temporelle sous la forme de contraintes ou d'obstacles dynamiques ;

[1] dans le sens le plus général du terme.

- la coordination de plusieurs robots lorsque $\mathcal{A}$ est composé de plusieurs sous-systèmes indépendants ;
- la prise en compte des incertitudes concernant l'espace de travail (liées à la modélisation du monde réel) ;
- et bien d'autres choses encore...

La variété de ces aspects permet d'appréhender la complexité de la planification de mouvement, et d'expliquer la nécessité d'utiliser des hypotèses simplificatrices pertinemment choisies selon les conditions dans lesquelles on se place. Celles-ci permettent de distinguer, dans le problème général, deux sous-classes que l'on appellera *la planification de chemin* et *la planification de trajectoire*[2]. La planification de chemin correspond aux premières recherches sur la planification de mouvement, et se limite aux questions géométriques du problème. La planification de trajectoire correspond à l'introduction de la dimension temporelle, ce qui permet de traiter des obstacles mobiles et d'imposer des contraintes dynamiques sur $\mathcal{A}$ (cela représente en fait un large spectre de problèmes).

Dans la suite de ce chapitre, nous allons présenter chacune de ces planifications, en nous appuyant sur [Fra92], [Lat90] et [Lau88]. Nous commencerons par la planification de chemin, pour laquelle nous introduiront le concept d'*espace des configurations*, avant de décrire les différentes classes de méthodes développées pour résoudre ce problème. Ensuite, nous aborderons la planification de trajectoire, ce qui nous amènera à définir *l'espace-temps des configurations* (une généralisation de l'espace des configurations).

1.2 La planification de chemin

1.2.1 Formalisation du problème

On considère, pour cette planification, que le robot est le seul objet mobile de l'espace, et on ne tient pas compte de ses propriétés dynamiques. On ignore aussi les interactions mécaniques qu'il pourrait avoir avec son environnement, et on considère qu'il est rigide (indéformable). Les obstacles étant eux même rigides et fixes, la planification de mouvement se ramène à un problème purement géométrique, qui peut être formulé ainsi :

> Soit $\mathcal{A}$ un objet rigide (ou un ensemble), en mouvement dans un espace euclidien $\mathcal{W}$, représenté par $\mathbb{R}^n$ (où $n \in \mathbb{N}^*$) ; $\mathcal{A}$ est appelé *le robot*.
>
> Soient $\mathcal{B}_1, \ldots, \mathcal{B}_{n_B}$ des objets (fermés) rigides et fixés dans $\mathcal{W}$, qui sont appelés *les obstacles*.

[2] en anglais "path planning" et "trajectory planning"

On suppose que la forme du robot et des obstacles est connue, ainsi que la position de chaque obstacle.

Etant données une position et une orientation initiale, et une position et une orientation finale de $\mathcal{A}$ dans $\mathcal{W}$, on veut générer *un chemin*, i.e. une séquence continue de positions et d'orientations de $\mathcal{A}$, évitant toute collision avec les obstacles $\mathcal{B}_i$ ($i \in \{1, \ldots, n_B\}$), commençant à la position et avec l'orientation initiales et finissant à la position et avec l'orientation finales (ou signaler l'erreur si un tel chemin n'est pas trouvé).

On note qu'aucune contrainte n'est imposée sur le type de déplacement, et que le temps n'est pas pris en compte : seul la géométrie des obstacles conditionne le déplacement du robot. Etant donné qu'il peut exister une infinité de solutions au problème posé, et qu'on ne tient pas compte du temps, on cherche en général le chemin qui va minimiser la distance parcourue par $\mathcal{A}$.

Pour simplifier ce problème, on introduit le concept d'*espace des configurations*.

1.2.2 L'espace des configurations

Ce concept est apparu en robotique (à notre connaissance) en 1977, dans [Udu77], mais n'a réellement été introduit que deux ans plus tard par Lozano-Perez et Wesley [LPW79, LP83]. L'espace des configurations désigne l'espace propre au robot, par opposition à l'espace euclidien $\mathcal{W}$ dans lequel il évolue ; le robot y est représenté par un point, et non par un objet comme dans l'espace euclidien. La planification dans un tel espace se ramène donc à la recherche d'une courbe continue de la configuration initiale jusqu'à celle finale.

Afin de définir cet espace, il faut énoncer ce que l'on entend par *configuration* : une configuration est, pour un objet quelconque, une spécification de la position de tous les points de cet objet (par rapport à un système de référence). Elle est représentée par une liste de paramètres indépendants liés à la nature du robot. L'espace des configurations de $\mathcal{A}$, noté $\mathcal{C}$, est l'ensemble des configurations que peut prendre le robot. Sa dimension m est donnée par le cardinal de cette liste des paramètres.

Nous allons illustrer cette définition par deux exemples. Tout d'abord, si le robot est un objet indéformable en translation (sans rotation), son espace des configurations sera isomorphe à l'epace euclidien, une configuration étant représentée par les coordonnées d'un point (le point de référence du robot) dans la configuration considérée. Par contre, si le robot est un bras à deux degrés de liberté dans le plan, son espace des configurations sera un tore (c'est-à-dire le produit de deux cercles, chacun représentant l'intervalle cyclique $]-\pi, \pi]$), dans lequel une configuration sera donnée par la valeur des angles des deux degrés de liberté (voir la figure 1.1).

Figure 1.1: un espace des configuration torique

L'espace des configurations $\mathcal{C}$ permet, comme on l'a vu, de ramener le problème de la planification de chemin à celui de la recherche d'une courbe continue. Cependant, avant de l'utiliser pour rechercher un chemin depuis la configuration initiale jusqu'au but, il faut tenir compte des obstacles.

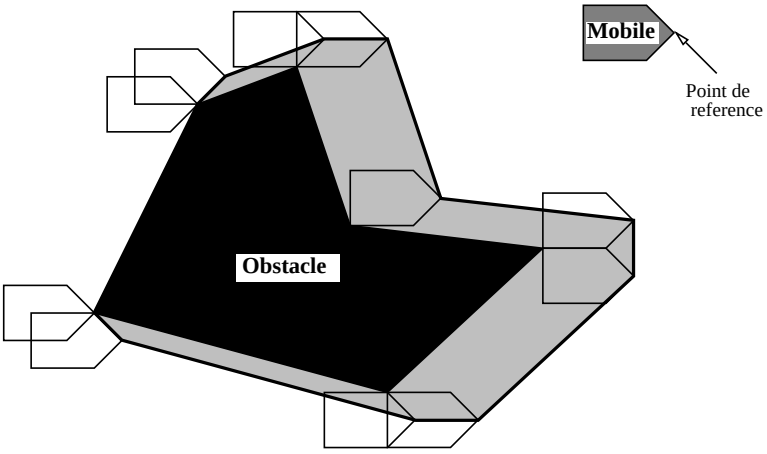

Figure 1.2: un exemple de $\mathcal{C}$-obstacle

On définit donc, pour chaque obstacle $\mathcal{B}_j$ de $\mathcal{W}$, un *$\mathcal{C}$-obstacle* noté $\mathcal{CB}_j$ dans $\mathcal{C}$: c'est l'ensemble des configurations dans lesquelles le robot est en collision avec l'obstacle $\mathcal{B}_j$. Si on note $\mathcal{A}(q)$ l'ensemble des points de $\mathcal{W}$ occupé par le robot dans la configuration q, on a :

$$\mathcal{CB}_j = \{q \in \mathcal{C} \ / \mathcal{A}(q) \cap \mathcal{B}_j \neq \emptyset\}$$

Une configuration q peut alors être d'un des quatre types suivants :

- *libre* si $q \notin \cup_{j=1}^{n_B} \mathcal{CB}_j$;
- *de contact* si $q \in \cup_{j=1}^{n_B} \partial(\mathcal{CB}_j)$;
- *admissible* si elle est libre ou de contact ;
- *de collision* si elle n'est pas admissible.

On note $\mathcal{C}_{libre}$, $\mathcal{C}_{contact}$, $\mathcal{C}_{admissible}$ et $\mathcal{C}_{collision}$ les ensembles correspondants. On cherche alors une courbe continue dans $\mathcal{C}_{libre}$ (ou $\mathcal{C}_{admissible}$ si on admet pour $\mathcal{A}$ un mouvement au contact des obstacles) qui relie les configurations initale et finale (si au moins une solution existe, on désire celle dont la longueur est minimale). Il existe pour cela plusieurs classes de méthodes, que nous allons présenter maintenant.

1.2.3 Les différentes méthodes

On peut regrouper les principales méthodes de planification de chemins en trois grandes classes (voir [Lat90]), les deux premières définissant un graphe dans l'espace des configurations (respectivement entre des points ou des secteurs singuliers) alors que la troisième construit incrémentalement un chemin à partir des données.

Les méthodes de type 'squelette'

Le terme anglais ("roadmap") est plus parlant. Le principe est simple : on définit dans $\mathcal{C}_{libre}$ (ou dans $\mathcal{C}_{admissible}$) un ensemble de courbes (structurées en graphe), puis on complète ce graphe pour raccorder les configurations initiale et finale. On cherche alors le plus court chemin dans ce graphe entre ces configurations. Un tel graphe peut provenir d'un *graphe de visibilité*, d'un *diagramme de Voronoï* ou d'une construction similaire.

Dans le cas d'un graphe de visibilité, les $\mathcal{C}$-obstacles sont supposés être polygonaux, et leurs sommets forment, avec l'origine et le but, les nœuds du graphe [Nil69]. Les arêtes sont les segments qui relient les nœuds sans intersection avec l'intérieur des $\mathcal{C}$-obstacles. Le plus court chemin dans ce graphe donne un *chemin de contact* entre l'origine et le but, c'est-à-dire un chemin contenant au moins une configuration sur la frontière d'un $\mathcal{C}$-obstacle (le chemin n'est pas inclus dans $\mathcal{C}_{libre}$, mais dans $\mathcal{C}_{admissible}$).

Une autre méthode utilise pour définir le graphe une fonction continue de $\mathcal{C}_{libre}$ dans une partie de dimension un de lui-même, et dont la restriction à cette partie est l'identité (une telle fonction est appelée une *rétraction* en topologie, d'où le nom de la méthode). Typiquement, en dimension deux, $\mathcal{C}_{libre}$ est 'rétracté' en son

diagramme de Voronoï[3] ; les configurations initiale et finale sont alors rétractées sur ce diagramme en deux points, qu'on cherche ensuite à relier par le graphe avec un coût minimal. Le chemin est alors formé (en dimension deux) de segments de droites et de paraboles. Des méthodes de ce type sont présentées dans [OY82, LS85].

Les décompositions en cellules

Cette méthode est probablement celle qui a été le plus étudiée. Elle consiste (comme son nom l'indique) en une décomposition de $\mathcal{C}_{libre}$ (ou de $\mathcal{C}_{admissible}$) en régions (ou *cellules*) au sein desquelles les chemins peuvent être générés sans contraintes. La recherche est alors transposée dans le *graphe de connectivité* (ou *graphe d'adjacence*), i.e. le graphe non orienté représentant les relations d'adjacence entre ces cellules. On déduit finalement de la liste des cellules sélectionnée le chemin optimal (par exemple, on effectue la transition d'une cellule à sa voisine en passant par le milieu de leur intersection).

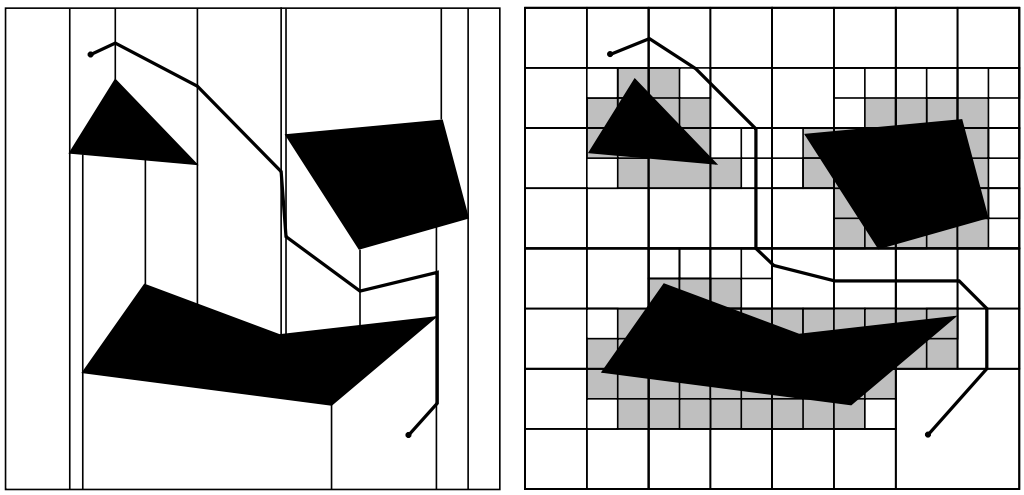

Figure 1.3: décompositions exacte (a) et approchée (b)

Cette décomposition peut être exacte (si l'union des cellules est égale à l'ensemble décomposé) ou au contraire approchée (si cette union est strictement incluse dans l'espace décomposé). Les **méthodes exactes** sont complètes (elles sont assurées de trouver une solution tant qu'il en existe) pourvu que l'algorithme d'exploration du graphe de connectivité le soit, mais sont souvent plus compliquées à implanter : la forme des cellules est alors plus complexe (pour pouvoir suivre les contours des $\mathcal{C}$-obstacles), et la détermination de leur limites difficile. Bien qu'une solution générale

[3] l'ensemble des points équidistants de deux frontières

ait été donnée dans [SS83a], on ne les utilise pratiquement qu'avec des obstacles polygonaux [SS83b, ABF88]. Les **méthodes approchées**, quant à elles, utilisent généralement des cellules de forme et de disposition régulières, aussi ne s'adaptent-elles pas parfaitement aux obstacles. Elles ont alors recours à une décomposition hiérarchique de l'espace des configurations, en redécomposant localement les cellules jusqu'à un niveau de résolution donné. Le choix de la forme ou de la disposition conduit à une variété de solutions proposées [BLP85, LG85, ZL89]. La plupart de ces méthode sont dites complètes *en résolution* : s'il existe une solution, elle pouront la trouver moyennant un niveau de décomposition assez profond.

Les méthodes de type 'potentiel'

Alors que les précédentes méthodes visaient à recouvrir l'espace des configurations libres (ou admissibles) grace à un graphe (exploré par la suite), les méthodes dites du type 'potentiel' utilisent un tout autre principe. En effet, elles consistent à considérer le mobile comme une particule chargée dans l'espace des configurations, et soumise à l'influence d'un potentiel : elle est attirée par le but, et repoussée par les $\mathcal{C}$-obstacles. Un tel potentiel a la propriété d'être indépendant de la position initiale. Il est composé d'une partie 'attractive', proportionnelle au carré de la distance au but, et d'une partie 'répulsive', qui tend vers $+\infty$ quand la distance à un $\mathcal{C}$-obstacle tend vers zéro (ce potentiel est généralement tronqué). Le chemin optimal est alors déterminé itérativement, de façon à suivre la plus grande 'pente' de la fonction potentiel.

L'inconvénient de telles méthodes est que, si le chemin construit mène toujours à un minimum, celui-ci peut être local et ne pas correspondre au but désiré. Une première solution pour résoudre ce problème est d'utiliser pour le potentiel une fonction qui, dans la partie connexe de $\mathcal{C}_{libre}$ (resp. $\mathcal{C}_{admissible}$) qui contient le but, n'admette pas d'autre minimum. On peut aussi avoir recours à des mécanismes variés pour sortir des minima locaux (par exemple des mouvements browniens).

Figure 1.4: la méthode des potentiels

Historiquement, ces méthodes ont été utilisées "on line", les potentiels étant calculés à partir de données fournies par des capteurs et le robot devant réagir en temps réel dans un environnement qu'il découvrait [Kha86]. En effet, l'influence des obstacles décroissant très vite avec la distance, la connaissance de l'environnement 'proche' était suffisant. Les méthodes utilisées dans ce cas sont dites *locales*, car elles ne disposent que d'une connaissance partielle de leur environnement, par opposition aux méthodes *globales* qui utilisent la connaissance de la totalité de leur environnement. L'avantage des méthodes de type 'potentiel' dans ce domaine était leur efficacité en temps de calcul.

Cependant, ces méthodes peuvent aussi être globales, lorsque la connaissance de la totalité de l'espace des configurations est utilisé pour la construction du potentiel. Une telle construction constitue un prétraitement semblable à la construction du graphe recouvrant l'espace dans les méthodes précédentes, mais elle permet de garantir la complétude de la solution si le potentiel est 'bien' construit [BL89].

1.3 La planification de trajectoire

Jusqu'à présent, nous avons parlé de planification de chemin, c'est-à-dire uniquement de l'aspect géométrique. On va maintenant rajouter l'aspect temporel, afin de s'intéresser à la façon dont le robot se déplace le long de la courbe qu'il suit. Cela va d'une part nous permettre de considérer des obstacles mobiles, et d'autre part nous autoriser à prendre en compte des contraintes dynamiques (force, accélération, vitesse).

1.3.1 Les obstacles mobiles

On considère toujours un robot $\mathcal{A}$ placé dans un espace $\mathcal{W}$, mais les obstacles $\mathcal{B}_j$ ($j \in \{1, ..., n_B\}$) sont maintenant mobiles. On note alors $\mathcal{B}_j(t)$ la région de $\mathcal{W}$ occupée par l'obstacle $\mathcal{B}_j$ à l'instant t.

Une première classe de méthodes consiste à rajouter la dimension temporelle à l'espace des configurations $\mathcal{C}$ ce qui donne un nouvel espace $\mathcal{CT} = \mathcal{C} \times [0, +\infty[$, appelé *espace-temps des configurations*[4]. La dimension de $\mathcal{C}$ étant m, celle de $\mathcal{CT}$ sera $m+1$. Les éléments de $\mathcal{CT}$ seront représentés par des couples (q, t), où q est une configuration de $\mathcal{C}$ et t est un instant donné. Il faut alors définir l'image dans $\mathcal{CT}$ des obstacles mobiles ; pour chaque obstacle $\mathcal{B}_j$ de $\mathcal{W}$, on crée donc *un $\mathcal{CT}$-obstacle* noté $\mathcal{CTB}_j$ et défini par :

$$\mathcal{CTB}_j = \{(q, t) \in \mathcal{CT} \ / \mathcal{A}(q) \cap \mathcal{B}_j(t) \neq \emptyset\}$$

La planification de trajectoire pour le robot $\mathcal{A}$, en présence d'obstacles mobiles, se ramène ainsi à une planification de chemin pour un point dans l'espace $\mathcal{CT}$. Il faut

[4] on peut rencontrer en anglais "configuration-time space" ou "configuration space-time".

cependant adapter les méthodes présentées dans la section 1.2, afin de tenir compte de la particularité de la dimension temporelle : celle-ci ne peut être parcourue que dans une direction (on n'a pas encore trouvé le moyen de remonter le temps). [RS85, FS90] présentent une extension du graphe de visibilité, alors qu'une adaptation de la décomposition cellulaire de $\mathcal{CT}$ est donnée par [SLG90].

Il existe une autre approche pour la planification de trajectoire en présence d'obstacles mobiles. Il s'agit de **la décomposition chemin–vitesse**, qui a été introduite par [KZ86]. Elle consiste à résoudre le problème en deux étapes : (a) on planifie un chemin sans collision en ne considérant que les obstacles fixes de $\mathcal{W}$, puis (b) on planifie la vitesse le long de ce chemin pour éviter les collisions avec les obstacles mobiles. La première étape correspond tout à fait au problème de la planification de chemin tel qu'il a été traité dans la section précédente. La seconde étape est encore un problème de planification de trajectoire, mais l'espace de travail a été réduit à une seule dimension (ce qui simplifie le problème).

Cette méthode est très efficace, mais elle a l'inconvénient de ne pas être complète : le chemin ayant été choisi sans considérer les obstacles mobiles, il n'existe pas forcément de profil de vitesse qui résolve l'étape (b), même si une trajectoire existe dans $\mathcal{W}$. Une solution consiste soit à réitérer le processus, soit à prévoir plusieurs chemins pour la planification de vitesse [PL90].

1.3.2 Les contraintes dynamiques

Le traitement des contraintes dynamiques est un problème très complexe. Il y a peu de résultats concernant la planification d'une trajectoire optimale en temps, alors que de nombreux travaux se sont consacrés à la recherche de solutions approchées de ce problème. Ces solutions sont dites sous-optimales. La planification de telles solutions utilise principalement deux outils : d'une part la restriction à une certaine classe de trajectoires (par exemple les trajectoires "bang-bang"), et d'autre part la discrétisation de l'espace (de travail, des configurations ou des phases[5]) afin de réduire le problème de planification à une recherche dans un arbre. On distingue deux classes d'approches, que nous alons détailler.

La première classe de méthodes est constitué par celles qui restent le long d'un chemin donné [BDG83, SM85]. Elles constitue la seconde partie d'une décomposition chemin–vitesse. Deux défauts s'ajoute à la non-complétude dont on a déjà parlé : la trajectoire optimale le long d'un chemin (même le plus court) n'est pas forcément la trajectoire optimale en temps et de plus, le chemin ayant été choisi sans tenir compte des contraintes dynamiques, son parcours peut être problématique.

Il existe aussi des méthodes plus générales, qui vise effectivement à déterminer des solutions optimales ou sous-optimales. Elles considèrent à la fois les contraintes dynamiques et celles statiques (les obstacles). Ainsi, [SD88] définit un graphe dans

[5]aussi appelé espace des états.

l'espace de travail du robot, en extrait un chemin optimal par recherche hierarchique sur la durée d'exécution, puis affine le résultat en fonction des contraintes dynamiques. Un travail similaire peut être effectué dans l'espace des configurations [SH85]. Ces méthodes ont cependant un temps de calcul exponentiel en la taille de la grille. Canny et al. [CDRX88] ont présenté une méthode dont le coût est polynômial en la taille de la grille (cette méthode a été reprise et approfondie dans [DX89] et [DX91], mais surtout étendue au cas d'un bras manipulateur dans [DX90]).

Chapter 2

Le cadre du travail

Nous avons jusqu'ici décrit le problème de la planification de mouvement de façon tout à fait générale. Nous allons désormais nous restreindre à la planification du mouvement d'un véhicule dans le réseau routier, dans le cadre du projet européen Eurêka Prometheus. Bien entendu, le système que nous avons développé n'est pas limité à ce seul cas, mais peut être utilisé pour toute planification en environnement structurable.

Dans ce chapitre, après une présentation du projet Prometheus, nous poserons le problème de planification qui en découle, puis nous présenterons la structure utilisée pour résoudre ce problème.

2.1 Le projet Prometheus

"Le projet européen Eurêka Prometheus est un programme de recherche qui a été lancé en octobre 1986 par les principaux constructeurs automobiles européens. Son but est d'étudier les possibilités d'amélioration du transport routier en Europe dans ses aspects sécurité, efficacité, économie, confort et impact sur l'environnement" [Fra92]. De telles améliorations supposent la mise en place de structures tant sur les véhicules qu'au niveau de l'infrastructure du réseau routier. Ces structures pourront en particulier fournir aux conducteurs une assistance, qui sera passive (sous la forme d'informations) ou active (c'est-à-dire une prise en charge partielle ou totale du véhicule). Les différents objectifs du projet Prometheus sont décrit en détails dans le document [Off89].

Pro-Art, l'un des sous-projet thématique de Prometheus, propose un pôle 'co-pilote', au sein duquel le LIFIA intervient. Son objectif est la mise au point, dans un premier temps, d'un système de 'co-pilotage de manœuvres' qui doit assister le conducteur pour des manœuvres délicates (traversée d'une intersection, insertion dans le flux autoroutier, etc...). A plus long terme, on envisage une prise en charge totale du véhicule, qui deviendrait alors totalement autonome (durant la réalisation

des manœuvres).

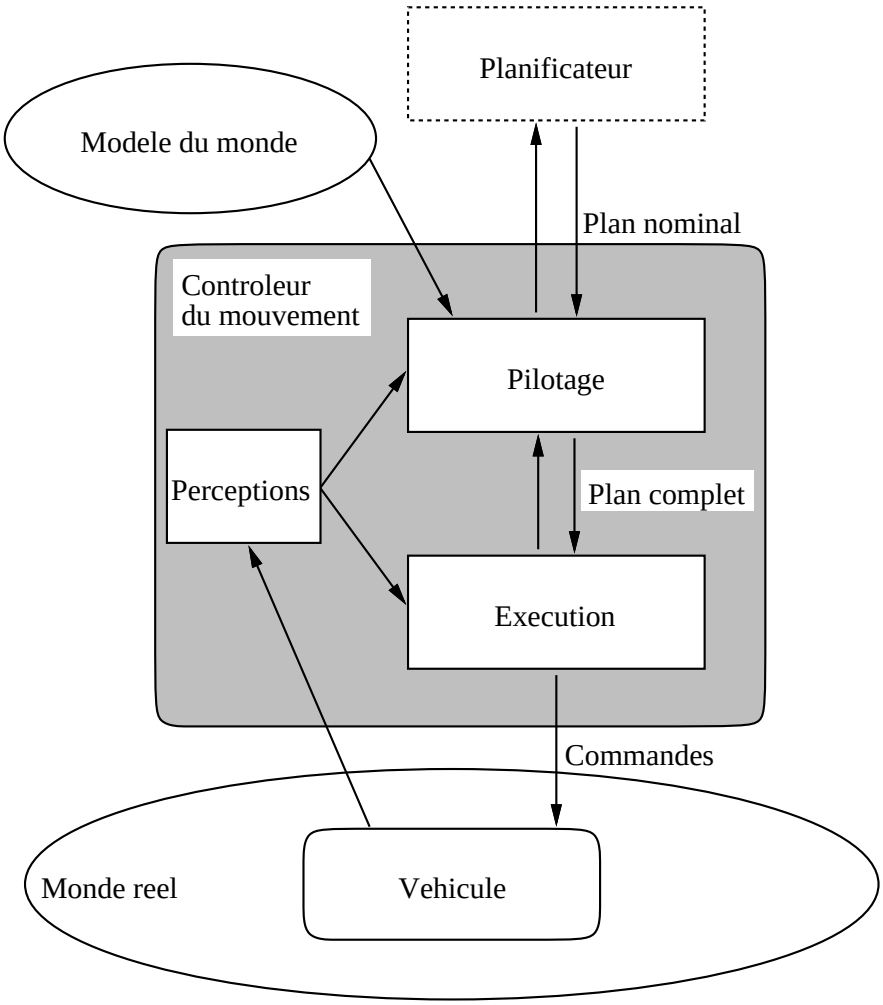

Figure 2.1: l'architecture du système

Dans les deux cas, le système considéré nécessite une planification de mouvement. Ce mouvement doit être calculé, à partir d'informations à priori sur les obstacles et l'espace de travail, puis exécuté en vérifiant sa validité. L'architecture globale du système est présentée dans la figure 2.1. Le pilote fait appel au planificateur de mouvement, afin de déterminer un plan nominal, qui est confié au contrôle d'exécution. Celui-ci exécute et amende le plan nominal, puis agit en retour (par "feedback") sur le pilote suivant le succès du plan (en lui indiquant par exemple la nouvelle position). Le problème de la planification sera décrit dans la section suivante. Le contrôleur d'exécution s'appuie sur une fonction potentielle (décomposée en trois parties, tenant compte respectivement du chemin planifié et des obstacles statiques, des obstacles mobiles, et des contraintes temporelles issue de la trajectoire) [FL91, HL92].

2.2 Le module de planification

Le module de planification doit résoudre un problème de planification de trajectoire dans un espace dynamique en présence d'une contrainte d'horizon temporel. On cherche en effet à relier deux configurations données pour le véhicule par un chemin exécutable (i.e. conforme aux contraintes cinématiques), tout en évitant des obstacles statiques et des obstacles mobiles, et en respectant des contraintes dynamiques.

L'approche que nous avons choisie consiste à utiliser la structure qui existe dans le réseau routier. L'espace de travail $\mathcal{W}$ est par conséquent organisé en un réseau de voies qui correspondent à celles du réseau routier, et que l'on s'oblige à suivre. Il convient de définir les informations de l'espace de travail $\mathcal{W}$. elles sont comme on le voit de trois types, et consistent précisément en :

- la géométrie et la topologie de $\mathcal{W}$, sous la forme d'un graphe dont les nœuds sont les voies et les arêtes des relations topologiques (adjacence ou croisement) ;
- la forme et la position des obstacles fixes ;
- la forme et la position des obstacles mobiles, sur la durée de *l'horizon temporel.*

L'horizon temporel indique le temps pendant lequel on peut faire confiance aux données concernant les obstacles mobiles. En effet, dans le cadre du projet Prometheus, toutes les informations connues sur l'environnement seront issues de capteurs instalés à bord du véhicule. Des bornes locales pourront fournir des renseignements sur l'espace $\mathcal{W}$ ou sur les obstacles fixes, mais les informations concernant les obstacles mobiles devront généralement être estimées. Ces estimations seront basées sur une approximation de la position, la vitesse et l'accélération instantanées (à l'instant présent), sur la nature de l'obstacle (piéton, voiture, etc. . .) et sur des informations symboliques (feux de véhicule, signes du bras, etc. . .). L'horizon temporel reflètera la durée de validité approximative de ces estimations.

Le module de planification se décompose alors en quatre planifications, comme le montre la figure 2.2. Chaque planification considère le problème de plus 'près' que la précédente, et affine le résultat déjà obtenu. Comme nous allons le montrer par la suite, elles utilisent des méthodes présentées dans le chapitre 1. Nous allons maintenant donner une description informelle de chacune d'entre elles.

2.2.1 Planification d'itinéraire

Cette étape correspond à la recherche, dans la structure du réseau routier, d'un itinéraire, i.e. une séquence de voies de $\mathcal{W}$, reliant la position initiale et celle finale. Les voies qui composent le réseau étant supposées rectilignes, cette planification fournit, à partir du parcours du graphe qui modélise la structure de $\mathcal{W}$, le squelette du chemin que va suivre $\mathcal{A}$, c'est-à-dire une suite de segments de droite (non nécessairement reliés).

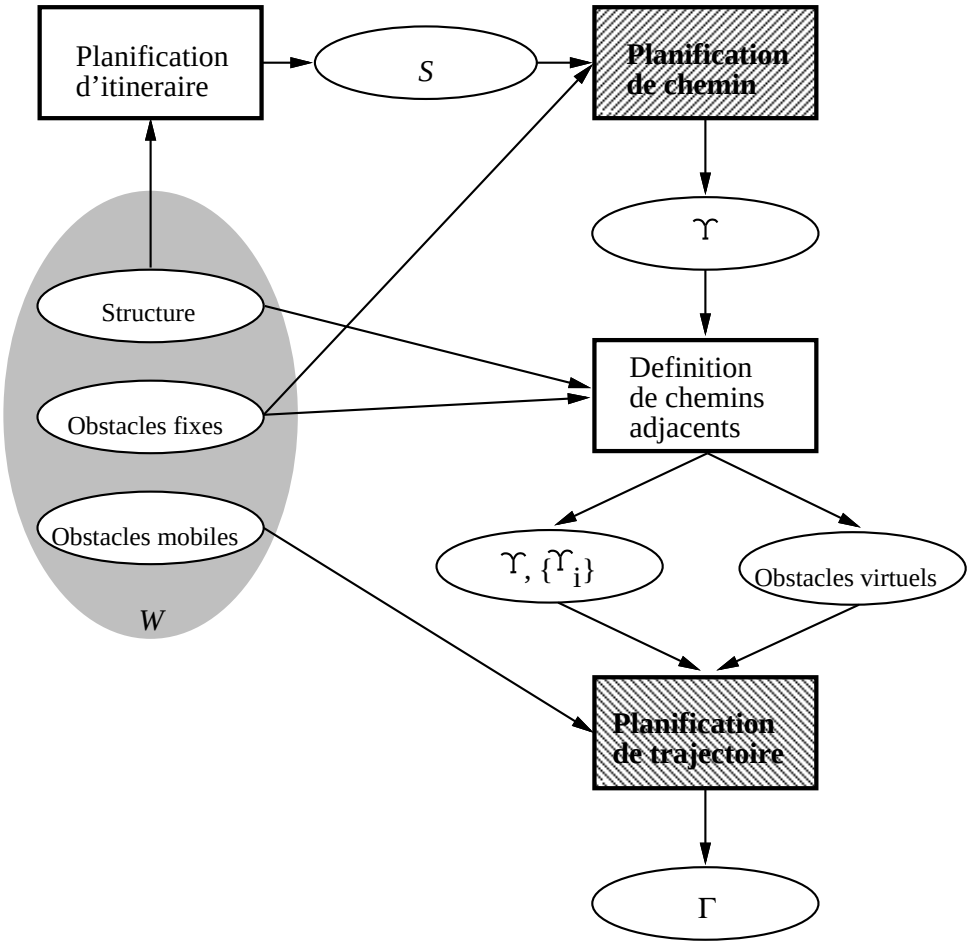

Figure 2.2: architecture du planificateur de mouvement

2.2.2 Planification de chemin

La planification de chemin affine le squelette rendu par la première partie. En effet, celui-ci n'est même pas forcément continu. Or, un véhicule tel que $\mathcal{A}$ ne peut pas suivre un chemin discontinu. Les roues arrières de $\mathcal{A}$ ayant une direction fixe par rapport au reste du véhicule, et le rayon de braquage du véhicule étant minoré, un chemin n'est *exécutable*par $\mathcal{A}$ que s'il est C^1 par morceaux (et que les points de discontinuité de la dérivé sont des points de reboussement) et si son rayon de courbure reste supérieur à une valeur notée ρ_{min}. Cependant, comme on ne veut pas effectuer de manœuvre (des changements de signe de la vitesse), on exige que le chemin soit C^1.

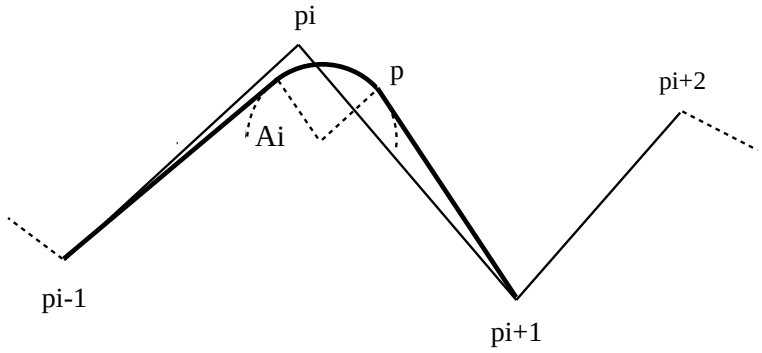

Figure 2.3: la création d'un virage

En fait, on remplace le squelette reçu en entrée par une séquence de segments de droite (chacun étant 'proche' d'une partie du squelette) et d'arcs de cercle (appelés *virages*), qui sont deux à deux tangents. La figure 2.3 donne un exemple de construction d'un virage entre deux composants du squelette. Le virage, ainsi que les segments qui l'entourent, doit être construit de manière à éviter toute collision avec les obstacles fixes.

2.2.3 Définition de chemins adjacents

La partie précédente nous a fourni un chemin sans collision, exécutable et sans manœuvre (c'est-à-dire C^1 et avec un rayon de courbure constament supérieur à ρ_{min}). Afin d'utiliser une autre caractéristique du réseau routier (l'existence de voies parallèles), on introduit la notion de *chemin adjacents*.

Si $\Psi : I \longrightarrow \mathcal{W}$ et $\Omega : J \longrightarrow \mathcal{W}$ deux chemins sans manœuvre et exécutable par $\mathcal{A}$, Ψ et Ω sont **adjacents**, i.e. disposés l'un parallèlement à l'autre avec un écart constant, si et seulement si (cf figure 2.4) :

1. $\forall i \in I, \exists j \in J$ unique et tel que :

$$\Psi(i)\overrightarrow{\Omega}(j).\dot{\Psi}(i) = 0, \quad \Psi(i)\overrightarrow{\Omega}(j) \wedge \dot{\Psi}(i) > 0, \quad \|\Psi(i)\overrightarrow{\Omega}(j)\| = \delta L$$

2. et $\forall j \in J, \exists i \in I$ unique et tel que :

$$\Omega(j)\overrightarrow{\Psi}(i).\dot{\Omega}(j) = 0, \quad \Omega(j)\overrightarrow{\Psi}(i) \wedge \dot{\Omega}(j) < 0, \quad \|\Omega(j)\overrightarrow{\Psi}(i)\| = \delta L$$

où δL est l'écart entre Ψ et Ω. Dans ce cas, Ψ (resp. Ω) est le chemin *de droite* (resp. *de gauche*). On dira aussi que $\Psi(i)$ est *le projeté* de $\Omega(j)$ sur le chemin Ψ (et inversement).

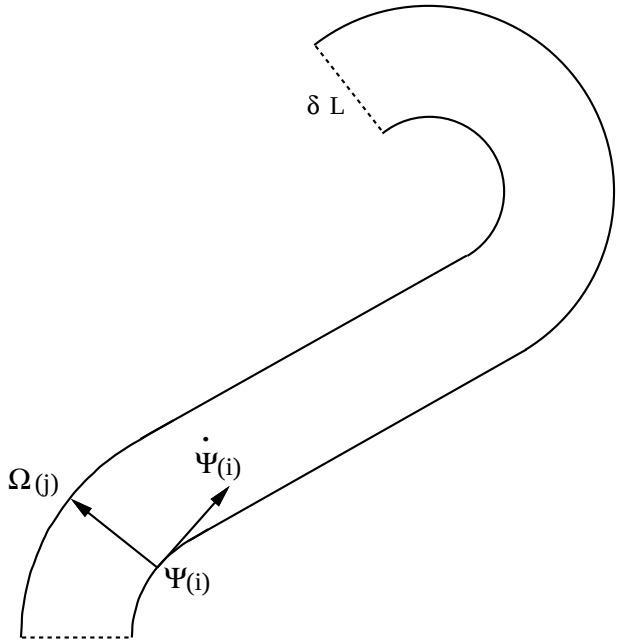

Figure 2.4: deux chemins adjacents

D'autre part, on associe au chemin Ψ **son couloir**, noté $\mathcal{R}(\Psi)$, qui est l'ensemble des points $p \in \mathcal{W}$ dont la distance à Ψ est inférieure à $\delta L/2$. On étend la notion de projeté sur un chemin à l'union des couloirs. On remarque d'abord que la perpendiculaire à un chemin Ω au point $\Omega(j)$ (i.e. la droite $(\Omega(j), \dot{\Omega}(j))$ est perpendiculaire à chacun des chemins adjacents (et aussi à leurs chemins adjacents, et...). La projection d'un point quelconque de l'union des couloirs sur un chemin Ω est donc l'intersection de ce chemin et de la perpendiculaire commune à tous les chemins, passant par ce point (voir la figure 2.5).

Cette notion de projection (définie dans l'union des couloirs) est utile pour construire *des obstacles virtuels*, là où les nouveaux chemins coupent les obstacles fixes.

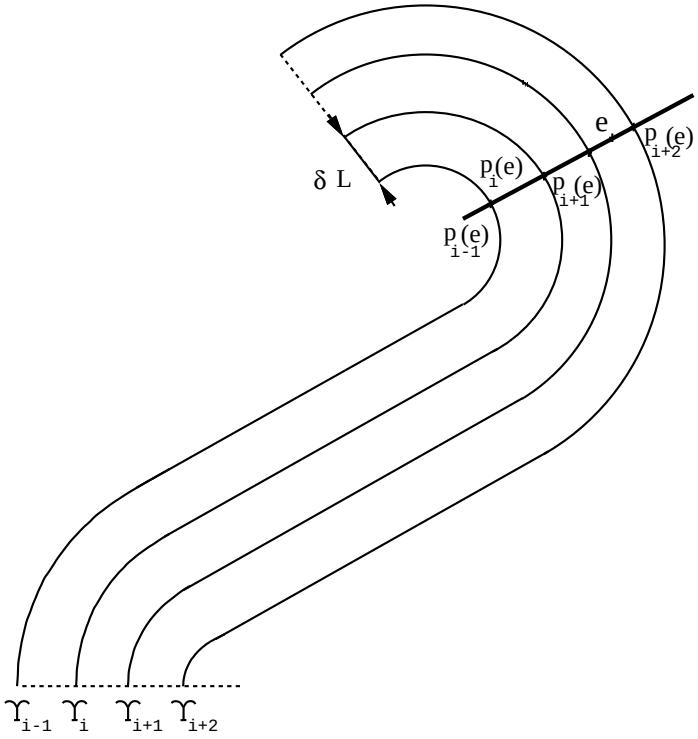

Figure 2.5: la projection sur les chemins

En effet, les chemins adjacents ne vérifient pas forcément les contraintes de non-collision, ni d'ailleurs celle qui porte sur le rayon de courbure. Il convient donc de fabriquer des obstacles qui empèche d'accéder aux parties non-conformes de chaque chemin. Ces obstacles sont des points qui possède une largeur telle qu'elle recouvre la zone interdite : toute la partie où le rayon de courbure est trop faible, ou la projection sur le chemin (perpendiculairement à sa dérivée) de l'intersection de l'obstacle avec son couloir, comme le montre la figure 2.6.

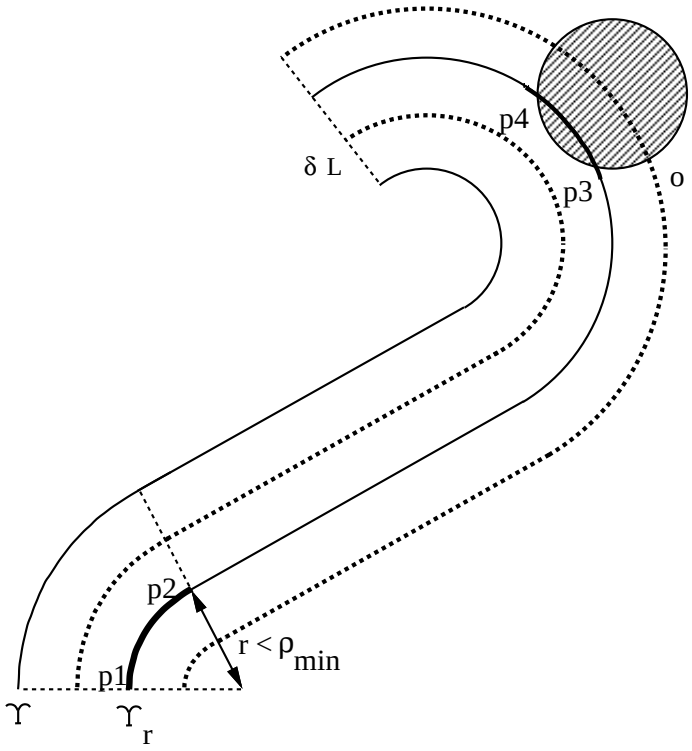

Figure 2.6: les obstacles virtuels

Lorsqu'on construit, à partir du chemin nominal (fournit par la planification de chemin), l'ensemble des chemins adjacents, on choisit δL tel que la zone que balaye $\mathcal{A}$ soit toujours incluse dans le couloir du chemin qu'il suit, tant que le rayon de courbure est assez grand. Ainsi, deux véhicules sur des chemins différents pourront se déplacer sans se soucier l'un de l'autre (le traitement des obstacles se limitera à ceux qui interseptent le couloir de la voie que l'on suit).

L'utilisation de chemins adjacents lors de la planification de trajectoire, afin de limiter l'effet de la non-complétude de la méthode, est une variante de l'utilisation de plusieurs chemins (proposée dans 1.3.1). Cette méthode est plus fine que le rejet global des chemins non-valides. Elle nous permet d'utiliser plus efficacement les chemins parallèles, et donc d'obtenir de meilleurs résultats pour la planification de trajectoire.

2.2.4 Planification de trajectoire

Enfin, la planification de trajectoire considère l'ensemble des chemins adjacents, et essaye de trouver une trajectoire les suivant (et passant de l'un à l'autre lorsque c'est nécessaire) et rejoignant le but en un temps minimal. C'est cette partie à laquelle nous nous sommes consacrés pendant la durée de ce stage, et que nous allons developper plus précisément dans la suite.

Chapter 3

Le problème abordé

Nous allons présenter dans cette partie le problème précis traité par le module de planification de trajectoire. Du fait de la complexité intrinsèque de ce problème, nous allons faire quelques hypothèses (dans la section 3.1) qui conduiront à une formulation particulière du problème (présentée dans la section 3.2).

3.1 Complexité du problème

Nous cherchons, dans un environnement plan parcouru d'obstacles mobiles, une trajectoire qui suive un chemin exécutable par le robot, qui vérifie certaines contraintes dynamiques et qui évite toute collision avec les obstacles. Hors, Canny a montré dans [Can87] que la planification de trajectoire pour un robot ponctuel à vitesse bornée dans un plan encombré de polygones en translation est un problème NP-dur. Cette planification est un sous-problème de celui que nous nous posons, et nous travaillons en plus avec une contrainte d'horizon temporel : notre modèle du monde extérieur n'étant valable que pour une durée t_{max}, nous voudrions obtenir une trajectoire avant la moitié de cet horizon (afin que l'exécution puisse se faire dans les limites de l'horizon temporel).

Il nous faut donc simplifier notre problème. Nous allons utiliser des chemins adjacents (définis par les modules précédents) : notre trajectoire prendra comme support l'un de ces chemins. Afin d'affiner un peu cette limitation, nous autoriserons des changements de chemin, c'est-à-dire le passage d'un chemin à un autre, sous certaines conditions. On passe donc d'un problème de dimension deux à un problème de dimension un (la possibilité de changement de chemin mise à part). Cela suppose que l'on redéfinisse précisément quelques points.

3.2 Formalisation précise

Nous allons d'abord indiquer comment sont modélisés le véhicule et les obstacles (mobiles ou fictifs) dans l'espace où nous travaillerons. Ensuite, nous allons expliquer comment seront traités le changement de chemin d'une part, et le traitement des obstacles d'autre part. Enfin, nous donnerons les contraintes dynamiques que nous devons respectées.

3.2.1 Le véhicule

L'objet $\mathcal{A}$ que l'on considère (*le robot* ou *le véhicule*) est modélisé par une particule ponctuelle et une longueur fixe l_A. Une position de cette particule est représentée par un triplet $(\Upsilon, p, \dot{p})$, c'est-à-dire respectivement le chemin que suit $\mathcal{A}$, son abscisse sur ce chemin et sa vitesse. On appelle ce triplet *état* de $\mathcal{A}$. Un *état-temps* correspondant à l'ajout de la dimension temporelle. C'est donc un quadruplet $e = (\Upsilon, p, \dot{p}, t)$, que l'on notera aussi quelquefois $e(t) = (\Upsilon(t), p(t), \dot{p}(t))$. On note $\mathcal{TS}$ l'espace des états-temps.

3.2.2 Les obstacles mobiles

Les obstacles sont, eux aussi, représentés par des particules ponctuelles munies d'une longueur constante. Nous allons montrer comment on calcule la position et la longueur d'une telle particule à partir de l'obstacle qui la génère.

Lorsqu'il suit le chemin Υ_i ($i \in \{0, \ldots, l-1\}$), le robot doit éviter les obstacles dynamiques qui coupent la région $\mathcal{R}(\Upsilon_\rangle)$, et eux seuls. A tout obstacle $\mathcal{B}_j$ ($j \in \{1, ..., n_B\}$) qui, grossit de $\frac{1}{2}\delta L$, coupe le chemin Υ_i (donc qui intersepte le couloir $\mathcal{R}(\Upsilon_\rangle)$), on fait correspondre à l'instant t l'abscisse $p^i_{B_j}(t)$ du milieu de l'intersection et $l^i_{B_j}(t)$ le rayon de cette intersection (sa demi-longueur le long de Υ_i). Les fonctions $p^i_{B_j}$ et $l^i_{B_j}$ ne sont pas définies pour tout t de $[0, t_{max}]$; on considèrera en fait que $l^i_{B_j}(t)$ est nul s'il n'est pas défini, et on prendra $w^i_{B_j} = \max_t l^i_{B_j}(t)$.

En conclusion, l'obstacle mobile $\mathcal{B}_j$ ($j \in \{1, ..., n_B\}$) est représenté sur le chemin Υ_i ($i \in \{0, \ldots, l-1\}$), s'il le coupe lorsqu'il est grossit de $\frac{1}{2}\delta L$, par une particule de longueur constante $w^i_{B_j}$ et de position, à l'instant t, $p^i_{B_j}(t)$.

Remarque : les obstacles fictif sont codés de manière similaire, si ce n'est qu'ils sont fixes, i.e. que leur centre ne dépend pas du temps.

3.2.3 Le changement de chemin

Dans le cadre de la planification de trajectoire, nous allons être amenés à changer de chemin (généralement pour éviter un obstacle). On a vu dans la section 3.1 que

l'on devait simplifier le changement de chemin (pour des raisons de complexité).

On impose d'abord qu'il ne se fasse qu'entre deux chemins adjacents. On suppose de plus que, pendant toute la durée de la transition d'un chemin à l'autre, le véhicule se trouve simultanément sur les deux chemins. En effet, la taille des couloirs correspondant à la largeur de $\mathcal{A}$, le véhicule 'empiète' sur le couloir du chemin voisin dès le début du changement de chemin, et 'dépasse' sur celui du chemin qu'il quitte jusqu'à la fin de ce même changement. Nous devons alors disposer d'une estimation de la trajectoire du changement de chemin (afin de pouvoir déterminer sa durée), pour vérifier sa non-collision.

3.2.4 L'évitement de collision

Nous avons vu dans la section 3.2.2 comment les obstacles mobiles sont représentés sur les chemins qu'ils coupent. Les obstacles fictifs étant représentés de la même manière, on peut traiter l'évitement de collision en imposant que la distance qui sépare $\mathcal{A}$ de chaque obstacle reste supérieure à une marge de sécurité.

Cette marge dépent de la longueur de l'obstacle et du robot, mais aussi de la vitesse de $\mathcal{A}$. Sur le chemin Υ_i ($i \in \{0, \ldots, l-1\}$), la marge relative à un obstacle $\mathcal{B}_j$ (défini par le couple $(p^i_{B_j}, w^i_{B_j})$) est donnée, pour une vitesse $\dot{p}$, par :

$$\delta\mathcal{B}_j^{\,i}(\dot{p}) = l_A + w^i_{B_j} + c_0 + c_1|\dot{p}|$$

où c_0 et c_1 sont des constantes imposées selon $\mathcal{A}$, qui reflêtent respectivement la marge statique et la distance de sécurité pour une vitesse donnée. Dans l'état-temps $(\Upsilon_i, p, \dot{p}, t)$, le robot $\mathcal{A}$ est en collision avec l'obstacle $\mathcal{B}_j$ (qui coupe le chemin Υ_i et est défini par le couple $(p^i_{B_j}, w^i_{B_j})$) si et seulement si :

$$|p - p^i_{B_j}(t)| < \delta\mathcal{B}_j^{\,i}(\dot{p})$$

Un état-temps est *sûr* s'il n'est en collision avec aucun obstacle (mobile ou virtuel). Le test de non-collision peut se limiter aux obstacles qui coupent le chemin correspondant à l'état-temps considéré.

3.2.5 Les contraintes dynamiques

La trajectoire que l'on va rechercher doit aussi respecter les contraintes dynamiques imposées à $\mathcal{A}$. Celles-ci ne portent que sur la vitesse et l'accélération que le robot peut subir. Cela étant, elles varient en fonction de la position dans $\mathcal{W}$, et donc de l'état dans lequel $\mathcal{A}$ se trouve. Pour un état $(\Upsilon, p, \dot{p})$, on exige donc que $\dot{p}$ soit compris entre 0 et $\dot{p}_{max}(\Upsilon, p)$, et on se limite à une accélération de module inférieur (ou égal) à $\ddot{p}_{max}(\Upsilon, p, \dot{p})$.

Les valeurs maximales de la vitesse et de l'accélération étant respectivement $\dot{p}_{max}$ et $\ddot{p}_{max}$, ces fonctions sont bien sûres telles que, pour tout état $(\Upsilon, p, \dot{p})$ de $\mathcal{A}$, on ai :

$$\begin{cases} \dot{p}_{max}(\Upsilon, p) & \leq & \dot{p}_{max} \\ \ddot{p}_{max}(\Upsilon, p, \dot{p}) & \leq & \ddot{p}_{max} \end{cases}$$

Conclusion : on a posé dans cette partie le problème que nous cherchons à résoudre : la détermination d'une trajectoire qui suit certains chemins, tout en évitant des obstacles mobiles et en respectant des contraintes dynamiques. Nous avons été amenés à faire des simplifications, pour des raisons de complexité. Nous devons maintenant définir exactement le type de trajectoire que nous voulons obtenir.

Une *trajectoire* de $\mathcal{A}$ est une fonction Γ qui associe à tout instant t de $[0, t_\Gamma]$ $(\subset [0, t_{max}])$ un état $\Gamma(t) = (\Upsilon(t), p(t), \dot{p}(t))$ de $\mathcal{TS}$. La *durée* de la trajectoire Γ est t_Γ. En fait, cette trajectoire est définie par deux fonctions indépendantes :

1. $\Upsilon : [0, t_\Gamma] \longrightarrow \{0, \ldots, l-1\}$, qui indique le numéro du chemin courant de $\mathcal{A}$;
2. $\ddot{p} : [0, t_\Gamma] \longrightarrow [-\ddot{p}_{max}, \ddot{p}_{max}]$, qui donne l'accélération instantanée de $\mathcal{A}$; les valeurs de $p(t)$ et de $\dot{p}(t)$ sont obtenues par intégration (au changement de chemin près, cf. 4.2.3).

Une trajectoire Γ est *sûre* si et seulement si, pour tout t de $[0, t_\Gamma]$, l'état-temps $(\Upsilon(t), p(t), \dot{p}(t), t)$ est sûr (comme on l'a définit dans la section 3.2.4).

Etant donné un état initial $s = (\Upsilon_s, p_s, \dot{p}_s)$ et un état final $g = (\Upsilon_g, p_g, \dot{p}_g)$, une trajectoire Γ est une *solution* au problème de planification si et seulement si :

- $\Upsilon(0) = \Upsilon_s$, $p(0) = p_s$, $\dot{p}(0) = \dot{p}_s$;
- $\exists t_\Gamma \in [0, t_{max}]$ tel que $\Upsilon(t_\Gamma) = \Upsilon_g$, $p(t_\Gamma) = p_g$ et $\dot{p}(t_\Gamma) = \dot{p}_g$;
- $\forall t \in [0, t_\Gamma]$, $\Upsilon(t) \in \{\Upsilon_i, i \in \{0, \ldots, l-1\}\}$;
- $\forall t \in [0, t_\Gamma]$, $0 \leq \dot{p}(t) \leq \dot{p}_{max}(\Upsilon(t), p(t))$ et $|\ddot{p}(t)| \leq \ddot{p}_{max}(\Upsilon(t), p(t), \dot{p}(t))$;
- Γ est sûre.

Naturellement, on cherche (si c'est possible) la solution optimale, i.e. celle dont la durée est minimale.

Chapter 4

La solution développée

Dans cette partie, nous présentons la solution que nous avons developpée pour la planification de trajectoire. On se place dans le cadre du problème énnoncé dans le chapitre précédent. On commencera par donner le principe général de la méthode, avant de décrire plus particulièrement deux points (la discrétisation et le parcours du graphe créé).

4.1 Le principe général

On utilise un principe très proche de celui présenté par Canny et al. dans [CDRX88, DX89, DX90]. On se place cependant en dimension un, puisque l'on suit un chemin nominal (ou un des chemins qui ont été définis à partir de celui-ci).

Ainsi, on discrétise le temps et l'accélération. Le pas de temps est noté τ, celui de l'accélération a_{step}. On va encore simplifier le type de trajectoire Γ que l'on cherche, en supposant que la fonction $\ddot{p}$ qui lui correspond est constante sur les intervalles de la forme $[k\tau,(k+1)\tau[, k \in \{0,\ldots,k_{max}-1\}$ (où $k_{max} = \left\lfloor \frac{t_{max}}{\tau} \right\rfloor$), et prend alors une valeur multiple de a_{step} (et qui vérifie les contraintes dynamiques). Un morceau de trajectoire de ce type, de durée t et d'accélération a est appelé trajectoire-(a,t). On cherche donc une trajectoire qui soit une réunion de trajectoires-$(a\ a_{step},\tau)$, a étant borné par les contraintes dynamiques. Avec de telles trajectoires, on ne peut atteindre que certains états de $\mathcal{A}$ lorsque le temps est multiple de τ. Cela permet de découper $\mathcal{TS}$ en une grille, et de créer un graphe $\mathcal{G}$ dont les nœuds sont des états-temps de la grille et dont les transitions sont étiquetées par l'accélération avec laquelle on passe d'un état-temps à l'autre. On va décrire en détail cette étape dans la section 4.2.

On recherchera ensuite dans ce graphe un chemin menant de l'état-temps $(s,0)$ (où $s = (\Upsilon_s, p_s, \dot{p}_s)$ est l'état initial) à un état-temps (g,t_{min}), $g = (\Upsilon_g, p_g, \dot{p}_g)$ étant l'état final et t_{min} étant le plus petit possible compte tenu des contraintes imposées. L'algorithme de parcours et l'heuristique utilisés seront décrit dans la section 4.3.

Ce chemin nous fournit une suite de paires (i_k, a_k), dont on déduit la trajectoire Γ cherchée en définissant les fonctions :

1. $\Upsilon : \begin{array}{ccc} [0, t_{min}] & \longrightarrow & \{0, \ldots, l-1\} \\ t & \longmapsto & i_{\lfloor \frac{t}{\tau} \rfloor} \end{array}$;

2. $\ddot{p} : \begin{array}{ccc} [0, t_{min}] & \longrightarrow & [-\ddot{p}_{max}, \ddot{p}_{max}] \\ t & \longmapsto & a_{\lfloor \frac{t}{\tau} \rfloor} a_{step} \end{array}$;

4.2 La discrétisation de l'espace-temps des états

Dans cette section, nous allons décrire précisément la forme de la grille des états-temps, et la construction du graphe correspondant. La grille est obtenue assez simplement, comme on le verra dans la section 4.2.1. Par contre, pour la création du graphe, il faut vérifier que les trajectoires intermédiaires sont sûres et qu'elles vérifient les contraintes dynamiques (cf. la section 4.2.2). On traitera le cas du changement de chemin dans une section séparée (4.2.3) ; il exige en effet plus de précautions, que ce soit au niveau de la validité de la transition que pour la valeur de l'état obtenu par cette transition.

4.2.1 La grille des états-temps

On désire construire une grille régulière ; on a en effet montré qu'un pas de discrétisation variable nous obligerait à faire de trop nombreuses approximations lors du parcours du graphe. Nous ne pourons donc pas tenir compte des limitations dynamiques locales dans la construction de la grille. Nous allons utiliser des conditions nécessaires pour la détermination des pas de discrétisation, en admettant qu'il est possible que l'on crée une grille trop fine.

On considère un état-temps $e_0 = (\Upsilon_0, p_0, \dot{p}_0, t_0)$. Une trajectoire-$(a\, a_{step}, \tau)$ issue de e_0 se termine à l'état-temps $e_f = (\Upsilon_f, p_f, \dot{p}_f, t_f)$ tel que :

$$\left\{ \begin{array}{rcll} t_f & = & t_0 + \tau & \\ \Upsilon_f & = & \Upsilon_0 & \text{(pas de changement de chemin)} \\ p_f & = & p_0 + \dot{p}_0 \tau + \frac{1}{2} a\, a_{step} \tau^2 & \\ \dot{p}_f & = & \dot{p}_0 + a\, a_{step} \tau & \end{array} \right.$$

On remarque que la vitesse est incrémentée d'un multiple de $a_{step}\tau$; on peut donc choisir cette valeur pour pas de discrétisation. L'abscisse est alors incrémentée d'un multiple de $\frac{1}{2} a_{step} \tau^2$, que l'on peut choisir comme pas de discrétisation. Ainsi, le choix des pas de discrétisation en temps (τ) et en accélération (a_{step}) permet de déterminer les pas de discrétisation en vitesse ($v_{step} = a_{step}\tau$) et en abscisse

($s_{step} = \frac{1}{2} a_{step} \tau^2$). On note respectivement k_{max}, a_{max}, v_{max} et s_{max} les limites supérieures des indices du temps, de l'accélération, de la vitesse et de l'abscisse. On a donc :

$$k_{max} = \left\lfloor \frac{t_{max}}{\tau} \right\rfloor, \quad a_{max} = \left\lfloor \frac{\ddot{p}_{max}}{a_{step}} \right\rfloor,$$

$$v_{max} = \left\lfloor \frac{\dot{p}_{max}}{v_{step}} \right\rfloor, \quad s_{max} = \left\lfloor \frac{p_{max}}{s_{step}} \right\rfloor$$

On considère l'ensemble $\{0, \ldots, l-1\} \times \{0, \ldots, s_{max}\} \times \{0, \ldots, v_{max}\} \times \{0, \ldots, k_{max}\}$, et on représente un état-temps par un élément de cet ensemble : à tout état-temps $(\Upsilon_i, p, \dot{p}, t)$ de $\mathcal{TS}$, on associe alors le quadruplé $(i, \left\lfloor \frac{p}{s_{step}} \right\rfloor, \left\lfloor \frac{\dot{p}}{v_{step}} \right\rfloor, \left\lfloor \frac{t}{\tau} \right\rfloor)$. Cette notation nous permet de simplifier considérablement les formules que nous utiliserons. Ainsi, le système précédent devient, lorsqu'on part de l'état (i_0, s_0, v_0, k_0) pour aller à l'état (i_f, s_f, v_f, k_f) avec une trajectoire-$(a\, a_{step}, \tau)$:

$$\begin{cases} k_f &= k_0 + 1 \\ i_f &= i_0 \\ s_f &= s_0 + 2v_0 + a \\ v_f &= v_0 + a \end{cases}$$

On remarque alors que $s + v$ garde la même parité : on peut donc diviser par deux le nombre des points de la grille. On choisit une partie ou l'autre, de façon à approximer au mieux les états initial et final[1], et on l'appelle $\mathcal{IS}$.

Sur cet ensemble, les contraintes dynamiques sont données par les fonctions suivantes (en reprenant les notations de la section 3.2.5) :

$$\begin{array}{rccc} a_{max} : & \{0, \ldots, l-1\} \times \{0, \ldots, s_{max}\} \times \{0, \ldots, v_{max}\} & \longrightarrow & \{0, \ldots, a_{max}\} \\ & (i, s, v) & \longmapsto & \left\lfloor \frac{\ddot{p}_{max}(\Upsilon_i, s\, s_{step}, v\, v_{step})}{a_{step}} \right\rfloor \end{array}$$

$$\begin{array}{rccc} v_{max} : & \{0, \ldots, l-1\} \times \{0, \ldots, s_{max}\} & \longrightarrow & \{0, \ldots, v_{max}\} \\ & (i, s) & \longmapsto & \left\lfloor \frac{\dot{p}_{max}(\Upsilon_i, s\, s_{step})}{v_{step}} \right\rfloor \end{array}$$

4.2.2 Le graphe correspondant

On construit ensuite un graphe, dont les nœuds sont les quadruplés de $\mathcal{IS}$. Les transitions issues d'un de ces nœuds doivent correspondre à des trajectoires-$(a\, a_{step}, \tau)$, a étant une valeur de $\{-a_{max}, \ldots, a_{max}\}$ telle que les contraintes dynamiques sont

[1] le choix τ et *step* permet de contrôler l'approximation que l'on va faire.

vérifiées et les obstacles mobiles évités sur la trajectoire-$(a\, a_{step}, \tau)$. On ne peut pourtant accepter toutes ces transitions : elles sont très nombreuses, or le coût de notre algorithme est proportionnel au nombre moyen de transitions issues d'un nœud (donc proportionnel au nombre total de transitions). Pour cette raison, on décide de n'utiliser que des stratégies extrèmes, c'est-à-dire de ne conserver pour chaque nœud que trois transitions sur la même voie : une correspondant à l'accélération maximale, une à vitesse constante, une à décélération maximale.

L'algorithme de construction du graphe est donc le suivant :

> Pour chaque nœud du graphe, c'est-à-dire pour tout élément (i, s, v, k) de $\{0, \ldots, l-1\} \times \{0, \ldots, s_{max}\} \times \{0, \ldots, v_{max}\} \times \{0, \ldots, k_{max}\}$ tel que $(s+v) \bmod 2$ vaut une valeur fixée initialement :
>
> - prendre pour a la plus grande valeur entière telle que :
>
> $$\begin{cases} a & \leq & a_{max}(i, s, v) \\ v + a & \leq & v_{max}(i, s) \end{cases}$$
>
> - pour ε décrivant $\{-1, 0, 1\}$, faire :
>   1. le nœud d'arrivé devant être $(i, s+2v+\varepsilon a, v+\varepsilon a, t+1)$, vérifier qu'il n'est pas en collision avec un obstacle ;
>   2. vérifier que la vitesse $(\max(v, v+a))$ et l'accélération (a) ne dépassent pas les limites autorisées pour ce nœud (sinon, recalculer a localement et reprendre à l'étape 1) ;
>   3. vérifier que la trajectoire-$(\varepsilon\, a\, a_{step}, \tau)$ est sûre ;

Le système d'équations et l'étape 2 permettent de vérifier la cohérence avec les contraintes dynamiques. Cela suppose cependant que les contraintes qui portent sur deux nœuds qui peuvent être reliés par une trajectoire-$(a\, a_{step}, \tau)$ sont représentatives des contraintes le long de cette trajectoire.

Les étapes 1 et 3 représentent les contraintes de non-collision, l'étape 1 permettant une élimination rapide des transitions évidemment impossible (comme on doit calculer l'état que l'on va atteindre, pour la vérification des contraintes dynamiques, autant vérifier la non-collision en ce point).

4.2.3 Le changement de chemin

Cet algorithme fournit trois voisins pour chaque nœud, ceux-ci étant sur le même chemin. Nous allons rajouter les transitions correspondant aux changements de chemin. Nous appliquerons pour cela un procédé similaire. Il nous faut déterminer, à partir de l'état de départ et du chemin visé, la représentation de l'état dans lequel

le changement de chemin s'achève. Nous devrons aussi avoir une description précise des positions intermédiaires pour le contrôle de non-collision.

Nous avons donc utilisé une version simplifiée du changement de chemin. Comme nous l'avons dit dans la section 3.2.3, nous nous limitons au passage d'un chemin à l'un des chemins adjacents. La transition se fera en deux arcs de cercles de rayon minimal, qui doivent vérifier les conditions suivantes :

- ils sont tangents l'un à l'autre ;
- l'un d'entre eux (que l'on appelle *le premier*) doit être tangent au chemin que l'on quitte ;
- l'autre (*le second*) doit être tangent au chemin que l'on veut rejoindre (le chemin *visé*) ;
- le sens de parcours du chemin initial induit un sens de parcours du premier cercle, et par conséquent un sens sur le second ; ce sens doit induire un parcours du chemin visé correspondant au sens initial (le changement de chemin n'est pas un demi-tour).

Les trois premières conditions sont pûrement géométriques, et très simples à résoudre. La contrainte de sens est déjà plus compliquée, d'autant plus que, si une solution existe, on désire celle qui minimise le chemin parcouru. L'annexe A décrit précisément les contraintes à vérifier et les paramètres donnant la forme du changement de chemin (quand il est réalisable). On remarque que le traitement est différent suivant que l'on veut rejoindre un segment de droite ou un arc de cercle. Dans l'algorithme, la transition se fait en supposant que le type de chemin ne change pas (l'état d'achèvement du changement de chemin étant calculé, on peut vérifier cette hypothèse et recalculer la transition si besoin est).

Le rayon de cercle minimal est définit par :

$$\rho_{chgt} = \max\left(\rho_{min}, \frac{\dot{p}^2}{g_{max}}\right),$$

où g_{max} est la force centrifuge maximale que peut subir le robot $\mathcal{A}$. Afin de calculer ce rayon, et d'obtenir une trajectoire simple le long de ce changement de chemin, nous avons choisit de fixer l'accélération pendant la transition (en utilisant les même valeurs que pour les transitions sur un chemin). Cela nous permet de déterminer la durée de chaque changement de chemin, et de vérifier les contraintes de non-collision pendant leur parcours.

Il faut préciser un dernier point. La vérification de la non-collision se fait, pour un état intermédiaire, sur les deux chemins entre lesquels il transite. Il faut donc vérifier, à chaque instant de la transition, que le projeté de l'état intermédiaire sur chacun des chemins est sûr.

Chaque nœud du graphe possède ainsi neufs voisins (s'il n'y a aucune contraintes dynamiques ou de non-collision) : les transitions ont le choix entre trois accélérations d'une part, et trois chemins (le chemin courant, celui de gauche ou celui de droite). On présente dans la figure 4.1 six de ces transitions.

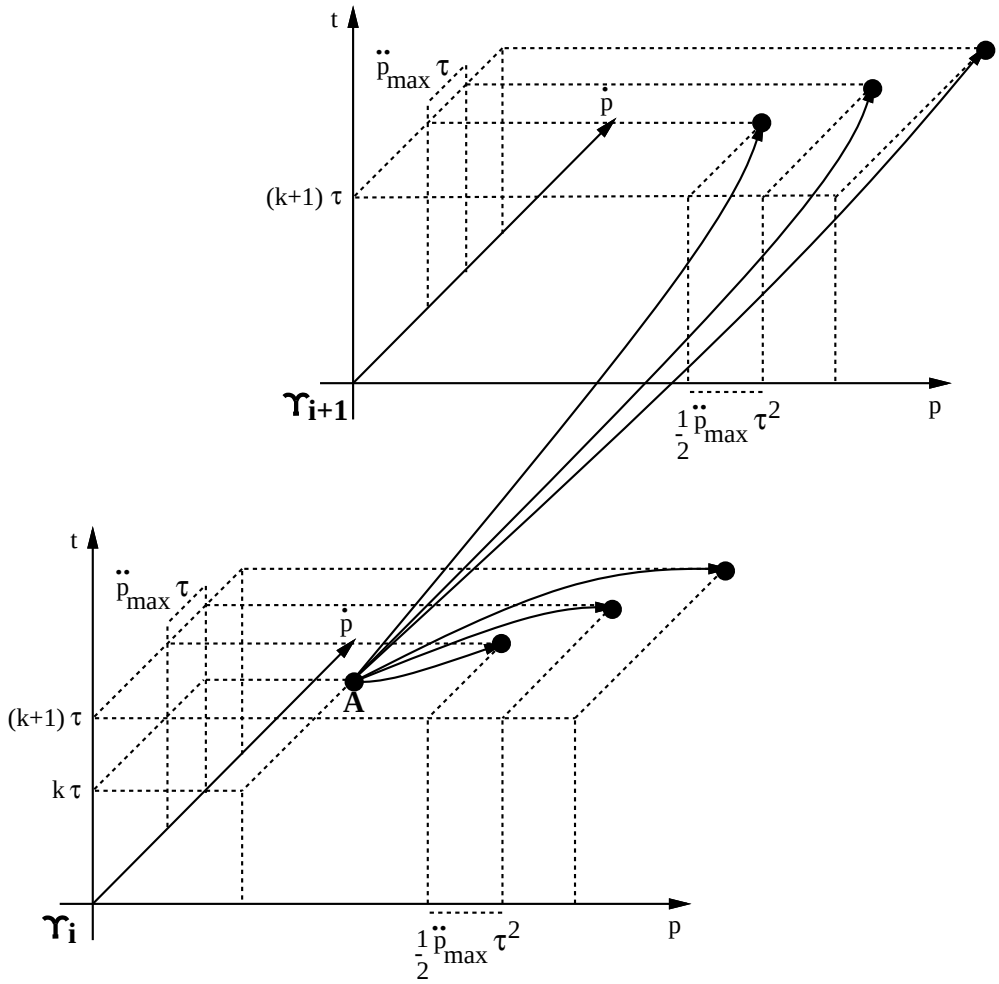

Figure 4.1: transitions dans le graphe $\mathcal{G}$

4.3 Le parcours du graphe états-temps

Le graphe $\mathcal{G}$ étant construit, mous allons décrire comment nous le parcourerons, à la recherche d'un chemin de l'état initial s à l'état final g. On définit par $\mathcal{N}_s$ le nœud de ce graphe correspondant à l'état-temps $(s, 0)$ initial. On note aussi G l'ensembles des nœuds de $\mathcal{G}$ qui correspondent à un état final, i.e. un état du type $(g, k\tau)$, où k appartient à $\{0, \ldots, k_{max}\}$.

4.3.1 L'algorithme de parcours

On utilise pour explorer $\mathcal{G}$ un algorithme du type A*, afin de n'explorer que la partie du graphe qui peut nous être utile. Cet algorithme construit un arbre de recherche recouvrant progressivement $\mathcal{G}$, et dont la construction est guidée par une heuristique. Il utilise une liste triée nommée OPEN, qui contient les feuilles non développées de l'arbre de recherche. Il utilise aussi l'arbre des nœuds déjà traités, appelé CLOSE. On peut formuler l'algorithme sous la forme suivante :

1. initialisation de OPEN à $\mathcal{N}_s$, et de CLOSE à $\emptyset$;
2. si OPEN est vide, on s'arrète en signalant l'échec ;
3. sinon, on extrait le premier nœud de OPEN, qui devient le nœud courant, et on l'insère dans CLOSE ;
4. on calcule tous les successeurs de ce nœud, et on vérifie leur validité (cf sections 4.2.2 et 4.2.3) ;
5. pour chacun de ces successeurs, on vérifie s'il appartient à G ;
   dans l'affirmative, on renvoie la branche de CLOSE menant au nœud courant prolongée par ce successeur (le chemin dans $\mathcal{G}$ qui mène de $\mathcal{N}_s$ à G avec le coût moindre) ;
   sinon, et si ce successeur n'appartient pas à CLOSE, il est inséré dans OPEN en fonction de son coût ;
6. on retourne à l'étape 2 ;

4.3.2 L'heuristique

La fonction heuristique est très importante, puisqu'elle guide la construction de l'arbre de recherche. Elle conditionne donc le temps mis par l'algorithme pour proposer une solution (s'il en existe). Cette heuristique doit donc être choisie de façon à estimer le plus précisément possible le temps que l'on mettra (au mieux) pour parvenir à un état final. Elle doit cependant ne pas être excessivement coûteuse, puisqu'elle sera estimée pour chaque nœud envisagé. On propose une solution dans l'annexe B.

On remarque que cette heuristique permet aussi d'éliminer les nœuds qui mènent à des sous-graphes de $\mathcal{G}$ qui n'ont aucune chance de contenir un élément de G. En effet, lorsqu'elle détecte de tels nœuds, elle retourne comme résultat de son estimation $+\infty$, ce qui conduit l'algorithme à ne pas les ranger dans OPEN (de tels nœuds sont ignorés). Ainsi, de grandes portions de $\mathcal{G}$ sont éliminées, ce qui diminue le temps de recherche : lorsque l'algorithme ne peut pas trouver de solution, il s'en rend compte sans avoir à parcourir le graphe en entier.

Chapter 5

Implantation

Nous allons présenter dans ce chapitre quelques exemples de planification, après avoir précisé les conditions d'expérimentation.

5.1 Expérimentation

Nous allons donner cinq exemples, chacun comportant deux chemins. Il faut remarquer que les obstacles mobiles ont été générés aléatoirement, et avec une vitesse constante. Ils entrent quelquefois en collision les uns avec les autres, ce qui peut rendre difficile la recherche d'une trajectoire sans collision (voir l'exemple de la figure 5.5). De plus, la contrainte d'évitement de collision ne tient pas compte de la vitesse ou de la taille de $\mathcal{A}$, mais seulement de celle des obstacles.

On a utilisé les valeurs suivantes, Ray étant le rayon des virages :

$$\begin{array}{cc} Ray = 100 \text{ m.}, & \delta L = 4 \text{ m.} \\ \rho_{min} = 4 \text{ m.}, & g_{max} = 1 \text{ m./s.}^2 \\ p_{max} = 500 \text{ m.}, & \dot{p}_{max} = 72 \text{ km./h.} \\ \ddot{p}_{max} = 1 \text{ m./s.}^2, & t_{max} = 100 \text{ s.} \\ \tau = 5 \text{ s.}, & a_{step} = 0,5 \text{ m./s.}^2 \\ (v_{step} = 9 \text{ km./h.}, & s_{step} = 6,125 \text{ m.}) \end{array}$$

De plus, on a limité le nombre de voisins dans le graphe à cinq : nous n'avons en effet considéré que les changements de chemin à vitesse constante (pour des questions de rapidité). D'autre part, les contraintes dynamiques sont gardées constante sur ce que l'on appelle un *type* de chemin. Dans le cas présent, nous avons trois types de chemin : la ligne droite (où les limitations sont les moins contraignantes) et deux types de virage, à droite et à gauche[1] (où la force centrifuge limite le mouvement). La réévaluation des valeurs limites n'est donc effectuée que lors d'un changement

[1] par rapport au chemin nominal (orienté).

de type ou de chemin (des chemins parallèles en virage n'ont pas le même rayon de courbure, donc pas la même contrainte liée à la force centrifuge).

Chaque exemple est constitué de quatre fenêtres (sauf le cas d'échec, nous verrons pourquoi par la suite). Les deux chemins sont numérotés 0 et 1, le chemin 0 étant celui de référence. Deux fenêtres sont alors associées à chaque chemin : une qui donne la trace de l'exploration de $\mathcal{G}$, et une qui présente le résultat finalement retenu. Dans ces fenêtres, le repère est en haut à gauche, l'axe horizontal (orienté vers la droite) donnant la position p, alors que l'axe vertical (orienté vers le bas) indique le temps. Les points de la grille de $\mathcal{TS}$ sont affichés, et les obstacles sont représentés par d'épais segments de droite noirs.

5.2 Résultats

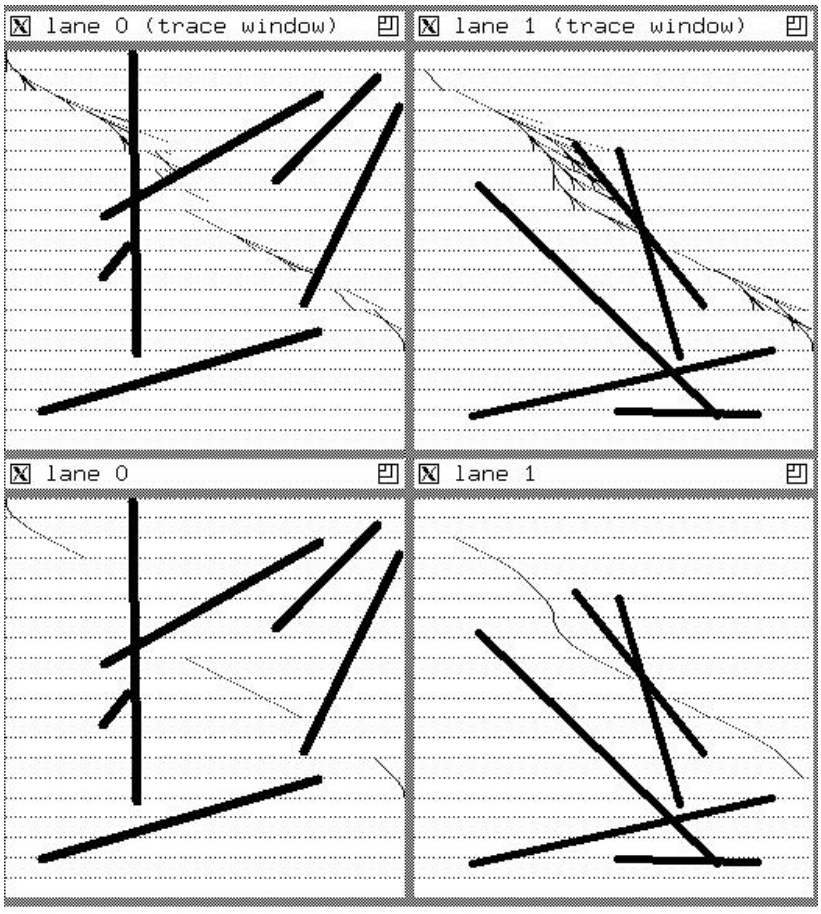

Figure 5.1: exemple en ligne droite

La première figure représente une planification de trajectoire en ligne droite. Il n'y a alors pas d'autre contrainte que les limitations sur l'accélération et la vitesse.

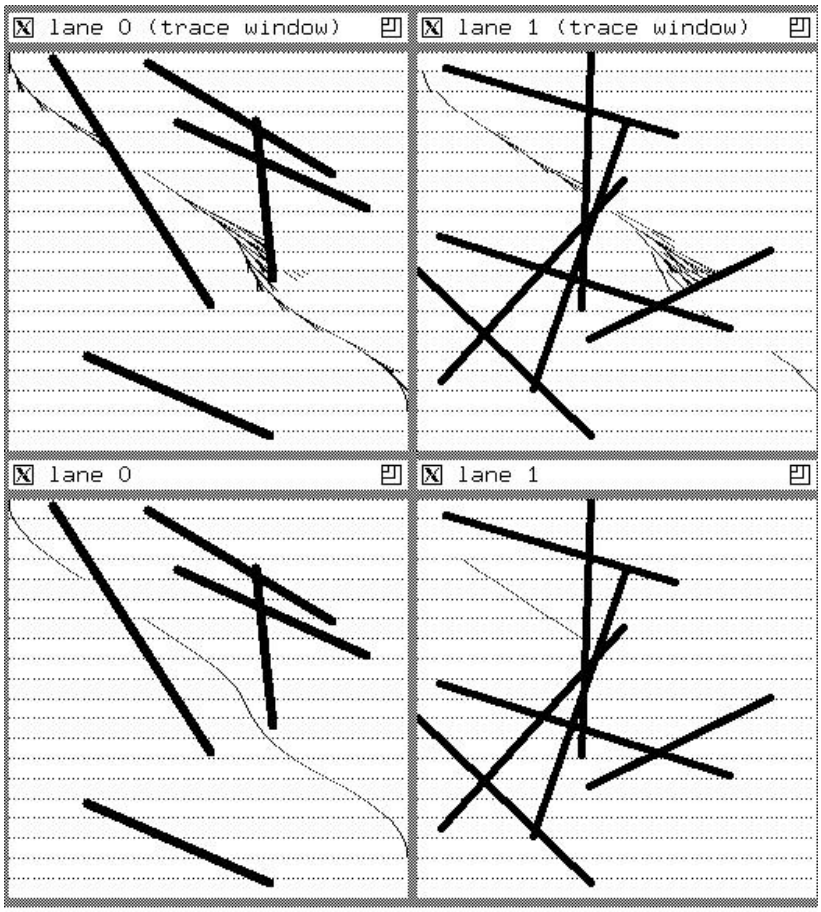

Figure 5.2: exemple en virage

Dans la seconde figure, on est en virage. La limitation sur la force centrifuge contraint les limites en vitesse et accélération, qui sont alors réduites de moitié. La marge de manœuvre s'en trouve réduite (un fort encombrement génère plus facilement un échec qu'en ligne droite).

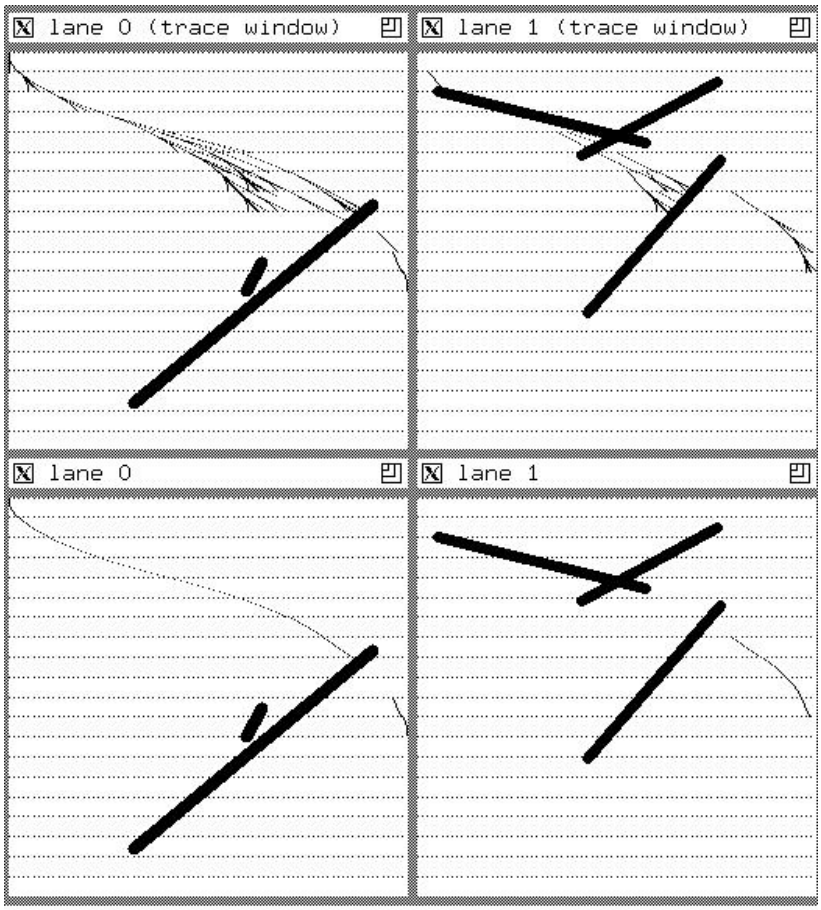

Figure 5.3: un premier exemple combiné

Dans la figure 5.3, ainsi que dans la suivante, on commence par trois cents mètres de ligne droite, avant d'arriver à un virage (de deux cents mètres). Cette rupture se voit sur le chemin 0, lorsque le planificateur semble 'hésiter' alors qu'aucun obstacle visible ne le gène. Cela correspond à un changement de valeurs limites, et donc à une modification (à la hausse) de l'heuristique. Le planificateur revient donc à des nœuds qu'il croit plus intéressants (à tort).

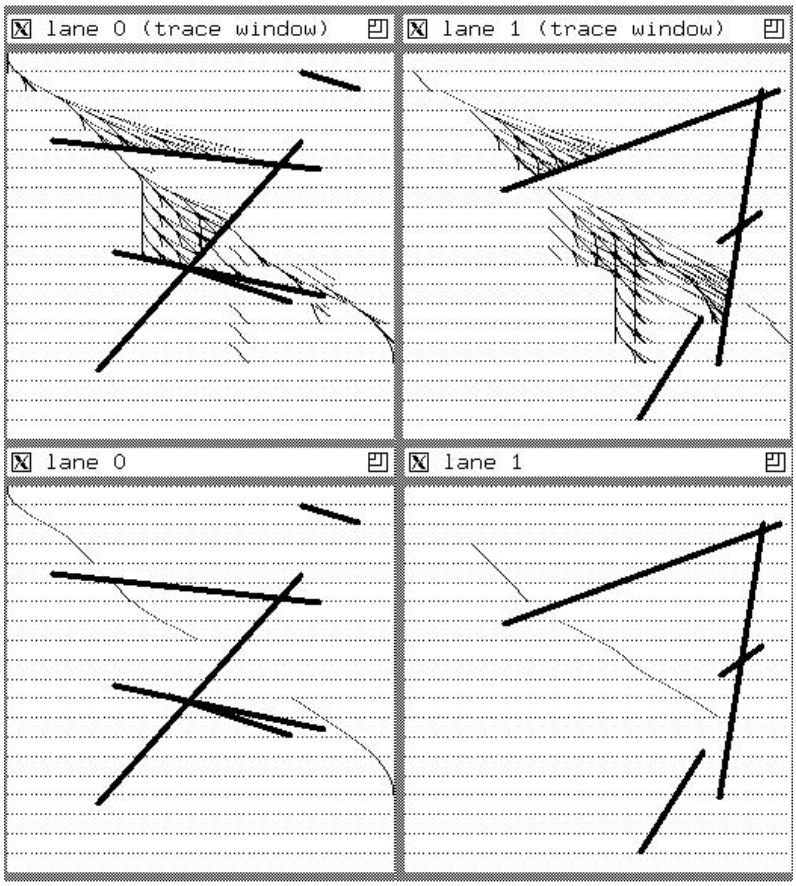

Figure 5.4: un second exemple combiné

On voit aussi, dans la figure 5.4, 'l'hésitation' remarquée dans la précédente. On voit surtout la manière dont le planificateur recherche la meilleure trajectoire : il privilégie les nœuds dont la position et la vitesse cumulées sont les plus intéressantes (lorsque le temps nécessaire au reste du parcours est le même).

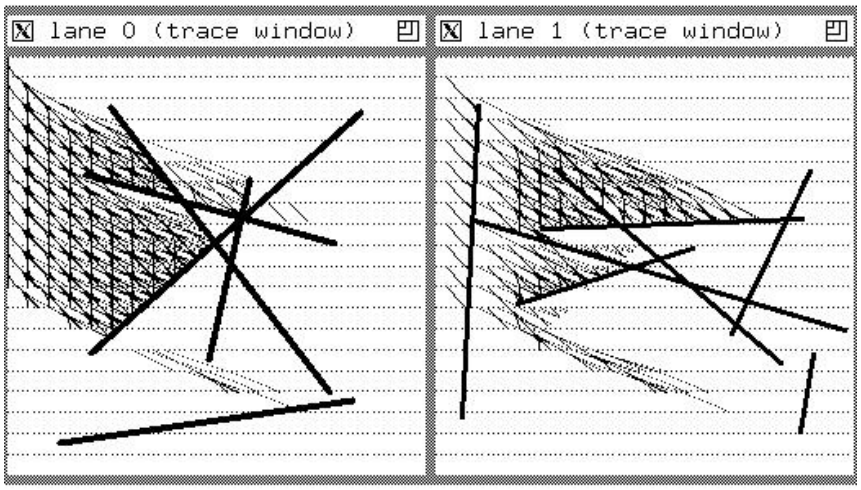

Figure 5.5: exemple d'un échec

On présente enfin un exemple d'échec : l'encombrement des deux chemins ne permet pas de rejoindre en temps voulu le but. On n'a pas représenté les figures finales (elles ne donnent pas de solutions et sont donc inintéressantes). On peut remarquer que le graphe $\mathcal{G}$ n'a pas été exploré dans sa totalité (les parties trivialement sans espoir ont été négligées).

Il a quand même fallut un peu plus de 13 secondes (temps CPU) pour conclure par un échec (nous avons eu affaire à un exemple qui a duré près de 30 secondes). En comparaison, un exemple trivial (un seul type de chemin, pas d'obstacle) est résolu en moins d'une demi-seconde ; les exemples présentés ici ont nécessité respectivement 1,28, 1,68, 1,30 et 5,27 secondes. Il est évident qu'il faut amméliorer l'implantation, le temps maximal que l'on puisse accepter devant être de l'ordre de $\frac{1}{2}\tau$ (cf section 3.1). Une première ammélioration pourrait être effectuée au niveau du changement de chemin, pour lequel les paramètres sont recalculés lorsque le besoin s'en fait sentir, au lieu d'être stockés. On pourrait aussi certainement trouver une meilleure heuristique.

Conclusion

Dans ce rapport, nous avons présenté une méthode de planification des mouvements d'un mobile $\mathcal{A}$ soumis à des contraintes cimématiques et dynamiques, dans un environnement dynamique $\mathcal{W}$. Cette méthode suppose que l'espace $\mathcal{W}$ est (ou peut être) structuré en un ensemble de couloirs de circulation. Elle se décompose en deux parties :

- une **planification de chemin** qui considère les contraintes cinématiques de $\mathcal{A}$ et les obstacles fixes de $\mathcal{W}$;
- une **planification de trajectoire** qui traite les contraintes dynamiques portant sur $\mathcal{A}$ et permet d'éviter les obstacles mobiles de $\mathcal{W}$;

Les deux planifications utilisent des approches existant dans la littérature, que nous avons présentées en premier lieu. Nous avons ensuite décrit une application réelle : la planification des mouvements d'une voiture dans le réseau routier. Enfin, nous avons formulé plus précisément la planification de trajectoire que nous avons développée.

L'originalité de la méthode consiste en l'utilisation poussée de la stucture existant au niveau de $\mathcal{W}$, et qui est 'adaptée' à $\mathcal{A}$. En effet, d'une part la recherche de chemins a profité de l'existence d'un graphe des couloirs de circulation dans $\mathcal{W}$, d'autre part la planification de trajectoire a eu recours à des chemins parallèles correspondant aux couloirs parallèles que contient $\mathcal{W}$.

Le travail que nous avons présenté est bien entendu inachevé (comme tout travail de recherche). Au niveau local, la planification de chemin pourrait être étendue afin de considérer des couloirs de circulation de forme plus générale (par exemple des unions de segments de droite et d'arcs de cercles, au lieu de simplesd segments rectilignes). Il faudrait aussi et surtout complèter l'implantation de la planification de trajectoire : intégrer tous les aspects présentés dans le chapitre 4, trouver une heuristique plus fiable (i.e. prenant en compte les limitations futures), peut-être

développer une planification hiérarchique (en affinant les pas de discrétisations du temps et éventuellement de l'accélération), etc...

En ce qui concerne la méthode de planification de mouvement présentée, il serait intéressant de connaître ses capacités lorsqu'il n'existe pas à priori de structure dans l'espace de travail $\mathcal{W}$. Le problème est bien sûr de déterminer ce qu'est une *structure adaptée* au mobile $\mathcal{A}$, puis de trouver comment construire une telle structure dans $\mathcal{W}$. Cela nous permettrait d'évaluer le champ d'application de cette méthode.

Annexe A

Description précise du changement de chemin.

Nous allons déterminer, dans cette annexe, comment rejoindre un chemin à partir d'un point donné, muni d'une vitesse initiale quelconque (en norme et direction). En effet, nous avions d'abord envisagé de pouvoir traiter le changement de chemin en plusieurs étapes (une plus longue réflexion nous a fait renoncer à cette idée pour l'instant), ce qui aurait généré des positions intermédiaires quelconques.

Les paramètres initiaux sont les suivants :

- la distance initiale au chemin, donnée par un réel d (positif ou non) ;
- la direction initiale, représentée par l'angle θ qu'elle fait avec la tangente au chemin en son projeté ;
- éventuellement, le rayon r du chemin (si celui-ci est un arc de cercle).

De plus, on a besoin de la valeur R du rayon de braquage du véhicule, qui est donné par les contraintes physiques ou cinématiques.

Cela étant fixé, on note ε le sens du premier virage (-1 si on commence par tourner à droite avant de virer à gauche, 1 dans le cas contraire), α l'angle du premier virage et, si le chemin est une droite, x la longueur du projeté du changement de chemin, ou χ l'angle que couvre ce projeté si le chemin est un arc de cercle.

A.1 Le changement de chemin vers un segment de droite

Dans le cas d'un chemin rectiligne, on définit le repère comme suit : la droite orientée supportant le chemin forme l'axe des abscisses, l'axe des ordonnées étant la perpendiculaire passant par la position initale. Le point de départ est donc $S\ (0, d)$,

alors que le centre du premier cercle de rotation est Ω_1 $(-\varepsilon R\sin\theta, d+\varepsilon R\cos\theta)$, et que le second est Ω_2 $(\varepsilon R\left[2\sin(\alpha+\theta)-\sin\theta\right], d-\varepsilon R\left[2\cos(\alpha+\theta)-\cos\theta\right])$. Pour que l'on puisse rejoindre le chemin, et ce de manière naturelle (continue et continuement dérivable), il faut que les coordonnées de ce dernier cercle soient aussi de la forme $(x, -\varepsilon R)$; on en déduit donc le système d'équation que vérifient les valeurs définissant le changement de chemin :

$$\left\{\begin{array}{rcl} d-\varepsilon R(2\cos(\alpha+\theta)-\cos\theta) & = & -\varepsilon R \\ \varepsilon R(2\sin(\alpha+\theta)-\sin\theta) & = & x \end{array}\right.$$

La première ligne de ce système conditionne la valeur de α, alors que la seconde donne celle de x. Cependant, pour que la première ligne admette une solution, il faut imposer une condition :

$$\begin{array}{rcl} 2\varepsilon R\cos(\alpha+\theta) = d+\varepsilon R(1+\cos\theta) & \Longrightarrow & -2R \leq \varepsilon d + R(1+\cos\theta) \leq 2R \\ & \Longrightarrow & -R(3+\cos\theta) \leq \varepsilon d \leq R(1-\cos\theta) \end{array}$$

On peut donc distinguer trois domaines de valeurs pour les paramètres initiaux :

- $\mathcal{D}_0 = \{|d| > R(3+\cos\theta)\}$, pour lequel il n'y a pas de solution au changement de chemin ;
- $\mathcal{D}_1 = \{R(1-\cos\theta) < |d| \leq R(3+\cos\theta)\}$, où la seule solution correspond à $\varepsilon = -\mathrm{sgn}(d)$ (sgn étant la fonction donnant le signe de son argument) ;
- $\mathcal{D}_2 = \{|d| \leq R(1-\cos\theta)\}$, dans lequel la détermination de α n'impose pas de contrainte sur ε.

On se restreint alors aux domaines $\mathcal{D}_1$ et $\mathcal{D}_2$, puis on pose $\phi = \arccos\left(\frac{R(1+\cos\theta)+\varepsilon d}{2R}\right)$. On veut alors que α soit minimal (en norme) tout en vérifiant les contraintes $(\varepsilon\alpha \geq 0)$ et $(\varepsilon(\alpha+\theta) \geq 0)$, pour que le sens le long des cercles soit compatible avec le sens initial ; la seconde condition implique $\alpha = \varepsilon\phi - \theta$, ou $\alpha = \varepsilon(2\pi-\phi)-\theta$ si la première solution ne convient pas.

On distingue encore deux cas :

- si $\phi > |\theta|$, ce qui est équivalent à dire que l'on est dans $\mathcal{D}_1$, alors $\varepsilon\phi - \theta$ est bien du signe de ε ;
- sinon, donc lorsqu'on est dans $\mathcal{D}_2$, $\varepsilon\phi - \theta$ est du signe de $-\theta$, ce qui implique que $\varepsilon = -\mathrm{sgn}(\theta)$.

En résumé, dans le cas d'un chemin rectiligne à rejoindre depuis une distance d avec une direction θ, et ce en suivant deux arcs de cercles de rayon R, on désigne la direction du premier virage par ε (-1 pour la droite, 1 pour la gauche, le second virage étant dans la direction opposée à celle du premier) et son angle par α (le sens initial imposant $\varepsilon\alpha \geq 0$) ; on distingue alors trois domaines de solution :

- $\mathcal{D}_0 = \{|d| > R(3 + \cos\theta)\}$, pour lequel il n'y a pas de solution au changement de chemin ;
- $\mathcal{D}_1 = \{R(1 - \cos\theta) < |d| \leq R(3 + \cos(\theta))\}$, où la meilleure solution est donnée par $\varepsilon = -\mathrm{sgn}(d)$;
- $\mathcal{D}_2 = \{|d| \leq R(1 - \cos\theta)\}$, dans lequel la meilleure solution est donnée par $\varepsilon = -\mathrm{sgn}(\theta)$.

La valeur de l'angle α, dans les cas où le changement de chemin est possible, est donnée par $\varepsilon \arccos\left(\frac{R(1+\cos\theta)+\varepsilon d}{2R}\right) - \theta$.

A.2 Le changement de chemin vers un arc de cercle

On passe maintenant au changement de chemin lorsque le chemin visé est un arc de cercle. Comme on l'a vu au début de cette partie, on rajoute des paramètres (le rayon r du chemin visé et l'angle χ correspondant au projeté du changement de chemin). De plus, on place cette fois le centre du repère au centre de l'arc de cercle que décrit le chemin, et on prend comme vecteur des abscisses la tangente à la courbe au projeté du point initial et comme vecteur des ordonnées celui dirigé vers le point initial. La position initiale est donc $S\ (0, r')$ (au lieu de $(0, d)$ dans le cas de la droite), où on note par r' la valeur $r + d$. On fixe ε arbitrairement, et pose $R' = \varepsilon R$ et $r'' = r - R'$, ce qui donne, pour les centres de rotation, $\Omega_1\ (-R'\sin\theta, r' + R'\cos\theta)$ et $\Omega_2\ (R'\left[2\sin(\alpha+\theta) - \sin\theta\right], r' - R'\left[2\cos(\alpha+\theta) - \cos\theta\right])$. La contrainte de continuité (et de dérivabilité continue) impose ici que ce second centre ait aussi pour coordonnées $(-r''\sin\chi, r''\cos\chi)$, ce qui donne comme système :

$$\begin{cases} r' - R'\left(2\cos(\alpha+\theta) - \cos\theta\right) &= r''\cos\chi \\ R'\left(2\sin(\alpha+\theta) - \sin\theta\right) &= -r''\sin\chi \end{cases}$$

On commence par déterminer α, en éliminant de ce système les secondes parties des égalités. Pour cela, on ajoute les deux équations élevées au carré, ce qui nous donne :

$$r'^2 + R'^2\left[(2\cos(\alpha+\theta)-\cos\theta)^2 + (2\sin(\alpha+\theta)-\sin\theta)^2\right] \\ -2r'R'(2\cos(\alpha+\theta)-\cos\theta) = r''^2$$

$$\Longrightarrow \quad r'^2 - r''^2 + 5R'^2 + 2r'R'\cos\theta \;= \\ 4R'\left[(R'\cos\theta + r')\cos(\alpha+\theta) + R'\sin\theta\sin(\alpha+\theta)\right]$$

On introduit alors un angle noté η, définit par $\eta = \arctan\left(\frac{R'\sin\theta}{R'\cos\theta+r'}\right)$, et donc tel que :

$$\cos\eta = \tfrac{r'+R'\cos\theta}{\sqrt{r'^2+R'^2+2r'R'\cos\theta}}, \qquad \sin\eta = \tfrac{R'\sin\theta}{\sqrt{r'^2+R'^2+2r'R'\cos\theta}},$$

en supposant :

$$r' + R'\cos\theta \geq 0. \tag{H1}$$

On cherche donc un angle α tel que :

$$4R'\sqrt{r'^2+R'^2+2r'R'\cos\theta}\cos(\alpha+\theta-\eta) = r'^2 - r''^2 + 5R'^2 + 2r'R'\cos\theta,$$

ce qui impose une condition sur les paramètres :

$$\left|r'^2 - r''^2 + 5R'^2 + 2r'R'\cos\theta\right| \leq 4R'\sqrt{r'^2+R'^2+2r'R'\cos\theta}.$$

Cette condition étant vérifiée, on pose $\phi = \arccos\left(\frac{r'^2-r''^2+5R'^2+2r'R'\cos\theta}{4R'\sqrt{r'^2+R'^2+2r'R'\cos\theta}}\right)$, et on prend α de la forme $\varepsilon'\phi+\eta-\theta$, où ε' prend une valeur dans {-1,1} faisant respecter la contrainte ($\varepsilon\alpha \geq 0$). Il ne se présente que trois cas, $\theta-\eta$ étant du signe de θ $\left(\sin(\theta-\eta) = \frac{r'\sin\theta}{\sqrt{r'^2+R'^2+2r'R'\cos\theta}}\right)$:

- $\mathcal{C}_0 = \left\{\frac{r'^2-r''^2+5R'^2+2r'R'\cos\theta}{4R'} > \sqrt{r'^2+R'^2+2r'R'\cos\theta}\right\}$, pour lequel il faut essayer avec l'autre valeur de ε ;
- $\mathcal{C}_1 = \{r'\cos\theta + R' < \frac{r'^2-r''^2+5R'^2+2r'R'\cos\theta}{4R'} \leq \sqrt{r'^2+R'^2+2r'R'\cos\theta}\}$, où $\phi < |\theta-\eta|$ (et donc α est du signe de θ), et dans lequel il faut vérifier que $\varepsilon = -\mathrm{sgn}(\theta)$, mais où on a le choix pour ε' (on essaie d'abord avec $-\varepsilon$ afin de minimiser α) ;
- $\mathcal{C}_2 = \{\frac{r'^2-r''^2+5R'^2+2r'R'\cos\theta}{4R'} \leq r'\cos\theta + R'\}$, enfin, où la seule contrainte est $\varepsilon' = \varepsilon$.

Cela étant, on peut en déduire χ, à partir de la première ligne du système original :

$$r'' \cos\chi = r' - R' \left[2\cos(\alpha+\theta) - \cos\theta\right],$$

ce qui impose :

$$\begin{array}{rrcl} & r' - R' \left[2\cos(\alpha+\theta) - \cos\theta\right] & \leq & r'' \\ \Leftrightarrow & r + d + \varepsilon R \left[2\cos(\alpha+\theta) - \cos\theta\right] & \leq & r - \varepsilon R \\ \Leftrightarrow & & & \\ & d \leq \varepsilon R \left[2\cos(\alpha+\theta) - \cos\theta - 1\right] & & \text{(H2)} \end{array}$$

Si jamais cette dernière condition n'est pas vérifiée, il convient de changer la valeur de ε' (si possible), ou celle de ε (on déclare forfait si aucune des valeurs de ε, et celles de ε' liées, ne permet de vérifier toutes les conditions).

Dans le cas contraire, on peut calculer χ, mais il faut encore s'assurer que le sens de parcours du second cercle correspond à celui du chemin, c'est-à-dire que l'on a finalement :

$$\varepsilon(\alpha + \theta - \chi) \geq 0 \qquad \text{(H3)}$$

En conclusion, dans le cas du changement de chemin depuis une distance d avec une direction θ, vers un arc de cercle de rayon r, et ce avec deux arcs de cercles de rayon R, la direction du premier étant donnée par ε, on pose :

$$\begin{array}{c} \varepsilon = -\mathrm{sgn}(d), \qquad \text{à priori,} \\ r' = r + d\ , \qquad R' = \varepsilon R\ , \qquad r'' = r - R', \\ \eta = \arctan\left(\frac{R'\sin\theta}{r' + R'\cos\theta}\right) \end{array}$$

en vérifiant l'hypothese H1. Ensuite, on considère ces paramètres afin de savoir dans quel cas on se trouve :

- $\mathcal{C}_0 = \left\{\frac{r'^2 - r''^2 + 5R'^2 + 2r'R'\cos\theta}{4R'} > \sqrt{r'^2 + R'^2 + 2r'R'\cos\theta}\right\}$, pour lequel il faut essayer avec l'autre valeur de ε ;
- $\mathcal{C}_1 = \{r'\cos\theta + R' < \frac{r'^2 - r''^2 + 5R'^2 + 2r'R'\cos\theta}{4R'} \leq \sqrt{r'^2 + R'^2 + 2r'R'\cos\theta}\}$, dans lequel il faut vérifier que $\varepsilon = -\mathrm{sgn}(\theta)$, et où on prend
  $\alpha = -\varepsilon \arccos\left(\frac{r'^2 - r''^2 + 5R'^2 + 2r'R'\cos\theta}{4R'\sqrt{r'^2 + R'^2 + 2r'R'\cos\theta}}\right) + \eta - \theta$
  (en gardant en réserve la possibilité d'utiliser un autre α, donné par une formule similaire où ε remplace $-\varepsilon$) ;
- $\mathcal{C}_2 = \{\frac{r'^2 - r''^2 + 5R'^2 + 2r'R'\cos\theta}{4R'} \leq r'\cos\theta + R'\}$, enfin, où on prend
  $\alpha = \varepsilon \arccos\left(\frac{r'^2 - r''^2 + 5R'^2 + 2r'R'\cos\theta}{4R'\sqrt{r'^2 + R'^2 + 2r'R'\cos\theta}}\right) + \eta - \theta$.

Il faut enfin vérifier que l'on peut calculer χ, c'est-à-dire que l'hypothèse H2 est correcte (sinon, on change α, si possible, ou ε), avant de prendre :

$$\chi = -\arccos\left(\frac{r' - R'\left[2\cos(\alpha+\theta) - \cos\theta\right]}{r''}\right).$$

et d'effectuer la dernière vérification, l'hypothèse H3.

Annexe B

Une heuristique pour la planification de mouvement

On a essayé de traiter ce problème dans le cadre le plus général, tout en effectuant les calculs sur des entiers. On note, dans cette partie, (s_i, v_i) la position actuelle (le temps étant t_i) et (s_f, v_f) la position à atteindre (l'horizon temporel étant t_f).

On s'intéresse aux positions les plus éloignées que l'on peut ralier à partir d'une position initiale [respectivement qui permettent de rejoindre une position finale]. Pour cela, il suffit d'étudier les cas limites, c'est-à-dire ceux correspondant à une trajectoire d'accélération maximale [respectivement minimale]. Ces cas sont répartis suivant une parabole dans le plan des indices d'abscisse et de vitesse, et sont de la forme :

$$(s_{ref} + 2.v_{ref}.t + a_{type}.t^2\ ,\ v_{ref} + a_{type}.t), \quad \text{où} \quad \begin{array}{l} ref \in \{i, f\}, \\ type = max \ \text{ si } \ ref = i, \\ type = min \ \text{ sinon,} \end{array} \tag{B.1}$$

t étant le temps nécessaire pour les atteindre depuis la position initiale [respectivement finale, le temps étant alors négatif] ; il faut donc que t soit inférieur (ou égal) à $t_f - t_i$ [respectivement supérieur à $t_i - t_f$] pour que la trajectoire à suivre reste dans l'horizon temporel.

Cela étant, on utilise la connaissance de ces cas limites pour déterminer un minorant du temps qu'il faudrait pour aboutir à la position finale. On commence par fixer des conditions d'accessibilité, déduites de la forme des paraboles données par l'équation B.1. La première vérifie que l'accélération maximale suffit pour atteindre la position finale (le problème ne se pose que lorsque la vitesse initiale est inférieure à la vitesse finale) :

$$s_f - s_i \geq \left\lceil \frac{{v_f}^2 - {v_i}^2}{a_{max}} \right\rceil. \tag{C1}$$

La seconde condition est symétrique, concernant l'accélération minimale, et ne se pose que lorsque la vitesse initiale est supérieure à la vitesse finale :

$$s_i - s_f \geq \left\lceil \frac{{v_i}^2 - {v_f}^2}{a_{min}} \right\rceil. \qquad \text{(C2)}$$

Ces deux conditions (C1 et C2) étant vérifiées, on peut aller de la position initiale à la position finale en accélérant autant que possible (la vitesse maximale étant donnée par v_{max}), puis en maintenant sa vitesse tant qu'on n'est pas obligé de décélérer (on suppose qu'il n'y a pas d'obstacle). La vitesse limite que l'on peut atteindre correspondrait (s'il n'y avait pas de vitesse maximale) à l'intersection de deux paraboles, l'une issue de la position initiale et correspondant a l'accélération maximale, l'autre aboutissant à la position finale avec une décélération maximale ; elle s'écrit donc sous la forme :

$$v_{lim} = min\left(v_{max}, \left\lfloor \sqrt{\frac{a_{max} v_f^2 - a_{min} {v_i}^2 - a_{max} a_{min}(s_f - s_i)}{a_{max} - a_{min}}} \right\rfloor\right)$$

On déduit de cette valeur l'abscisse où cette vitesse est atteinte lorsqu'on part de la position initiale, et celle où on doit commencer à freiner pour arriver à la position finale, notées respectivement s_1 et s_2 :

$$s_1 = s_i + \left\lfloor \frac{v_{lim}^2 - {v_i}^2}{a_{max}} \right\rfloor, \qquad s_2 = s_f + \left\lceil \frac{v_{lim}^2 - {v_f}^2}{a_{min}} \right\rceil.$$

Finalement, on obtient l'estimation du temps de parcours (tenant compte des conditions limites, des bornes d'accélération et de vitesse, mais pas des éventuels obstacles) en cumulant le temps d'accélération, celui à vitesse limite et celui de décélération :

$$t_{par} = \left\lfloor \frac{v_{lim} - v_i}{a_{max}} \right\rfloor + \left\lceil \frac{s_2 - s_1}{v_{lim}} \right\rceil - \left\lceil \frac{v_{lim} - v_f}{a_{min}} \right\rceil$$

Il est évident que lorsqu'un nœud génère une estimation de temps de parcours tel que $t_{par} > t_f - t_i$, la position finale n'a aucune chance d'être rejointe ; on peut alors éliminer le nœud de l'arbre de recherche.

Bibliography

[ABF88] F. Avnaim, J-D. Boissonnat, and B. Faverjon. A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 1656–1661, Philadelphia, PA (USA), Apr. 1988.

[BDG83] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. On the optimal control of robotic manipulators with actuator constraints. In *Proc. of the American Control Conf.*, pages 782–787, San-Francisco, CA (USA), June 1983.

[BL89] J. Barraquand and J-C Latombe. On non-holonomic mobile robots and optimal maneuvering. *Revue d'intelligence Artificielle*, 3(2):77–103, 1989.

[BLP85] R.A. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Trans. Systems, Man and Cybernetics*, 15(2):224–233, March/April 1985.

[Can87] J. Canny. *The complexity of robot motion planning.* PhD thesis, Dept of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, MA (USA), 1987.

[CDRX88] J. Canny, B. Donald, J. Reif, and P. Xavier. On the complexity of kynodynamic planning. In *Proc. of the IEEE Symp. on the Foundations of Computer Science*, pages 306–316, White Plains, NY (USA), Nov. 1988.

[DX89] B. Donald and P. Xavier. A provably good approximation algorithm for optimal time trajectory planning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 958–963, Scottsdale, AZ (USA), May 1989.

[DX90] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for cartesian robots and open-chain manipulators. In *Proc. of the ACM Symp. on Computational Geometry*, pages 290–300, Berkeley, CA (USA), 1990.

[DX91] B. Donald and P. Xavier. Time-safety trade-offs and a bang-bang algorithm for kinodynamic planning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 552–557, Berkeley, CA (USA), 1991.

[FL91] Th. Fraichard and C. Laugier. On-line reactive planning for a non-holonomic mobile in a dynamic world. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 432–437, Sacramento, CA (USA), Apr. 1991.

[Fra92] Th. Fraichard. *Planification de mouvements pour mobile non-holonome en espace de travail dynamique.* PhD thesis, Inst. Nat. Polytechnique de Grenoble, 1992.

[FS90] K. Fujimura and H. Samet. Motion planning in a dynamic domain. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 324–330, Cincinnatti, OH (USA), May 1990.

[HL92] M. Hassoun and C. Laugier. Reactive motion planning for an intelligent vehicle. In *Proc. of the International Symposium on Intelligent Vehicles*, Detroit, MI (USA), June 1992. IEEE.

[Kha86] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. Journal of Robotics Research*, 5(1), Spring 1986.

[KZ86] K. Kant and S. Zucker. Toward efficient trajectory planning: the path-velocity decomposition. *Int. Journal of Robotics Research*, 5(3):72–89, Fall 1986.

[Lat90] J-C. Latombe. *Robot motion planning.* Kluwer Academic Press, 1990.

[Lau88] J-P. Laumond. L'algorithmique du mouvement en robotique. In *Proc. of the colloquium "Géométrie discrète, géométrie algorithmique, passage du discret au continu"*, Grenoble (F), Nov. 1988.

[LG85] C. Laugier and F. Germain. An adaptive collision-free trajectory planner. In *Proc. of the IEEE Int. Conf. on Advanced Robotics*, Tokyo, (J), Sep. 1985.

[LP83] T. Lozano-Perez. Spatial planning, a configuration space approach. *IEEE Trans. Comput.*, 32(2), Feb. 1983.

[LPW79] T. Lozano-Perez and M.A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM*, 22(10):560–570, Oct. 1979.

[LS85] D. Leven and M. Sharir. An efficient and simple algorithm for a ladder moving in a two-dimensional space amidst polygonal barriers. In *Proc. of the ACM Symp. on Computational Geometry*, pages 211–227, 1985.

[Nil69] N.J. Nilsson. A mobile automaton: an application of artificial intelligence techniques. In *Proc. of the Int. Joint Conf. on Artificial Intelligence*, pages 509–520, Washington, DC (USA), 1969.

[Off89] Prometheus Office. "Functions" or how to achieve PROMETHEUS "objectives". Technical report, Stuttgart (FRG), July 1989.

[OY82] C. Ó'Dúnlaing and C. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111, 1982.

[PL90] T-J. Pan and R.C. Luo. Motion planning for mobile robots in a dynamic environment with moving obstacles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 578–583, Cincinnati, OH (USA), May 1990.

[RS85] J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *Proc. of the IEEE Symp. on the Foundations of Computer Science*, pages 144–154, Portland, OR (USA), Oct. 1985.

[SD88] Z. Shiller and S. Dubowsky. Global time optimal motions of robotic manipulators in the presence of obstacles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 370–375, Philadelphia, PA (USA), Apr. 1988.

[SH85] G. Sahar and J. H. Hollerbach. Planning of minimum-time trajectories for robot arms. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 751–758, St Louis, MI (USA), March 1985.

[SLG90] C.L. Shih, T.T. Lee, and W.A. Gruver. Motion planning with time-varying polyhedral obstacles based on graph search and mathematical programming. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 331–337, Cincinnatti, OH (USA), May 1990.

[SM85] K.G. Shin and N.D. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Trans. Autom. Contr.*, 30:531–541, June 1985.

[SS83a] J. T. Schwartz and M. Sharir. On the piano movers' problem: II. general techniques for computing topological properties of real algebraic manifold. *Advances in Applied Mathematics*, 4:298–351, 1983.

[SS83b] J.T. Schwartz and M. Sharir. On the piano movers' problem: I. the special case of a rigid polygonal body moving amidst polygonal barriers. *Commun. Pure Appl. Math.*, 36:345–398, 1983.

[Udu77] S.M. Udupa. Collision detection and avoidance in computer-controlled manipulators. In *Proc. of the Int. Joint Conf. on Artificial Intelligence*, pages 737–748, Cambridge, MA (USA), Aug. 1977.

[ZL89] D. Zhu and J-C. Latombe. New heuristic algorithms for efficient hierarchical path planning. Research Report STAN-CS-89-1279, Robotics Lab., Computer Science Dept, Stanford Univ. CA (USA), 1989.