Safe navigation within dynamic environments: a partial motion planning approach

Stéphane Renaud Petti

ECOLE DES MINES DE PARIS
Collège doctoral

N° attribué par la bibliothèque

|__|__|__|__|__|__|__|__|__|__|

T H E S E

pour obtenir le grade de
Docteur de l'Ecole des Mines de Paris
Spécialité «Informatique temps réel – Robotique - Automatique»

présentée et soutenue publiquement
par
Stéphane Renaud PETTI

le 11 JUILLET 2007

SAFE NAVIGATION WITHIN DYNAMIC ENVIRONMENTS: A PARTIAL MOTION PLANNING APPROACH

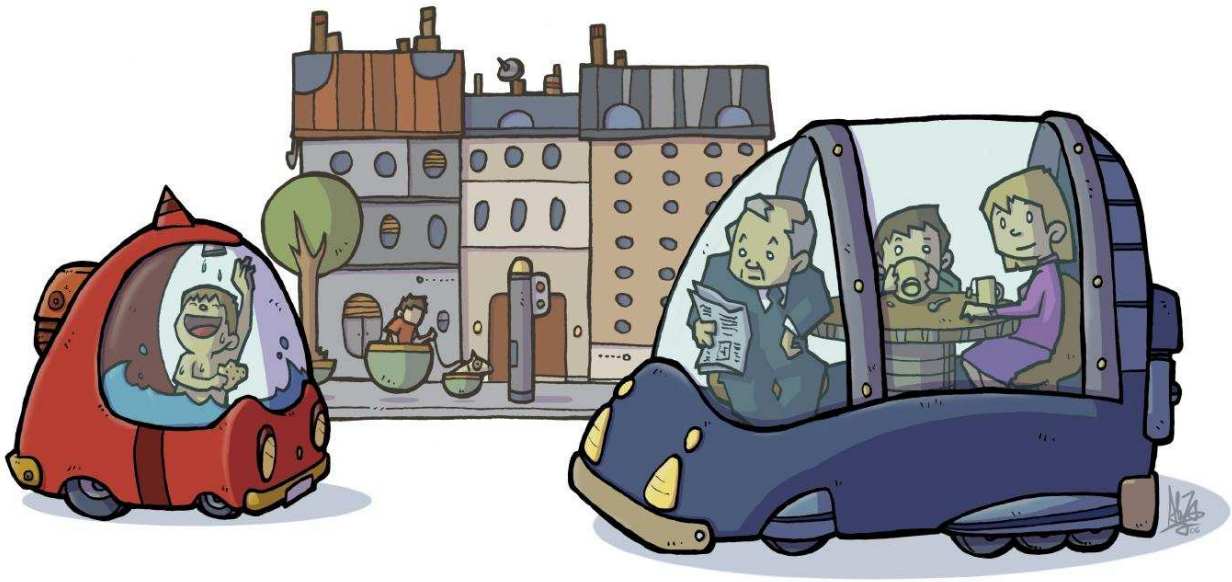
Directeur de thèse : Christian LAUGIER
Co-Directeur de thèse : Thierry FRAICHARD

Jury

M. Claude LAURGEAU	Président
M. Luis MONTANO	Rapporteur
M. Thierry SIMEON	Rapporteur
M. Michel PARENT	Examinateur

Safe Navigation within Dynamic Environments: A Partial Motion Planning Approach

(Navigation sûre en environnement dynamique: une approche par planification de mouvement partiel)



Au tout début de la thèse on s'autorise parfois à rêver à ce moment où il faudra écrire cette page, signe de l'accomplissement de plusieurs années de travail. On se remémore alors celles et ceux qui ont rendu cette aventure possible, pour prendre le temps de les remercier.

Au tout début, il y a une Société, Aisin AW, et certains de leurs membres qui ont cru et soutenu avec conviction mon projet. Je voudrais en particulier remercier mes collègues Européens M.Cervello, M.Knop, M.Diels et Japonais M.Kawa et, M.Fukaya. Puis il y a des rencontres capitales, avec l'INRIA tout d'abord, avec Michel Parent, qui a su immédiatement me faire confiance, Christian Laugier qui m'a accueilli sans hésitations dans son équipe et puis avec les Mines de Paris et Claude Laurgeau qui m'a ouvert les portes du centre de robotique CAOR, que je remercie sincèrement pour leurs marques de confiance et soutien sans failles.
Avec le travail qui avance de concert avec ses problèmes et doutes, ce sont les proches collègues de ces équipes qui permettent de reprendre confiance. Je remercie chaleureusement l'incontournable Chantal, reine d'IMARA et Armand, le Mc Gyver des Cycabs, pour leur humour, soutien et tous ces conseils avisés dont ils me faisaient part lors de nos pauses café matinales. J'ai à cœur de remercier également les talentueux Tony, Mickael, Laurent, Angele et mon vaillant collègue de bureau, l'extravagant mais non moins talentueux Rodrigo, pour toutes ces discussions passionnées sur des sujets aussi bien techniques que philosophiques, qui m'ont été si précieuses. Je voudrais également remercier Konaly et Fawzi pour la bonne humeur qu'il m'ont toujours communiquée aux Mines, merci également à Olivier, Arnaud, Yotam, et tous les autres membres d'IMARA, d'eMotion et du CAOR que je n'ai pas salué ici et pourtant qui m'ont permis grâce à chacune de nos conversations d'avancer un peu plus.
Bien entendu je n'oublie pas le plus vaillant de tous, Thierry Fraichard, qui armé de patience et conviction, m'a soutenu, conseillé et aidé tout au long de ce travail, durant plus de trois années, depuis le premier jour, jusqu'à ce dernier jour de soutenance et à qui je voudrais exprimer ma sincère gratitude.
Je voudrais également remercier Julien Akita, pour m'avoir offert la talentueuse illustration, unique, originale et pertinente qui préface ce manuscrit de thèse.

Et puis, par-dessus tout,
à mes parents,
et leur foi en moi, indéfectible, qui m'a sans nulle doute portée jusque là,
à ma femme, Ioulia,
sa gentillesse et sa patience, sans égale,
et à mon fils, Alexandre
son énergie rafraîchissante et sa comprehension…
et pour tout le reste, l'essentiel vraiment, qui ne trouve plus la place en ces quelques lignes.

Ainsi, je regarde aujourd'hui ces années de thèse, ces périodes de doutes, de stress mais aussi de satisfaction et voulais prendre le temps ici de rendre hommage à tous ceux qui ont fait que ces années ont été agréables et profitables et restent, plus qu'un souvenir, une expérience unique.

Contents

List of Figures

Chapter I

Introduction

1 Introduction

Driven by the dream of creating machines that would autonomously accomplish specific tasks, the interest to robotics goes back to early ages. From early automatons, to more recent *robots*[1] [Cap20], these machines always inspired fascination. Nowadays, a robot is defined as a complex mechanical device equipped with sensors and actuators. It is designed to perform complex tasks autonomously or with human supervision, that are too dull, dirty, dangerous or accurate for humans.

A robot can perform several tasks, like mating or grasping parts, moving around to explore or survey. Regardless of the task, the robot performs a sequence of actions, *eg* moving an arm, closing grippers, or propelling itself. Then, each action will result in a motion. Thus, in order to successfully accomplish its task, a robot must be able to plan, *ie* find on its own and in advance, the sequence of motions to execute. In robotics, this specific problem of planning a motion a priori is addressed by *motion planning* and is at the heart of the work presented here.

In its general form, the motion planning problem consists in finding a priori, a sequence of motions that guides a system to a predefined objective. The calculation of a motion plan relies on models of both the system and the environment in which it is placed.

The basic motion planning problem is often referred to as *path planning*. The path planning problem consists in finding a collision free *path*, *ie* a continuous sequence of *configurations*. A configuration is the set of independent parameters representing the po-

[1] Term coming from the czeck *robota*, meaning drudgery.

sition and the orientation of every part of the robot. Path planning, in its basic form, deals with free flying robots moving amidst stationary obstacles. The mechanical design of a free flying robot allows every part of it to freely rotate and translate such that its configuration changes are not constrained. In the late seventies, the concept of *configuration space* is introduced as a useful framework for the basic motion planning problem. In the configuration space, the robot is represented as a point and stationary obstacles as forbidden regions. The basic problem is therefore essentially geometric and deals with collision avoidance of stationary obstacles.

However, as soon as one of the main path planning hypotheses is broken, path calculation might not provide sufficient information for the robot to achieve its task. To begin with, the obstacles in the environment might move. In such a case, the time dimension must be considered. A time parametrization of the sequence of configuration becomes necessary. This introduces the notion of *trajectory*. In addition to the temporal aspects, the robot might be constrained by its kinematics or dynamics, which will limit its motion capability. Finally, the models on which motion planning relies might differ with the reality. The possible error, *ie uncertainty* impacts the motion at execution and might be incorporated at the planning level which further complicates the problem.

2 Problem

In robotics, it is important to always consider a robot as a real device with specific mechanical architecture and constraints, intended to perform real tasks within a real world. Industrial robots used in manufacturing lines used to be the most common form of robots and motivated most of the work in path planning. Manipulators are built to ensure that the end effector can rotate or translate freely. They generally operate in a known fixed environment. Nowadays, other robots are emerging, such as servicing robots, that are also finding their way into entertainment, home, health care.

As for transportation, a large effort has been put in major industrial countries in the last decade, into developing new kinds of intelligent transportation systems, the Cybercars [Par97], as a mean to address the problems of congestion, pollution and safety raised by the increasing usage of personal cars. In the long term, these innovative transportation systems are envisioned to autonomously drive people within a city. The mechanical platform of these car-like robots, exhibit constraints (kinematic, dynamic and actuator constraints) that restrict their motion capabilities. Besides, they evolve in dynamic environments, cluttered with other cars or pedestrians. Cybercars are at the center of our research

interest. The main goal of this research work is to provide this vehicle, the capability to autonomously plan its trajectory to a predefined goal while avoiding obstacles. We consider the real urban context within which these vehicles are aimed to work.

Such an environment brings a major constraint that has not been mentioned until now. A system placed in the real world, cluttered with moving obstacles, has the obligation to plan a trajectory within a limited time, otherwise it might be in danger by the sole fact of being passive. This limited available time is a *decision time constraint,* imposed by the nature and dynamicity of the environment. This constraint must be strictly fulfilled at execution for the system's safety and is therefore a **hard real-time constraint**. Although this constraint is of crucial importance, there is very little work in the literature taking this constraint into account explicitly. There are a few methods based on dynamic programming [PNIV01, Ste02] that succeed to plan very fast. The constraints of the system are however not easily incorporated, and the problems handled are usually of low dimension. Latest probabilistic methods as well have shown efficient schemes [HKLR02, BV02] for simple systems. Nevertheless, no matter how fast these methods are, none of them provide a guarantee on the computation time upper bound and therefore none of them account for the real-time constraint explicitly.

Therefore our problem turns into trajectory planning for a system :

1. under kinematic and dynamic constraints of the system
2. evolving within a dynamic environment
3. under a real time constraint

3 Contributions

- In our work we address the problem of navigation within a dynamic environment from a motion planning perspective. The main contribution of this work is to explicitly consider the real-time constraint dictated by a dynamic environment. Now, given the intrinsic complexity of the motion planning problem, computing a complete motion to the goal within the time available is impossible to achieve in most real situations. Partial Motion Planning (PMP) is the answer to this problem proposed in this work. PMP is a motion planning scheme with an anytime flavor: when the time available is over, PMP returns the best partial motion to the goal computed so far.
- **Safety** becomes a major issue for partial planning. When collision free condition might suffice for path planning problems, this is not the case when dealing with

dynamic environments. Safety must be considered from a different perspective and other criteria must be considered. At first the dynamics of the system must be taken into account. Indeed a dynamic system has a drift caused by its inertia, that moves a system even when commands are applied. Furthermore, when dealing with a dynamic environment, it is fundamental to account explicitly for the motion of the obstacles. Otherwise, even though the last state might be collision free, it might still be in danger. A guarantee that the system will never end up in a critical situation yielding an inevitable collision must be given. The second contribution of our work addresses this aspect and studies the safety issue from a more complete perspective suitable for our problem. The answer proposed in this work to this safety issue relies upon the concept of Inevitable Collision States (ICS) [FA04]. ICS is a concept that encompasses the dynamics of both the system and the moving obstacles. A strong safety guarantee is given to a partial plan by computing ICS-free partial motions.

- Finally, the last contribution of this work is to **demonstrate** the efficiency and robustness of the approach and **integrate** the algorithm on a real platform, a Cycab. We present the integration of PMP within a real navigation architecture. This architecture mainly rely on a laser scanner for the model perception and construction and a Real Time Kinematics GPS for the localization. As for the model prediction, we assume it is provided to our planner. We believe this assumption realistic considering latest results on model's prediction [WTT03, VF04], however we do not address it in this work. Nevertheless, we present the advantage of coupling PMP with such an approach, a Simultaneous Localization, Mapping and Moving Objects Tracking (SLAMMOT) algorithm. Furthermore, we detail how the planning and execution interleave. The execution of the trajectory is handled by a low level controller, a trajectory tracking controller, aimed at properly execute the planned trajectory. Actual experiments on a real platform, the Cycab are presented to validate the approach and the overall architecture.

4 Document Layout

We complement in chapter §2 the description of the problem and present a state of the art of the related work. Our approach is then presented in chapter §3. In chapter §4 we present an adaptation of PMP to the case study of a car-like robot and we present the results of our simulations as well as our experimentation on a real platform, that we have carried out for this case study in chapter §5. Finally, in chapter §6 we draw our final conclusions on the approach and, from the return of experience of our experiments, we discuss the perspective of this work.

Introduction

Le rêve de créer des machines qui pourraient accomplir des tâches spécifiques de manière autonome, et l'intérêt porté à la robotique plus généralement remonte à très longtemps. Des premiers automates aux plus récents robots, ces machines ont toujours engendré de la fascination. De nos jours, un robot est défini comme un dispositif mécanique complexe équipé de capteurs et d'actuateurs. Ils sont conçus pour accomplir, soit de manière autonome, soit sous la supervision d'êtres humains, des tâches complexes qui sont trop ennuyeuses, sales, dangereuses ou trop précises pour l'homme.

Un robot peut effectuer de nombreuses tâches comme l'assemblage et la saisie d'objets, ou l'exploration. Indépendamment de la tâche, le robot doit effectuer une séquence d'actions, eg. bouger un bras, fermer des pinces ou se propulser. Puis, chaque action conduit à un mouvement. Ainsi, afin d'accomplir avec succès une tâche, un robot doit être capable de planifier, cad. trouver par lui même et à l'avance, la séquence d'action à exécuter. En robotique, le domaine qui aborde ce problème de déterminer à priori un mouvement, s'appelle la planification de mouvement et se trouve au coeur du travail présenté ici.

Dans sa forme générale, le problème de planification de mouvement consiste à trouver a priori une séquence de mouvements qui amènent un système à un objectif prédéfini. Le calcul de ce plan s'appuie sur un modèle du système et de l'environnement dans lequel il évolue.

Dans sa forme de base, le problème de planification de mouvements est appelé planification de chemin. Le problème de planification de chemin consiste à trouver un chemin sans collision, cad. une séquence continue de configurations sans collision. Une configuration est un ensemble de paramètres indépendants qui représentent la position et l'orientation de chaque partie d'un robot. Dans sa forme primitive, le problème de planification de chemin considère des objets qui évoluent sans contraintes dans un environnement statique. Le concept mécanique d'un robot "free flying" permet à chacune des parties de ce robot d'effectuer rotation et translation librement, de telle sorte que chaque changement de configuration ait lieu sans contraintes. Dans la fin des années '70, le concept d'espace des configurations est introduit comme un formalisme très utile pour le problème de la planification de chemin. Dans l'espace des configurations, le robot est représenté par un point et les obstacles comme des régions de l'espace interdites. Le problème de base est alors essentiellement géométrique et porte sur les évitements d'obstacles statiques.

Cependant, des que une des hypothèses de base du problème n'est plus remplie, il est probable que le calcul de chemin ne fournisse plus alors toutes les informations nécessaires au robot pour accomplir sa tâche. Tout d'abord, les obstacles dans l'environnement peuvent

bouger. Dans ce cas, l'aspect temporel doit être pris en compte. Une parametrisation en fonction du temps de la séquence de configuration devient alors nécessaire. Cela introduit la notion de trajectoire. En plus des aspects temporels, le robot peut être contraint par sa propre cinématique ou dynamique, ce qui limite ses mouvements. Enfin, le modèle sur lequel la planification de mouvement se base, peut être différent de la réalité. Cette erreur possible, ou incertitude a un impact sur le mouvement lors de son exécution et peut être prise en compte dés la planification ce qui complique largement le problème.

Problème

En robotique, il est important de toujours considérer le robot comme un dispositif réel avec une architecture et des contraintes mécaniques spécifiques, dans le but d'accomplir des tâches réelles dans un monde réel. Les robots industriels utilisés dans les usines réelles étaient jusqu'à présent les plus communs et ont motivé l'essentiel du travail en planification de chemin. Les bras manipulateurs sont conçus pour permettre à son extrémité de tourner ou translater librement. Ils opèrent en règle générale dans un environnement fixe et connu. De nos jours, de nouveaux types de robots émergent, comme des robots grands publics que nous trouvons dans le divertissement, à la maison ou les soins de santé.

Quant au transport, un effort important, dans la majeure partie des pays industrialisés durant les dix dernières années, a été fait pour développer de nouveaux types de systèmes de transports intelligents, les Cybercars, comme moyen pour affronter les problèmes de congestion, pollutions et sûreté, soulevés par l'usage toujours grandissant des véhicules personnels. A long terme, il est envisagé que ces nouveaux systèmes de transports intelligents, soient capables de transporter les gens au sein de la ville, de manière totalement autonome. Ces robots voitures sont soumis à des contraintes (mécaniques, dynamiques, au niveau des actuateurs) qui restreignent la capacité de mouvement du système. De plus, ils évoluent dans un environnement dynamique, ou d'autres objets (voitures ou piétons) évoluent. Les Cybercars sont au centre de notre travail de recherche. L'objectif essentiel de ce travail est de pouvoir fournir à ce type de véhicule, la capacité de planifier de manière autonome sa trajectoire vers un but choisi tout en évitant les différents obstacles. Nous considérons l'environnement urbain réel, comme étant celui où ces véhicules doivent pouvoir progresser.

Ce type d'environnement apporte une nouvelle contrainte, dont nous n'avons pas encore parlé. Un système placé dans un environnement réel, peuplé d'obstacles mobiles, a l'obligation de planifier une trajectoire dans un temps limité, sinon il peut se retrouver en danger, par le seul fait d'être passif. Cette limite de temps disponible est une contrainte

de décision imposée par la nature et dynamicité de l'environnement. Cette contrainte doit être remplie strictement durant l'exécution du mouvement et est pour cette raison une contrainte temps réel dure. Bien que cette contrainte soit d'importance cruciale, il y a paradoxalement peu de travaux dans la littérature qui prennent cette contrainte en compte. Il y a certaines méthodes basées sur la programmation dynamique qui réussissent à planifier très rapidement. Les contraintes liées au système sont cependant difficiles à prendre en compte et les problèmes abordés sont généralement de dimension peu élevée. Les récentes méthodes probabilistes ont également montré qu'elles peuvent être très efficaces pour des systèmes simples. Néanmoins, aussi rapides soient elles, aucune de ces méthodes ne garantit une borne en temps de calcul et ainsi aucune ne prend en compte de manière explicite cette contrainte temps réel.

Ainsi notre problème devient un problème de planification de trajectoire pour un système qui :

1. *est soumis à des contraintes cinématiques et dynamiques*
2. *évolue dans un environnement dynamique*
3. *est soumis à une contrainte temps réel*

Contributions

- *Dans ce travail, nous adressons le problème de navigation en milieu dynamique d'un point de vue planification de mouvement. La contribution principale de ce inclutravail est la prise en compte explicite de cette contrainte temps réel définie par l'environnement. Compte tenu de la complexité intrinsèque au problème de planification de mouvement, calculer un plan vers le but dans un temps limité est impossible dans la plupart des cas. La planification de mouvement partiel (PMP) est la réponse apportée à ce problème que nous proposons dans ce travail. PMP est une méthode de planification de mouvement ayant la faculté de retourner à tout moment le meilleur plan possible vers le but qui a été calculé jusqu'ici.*

- *La sûreté devient un élément essentiel pour la planification partielle. Quand la condition de non collision est suffisante pour les problèmes de planification de chemin, ce n'est plus le cas lorsqu'un environnement dynamique est pris en compte. Dans ce cas la notion de sûreté doit être étudiée sous une autre perspective avec d'autres critères. Tout d'abord, la dynamique du système doit être prise en compte. Par exemple, l'inertie d'un système dynamique peut le faire dériver sans qu'aucune commande ne soit appliquée sur celui-ci. De plus, quand un système évolue dans un*

environnement dynamique, il est fondamental de prendre en compte de manière explicite le mouvement des obstacles. En effet, quand bien même un état est sans collision à un moment donné, il se peut qu'il soit en danger à l'étape suivante. Une garantie sur le fait qu'un système ne va jamais se mettre dans une situation critique qui aboutirait à une collision inévitable, doit être fournie. La seconde contribution de ce travail traîte cet aspect et étudie la sûreté sous un angle plus complet et approprié à notre problème. La réponse proposée dans ce travail concernant le problème de sûreté réside dans l'utilisation du concept des Etats de Collisions Inévitables (ICS). ICS est un concept qui inclut la dynamique du système et de l'environnement. Une garantie de sûreté forte est donnée à un plan partiel en calculant un plan sans ICS.

- *Enfin la dernière contribution de ce travail consiste à démontrer l'efficacité et robustesse de l'approche et d'intégrer l'algorithme sur une plate forme réelle, un Cycab. Nous présentons l'intégration de PMP au sein d'une architecture de navigation réelle. Le couplage avec la perception est essentiellement basé sur l'utilisation d'un laser scanner et d'un GPS cinématique temps réel (RTK). L'avantage de l'intégration avec un algorithme de localisation et construction de carte simultanée (SLAM) est également présenté. Quant à la prédiction du modèle, nous partons du principe que cette information est fournie au planificateur. Nous pensons que cette hypothèse est réaliste compte tenu des récents travaux en la matière, cependant nous n'abordons pas ce sujet dans ce travail. De plus nous détaillons comment la planification et l'exécution se synchronisent. Le contrôleur bas niveau est un algorithme de suivi de trajectoire qui a pour but d'exécuter la trajectoire planifiée de référence. Des résultats d'expérience sur plateforme réelle sont présentés pour valider l'approche et l'architecture générale.*

Plan du document

La description plus détaillée du problème ainsi que de l'état de l'art sont présentés dans le chapitre §2*. Notre approche est ensuite présentée dans le chapitre* §3*. Dans le chapitre* §4 *nous présentons l'adaptation de notre approche à un cas particulier, celui du robot type voiture et présentons dans le chapitre* §5 *les résultats de simulations et expérimentaux sur plateforme réelle, que nous avons pus faire pour ce cas d'étude. Finalement, dans le chapitre* §6 *nous présentons nos conclusions finales et discutons des perspectives possibles de ce travail compte tenu du retour d'expérience des tests entrepris.*

Chapter II

Problem and existing works

1 The General Problem

From a general point of view, this work is aimed at developing a navigation method for **autonomous vehicles** evolving within **dynamic environments**. A navigation method consists in generating and executing a motion to a predefined objective while avoiding the obstacles present in the environment. Such methods are at the heart of the motion strategy of autonomous robots. In the literature, the general navigation problem has been essentially addressed from two distinct perspectives, the *reactive* one and the *deliberative* one.

The reactive approaches compute one action at a time to be performed during the next time step. The deliberative approaches, at the opposite, are aimed at calculating the **complete** sequence of actions to reach the goal.

In the following section, we propose to review the most significant reactive approaches and discuss their disadvantages. Following this presentation we concentrate on the deliberative approaches, namely motion planning. We review the different methods and their capability to handle different constraints. We finally discuss recent techniques that have motivated us to formulate the problem addressed in this work from the deliberative perspective.

2 A Reactive Perspective

Initial work on real robots required the ability to "react" to the environment in order to avoid obstacles. The necessity to build schemes fast enough such that the robot does not collide with its environment led several authors to propose algorithms later referred to as reactive approaches [BK91, Kha96, FS98, KS98]. In order to be fast and therefore limit the computation time, a reactive scheme consists in calculating at each time step only the next action that will move the robot closer to the goal without colliding with its surroundings. The complete knowledge of the environment is not necessary to compute a single motion, and in most cases the local information of the direct surrounding of the robot will be sufficient. Thus, reactive methods are very well suited for the navigation of real robots equipped with on-board sensors (ultrasonic, laser, ...). This navigation refers also to as sensor based navigation. However, the calculation of one single motion step limits the lookahead *ie* the ability to anticipate and judge whether a better path would be available. As a consequence, robots can be navigated to areas from which it will never escape. In fig. II.1 the robot might be trapped at execution in two obstacles. These ¨traps¨ or local minima can be formed by obstacles of concave shape and cannot be anticipated using reactive schemes. This certainly is their major disadvantage.

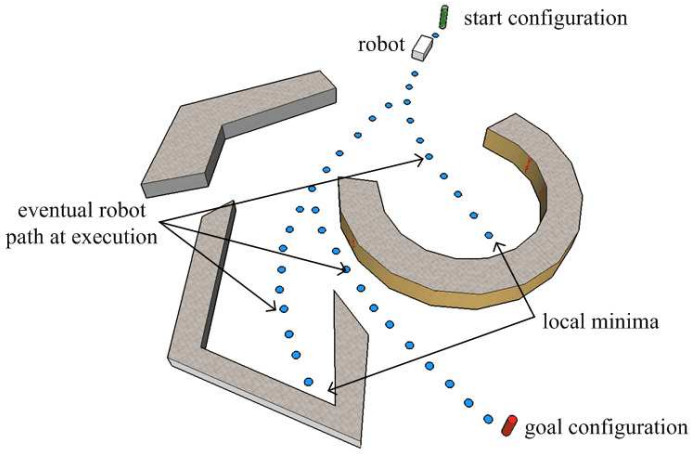


Figure II.1: Obstacles that form local minima for the robot's path.

Another consequence stemming from the lack of lookahead is the problem these schemes exhibit in some situations to converge to the goal rapidly. Nevertheless, several reactive methods paved the road to recent complex and efficient techniques. We propose to discuss these methods, that mostly differ from the assumptions on the characteristics of environment and the robot itself.

- The **Potential Field** method [Kha86] consists in generating an action calculated

by combining the attractive force of the goal and repulsive forces of the obstacles. It is very well suited for small mobile robotic platforms equipped with ultrasonic sensors for instance. A potential field (represented as a mesh in fig. II.2 of value P) is calculated online from the obstacle information given by the sensors and the next action is calculated. Interestingly, in case a complete model of the environment is provided, a complete potential field of the world can be computed. In this case a global form is derived using a navigation function which is also referred to as a feedback motion planning scheme [RK88]. However, the calculation of such a function, without other local minima than the goal, is extremely difficult in general, and has been developed for circular obstacles mainly. Another disadvantage of this method is that it is not suitable for dynamic environment as the force calculation does not account explicitly for the obstacles motion.

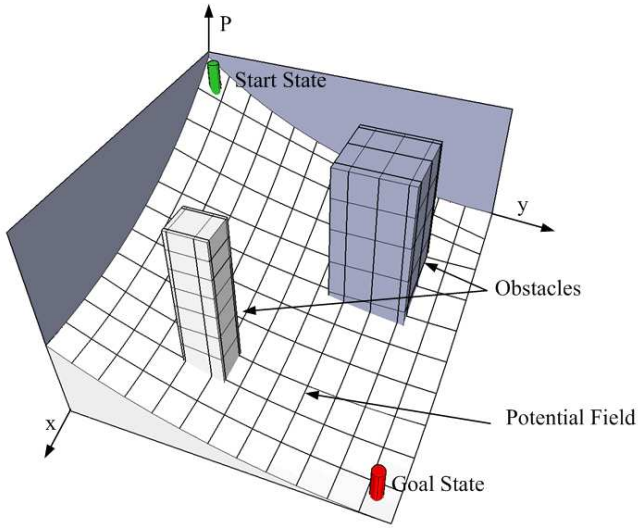
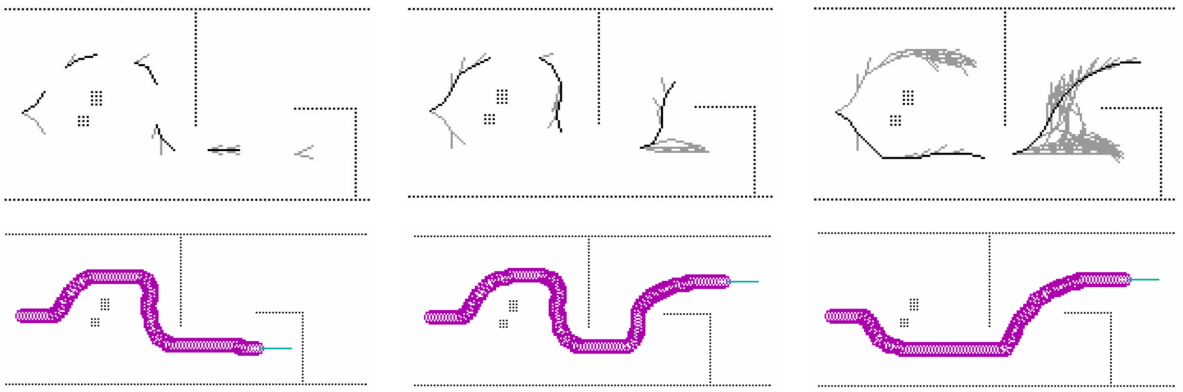


Figure II.2: Potential Field of an environment with two obstacles.

- The **Vector Field Histogram** (VFH) method introduced in [BK91] uses a 2D Cartesian histogram as a world model. It can be updated in real time using information from on board sensors. This histogram grid is based on earlier certainty grid [Mor88] and occupancy grid [Elf89] where each cell information represents the probability of having an obstacle at this point. This grid is translated into a polar histogram and the possible decision results from using the grid below a certain threshold value. A cost function based on the goal direction, the robot's current orientation and the wheels angle, is used to make the decision between the potential authorized directions and define the orientation of the next step motion. The speed is calculated at a second stage, as a function of the distance from the mobile robot and the goal state. This approach is limited by the use of arbitrary heuristics the

tuning of which largely influence the behavior of the algorithm. Besides, the use of polar grids reduce the field of view and thus prevent sometimes the system from choosing the most suitable orientation. An extension, VFH* method, is proposed later in [UB00] to improve the local nature inherent to the original scheme. The idea consists in using a map and calculate a small tree in $\mathcal{CS}$, of a sequence of collision free pose, *ie* position and orientation of the system, a few steps ahead and from different starting configuration (see fig. II.3). This sequence is therefore geometric and cannot be used directly as a control input for the robot. The local search is based on a graph search using dynamic forward programming. This augmented lookahead improves indeed the faculty of the robot to escape local minima, while providing a smarter indication on the next orientation to use, thus improving the convergence of the scheme. This method however does not consider dynamic environments. Even though, kinematic constraints are considered in a variant of this scheme [UB98], this method does not account for dynamic constraints (from the system and the environment) which is its major limitation.



(a) exploration tree and executed trajectory (1 step ahead)

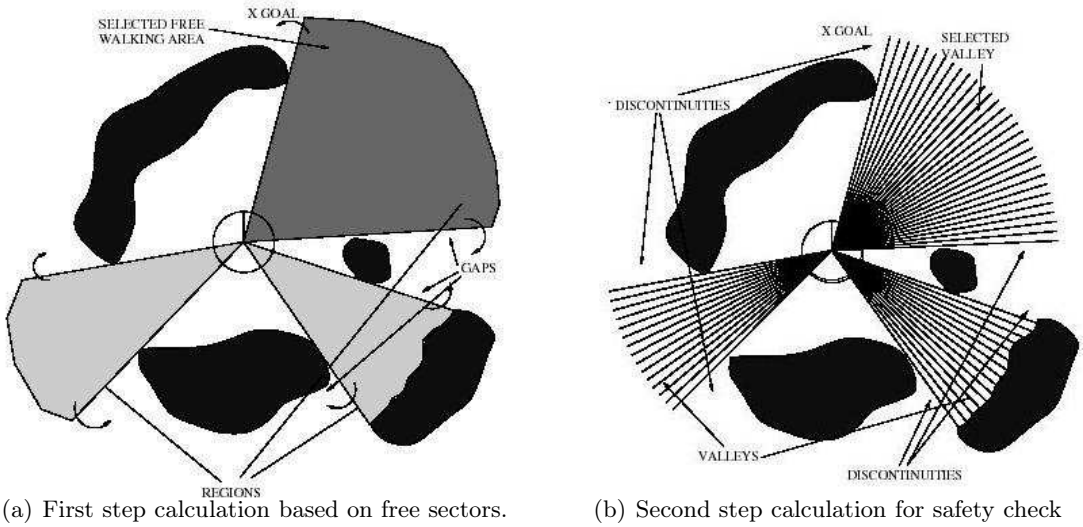
(b) exploration tree and executed trajectory (2 steps ahead)

(c) exploration tree and executed trajectory (5 steps ahead)

Figure II.3: VFH* method. (source: [UB00])

- The **Nearness Diagram** (ND) [MM00] method consists in analyzing a situation from two polar diagrams. These diagrams are built by means of on board sensors. One diagram is used to extract information of environmental characteristics and identify the immediate goal valley (see fig. II.4(a)), and the second one is used to define the safety level between the robot and the obstacles by identifying the closest one (see fig. II.4(b)). ND is similar to the VFH method that uses polar histogram. The kinematic constraints of mobile robots are taken into account in [MMSV02] similarly to all other previously presented methods, by approximating its path by straight segments and arc of circles. The decision is taken by a choice of suitable

controls in the ego-kinematic space, which is a representation of the world in terms of distance and radius of curvature that describe the arc of circle from the robot to an obstacle. The Global ND [MMSA01] then is a hybridization of ND combined with a NF1 function in order to get more lookahead and avoid traps similarly to other methods. An ego dynamic space is presented to account for dynamic constraints in [MMK02].



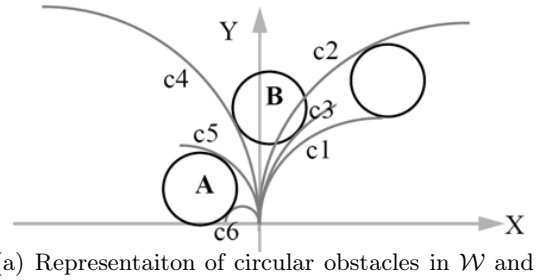
(a) First step calculation based on free sectors. (b) Second step calculation for safety check

Figure II.4: Nearness Diagram method (ND).

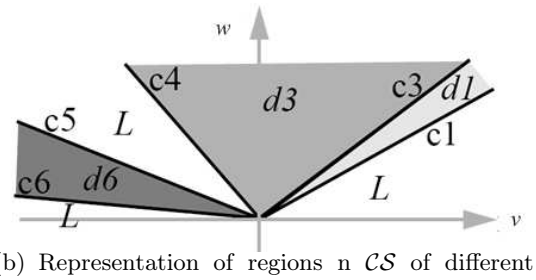
- The **Curvature Velocity** method (CVM) was introduced in [Sim96] as a local obstacle avoidance technique for indoor mobile robots accounting for their kinematic and dynamic constraints (acceleration and velocity bounds). This technique consists in finding the next suitable speed (rotational w and linear v speed) as this information is directly a command for the mobile robot. Given the physical and environmental constraints on the velocities, commands for the robot are chosen by optimizing an objective function. The objective function is designed so as to prefer high speeds, curvatures that travel longer before hitting obstacles and should try to orient the robot to head in the desired goal direction. The main principle consists in directly exploring the velocity space $\mathcal{VS}$, the space of all possible speed for the robot. The main difficulty consists in defining obstacles of the real environment in this $\mathcal{VS}$. The CVM method uses arc of circles of different curvatures (noted ci's) in order to consider the robot's kinematics. Distances to the obstacle (noted di's) that the robot would travel before hitting the obstacle, are calculated for all curvatures (in fig. II.5(a), obstacles are represented by circles and the distance between the robot, located at the origin, and the obstacles, by the distance of the arc of circles

hiting the obstacles). Then, the method consists in mapping this distance and curvature information from the workspace $\mathcal{W}$ to the velocity space $\mathcal{VS}$. Interestingly, in $\mathcal{VS}$, the curvature information is described by a half-line. This half-line defines the boundary of a region of a specific weight, which is the shortest distance to an obstacle as illustrated in fig. II.5(b). In order to account for the physical limitation of the system, a maximum acceleration constraint is set, and a maximum velocity defined in $\mathcal{VS}$. In fact, at each step, a new velocity bound is calculated from the system's acceleration bounds and the time step (see fig. II.5(c)). Finally, the candidate velocity pair that maximizes the objective function is chosen as the best candidate for the next step. The disadvantage of the weighted cost function is to depend on the choice of weights which might result in a non uniform robot's behavior, depending on the situation. The Steering Angle Method [FBL94] is very similar to this method but only rotational velocity is calculated. The Lane-Curvature method is presented later in [KS98], to overcome the main shortcomings of the CVM. For instance, at an intersection, CVM method fails to guide the robot into an open corridor toward the goal direction. In fact it often passes over some paths which are at right angles to the current robot's orientation. Besides, as all local methods, it lets the robot head towards an obstacle, even if there is a clear space around it. These problems stem from the fact that CVM chooses commands based on the collision-free length of the arcs assumed to be robot's trajectories and it does not pay much attention to collision free directions. It seems that the Lane Method is here to finally pre-process the environment without considering the robots dynamic. Therefore LCM turns into a two step approach that takes the free directions into account in a first step and then the collision free arc length. In order to find a heading direction the lane method divides the environments into lanes oriented in the direction of the desired goal heading. This methods account for more constraints as VFH, however, the dynamic environment is not handled more explicitly.

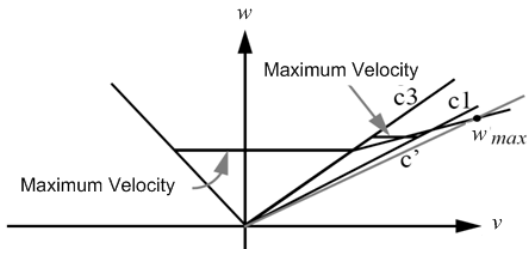
- The **Dynamic Window Approach** (DWA) introduced in [FBT97] is certainly one of the most popular reactive approach. The DWA is in many ways similar to the CVM method, with the difference that a discretized velocity space is built. A dynamic window of feasible velocities is built around the robot's velocity. A velocity is considered solution if it fulfills the systems constraints of maximum speed and maximum acceleration. Furthermore, the velocity must allow the robot to stop before hitting the obstacle. In case nonholonomic systems are considered, arc of circles are considered instead of straight lines. To overcome the lack of lookahead, an augmented method, the Global DWA, is presented by [BK99]. In addition to the sensor readings used to built the model, a map is incorporated in order to allow



(a) Representaiton of circular obstacles in $\mathcal{W}$ and the distance of the robot to the obstacles.



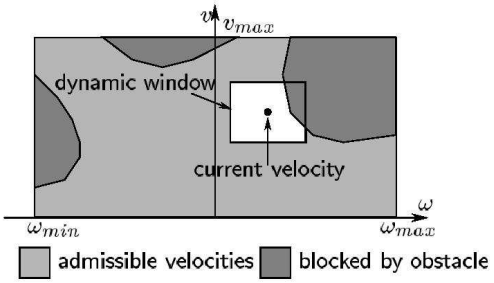
(b) Representation of regions n $\mathcal{CS}$ of different weights accounting for the proximity of an obstacle to the robot.



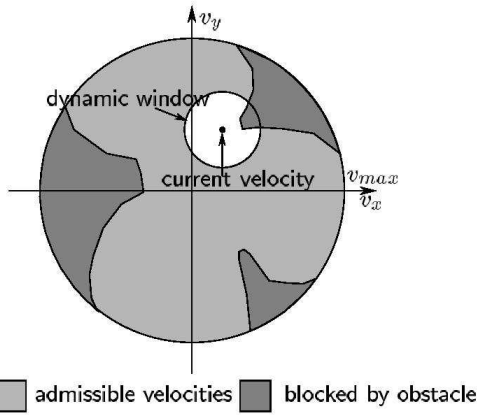
(c) Repesentation in $\mathcal{CS}$ of the velocity constraint and physical limitations.

Figure II.5: Curvature Velocity method (CVM).

better lookahead. Thus, the DWA is combined with a grid based global navigation function in order to get a more goal directed scheme and get information about the free space connectivity in order not to be trapped in local minima. This method is limited by the model construction as well as the parameters tuning. This method finally is mostly suitable for static or low dynamic environments.



(a) dynamic window for a holonomic system



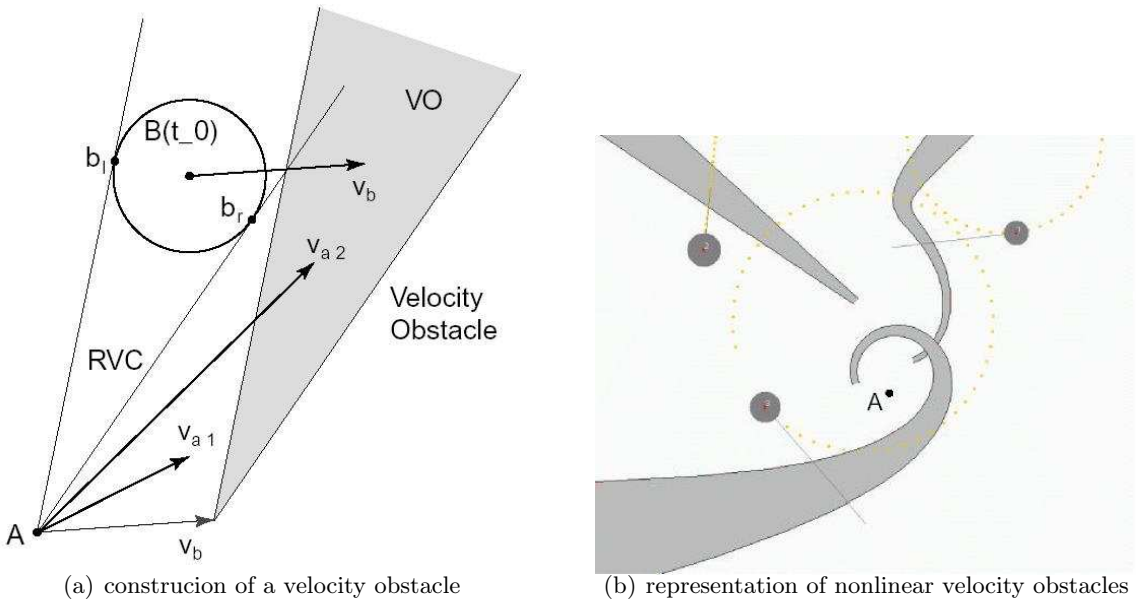
(b) dynamic window for a nonholonomic system

Figure II.6: Dynamic Window approach (DWA).

- The concept of **Velocity Obstacles** (VO) introduced in [FS98], allows the representation within the velocity space of moving obstacles (see fig. II.7(a)). This technique differs from the previous one by the capability of taking explicitly into account the environment dynamicity. Velocities (*ie* speed and direction) of the surrounding objects are assumed to be known. Having this information, moving obstacles are transposed into the velocity space (the velocity pair v_x and v_y, resp. along the x-axis and y-axis in this case) and represented by velocity obstacles, regions of forbidden velocities. A velocity if forbidden if it eventually yields a collision with the moving object. Velocity for the next time step is selected among the subset of allowed velocity vectors. This method also permits to consider objects moving along arbitrary trajectories (see fig. II.7(b)). An incremental motion planning strategy is used in [LSSL02] using this concept. The incremental approach calculates at each time step, a collision free velocity vector. A sequence of such admissible velocities is chosen and supplied to the system. From the resulting position another free velocity vector is calculated according to the model of the environment, one step further. A tree of free velocity controls is built defining thus a complete plan. This technique is the first approach to our knowledge that is aimed at calculating a few motions in advance. In order to account for dynamic and kinematic constraints, a pre-calculation of admissible velocities V_{adm} has to be done. Each node is determined by applying a safe linear constant velocity. For real world applications however, the errors introduced during the transition between two different linear velocity control, might put the robot in danger as the real robot displacement will differ from the piecewise linear one which is assumed when calculating safe controls.

- A more recent approach [OM05] presents a method that addresses robot subject to kinematic and dynamic constraints evolving within a dynamic environment. Similarly to several methods already described, the key of this method consists in mapping these constraints into a velocity-time space $\mathcal{VS}xT$. Unlike DWA, the proposed method accounts for the motion of the surrounding obstacles. The possible paths are represented using a similar path discretization as in CVM or DWA (arc of circles and straight segments). Then, for every surrounding moving obstacle a surface of forbidden velocities is calculated based on possible collisions. Limiting the window of feasible velocities to account for the dynamics of the system, the best velocity is chosen within the free velocity space while maximizing the convergence to the goal. Interestingly, this method enhances the concept of the DWA approach by considering dynamic environments.

Thus, reactive approaches have been very popular and widely used on several real platforms, thanks to their ease of implementation, a low computation cost and the actual



(a) construcion of a velocity obstacle (b) representation of nonlinear velocity obstacles

Figure II.7: Velocity Obstacle approach.

difficulty to observe and model the environment. However, their inherent limitation is poor lookahead that conducts the robot to be trapped in local minima during its trip, and as a consequence a weak convergence to the goal. Besides, reactive methods are very efficient when they directly work in the velocity space that is in most cases the natural command space of the robot (steering angle control homogeneous to the rotational speed w and longitudinal speed v). However, certain systems exhibit constraints that are difficult to model explicitly in this $\mathcal{VS}$. Car-like robot for instance have kinematic constraints that are taken into account by considering its path as a sequence of arc of circles and straight segments [Sim96]. This approximation however introduces a discontinuity in the path curvature, which represents an instantaneous change in the wheels' orientation. At execution, the system has to stop at each curvature discontinuity, *ie* at each junction between a straight segment and an arc of circle, otherwise the real motion cannot comply with the one calculated by the reactive scheme. Dynamic constraints as well must be approximated to be taken into account. It is done by reducing the control space to a set of physically possible controls in order to account for the bounded acceleration and deceleration of a system [FBT95, Sim96, MMSV02]. As we have described, even though most of the reactive methods have been recently modified so as to improve these limitations, the most important issue remains the fact that, at the exception of [FS98, LSSL02] and [OM05], none of the presented schemes do account explicitly for the dynamic nature of the environment, hence becoming useless in such an environment. Therefore, in our opinion, reactive methods present significant weaknesses in the context of our work. This

observation is the ground motivation for us to address our problem from the deliberative perspective.

3 A Deliberative Perspective

3.1 The Basic Problem

The paradigm of *deliberative approaches* is usually referred to as *motion planning*. We define the motion planning problem as:

Definition 1
Motion planning is a priori determination of the motion strategy based on a model of the world that will take the robot from its current position to a goal position.

The basic motion planning problem has been essentially motivated by the industrial use of manipulator arms robots ever since 1961 with the introduction of Unimate [1]. Indeed, widely used in production lines, these robots operate via an end effector which is mechanically designed to freely translate and rotate in usually a known and fixed environment. Under these two conditions (a robot free of move and a static environment), the basic motion planning problem consists in defining a sequence of *pose* or *configuration* of a robot, collision free to the goal. Formally, a configuration is the set of independent parameters that uniquely define position and orientation of every part of a robot, and the *configuration space* $\mathcal{CS}$, the set of all possible configurations [LP81]. Such a sequence is usually referred to as a *path* and the basic problem as *path planning*.

The context is particularly important in robotics. As soon as one goes beyond the scope of manipulator arms, the main hypotheses of path planning are violated. This is the case in our work since it focusses on intelligent cars evolving within cities. Indeed, cars are robots constrained by their kinematics and dynamics. Besides, in our work they are assumed to evolve in an environment cluttered with moving obstacles (pedestrians, other cars). This latter constraint, the dynamic nature of the urban environment, particularly complicates the motion planning problem as we will see in this chapter.

In the following of this chapter, we first present in section 3.2 the main types of motion planning strategies and their motivation from a complexity point of view. We explain the basics of these techniques, as this will be necessary latter in this work. At second, detail the extensions of the basic problem that we just mentioned, as we believe this will

[1] In 1961 the first Unimate robot, manufactured by Unimation Inc., is shipped from Danbury, Connecticut and installed in a plant of General Motors in Trenton, New Jersey. The robot lifts pieces of metal from die casting machine and stacks them.

ease the understanding of the remaining of this work. Thus, in section 3.3.1 we detail the constraints that impact the original motion planning problem, namely the kinematics and the dynamics. Then, in section 3.4, we introduce formally the main concern of this work,*ie* the problem of motion planning in a dynamic environment and present the related existing works. Finally, we conclude this chapter by pointing out fundamental observations in section 3.5 and 3.6 that lie at the heart of this work.

3.2 Complexity and Main Strategies

In this section, we review the main motion planning strategies that have been developed over the last years. We study the evolution of these basic methods, from the perspective of algorithm's completeness and detail the three major classes of motion planning strategies, namely the complete approaches, the resolution complete approaches and the probabilistic complete approaches.

Complete Algorithms Approaches Motion planning algorithms are evaluated in terms of their *completeness*, which is defined as follows:

Definition 2
An algorithm is said to be complete if it returns a valid solution to the motion planning problem if one exists or returns failure if there is no solution.

We do not review in this work the well-known complete motion planning approaches addressing the basic motion planning problem. For a thorough study and state of the art of these methods, we refer to the work of [Lat91]. Even though, the study of the extension of the basic motion planning problem will be presented in the next section, let focus just a moment on complete schemes that address such extensions to make the following observation: there exist very little work, and to our knowledge only the work of [O'D87] and [FS89], which propose complete motion planning approaches, for extensions of the basic problem in very simple cases. In [O'D87] a body moving from a state A to a state B is considered, while avoiding collision with a set of moving obstacles. The motion must satisfy an inertial constraint: the acceleration cannot exceed a given bound. In this work a polynomial-time motion-planning algorithms is given for the case of a particle moving in one dimension. In [FS89] a path planning technique in the presence of moving obstacles is presented, based on the exact cell decomposition technique. The methodology is to include time as one of the dimensions of the model of the world. This enables the authors to regard the moving obstacles as being stationary in this extended world. For a solution to be feasible, the robot must not collide with any other moving obstacle, and

also must navigate without exceeding the predetermined range of velocity, acceleration and centrifugal force. The authors investigate an appropriate model to represent the extended world for the path-planning task and give a time-optimal solution using this model.

An explanation for such a few available techniques comes from the study of the complexity of the complete motion planning algorithms. It is interesting indeed to have a better understanding on the difficulty of the problem. Both *lower bounds* which give indication on the difficulty of the problem itself, and *upper bounds* given by the existence of algorithms, have been presented in the literature. In 1979, the first lower bound complexity of motion planning is proved to be PSPACE-hard [Rei79]. Several following results for a variety of extensions also showed PSPACE complexity (Planar linkage [HJW84], multiple rectangles [HW86], planar arm [JP85]). The *decidability* of the motion planning problem, which relates to the fact that a solution to the problem exists, was established by [SS83] using the cylindrical algebraic decomposition in the 3D workspace. This algorithm set the first upper bound for the global motion planning problem and gives a time complexity to be twice exponential in the dimension of the space, and polynomial in the geometric complexity of the obstacle. A few years later, a roadmap based algorithm solved the same problem with singly exponential complexity in the space dimension [Can87]. This algorithm remains the most efficient algorithm currently available for solving the general motion planning problem. Good solutions exist, however for limited cases only, *eg* polynomial-time algorithms when the workspace is a plane [OY82, LS85, ABF88]. As soon as further constraints are added, the problem becomes intractable (*eg* kinematic constraints NP-hard [RW98], dynamic constraints NP-hard [RS85], [CR87]). As a consequence, the discouraging complexity of the problem and the need for practical algorithms motivated some authors to weaken its requirements.

Resolution Complete Approaches The main idea is to discretize the search space and build a conservative approximation of the free space. Some approaches conduct a search over grids of fixed resolution in the explored space. The size of the grid defines the resolution of the algorithm. They result in weaker guarantees that the problem will be solved. Indeed, for a given resolution, if a solution exists, the algorithm will find it, if no solution is found, it does not mean the problem does not have a solution, since the solution might have been found for a smaller resolution level. These approaches are called *resolution complete*. In [DXCR93] the first polynomial-time algorithm with a proved goodness and running time bound is presented, for a robot moving from a starting position to a goal position while obeying both kinematic (joint limitations and obstacles) and dynamic (velocity and acceleration) constraints. These planning techniques enable near optimal solution for practical problems to be found [SD88, SH85, JC89]. Using this

approach, the work of [Fra99] presents a near time optimal approach that searches the solution trajectory over a restricted set of "canonical trajectories" accounting for both dynamic of the system and the environment.

Probabilistically Complete Approaches The first randomized approaches appeared in the mid of 90's, in order to improve existing methods [BLL92]. In the randomized potential field approach, a heuristic function is defined on $\mathcal{CS}$ that attempts to steer the robot toward the goal through gradient search. If the search becomes trapped in local minimum, random walks are used to help escape. The first planning technique completely based on random sampling is the Probabilistic Roadmap Planner (PRM) [KSLO96, OS96], now recognized as a major popular and efficient technique.

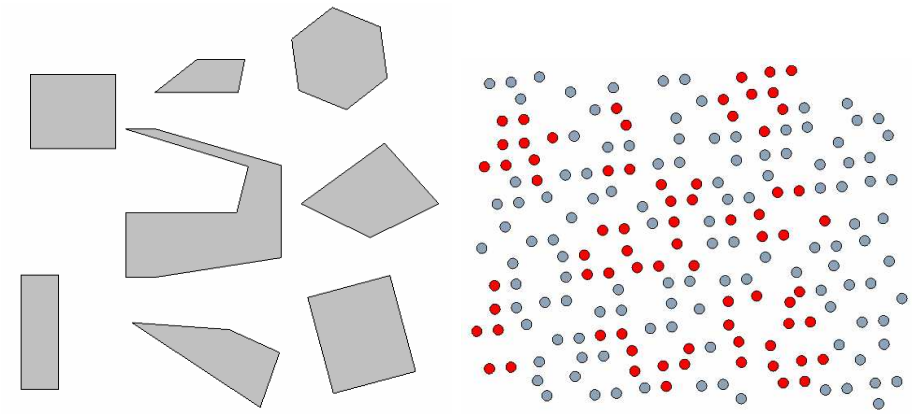


Figure II.8: Exact vs. approximate representation.

The main idea for PRMs is to avoid the explicit construction of the free space. It is approximated by random sampling instead. This approach consists in probing the space with a random sampling scheme and a detection collision module. Fig. II.8 illustrates an explicit representation of the free space on the left, where each obstacle is completely defined as opposed to an approximated one on the right where states are individually tested and considered as part of an obstacle, or part of the free space. Nearby configurations are connected by computing a path using a local planner so as to construct a roadmap *ie* a graph within $\mathcal{CS}$ (see fig. II.9). The roadmap is later used to solve path planning problems. Probabilistic planners have been proved to be *complete* in a *probabilistic* sense, *ie* the probability of correct termination approaches unity as the number of milestones increases.

Despite the differences between the different probabilistic approaches, the key element is the sampling distribution. It is expressed in terms of two criteria called the *dispersion*, and *denseness*. The dispersion reflects the size of the largest uncovered area. This

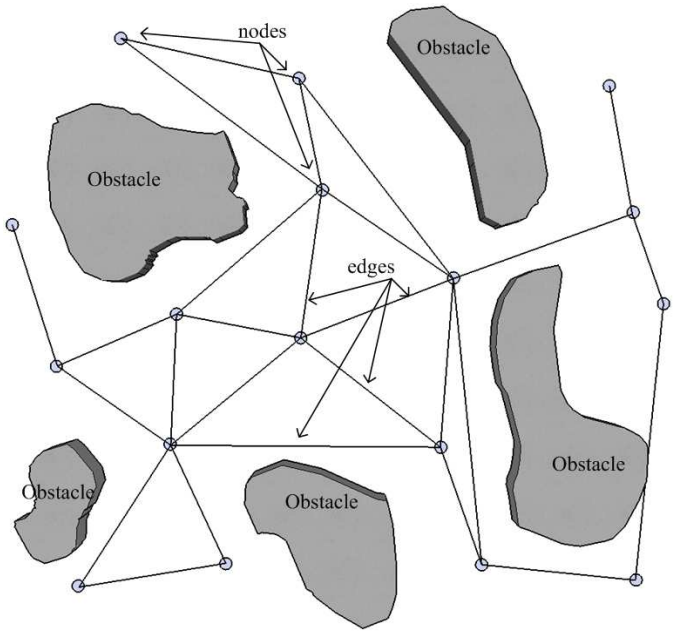
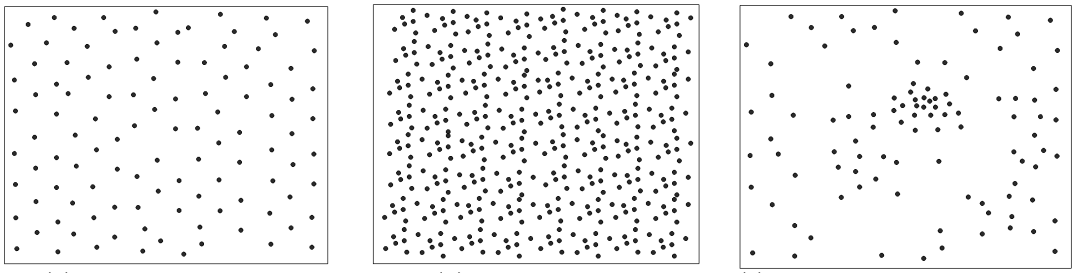


Figure II.9: Illustration of a random sampling based roadmap.



(a) not dense sampling (b) dense sampling (c) sampling with poor dispersion

Figure II.10: Denseness & Dispersion of a sampling.

generalizes the idea of grid resolution. The denseness relates to the techniques where samples come arbitrarily close to any state, as the number of iterations tends to infinity (see fig. II.10). One can choose uniform sampling, but this most likely will fail within an environment where obstacles do not lie uniformly over the scene, or Gaussian sampling technique [BOvdS99]. [KSLO96] suggested to memorize the failures of the local planner in order to increase the area where this would have occurred. A few approaches require more geometrical computation, and place addition samples near to edges and vertices of obstacles [ABD+98], [SO97] or allow for samples inside obstacles and push them to the outside [WAS99], [HKL+98].

The roadmap can then be used for *multiple query* problems, *ie* multiple path planning

problems in the same environment. Once it has been constructed, the planning problem becomes one of searching a graph for a path between two nodes. The roadmap construction step can be very expensive in the PRM algorithm, specially in complex environment. Several approaches have been developed, aimed at minimizing the number of collision checks and hence minimize the running time of the planner. In [SLN00] only nodes that can be connected to two components or to no components are added. The reason is that a node that can be connected to just one component represents an area that can already be "seen" by the roadmap. The concept of lazy approaches presented in [BK00a] assumes that all nodes and edges in the roadmap are initially collision-free, and searches the roadmap at hand for a shortest path between the initial and the goal node. The nodes and edges along the path are then checked for collision. If a collision with the obstacles occurs, the corresponding nodes and edges are removed from the roadmap. The major difficulty with these techniques is that although powerful for standard path planning their ability to extend to general problems that involve differential constraints depends upon the existence of a local planner. This method was successfully applied to a nonholonomic planning problem using Reeds-Shepp curves for car like robots. This result directly enables the connection of two configurations with the optimal length path. For more complicated systems, a steering method can be used to built the local planner. However, the connection problem can be as hard as designing a non linear controller. The PRM technique might require the connections of thousand of states to find a solution and if each connection is akin to a nonlinear controller, the problem seems impractical.

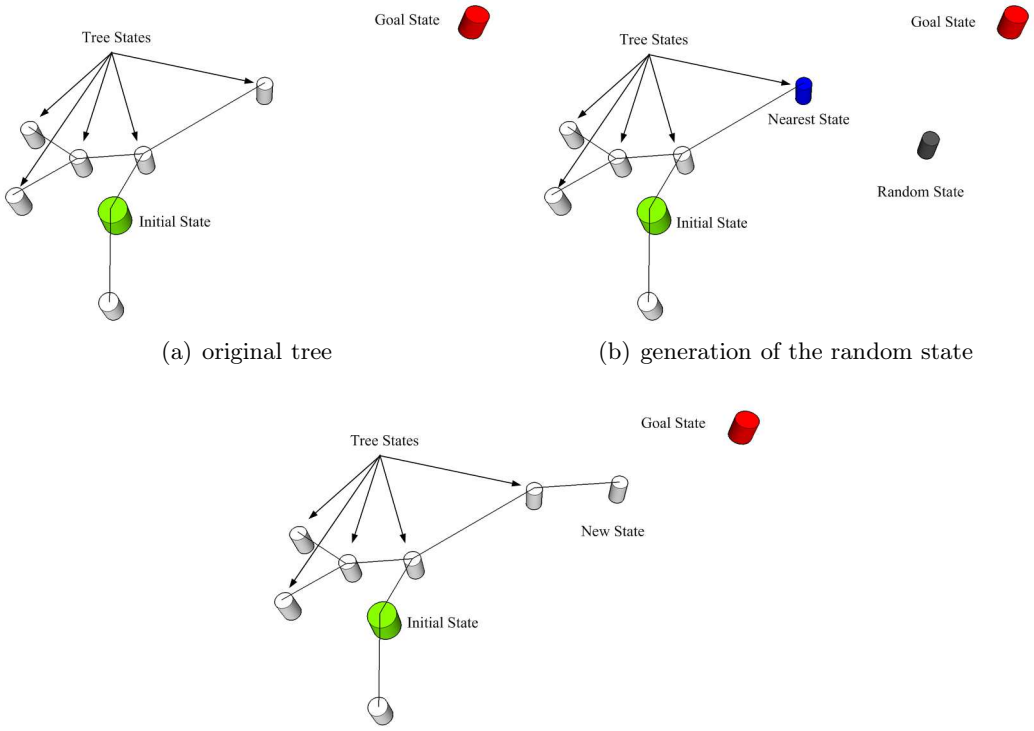
As for *single query* problems, they use either fast roadmap construction techniques or a diffusion technique. These diffusion techniques are based on the incremental construction of a tree. The Rapidly-exploring Random Tree (RRT) technique introduced by [LK01b] is a recent and popular diffusion technique. In essence, this method is based on the incremental construction of random trees (RRTs) in a way that quickly reduces the expected distance of a randomly-chosen point to the tree. The construction operates as follows over a given tree (see fig. II.11(a)) :

1. A random state is chosen. The nearest state from this state and on the tree is selected (II.11(b)).

2. A new state is calculated along the direction connecting the random node to the nearest neighboor in the tree, untill a collision possibly occur and inserted in the tree. (II.11(c)).

This approach is very efficient. Besides, it has the ability to handle explicitly the system's differential constraints. Indeed, as the state calculation can be performed by mean of an

incremental simulator. This generally reduces to the integration of a given differential model of the system considered over a predefined time interval. Depending upon the model of the system which is used, the kinematic and dynamic constraints of the system are explicitly taken into consideration. However, the design of the metric to efficiently select the best state of the tree is usually involved, specially for nonholonomic systems. Nevertheless, several recent approaches [HKLR00, BV02] based on this technique have shown impressive results.



(a) original tree

(b) generation of the random state

(c) insertion of a new state

Figure II.11: Rapidly-exploring Random Tree principle.

We now have a sufficient background to state the main problem addressed in this work.

3.3 Extension of the Basic Problem

3.3.1 Kinematic Constraints

Holonomic Constraints A manipulator arm moving a glass full of water is not free to move its hand in any direction if it has to keep the water inside the glass. This kind

of constraint is called *holonomic.* A holonomic constraints is of the form $F(q) = 0$ with q being the configuration of the system. Such constraints reduce the configuration space of the system and define a subset of $\mathcal{CS}$ that can be reached, of same dimension as $\mathcal{CS}$. In the case of the robot carrying a glass of water, the constraint consists in maintaining the orientation of the arm upright. It is important to not that such constraints do not affect the shape of the path of the system. In fact, holonomic constraints can be compared to static obstacles. Therefore the standard path planning algorithms apply to this class of systems.

Nonholonomic Constraints Wheeled robots are another type of very popular robotic system. In 1966, the first wheeled mobile robot, "Shakey" [Nil84], was presented. Easy to integrate on a robot and to control, the use of wheels, of different type, number or pattern [CBDN96], have become widespread. The presence of a wheel in a robotic systems raises however a new problem. *This problem is related to the mechanical concept of nonholonomy. Nonholonomic constraints* [Lau86], are differential constraints, that reduce the instantaneous motions that the robot can perform. Many nonholonomic systems still have sufficient control inputs to allow the system to move anywhere in $\mathcal{CS}$, in such case they are said to be *controllable* [BL93]. Fig. II.12 depicts a wheel whose contact point with

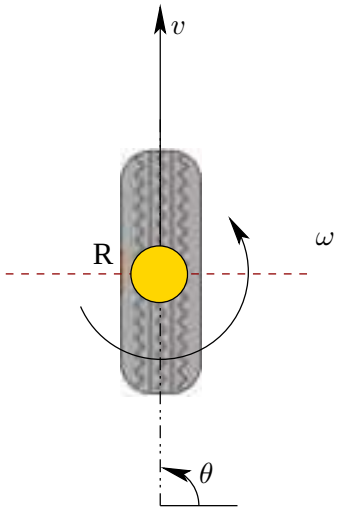


Figure II.12: Unicycle system.

the ground is R of coordinates (x, y) and whose orientation is θ. Assuming pure rolling and no slipping, the wheel can only move in a direction perpendicular to its rotation axis. This constraint can be written as follows :

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0 \tag{II.1}$$

II.1 is a nonholonomic constraint. It features first order derivative of the configuration variables that cannot be integrated. The general form of a nonholonomic constraint is :

$$G(q, \dot{q}) = 0 \quad \dot{q} \in \mathcal{T}_q(\mathcal{CS}) \tag{II.2}$$

where $\mathcal{T}_q(\mathcal{CS})$ is the *tangent space* of $\mathcal{CS}$ at q. The tangent space $\mathcal{T}_q(\mathcal{CS})$ has the same dimension as $\mathcal{CS}$ and represents the space of the possible velocities. A set of m linearly independent nonholonomic constraints reduce the space of the reachable velocities for $\mathcal{A}$ at all q of $\mathcal{CS}$ to a subspace of $\mathcal{T}_q(\mathcal{CS})$ of dimension $(m-n)$ without affecting the dimension of $\mathcal{CS}$. Thus, at every point q in $\mathcal{CS}$ is defined a tangent space of dimension $(m-n)$ defining the space of allowed velocities for the wheeled mobile robot system.

The path has to fulfill these differential constraints, in order for the system to properly follow it at execution. In this case, the path is said to be *feasible*. The problem of finding a feasible path for a nonholonomic systems has solution for specific systems (*eg* kinematic car model [Dub57, RS90], flat, chained, nilpotent[Lau86]) and usually rely on complex maneuvers. For general systems however like non small time controllable systems, the general problem remains largely open.

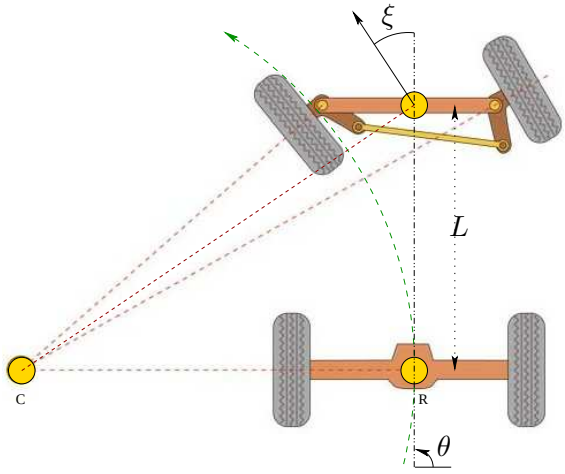


Figure II.13: Car-like system.

Car-like robots (fig. II.13) are an interesting class of wheel mobile robots, due to the tremendous potential of robotic applications on real vehicles. A car like robot is equipped with four wheels, two fixed rear wheels and two front steering wheels. It has one center of gyration C by mean of the Ackerman's steering system, which turns the wheels adequately such as to intersect all rotation axis. In terms of kinematics, the model of a car-like robot is therefore equivalent to the bicycle model. Let R be the center point of the rear axis, of coordinates (x, y), θ the orientation of the car and ξ the front wheel angle with respect to

its orientation. A configuration of the system is given by the 4-tuple $q = (x, y, \theta, \xi)$. This type of system presents another nonholonomic constraint, in addition to the one exhibited by the contact wheel ground, which is due to the mechanical limitation in the steering wheel angle. Indeed, on a real system, the steering angle has a physical stop expressed as follows:

$$\dot{x}^2 + \dot{y}^2 - (\frac{b}{\tan \xi_{max}})^2 \dot{\xi}^2 \geq 0 \tag{II.3}$$

This constraint upper bounds the maximum curvature of the path.

3.3.2 Dynamic Constraints

There are two different types of dynamic constraints. The first one concerns the environment, where surrounding objects are moving. The second one concerns the system itself and its physical capabilities (acceleration, velocity, etc.). The physical constraints are differential constraints for which higher order differential equations are needed. Indeed, the models for dynamics involve acceleration, in addition to the velocity and configuration. These constraints can be written as follows :

$$G(q, \dot{q}, \ddot{q}) = 0 \tag{II.4}$$

A technique to handle these constraints consists in converting these high order derivatives into a new set of constraints that are first order only, but in a enlarged space, the **state space,** we note $\mathcal{S}$. In such a space, the explicit representation of these constraints is of the form :

$$H(s, \dot{s}) = 0 \tag{II.5}$$

For trajectory planning therefore, the configuration space $\mathcal{CS}$ framework is replaced by the state-space $\mathcal{S}$ counterpart. A point in $\mathcal{S}$ may include both configuration parameters and velocity parameters. If we look at the problem from a control perspective and express the dynamic constraint through their parametric form :

$$\dot{s} = f(s, u) \tag{II.6}$$

where $u \in \mathcal{U}$ is a control *ie* a command sent to the system's actuator to execute the motion. Such a function is also called a transition function as it represents how a control applied to a state over time modifies this state.

Both dynamic constraints involve the time dimension. The time dimension brings the notion of *trajectory*. Whereas path planning is characterized by the search of a continuous sequence of configurations, *trajectory planning* is concerned with *the time history* of such

a sequence. A path informs about geometry and does not have any information about the way the robot actually moves on a path. This is the reason why path planning applies only in static environment. Thus, the problem of trajectory planning consists in finding the sequence of admissible controls among $\mathcal{U}$, the set of all possible controls, that drives the robot toward its goal. This sequence is *an open-loop trajectory*, *ie* a sequence of action calculated a priori, without real time observation feedback.

Interestingly, the two dynamic constraints are in fact not completely independent. Indeed, when a trajectory is calculated within a dynamic environment, the real physical limitations of the system in terms of maximum speed (and acceleration) must be also taken into account in order for the system to actually being able to execute it. Most of related works treat however these constraints separately. In our work, we address both constraints simultaneously and tackle the problem of motion planning within a dynamic environment, while obeying differential constraints of the system.

3.4 Motion Planning in Dynamic Environments (MPDE)

3.4.1 Problem Formulation

Let first formally pose a few useful notations and a definition before to state the problem. Let $\mathcal{A}$ be the robot evolving in $\mathcal{W}$. $\mathcal{A}(s)$ is the subset of $\mathcal{W}$ occupied by $\mathcal{A}$ at a state s. Furthermore, $\mathcal{A}(s,t)$ is the subset of $\mathcal{W}$ occupied by $\mathcal{A}$ at a state s and time t. Let the model of $\mathcal{A}$ be described by a differential equation of the form $\dot{s} = f(s,u)$ where $s \in \mathcal{S}$ be the state of the system, $\dot{s}$ its time derivative and $u \in \mathcal{U}$ a control. We formally define a trajectory as follows:

Definition 3
Let $\phi \in \Phi$: $[t_0, t_f] \longmapsto \mathcal{U}$ denote a control input, ie *a time-sequence of controls. An initial state and a control input define a trajectory for $\mathcal{A}$,* ie *a time sequence of states. For sake of clarity, we denote by $s = \phi(s_0, t)$, the state of the system $\mathcal{A}$ at time t, starting from an initial state s_0 at time $t_o = 0$ and under the action of a control input ϕ.*

.

Let a static obstacle $\mathcal{B}$ be a closed subset of the workspace $\mathcal{W}$ we note the (time-dependent) moving obstacle $\mathcal{B}(t)$. Then, $\mathcal{B}(t_o, \infty)$ denotes the obstacle $\mathcal{B}$ from time t_0 to ∞. It models the future behavior of $\mathcal{B}$.

We further introduce the notion of collision state as follows :

Definition 4
A state s is a collision state at time t if and only if $\exists \mathcal{B}$ such that $\mathcal{A}(s) \cap \mathcal{B}(t) \neq \emptyset$.

We can now introduce the notion of collision-free trajectory.

DEFINITION 5
A trajectory is collision free if $\forall t$, $\mathcal{A}(\phi(s_0, t)) \cap \mathcal{B}(t) = \emptyset$

For a complex system, *ie* constrained by its kinematics and dynamics, moving within a dynamic environment, the general problem addressed from the motion planning perspective is formulated as follows :

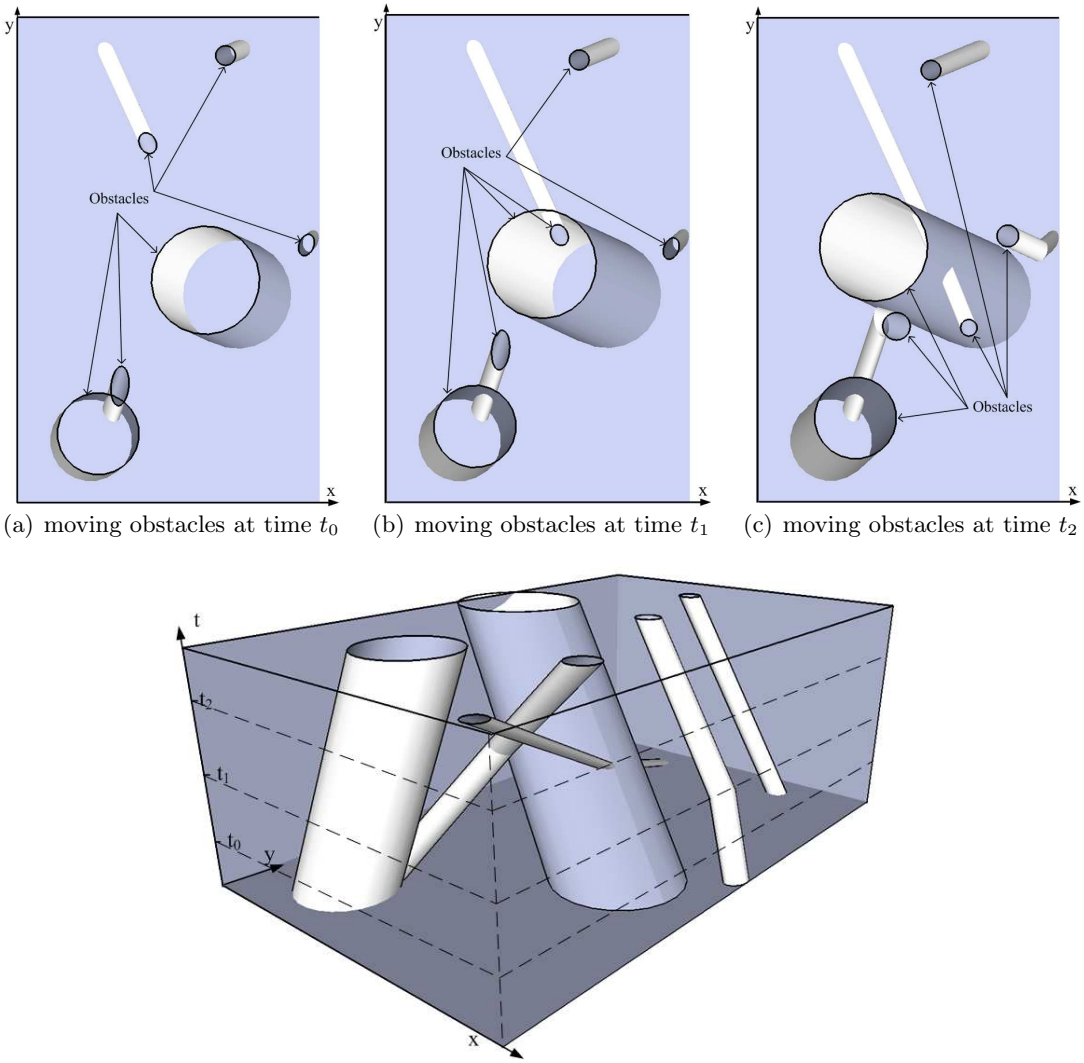
STATEMENT 1 (MOTION PLANNING IN DYNAMIC ENVIRONMENTS (MPDE))

1. *Let a workspace* $\mathcal{W} \in \mathbb{R}^2$ *(or* $\mathbb{R}^3$*)*
2. *Let a state* s_0 *designates the initial state at time* t_0
3. *Let* S_f *designates the set of goal states*
4. *An algorithm must compute a collision-free trajectory* $\phi \in \Phi : T \longmapsto \mathcal{U}$*, from* $s_0 = s(t_o) = \phi(s_0, t_0)$ *to* $\phi(s_0, t_f) \in S_f$

3.4.2 Works Related to MPDE

Dynamic Environment The main idea in the work that addressed moving obstacles, lies in adding the time dimension and construct an extended space [ET87, FL92]. Fig. II.14 illustrates such a representation in a simple space. The moving obstacles depicted at different time slices II.14(a-c) are represented in fact in the time-extended space as static obstacles (cylinder in fig. II.14(d)). In a time extended space, moving obstacles are considered explicitly. This means that dynamic forbidden areas turns into static ones in the enlarged space. Given the particularities of the time dimension (impossible to go backward and speed is not infinite), usual planning schemes might be still applicable with little adaptation. Some authors have proposed an adaptation of the visibility graph [RS85, KZ86, ET87], cell decomposition method [FS89] or similar concepts, using a network of lanes and crossing [FL91], or the traversability vector concept [LP91]. Other methods preferred to resort to decoupled techniques. In a dynamic environment, the path velocity decomposition technique was proposed by [KZ86]. It first reduces the problem to a static traditional path problem, then a temporal velocity-planning problem is solved. The second part consists in defining a proper velocity profile incorporating the moving obstacles which becomes a much easier task to handle since this velocity planning is performed on the path-time space, a 2D space.

System's Dynamics The major work dealing with dynamic constraints has involved $\mathcal{S}$ [BDG85a, Can88], even though some authors have proposed trajectory planning meth-



(a) moving obstacles at time t_0 (b) moving obstacles at time t_1 (c) moving obstacles at time t_2

(d) complete model

Figure II.14: Representation of moving obstacles within $\mathcal{CS} \times T$.

ods operating within $\mathcal{CS}$ [SH85] or $\mathcal{W}$ [SD88]. A common approach is to decouple the problem in a two-stage optimization process formulation. Given a particular trajectory, its velocity profile is rescaled so as to respect dynamic constraints and to be time-optimal [Hol83, BDG85b, SD85, SM85, O'D87, SD89]. [SD85, SM85] provide exact time optimal algorithms in the case of robots with full dynamics (bounds on the velocity and acceleration) and moving along a given path. This path is deformed by means of variational technique. [DSY87] establishes an exact algorithm for 1D kinodynamic planning, method that can be extended to 2D and 3D cases. From these results new methods have been presented introducing local time optimal methods by [SD89].

In fact, there is very little work that addresses both types of dynamic constraints simultaneously. Beside early work [O'D87, FS89, FL92] dealing with low dimensional problem often accounting for dynamic constraints given a collision free path, there is really to the authors knowledge only the recent work [HKLR02, Jai05, vdB07] that addresses simultaneously both constraints using a method operating within the state space $\mathcal{S}$ augmented with the time dimension. We refer to this space as the state-time space $\mathcal{ST}$.

3.5 Real-Time Constraint

3.5.1 Introduction

There is an important consequence stemming from dynamic environments. In such environment the calculated trajectory accounts for the time dimension. But more precisely, it has the constraint to start at a specific time and last during a precise period. We can say that this plan or trajectory has the characteristic to be *anchored in time*. In a real environment, these time boundary condition have a direct impact on the computation time of the trajectory. Thus, a motion has to be calculated within a limited period of time, lying from the current time to the time the calculated motion is planned to start. If it were possible to start the plan at an arbitrary time, this constraint would not affect the motion planning problem. However, this is not the case. Indeed, this time constraint is related to the constraint of collision avoidance with the moving obstacles. We define this time constraint, the *decision time constraint*. This constraint is a **hard real-time constraint**. The system must return a plan of its future motion within this limited amount of time, and move from its current place before it gets hit. This time constraint depends upon a factor we call the *dynamicity* of the environment, which is defined by the environment itself and is function of the system and the moving obstacle's dynamics. Even though this constraint is of utmost importance for real applications, surprisingly is has attracted very little attention in past research. As a practical aspect, this certainly relates to the complexity of deliberative schemes that do not give much hope on fulfilling such a real-time constraint. Nevertheless, for real applications, the real-time constraint must be fulfilled and the problem we address in this work turns into a motion planning problem under real time constraints.

3.5.2 MPDE and Real-Time Constraint

We can now reformulate the MPDE problem so as to account for the Real-Time constraint.

Statement 2 (MPDE+RT)

1. *Let a workspace* $\mathcal{W} \in \mathbb{R}^2$ *(or* $\mathbb{R}^3$*)*

2. *Let δ_d be the decision time constraint*

3. *Let a state s_0 designates the initial state at time t_0*

4. *Let S_f designates the set of goal states*

5. *An algorithm must compute a collision-free trajectory $\phi \in \Phi : T \longmapsto \mathcal{U}$, from $s_0 = s(t_o) = \phi(s_0, t_0)$ to $\phi(s_0, t_f) \in S_f$, within a computation time δ_c such that $\delta_c < \delta_d$*

3.5.3 Works Related to MPDE + RT

The way this real-time constraint has been taken into account in past research varies depending upon the approaches used. Most of the work in the literature abusively refer to as real time, a motion strategy that provides a continuous, or near-continuous robot motion in the real world. In many situations indeed, a robot that stops frequently to plan is not useful; in military applications it may be fatal. The main goal of these techniques in fact, reduces to develop algorithms that can make fast enough decisions. This proposed notion of real time however, heavily depends upon the environment or the system and appears in reality extremely vague. Originally, the different work to address the time issue from a planning perspective was motivated by the problem of adapting a plan to an unpredictable change in the model of the environment [QK92, Ste94, TKA95]. The question of the necessary time computation required to calculate a new plan came later and generated a lot of various work aimed at reaching very fast planning capability, sometimes abusively called real time planning. We describe here the significant related approaches.

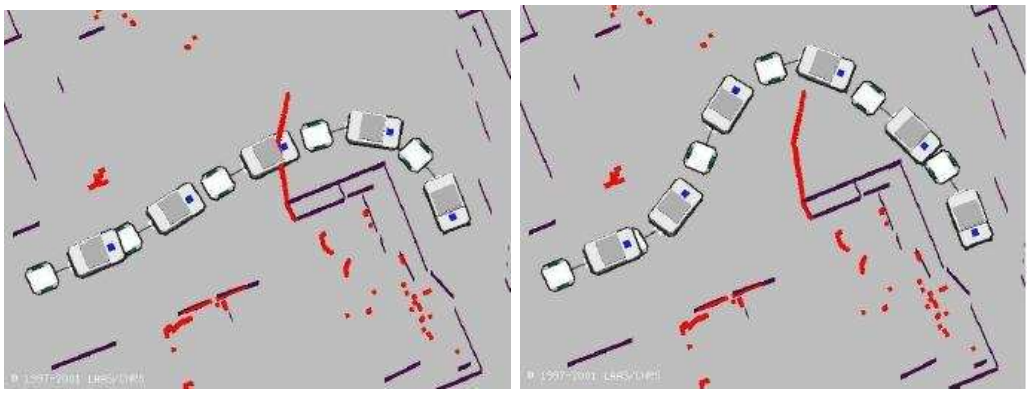
- One of the early approaches was to adapt the popular **dynamic programming** methods. These efficient graph search techniques, are particularly well adapted to the approximated planning techniques. Indeed, the discretization of the explored space result in a grid which is used by these techniques. From early methods based on A* algorithm, several adaptation have appeared in order to rapidly find the optimal path in complex and changing environment. As a result, very fast techniques have been developed like the D* algorithms (which stands for Dynamic A*) [Ste94, Ste95] and more recent adaptations [KL02]. These techniques achieve fast planning and allow an adaptation to unknown environments. Furthermore, even though all these techniques claim to be fast, they do not provide any indication on the actual computation time nor any guarantee on its upper bound. In very recent work only, these issues are discussed. In [Ste02] a variant of the D* algorithm is presented. The experiments indicate a replan calculation time to be less than a second. From this observation however, no formal guarantee can be given. At such a rate, this

algorithm is argued to be real time, as it is usable online by a real robot, however no evidence is given on the fact that this bound will always be respected nor that this time will always be sufficient for any real experiment.

- The recent work [LFG+05] derived from the dynamic programming paradigm combined with incremental optimization algorithms, the anytime algorithm, is to the authors knowledge the only complete approach to address explicitly the real time constraint within the motion planning scheme. However, the completeness is achieved at the expense of the plan optimality. Besides, this work limited to low dimension problems, does not explicitly account for the dynamics of both the system and the environment. In the more recent work of [vdBFK06], this technique is used to repair broken probabilistic roadmap due to dynamic changes.

- **Deformation techniques** are interesting approaches and started with the Elastic Band concept, introduced by [QK93], developed for holonomic systems. Given a collision-free path within the configuration space, provided by a planner, virtual forces created by newly detected obstacles are calculated and applied on the path and deform it as it was an elastic band, in such a way that the system is pushed away from obstacles. The concept of balls covering the path is used in [Qui94, Kha96] for holonomic systems and in [KJCL97] for nonholonomic ones. A much more efficient and fast algorithm based on this technique is presented in [BK00b]. The claim of this latter method to be real time stands from the potential capability it has to handle real-world application. However, there is no guarantee on planning within an arbitrary small time computation. Another method, more involved and designed mainly for nonholonomic systems, uses the principle of virtual force [LB02]. A virtual force is applied to a path in order to deform it, while keeping nonholonomic constraints (see fig. II.15). Even though this approach is very elegant and gives good quality path results, it remains computationally involved and suited for unexpected stationary obstacles but not dynamic environment as future motion of the surrounding obstacles is not explicitly considered.

- A tremendous effort has been put recently into developing **fast random algorithms**. The probabilistic random planners achieve fast roadmap construction and update and therefore allowing fast random path planning. One significant recent work is presented in [HKLR00] where fast replanning is reconsidered using the new tools of random based planners. The roadmap is based on the random choice within the control space, and the related state calculation by model integration. This single query probabilistic roadmap planner achieve a specified goal under kinematic and/or dynamic constraints while avoiding collision with moving obstacles with known trajectories. The principle of this algorithm consists in constructing a roadmap for a



(a) path followed without deformation (b) path after deformation

Figure II.15: Nonholonomic trajectory deformation (source [LB02])

single query, built with milestones sampled in the state time space $\mathcal{ST}$ yielding to a tree shaped based roadmap T. At each iterations, it first picks at random a milestone from T and also picks at random a point in the space of admissible control functions. It then computes the trajectory by integrating the equations of motion. The selection of a milestone m of the tree T to be expanded is done at random with a probability inversely proportional of the current density of milestones around m. The bounds of convergence for this algorithm reside in the assumption of uniform sampling, which cannot be satisfied for kinodynamic constraints. In the more recent work of [JS04] the technique of fast roadmap construction is further explored using lazy-evaluation mechanisms in order to rapidly update the roadmap according to the dynamic changes. In [BV02] the single query RRT technique is used and improved by mean of a modification of the sampling technique and achieves very impressive results. Even though both later algorithms successfully replans at a high rate the path of a point mass robot while evolving among a highly dynamic environment there is no guarantee of an upper bound of the complete planning computation time. [Fra01] introduces an algorithm of kinodynamic motion planning within a dynamic environment integrated within robust hybrid control architecture. This algorithm presupposes the existence of a closed loop architecture that enables the guidance of the vehicle from any initial conditions to any target location at equilibrium. Thus rather than working with an "open-loop" system as presented in earlier publications, his basic dynamical system is a closed-loop one, thanks to this guidance law that can synthesize control inputs function of current state. In many cases, efficient, obstacle free guidance laws may be computed analytically. In addition, many of these problems, although they may not admit closed-form solutions, may be solved numerically via the approximate or exact solution to an appropriate optimal control

problem, computed and stored using iterative methods for instance.

3.6 Safety Concerns

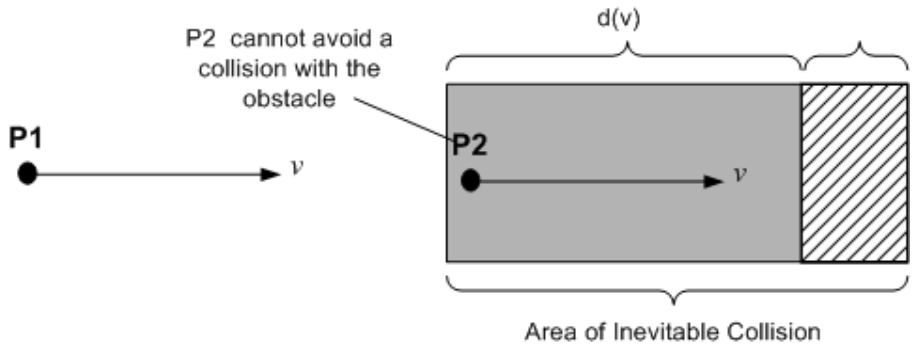
As a corollary of the time constraint, we already mentioned the safety issues that face most of the planning schemes under time constraints. Indeed, a computed trajectory is a sequence of decisions or actions aimed at moving the system safely towards the goal. In fact all reactive or deliberative planning schemes are affected with the problem of safety as soon as the environment is dynamic. Indeed, a planning scheme is aimed at calculating states or commands for which a motion with no-collision must be guaranteed.

3.6.1 General Concept

Safety issue for systems with dynamics is one of the most challenging aspect of kinodynamic planning. The notion of configuration space is appropriate for problems focusing on geometric aspects of motion planning. The safety issue addressed by these techniques lies on the notion of forbidden (or collision) configuration *ie* a configuration for which a system is in collision with its environment (or itself). A configuration obstacle (C-obstacle) is the representation in $\mathcal{CS}$ of an obstacle $\mathcal{B}$ present in the workspace $\mathcal{W}$. It represents the set of all configurations yielding to a collision with a particular obstacle $\mathcal{B}$ of the environment $\mathcal{W}$. The calculated plans are **safe** when they are **collision-free** with the C-obstacles of the configuration space $\mathcal{CS}$. The notion of forbidden configuration is well described in the literature and extensively used for path planning problem.

The trajectory planning problem however, involves a description of the obstacles within the state space $\mathcal{S}$ of the system, which is the space of all configurations augmented by their derivatives. Of course a straightforward description of the obstacles within $\mathcal{S}$ brings the notion of **collision state**. In terms of safety, even though the transposition of collision configuration to the state space or time-state space is straightforward, it takes simple examples to illustrate the limits of this simple concept.

Example 1
In II.16 we consider a point mass robot with non zero velocity moving to the right (a state of **P** is therefore characterized by its position (x, y) and its speed (v). Depending upon its speed there is a region of states, for which collision will not be avoided. This area represents the set of states for which the system **P** (as **P2** in the figure), even though it does not collide, will not have the time to brake and avoid collision, and thus for which no matter what the future trajectory followed by the system is, a collision with the obstacle eventually occurs.

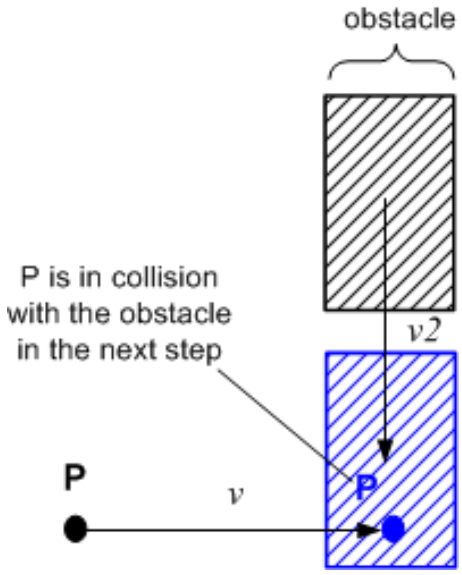


(a) collision state vs. inevitable collision state

Figure II.16: Influence of the dynamics of a point mass robot on its safety.

Example 2

In our second example, a point mass robot **P** moves to the right. An obstacle upfront is moving downward. In case the system does not account for the obstacle future motion, it will proceed to move to the right during the next step, of arbitrary duration. However due to the obstacle motion downward, our system **P** will collide the obstacle as it did not consider its motion (fig. II.17). This very simple example is however sufficient to illustrate that the for the system's safety, it is crucial to explicitly account for its surrounding obstacles' motion.



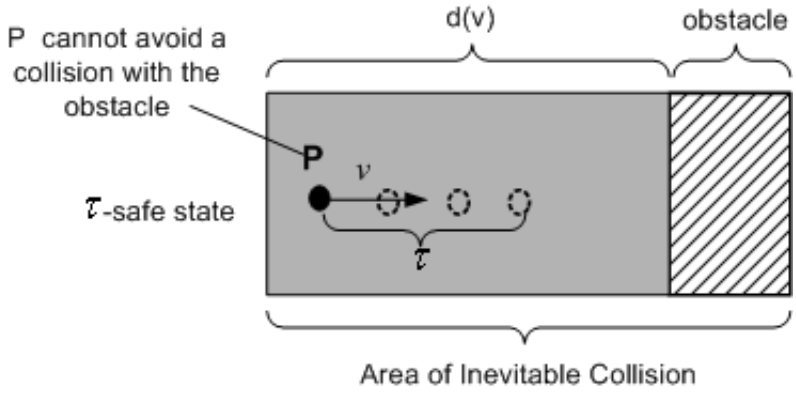
(a) collision state vs. inevitable collision state

Figure II.17: Influence of the world's dynamics on a system's safety.

Example 3

This last example describes the case of a similar system **P** as before, moving to the right for which is guaranteed to be collision free for an arbitrary duration. Our observation is that in

case this time period has no obvious physical relation with the dynamics of the system, this guarantee appears clearly unsatisfying. Indeed, a decision process reasoning on a limited time duration might bring the controlled system to a state where a collision might occur afterwards. In figure II.18, the state P which is guaranteed to be collision free over τ seconds will however collide the obstacle as it will not have time to brake.



(a) collision state vs. inevitable collision state

Figure II.18: Influence of the time horizon on a system's safety (τ-safety concept from [Fra01]).

Therefore for a system evolving within a dynamic environment, safety is to be considered with respect to several criteria that bring us far beyond the traditional collision free paradigm.

1. At first, it is important that the notion of system's dynamics is taken into account, *eg* the inertia of a robot.
2. Besides, the future motion of the surrounding obstacles must be considered.
3. Finally, the safety guarantee should not depend upon an arbitrary temporal horizon.

3.6.2 Related Works

There is little work on safety issue itself and most of the work addressing this issue do not provide in fact any guarantee in terms of safety. The methods rely to heuristics or their intrinsic reactive nature. There is very little work that formally describes and explicitly consider the problem of safety, that can be separated into three main categories:

- There are approaches that calculate **braking trajectories** and therefore guarantee that the system will always be able to stop before collision. The work presented

in [BK99][FBT96] does not however consider explicitly the future motion of the moving obstacles, which reduces significantly the pertinence of the safety guarantee unlike the work of [ASMK02] where a conservative worst case scenario on the moving obstacles motion is taking into account.

- A more common approach that have been investigated is to provide a guarantee of no collision over a specific, empirical time period sometimes referred to as **time horizon** [Sim96][FS98][LSSL02] or τ**-safety** [Fra01].

- Some other work discussed the notion of **evasive trajectory**. Mainly coming from the avionics the main idea lies in the verification at each time that a specific control sequence can be applied to escape a situation of danger. The evasive plan can be explicitly calculated [Pri99] or a danger zone [WBN93, TT03], can be defined such as to guarantee these maneuvers to be applicable. This notion is used in the work of [HKLR02] where they define safe planning as the capacity of the planner to calculate an escape plan in case it has failed finding a complete trajectory to the goal during the allotted time. In order to avoid an undeterministic calculation overhead while moving, the escape plan is systematically calculated with the trajectory. But no details with respect to the valid time length of the escape plan is given.

4 Conclusion

In our work, the problem of navigating a system while considering both dynamics of the environment and the system is addressed from the motion planning perspective. Motion planning has been well studied and strong mathematical framework has been brought to account for the various constraints of a complex system and the dynamic nature of the environment. Even though early complete techniques did not give much hope, latest probabilistic techniques have certainly opened a new way of tackling the problem as demonstrated in recent work. Furthermore, when the robot is placed in a real dynamic environment, the related real time constraint becomes of fundamental importance and raises a new great challenge. This real time constraint is a hard constraint which must be considered explicitly. Surprisingly, even though this aspect is crucial for real applications, it has not attracted a lot of attention in the literature. Finally, the dynamics brings the new safety requirements to account for in order to insure safe motion.

In the next chapter, we will present the concept of the approach we chose to tackle this problem.

(Présentation du probème et état de l'art)

Dans la littérature, le problème général de la navigation est abordé principalement suivant deux perspectives, celle des techniques délibératives et celle des techniques réactives. Les techniques réactives ont été originellement motivées par la nécéssité des robots réels à pouvoir réagir à l'environnement afin d'éviter les obstacles. Ces techniques nécessitent d'être rapides ce qui limite fortement le temps de calcul disponible. Ainsi les méthodes réactives consistent à calculer à chaque itération une seule et unique nouvelle action à exécuter qui approche le robot de son objectif tout en évitant de rentrer en collision avec les obstacles de l'environnement. Ces techniques connues ont une limitation essentielle qui est la limite de l'horizon exploré. Cette vision courte peut amener les robots à être emprisonnés dans des obstacles de formes spéciales concaves. Une autre conséquence est la faible convergence vers le but que peuvent parfois montrer ces méthodes. Les principales méthodes réactives sont revues. Pour la plupart de ces méthodes des évolutions récentes ont été proposées afin d'améliorer l'étendue de l'horizon exploré, la prise en compte d'autres contraintes telles que la cinématique ou dynamique des systèmes et la dynamique de l'environnement s'avère être beaucoup plus complexe. Certaines méthodes ont développé un formalisme qui permet de prendre en compte la dynamique de l'environnement de manière explicite, mais ces techniques restent peu nombreuses. De plus, la plupart des méthodes utilisent l'espace des vitesses comme espace de représentation. Cela rend l'expression des contraintes cinématiques ou dynamiques du système difficiles et oblige les méthodes réactives à avoir recours à des approximations, qui dans certaines conditions peuvent avoir de graves conséquences. Ces observations nous ont motivé à aborder notre problème sous l'angle des méthodes délibératives.

Les approches délibératives, ou de planification de mouvement, consistent à déterminer, à priori, une séquence complète de mouvements, basée sur les modèles du système et du monde, qui amène le robot à son objectif final. Le travail présenté ici se concentre sur les voitures intelligentes évoluant dans un environnement dynamique. Le problème de planification de mouvement de base qui traite d'objets sans contraintes évoluant dans un environnement fixe, devient extrêmement plus complexe quand il s'agit de prendre en compte ces contraintes, ce que nous faisons dans notre travail. Le problème de base est revu puis les extensions de ce problème sont décrites, en particulier l'impact de la cinématique du système, puis de la dynamique, du système et de l'environnement.

En faisant un état de l'art du point de vue de la complétude, nous observons qu'il existe en fait très peu d'algorithmes complets, cad. garantissants de trouver une solution si elle existe, et dans ce cas pour des problèmes très simples seulement. Dès que la dimension du problème augmente soit la complexité augmente et devient exponentielle soit aucun

algorithme n'a encore été trouvé. C'est dans ce contexte que les techniques approchées (en résolution ou probabilistes) ont vu le jour afin d'aborder ces extensions au problème de base et ont montré des résultats très impressionnants pour des problèmes à dimension élevée.

Le problème de base que nous examinons dans ce travail est donc celui de trouver un algorithme qui doit calculer une trajectoire sans collision d'un état initial à un état but.

Il y a cependant une conséquence primordiale au fait de travailler dans un environnement dynamique. En effet, dans un tel environnement, la trajectoire calculée doit commencer et finir à un temps précis, et est en ce sens ancrée dans le temps. Pour un robot placé dans un environnement réel, cela implique que le plan doit être calculé pendant une durée qui s'étale du moment présent au moment où le robot doit commencer à se mouvoir. S'il était possible de commencer le mouvement à un temps arbitraire, cette contrainte n'affecterait pas le problème de planification de mouvements en environnement dynamique. Mais tel n'est pas le cas, car cette contrainte est liée à la nécéssité d'éviter les obstacles mobiles. Nous définissons cette contrainte de temps comme la contrainte de temps de décision. Cette contrainte est une contrainte temps réel dure. Même si cette contrainte est cruciale pour tout robot évoluant dans un environnement réel dynamique, elle n'a pas retenue beaucoup l'attention à en regarder la littérature jusqu'a présent. Cela est certainement du à la complexité intrinsèque du problème de planification de mouvement qui donne peu d'espoir pour satisfaire cette contrainte. Cependant, pour des applications réelles, cette contrainte temps réel doit être prise en compte et le problème abordé dans ce travail devient un problème de planification de mouvement en environnement dynamique sous contrainte temps réel. La problème devient celui de trouver un algorithme qui doit calculer une trajectoire sans collision, d'un état initial à un état final, dans un temps de calcul limité.

La conséquence majeure de cette contrainte de temps est certainement le problème de sûreté qui en découle. Pour un système évoluant dans un environnement dynamique, la sûreté doit être considérée par rapport à différents critères qui nous amènent bien au delà du traditionnel paradigme d'état sans collision.

1. Tout d'abord, il est important que la dynamique du système, cad. son inertie, soit prise en compte.

2. De plus, le mouvement futur des obstacles environnants doit être considèré.

3. Finalement, la garantie de sûreté ne devrait pas dépendre d'un horizon temporel arbitraire.

Dans notre travail, le problème de navigation d'un système, soumis à sa propre dynamique et celle de son environnement est abordée sous l'angle de la planification de trajectoire. La planification de mouvement a été abondamment étudiée et dispose d'un

formalisme mathématique important qui permet de prendre en compte diverses contraintes telles que les contraintes dynamiques qui nous intéressent dans ce travail. Même si les premières méthodes ne donnaient que peu d'espoir, les récentes méthodes probabilistes ont certainement ouvert une voie en vue de considérer des problèmes complexes tel que celui abordé dans ce travail.

De plus, quand un robot est placé dans un environnement dynamique réel, la contrainte de temps qui en découle devient d'une importance cruciale et soulève un nouveau défi. Cette contrainte de temps est une contrainte temps réel dur qui doit être prise en compte de manière explicite. Etonnamment, même si cet aspect est fondamental, il n'a pas beaucoup attiré l'attention. Enfin, la prise en compte de la dynamique apporte de nouvelles exigences sur la sûreté qu'il faut prendre en compte afin de garantir la sûreté de mouvement.

Chapter III

The Approach

1 Introduction

1.1 Real-Time Constraint Problem

When placed in a dynamic environment, an autonomous system has a limited time to make a decision about its future course of action. This real time constraint, imposed by the environment's dynamics must be taken into account explicitly by the planning technique.

In this context, addressing the problem from the motion planning perspective does not give much hope, given the computational complexity of motion planning problem as the dimension of the problem increases, also referred by some authors to as the "curse of dimensionality". This certainly explains why so many reactive methods have been developed in the past. However these methods exhibit severe limitations that we described in the previous chapter. The most important is certainly the lack of lookahead of these schemes. We observe that new deliberative techniques have been introduced recently, dealing with a dynamic environment, and aim at producing efficient and fast planning schemes in terms of computation. Among the recent techniques, the probabilistic approaches have shown very impressive results.

Nevertheless, no proof is given on the algorithm's capability to deterministically fulfill the real time constraint in general conditions. It is most likely that for more complex systems in an arbitrary dynamic environment, these schemes fail. Figure III.1, depicts the evolution of the running time of two probabilist motion planners with respect to the number of nodes expanded. Even though the algorithms show a global trend, it also exhibits a large number of outliers for which the running time is of much greater magnitude. Thus, this

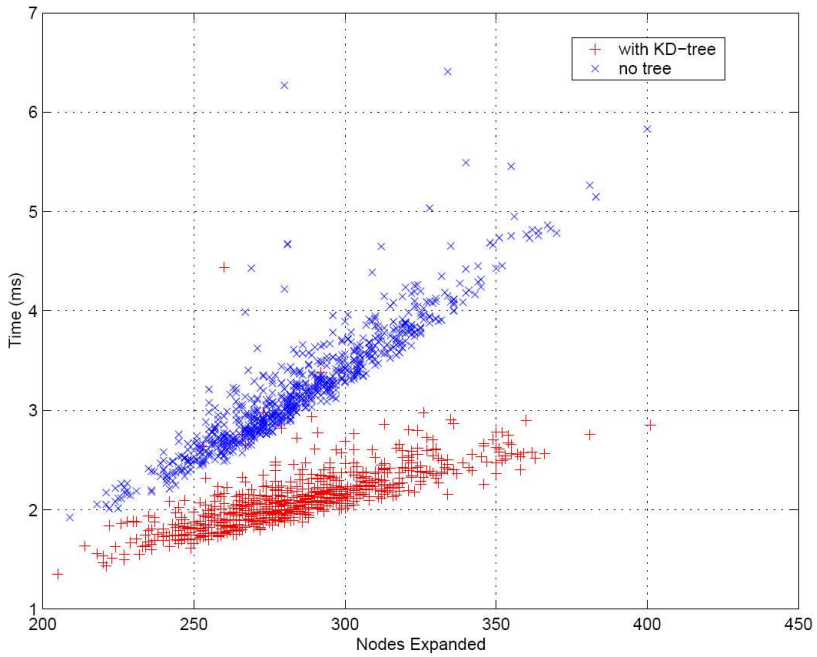


Figure III.1: Running time versus number of nodes developed of two "Rapidly-Exploring Random Tree"-based randomized motion planners. (source: [BV02])

figure clearly illustrates that running time upper bound for probabilistic algorithms can not be guaranteed.

In fact, the real time designation given to a few techniques comes from the observation that on a particular platform and environment, a continuous motion has been executed. To our knowledge, at the exception of [Fra01], dealing with fully known environment, none of these techniques handle explicitly the real time constraint stemming from the dynamic environment.

1.2 The Answer To An Ill-Posed Problem

We have discussed in the former chapter how the inherent complexity of the motion planning problem prevents to provide any arbitrary low computational time upper bound when the problem involves a high dimension space description. Therefore, we claim that the problem 2 of Motion Planning within a Dynamic Environment and Real-Time constraint (MDPE+RT) as formulated in chapter II, section 3.5.2, is impossible to solve in general. There is an intrinsic contradiction and therefore impossibility to fulfill deterministically a hard real-time constraint while tackling an intractable problem.

The Partial Motion Planning (PMP) algorithm is the answer we propose to the problem

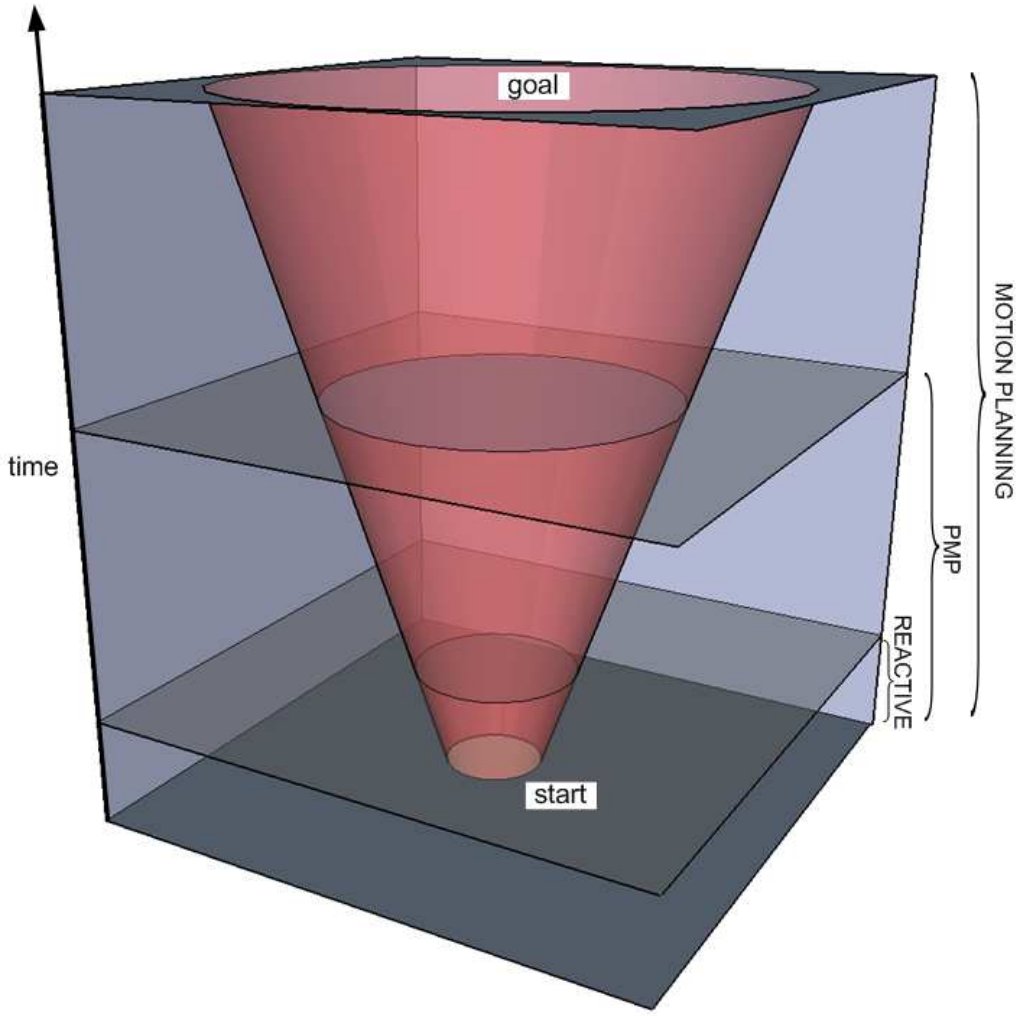


Figure III.2: Partial Motion Planning vs. Reactive and Deliberative methods in $\mathcal{ST}$.

of navigation in dynamic environments. It is especially designed to account explicitly for the real-time constraint stemming from such environments. PMP is a motion planning scheme with an anytime flavor : when the time available to compute a new trajectory is over, PMP returns the best partial motion to the goal computed so far. As opposed to reactive methods that calculate the next time step at a time, and deliberative methods that calculate the complete sequence to the goal, PMP consists in planning as many steps as possible within a fixed available time. Thus, PMP is a paradigm lying in between the reactive and deliberative paradigm (see fig. III.2).

We will now present our theoretical answer to this problem. This chapter focuses on the general principles of our approach. A practical instantiation of these principles will be detailed later in this work, in the next chapter where the case of the car will be studied.

2 Theoretical Approach

The Partial Motion Planning (PMP) is a new approach to the problem of planning in real dynamic environments as it is built around time requirements. We would like to make two remarks at this point. First, the time constraint does not necessarily implies extremely fast planning. The time requirement is defined by the decision time constraint, which is defined in turn by the nature of the real environment. In fact, in most realistic situation, the environment is not very aggressive, and the decision time not small which allows us to introduce the notion of cycles. In section 2.1 we present the principle of PMP and detail this concept of PMP cycles. Second, the necessity to guarantee the computation time is achieved at the expense of a complete calculation of the plan to the goal. The necessity to change the requirements of the original MPDE+RT problem to an incomplete planning scheme in order to meet the real time decision constraint brought new challenges, to begin with, safety issues. We present the concept of Inevitable Collision State (ICS) in the section 2.2 and explain how our planning scheme can benefit from this concept.

2.1 Partial Motion Planning (PMP) Algorithm

In this section we first describe the overall principle of a PMP cycle. As a complete PMP algorithm consists in a sequence of PMP cycles, we then discuss the arrangement of these cycles that define the PMP scheduling. Then, the method to generate a trajectory during a cycle is explained. In Particular, we discuss the diffusion technique on which it relies. This diffusion technique consists in the exploration of $\mathcal{ST}$. We finally explain how the world's obstacles are modeled and incorporated within this space.

2.1.1 General PMP Cycle

We introduced in the previous chapter the notion of decision time constraint. A planning technique aimed at navigating a robot within a dynamic environment necessarily faces this real-time constraint. This real-time constraint impacts the planning algorithm as it has to return a decision within this bounded time, which depends upon the current situation of the environment. To take into account explicitly this constraint, we propose a technique that plans motions according to a cycle of limited duration. The duration of this cycle can not exceed the decision time constraint δ_d. A general cycle of duration δ_d is described as follows (Table III.1) :

1. At instant t_i, get a model of the environment. Like motion planning, partial motion planning requires a model of the environment. The first step is aimed at getting

Table III.1: Partial Motion Planner (PMP) ith Cycle.

1	`GetModelOfTheFuture` $\mathcal{B}(t_{i+\delta_d}, \infty)$,
2	`Launch PMP`
3	`Interrupt PMP at time` δ_d
4	`Return Best Trajectory`

this model. This model contains information on the geometry of the world, *ie* a description of the static forbidden areas together with a description of the future states of the moving obstacles $\mathcal{B}(t)$.

2. Plan a *safe* trajectory from the state $s(t_{i+1})$ of the current nominal trajectory, at time $t_{i+1} = t_i + \delta_d$, toward the goal state s_g. The computation time is defined by the cycle duration, *ie* at most δ_d. At this point of the work and for sake of simplicity, it is sufficient to understand safe trajectory as collision-free trajectory. The notion of safety will be further developed in section 2.2.

3. At time t_{i+1}, the available calculation time is up. ϕ_i the best partial trajectory,*ie* the one that optimizes a given cost function, is returned. This trajectory has a specific duration δ_{h_i}. This trajectory becomes the new plan to be executed in the real world by the robot during the next cycle.

2.1.2 PMP Scheduling

PMP algorithm consists in a sequence of PMP cycles. Depending on the nature of the environment, the sequence is different, resulting in different PMP scheduling. More specifically, we detail in this section two interesting cases where the motion of the surrounding dynamic obstacles is known and the case where it is partially predictable only.

Planning in a Known Environment In some circumstances, the model of the future can be known a priori (*eg* space applications use the physics of Kepler laws). We refer to this type of environment as *known* environment. Such an environment is however not usual, beside rare exceptions (*eg* space application [Fra01]).

1. At the beginning of the first cycle, the model of the environment $\mathcal{B}(t_1, \infty)$ is obtained. During the first cycle of duration δ_d as imposed by the environment, the first plan ϕ_{h_1} is computed, within a completely known environment (cycle 1 in fig. III.3). This trajectory has a duration δ_{h_1}. This trajectory does not necessarily reach the goal and might be a partial plan only. Indeed, the allotted time δ_d to compute the

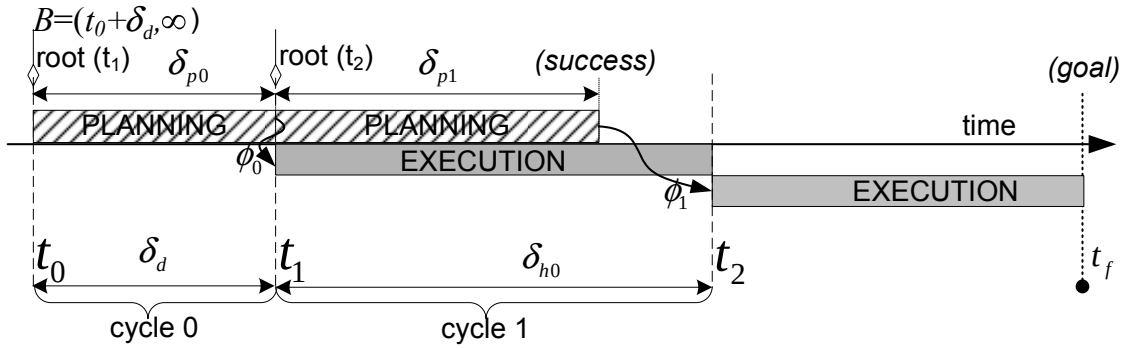


Figure III.3: Partial Motion Planning in a known environment.

trajectory during the first cycle might be too short to compute a complete plan to the goal. Therefore, a partial plan only is computed and returned to the robot.

2. During the second cycle, while the first plan is being executed, a new plan is being computed. Since the environment is known the model of the obstacle future motion remains valid over an arbitrary long period of time. Thus, the second cycle can last as long a it lasts to execute the planned trajectory during the first cycle, namely δ_{h_1}. Such a cycle is repeated until the calculated plan reaches the goal. In fig. III.3, a complete trajectory to the goal is already found during the second cycle. The computation stops when the new planned trajectory reaches the goal. In the case, the computation or planning time, noted δ_p is shorter than the available cycle duration. The complete trajectory is returned to the robot for execution after completion of the first trajectory execution.

3. The third cycle in fig. III.3 consists in the execution to the goal of the complete trajectory planned during the second cycle. Thus, in a known environment, PMP is an aperiodic scheme.

Planning in a Partially-Predictable Environment In most real situations, the environment cannot be known over an unlimited period and the model will have to rely on prediction of limited duration only (*eg* urban environment). Such an environment is said *partially predictable*. We define the period over which the prediction will be considered as valid, the *model prediction validity duration* and note it δ_v. The model of the environment in such a case is noted $\mathcal{B}(t_0, t_{\delta_v})$. This duration imposes a constraint on the planning/execution coupling (Fig. III.4).

1. At the beginning of the first cycle, the model of the environment $\mathcal{B}(t_1, t_{\delta_v})$ is obtained. A trajectory is computed, during the planning cycle of duration δ_d and is

returned to the robot.

2. During the second cycle, this trajectory is executed, within the real world for which a valid predictive model was provided for planning. As the world is partially predictable only, the duration over which the model remains valid is limited to δ_v. This duration must cover the planning time δ_d necessary to produce a plan and the execution duration of this plan δ_h as illustrated in fig. III.4. In other words, the duration of the model prediction δ_v imposes a constraint over the combination of the planning time δ_d and its following execution time δ_h, namely $\delta_d+\delta_h \leq \delta_v$. Indeed, in case $\delta_d+\delta_h$ exceeds δ_v, the plan is being executed in a different world as the one used for planning, which will put the system in danger.

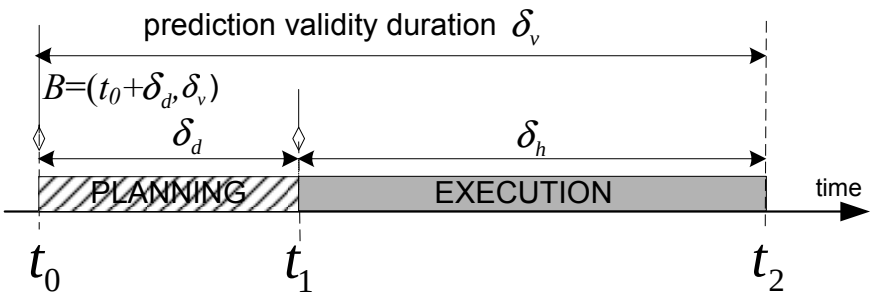


Figure III.4: Prediction validity model.

We assume the model prediction validity duration δ_v to be constant in a given homogeneous partially predictable environment. As a consequence, each cycle must be of equal duration and PMP becomes a periodic scheme. Thus, PMP algorithm iterates over a constant cycle of duration we note δ_c, with $\delta_c \leq \delta_d$ so as to fulfill the decision time constraint δ_d while allowing regular update of the model. The PMP algorithm operates until the last state of the planned trajectory reaches the goal. In case the planned trajectory has a duration $\delta_h < \delta_c$ the PMP cycle duration δ_c must be set to this new lower bound. In practice however, the magnitude of δ_h is much higher than δ_c (see fig. III.5).

It is important to notice that in general, during a cycle, a complete plan to the goal might not be calculated within the allotted time. In such a case, a partial plan only is computed and executed by the robot.

Finally, as our work mainly focuses on real world application within real environments, *ie* environments for which a complete knowledge of the future moving obstacle's motion can not be fully known in advance in general, we consider in the remaining of our work that the world is partially predictable.

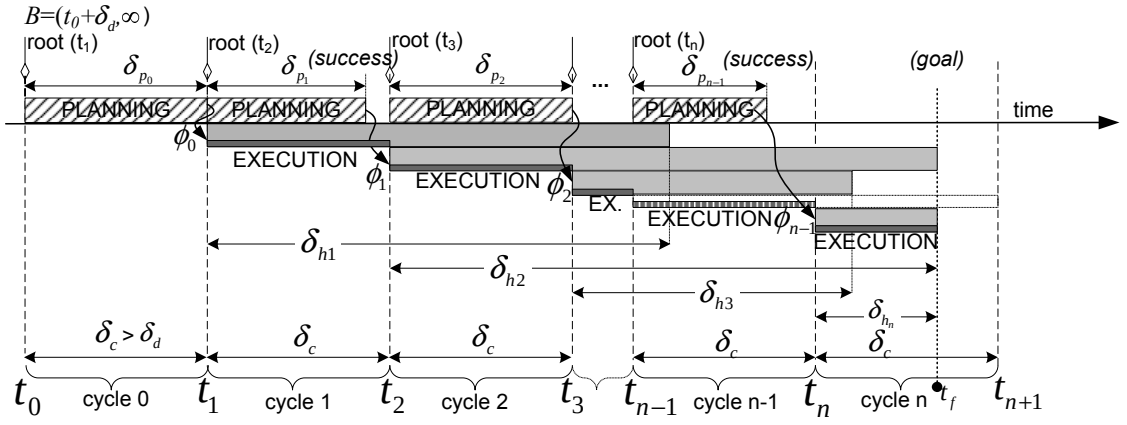


Figure III.5: Partial Motion Planning in a partially predictable environment.

2.1.3 Diffusion Technique

There are several possible ways to implement the diffusion technique, depending on the problem under study. In the case of a partially predictable environment, PMP has to use a new updated model of the world at the beginning of each cycle to generate a new trajectory. Therefore, each cycle is a new single query problem. As we reviewed in the former chapter, there are two approaches to address this problem. One relies on tree construction, the other on the construction of a roadmap. The latter one, usually used for known spaces, samples the space during a precomputation phase, in order to built a roadmap. For single query problem where both initial and goal configurations are known, the roadmap computation is however expensive. There are recent techniques that we discussed in the previous chapter, that use no precomputation and built a roadmap on the fly (*eg* [HKLR00]). These methods are suitable for dynamic environment however, such a fast construction is done at the expense of the roadmap denseness which does not guarantee then, to sufficiently capture the space connectivity, within an arbitrary computation time. Consequently, there is no guarantee the plan can reach the goal or has a specific duration, in case it is partial, as it depends on the randomly constructed roadmap.

A tree construction technique does not involve any roadmap precomputation, which is the first advantage. The second main advantage is that the tree construction is incremental. The benefit of incremental approaches is their ability to be interrupted anytime. Indeed, the duration of a PMP cycle is limited and when time has elapsed, it triggers the end of the trajectory computation. The planning technique is therefore capable to be interrupted and properly terminate so as to always return a current trajectory.

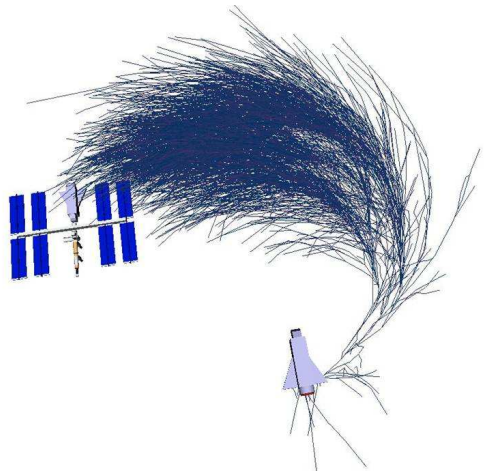


Figure III.6: Example of a tree of approaching trajectories of a spaceship to a satellite. (source: [PKB03])

The main purpose of the diffusion technique is the incremental exploration of $\mathcal{ST}$ by constructing a tree. Figure III.6 illustrates such a tree within $\mathcal{ST}$. During a PMP cycle, the major task is the trajectory generation, which relies on the diffusion technique. In our work the diffusion technique is based on a *tree construction method*. A general incremental tree construction operates as follows:

1. Initialize the tree $\mathcal{T}$, which can be seen as a directed graph, as the initial state, the singleton s_I.
2. Choose a vertex v_c of the tree that optimizes a metric or cost function (*eg* distance to the goal).
3. Expand the tree $\mathcal{T}$. Calculate a new vertex v_n by generating a small displacement from v_c and check that this simple piece of trajectory is safe. The goal is not to try to solve the entire planning problem, but rather to build an increment of $\mathcal{T}$. We note that the direction of the displacement depends upon the practical implementation and recall that for the RRT algorithm for instance, the displacement heads for a randomly selected state.
4. Insert the new vertex v_n in $\mathcal{T}$ (if it does not exist already).
5. Check for a solution in $\mathcal{T}$, from the initial state s_I to the goal state s_G.
6. If no solution is found in step 5, return to step 2, else stop.

In the next chapter, we will instantiate this technique to the case-study of a car-like robot, and detail the tree construction algorithm for this case.

We propose now to have a closer look to the third step of the diffusion technique, the tree expansion.As all deliberative methods, PMP relies on a model of the free set of states within $\mathcal{ST}$, *ie* the set of states for which the system is safe. We note $\mathcal{ST}_{free}$ the set of safe states within $\mathcal{ST}$. Formally, given a system $\mathcal{A}$ and obstacles $\mathcal{B}$, $\mathcal{ST}_{free} = (s,t) \in \mathcal{ST} | \mathcal{A}(s) \cap \mathcal{B}(t) = \emptyset$. In our diffusion technique, a new model of the world is obtained at each cycle, prior the trajectory generation. Complete methods rely on the *explicit* construction of $\mathcal{ST}$, which is a complex task for low dimension spaces and becomes as difficult as the motion planning problem itself for higher dimension spaces.

In order to avoid such a construction, one might resort to sampling techniques. The main idea of sampling techniques is to avoid the explicit construction of $\mathcal{ST}_{free}$ by probing the space with sampling schemes. Early methods used grids within $\mathcal{ST}$. We recall from previous chapter the term of *resolution complete* for these approaches that use deterministic sampling. From this perspective, a grid within $\mathcal{ST}$ is simply a particular case of sampling. Recent *probabilistically complete* methods use randomized sampling and create *probabilistic roadmaps*. These techniques have demonstrated they could solve difficult problems that could not be solved by means of complete methods based on an exact space discretization.

In our diffusion technique, the tree expansion somehow consists in incrementally build a similar grid. Indeed, at each step, the tree is expanded by adding a new vertex. A new vertex is added by applying a small displacement on a selected one already existing within the tree. This displacement might be arbitrary simple. Interestingly, for systems evolving under differential constraints (kinematics and dynamics of the system) that can be described by a differential equation, *ie* a transition function, of the form $\dot{s} = f(s,u)$ (Cf. section 1.2 of this chapter), instead of directly sampling the state-time space $\mathcal{ST}$, the method is to partition the time interval T into intervals of length Δt and chose a finite subset $\mathcal{U}_d$ of the control space $\mathcal{U}$. Then sampling states are not directly chosen but calculating by integrating over a time interval Δt the transition function $\dot{s} = f(s,u)$ with a given action $u(t) \in \mathcal{U}_d$ constant over the time interval Δt. This method is referred to as the discrete-time model ([LaV06]). No matter the approach that is chosen, the most important remains that any sample sequence is dense in the space on which sampling occurs and that this sequence leads to a dense set in $\mathcal{ST}_{free}$.

Thus, this tree expansion allows to incrementally and efficiently capture $\mathcal{ST}_{free}$. It is the heart of the tree construction technique integrated within the PMP.

2.2 Safety Issue

The partial motion planner explores the state-time space $\mathcal{ST}$ by constructing a tree toward the goal, within a limited time. In case no complete trajectory is calculated within the given time, a partial plan is returned to the robot. Let concentrate now on the last state of this partial trajectory as this is our major interest. For instance if the system is too close from an obstacle and he might not have the physical capability during the next PMP cycle to actually avoid a collision with this obstacle. We illustrated such a situation in the previous chapter, section 3.6. The most important point is that even though the state is collision-free, it remains in danger. This observation obliges us to refine the notion of safety and does not allow us anymore to consider a safe state as collision-free state.

In fact, PMP faces the same safety problem as reactive methods. The primary concern of navigation however it to ensure the safety of the robotic system. To ensure such a safety guarantee, the trajectory must account for the system's dynamics as well as the future behavior of the moving obstacles (Cf. chapter II, section 3.6). In our work, we use the concept of Inevitable Collision States (ICS) formally presented in [FA04] as the theoretical answer to our problem of safety. We will introduce in the next section this concept and detail how this concept can be carried over a PMP approach.

2.2.1 Concept of Inevitable Collision States (ICS)

One aspect of ICS is to extend the notion of configuration obstacle so as to explicitly account for both dynamics of the system and the environment. An ICS for a robotic system can be defined as a state for which, no matter what the future trajectory followed is, a collision with an obstacle eventually occurs.

Using the notations and definitions introduced in the previous chapter, an ICS is formally defined as follows:

Definition 6 (Inevitable Collision State)
We recall that $\phi \in \Phi$: $[t_0, \infty] \longmapsto \mathcal{U}$ denote a control input, ie *a time-sequence of controls. Thus, a state s is an Inevitable Collision State (ICS) if and only if $\forall \phi$, $\exists t_c$ (for a given ϕ) such that the state $\phi(s, t_c)$ is a collision state.*

An Inevitable Collision Obstacle (ICO) can be determined for a given obstacle $\mathcal{B}$ and control input applied to a system.

Definition 7 (Inevitable Collision Obstacle)
Given an obstacle $\mathcal{B}$ and a control input ϕ, $ICO(\mathcal{B}, \phi)$, the inevitable collision obstacle of $\mathcal{B}$ for ϕ is defined as:

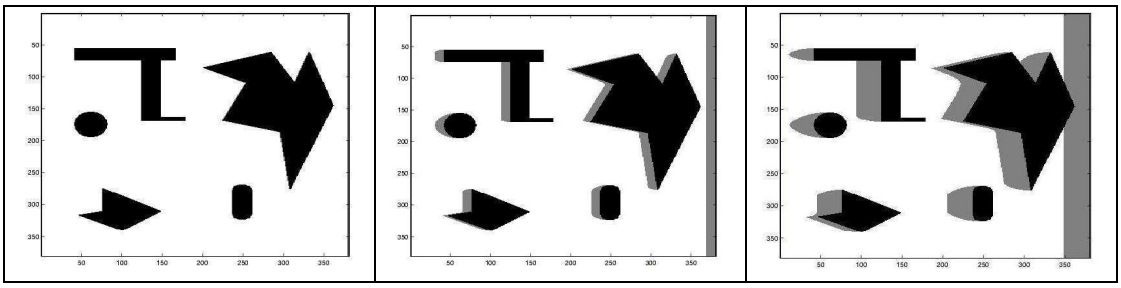


Figure III.7: Inevitable Collision Obstacle for a mass point robot system. (source: [LK01a])

$ICO(\mathcal{B}, \phi) = \{s \in S, \exists t, \phi(s,t)$ *is a collision state with* $\mathcal{B}\}$

The inevitable collision obstacle of $\mathcal{B}$ *is defined as follows :*

$ICO(\mathcal{B}) = \{s \in S | \forall \phi, \exists t, \phi(s,t)$ *is a collision state with* $\mathcal{B}\}$

The characterization of the inevitable collision obstacle of an obstacle $\mathcal{B}$ for a system with several control inputs ϕ, can be derived from the characterization of $ICO(\mathcal{B}, \phi)$ for every control input ϕ of the system using the following property ([FA04]):

Property 1
$ICO(\mathcal{B}) = \bigcap_{\phi} ICO(\mathcal{B}, \phi)$

In [LK01b], the authors also believe that the problem of safety within the kinodynamic motion planning problem is to be addressed from the concept of the region of inevitable collision, without actually doing it, the task being complex. In figure III.7 the forbidden area for a system moving to the right is represented, at three different speed. The authors illustrate here the consequences of the speed, *ie* the system's dynamics, on its safety. As the speed increases, $\mathcal{ST}_{free}$ reduces. From the ICS point of view, the physical obstacles augmented by the forbidden area due to the system's dynamics represent in fact the ICO for the system.

In general, a typical system has an infinite number of control inputs, which leaves little hope of being actually able to compute $ICO(\mathcal{B})$. Fortunately the approximation property established in the work of [FA04] enables to compute a conservative approximation of $ICO(\mathcal{B})$ by using a subset of the whole set of possible control inputs. In a formal way :

Property 2 (ICO Approximation)
Let I *denote a subset of the set of possible control inputs* Φ

$$ICO(\mathcal{B}) \subset \bigcap_{I} ICO(\mathcal{B}, \phi)$$

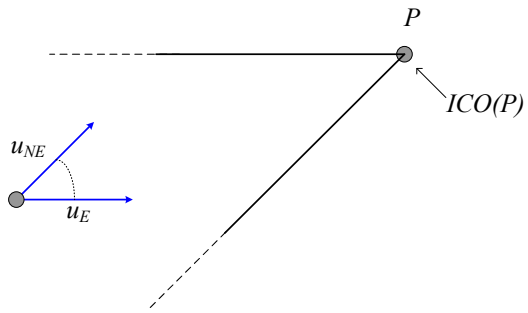


Figure III.8: North-East East system and a static point obstacle.

Using these first properties, we can now introduce the principle of the construction of ICOs.

2.2.2 ICO Representation

Let us now consider a simple example so as to illustrate more precisely this concept. We consider the case of a planar point system $\mathcal{A}$ that can move in two directions only (North-East (NE) and East (E)) at constant unit speed . A state of $\mathcal{A}$ is $s = (x, y) \in \mathbb{R}^2$ and a control u can take only two values associated to each direction : either u_{NE} (North-East direction) velocity vector of orientation $\frac{\pi}{4}$ or u_E (East direction) velocity vector of orientation 0. This system has only two possible constant control inputs: ϕ_{NE} and ϕ_E that respectively correspond to motions in the North-East and East directions.

Let consider $\mathcal{B}$ a static point obstacle P (see fig. III.8). The two constant controls are represented by the two velocity vectors of unit speed (namely u_{NE} and u_E). In such a case, for the constant control input ϕ_{NE} , the set of inevitable collision states of P for ϕ_{NE}, namely $ICO(P, \phi_{NE})$ using the previous notations, is the half-line of orientation $\frac{\pi}{4}$ that represents the set of points reachable by the system if it follows ϕ_{NE}, until it reaches and collides P. Similarly, in case the system's motion follows the constant control input ϕ_E, $ICO(P, \phi_E)$ represents a semi-line (see fig. III.8). Then, for our complete system that use both controls u_{NE} and u_E, the Inevitable Collision Obstacle is determined, using the approximation property (Cf. section 2.2.1, property 2). Namely, $\forall \phi$, $ICO(P, \phi) = ICO(P, \phi_E) \cap ICO(P, \phi_{NE}) = P$. Therefore the Inevitable Collision Obstacle of P for our system reduces to the point P itself .

Then, let $\mathcal{B}$ be a rigid bar. In this case for each constant control input u_{NE} and u_E, $ICO(B, \phi_{NE})$ and $ICO(B, \phi_E)$ represent respectively a band of forbidden states of orientation respectively $\frac{\pi}{4}$ and 0 terminating in the obstacle. In this case, the inevitable

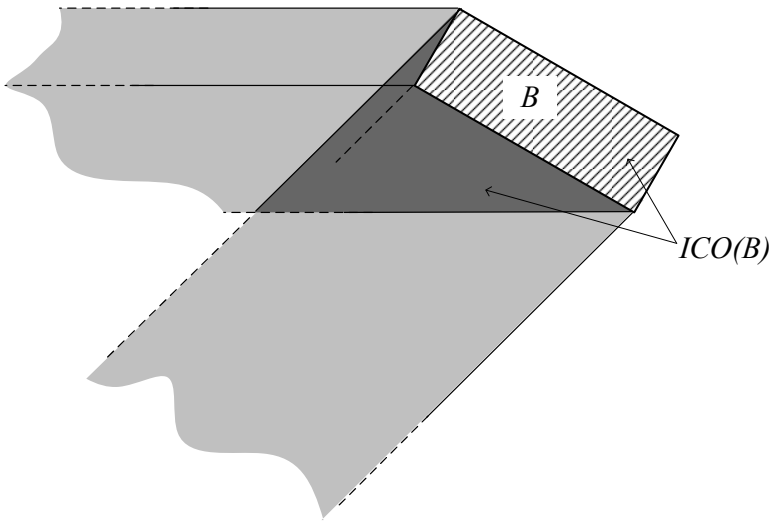


Figure III.9: North-East/East system and a static obstacle.

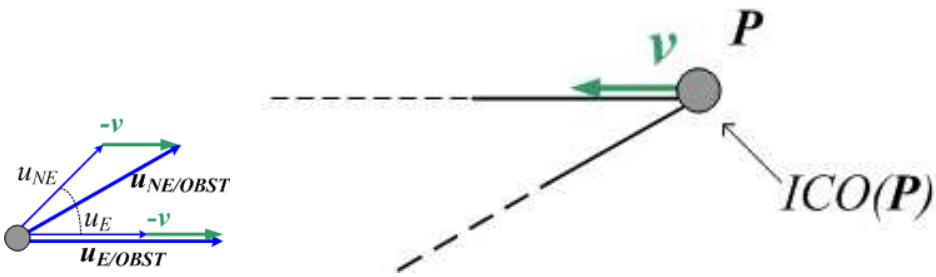


Figure III.10: North-East/East system and a moving point obstacle.

collision obstacle of B $ICO(B, \phi)$ is the intersection area of this two bands forming a triangle (see fig. III.9).

The concept of inevitable collision states handles moving obstacles as well. In case our point obstacle P is moving under a control v, the construction of $ICO(P, \phi)$ is slightly different than in the static case. The method we propose is to work in relative control inputs. This concept is similar to the *relative-velocity* paradigm used in the Velocity Obstacles (VO) approach ([FS98]). If we consider the control of the system u_{NE} relatively to the obstacle P of velocity v we can sum the vectors and construct a new control $u_{NE/obst} = u_{NE} + (-v)$ which can be assumed as the new control for the system with respect to a static obstacle P. The orientation of the line that represents all states of the system under this control before it collides P therefore changes (fig. III.10).

Let consider the rigid bar moving at a velocity v. Similarly than in the previous example the representation of $ICO(B, \phi)$ consists in the intersection of two bands of orientation

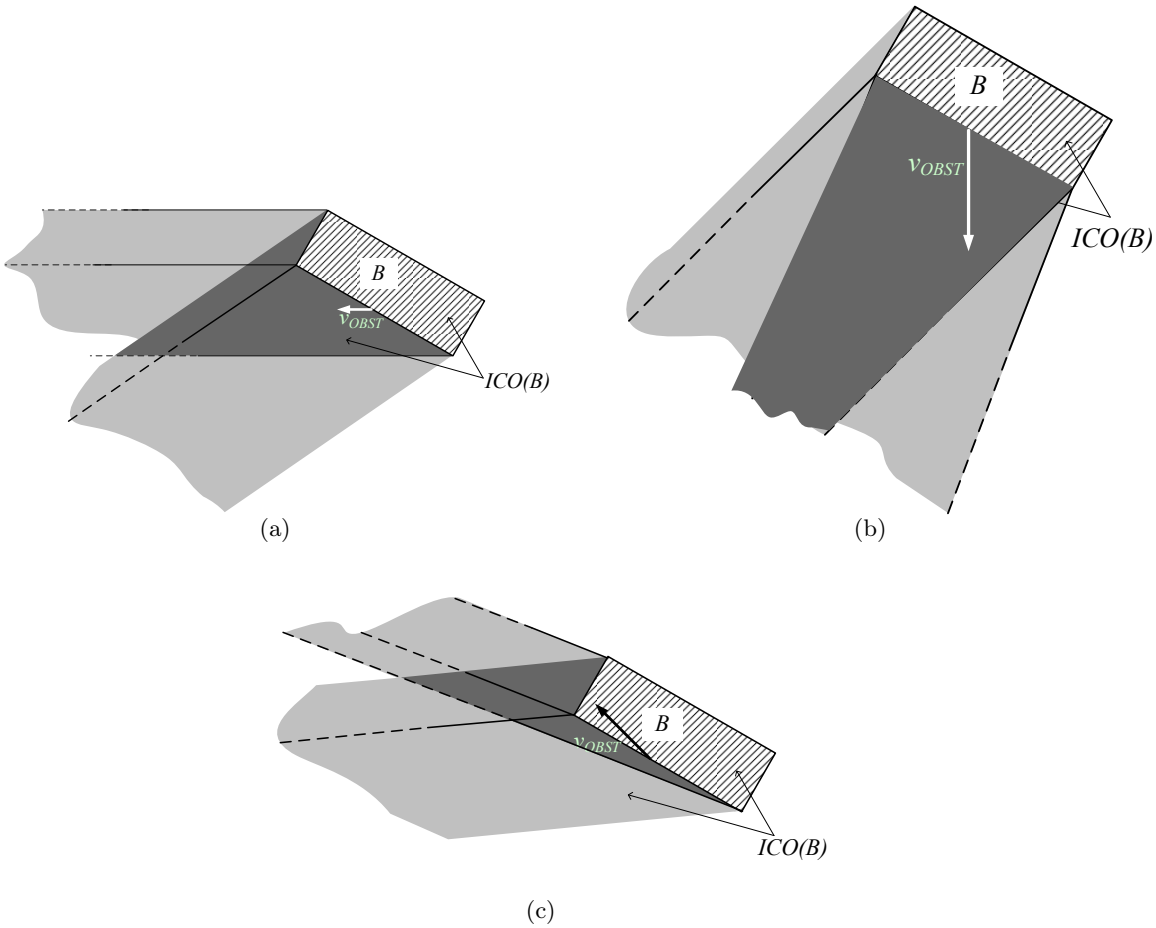


Figure III.11: North-East/East system and a moving obstacle of different velocities.

respectively $u_{NE/obst}$ and $u_{E/obst}$ (see fig. III.11).

In fig. III.11 we illustrate the inevitable collision obstacle of B for our system, when the obstacle is moving under different velocities v. We can observe how the velocity of the obstacle impacts the inevitable collision obstacle for a given obstacle. Interestingly, we can observe that the size of the area of the Inevitable Collision Obstacle gives an indication on the danger the obstacle represents to the system, For instance, we observe that when the obstacle moves toward the system, $ICO(B)$ logically increases, whereas when the obstacle moves away from the system, $ICO(B)$ decreases. This illustrates how the concept of Inevitable Collision Obstacle accounts for both dynamics of the system and the obstacle.

Let our system have a third control input u_{SE} in the South-East direction. Interestingly if we compare $ICO(B)$ for the system with three control input, we observe that $ICO(B)$ in all cases is smaller. This is another illustration of the approximation property. The

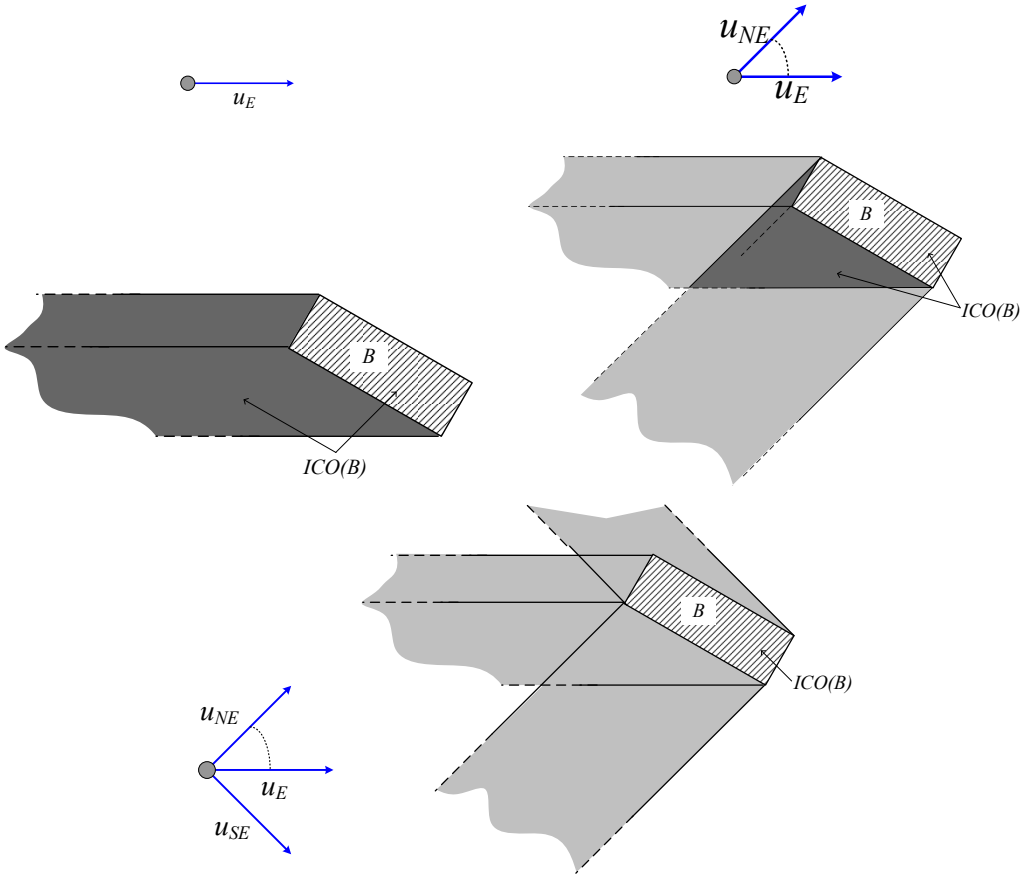


Figure III.12: Comparison between East, North-East/East and North-East/East/South-East systems and a static obstacle.

more control inputs in the subset $\mathcal{I}$, the better the approximation of the true set of ICS. By adding maneuvers, the system's ability to avoid collision increases (see fig. III.12) hence the reduction of the ICS set.

When the different solutions to the safety problem, presented in the previous chapter, are evaluated from the perspective of Inevitable Collision States, they all appear weaker and therefore not satisfying. The concept of Inevitable Collision States in fact encompasses all these approaches. For all partial or reactive planning schemes, the main risk is to bring the system in an Inevitable Collision States. Insuring safety for a robotic system therefore requires to guarantee that all calculated states of the planned trajectory are not ICS. Indeed, where motion planning scheme could be satisfied with collision-free states calculation, partial motion planning requires a much stronger requirement, that generated trajectories are ICS-free.

2.2.3 ICS Concept in the PMP Approach

The ICS concept is very general. It applies to any type of obstacles (static and dynamic) and accounts for the system's dynamics as well. This concept fulfills in fact the three requirements listed in the previous chapter, section 3.6. If we can guarantee that the robotic system never enters an inevitable collision states, then the safety problem is solved. Thus, as PMP faces a strong safety issue, our goal in our work is to construct an ICS-free tree within $\mathcal{ST}$. Ideally, a trajectory is safe, if all its states are not inevitable collision states. Formally,

Definition 8 (Safe Trajectory)
A trajectory, defined by the initial state s_0 at time t_0 and the control input ϕ over $[t_0, t_f]$, is safe if and only if $\forall t \in [t_0, t_f]$, $s(t) = \phi(s_0, t)$ is not an ICS.

In order to decrease the computational burden of PMP we present a reduction property that allows to check the safety of a partial trajectory by simply considering the last state of this trajectory. To begin with we prove that for a given control input, all the states between an ICS and the corresponding collision state, are ICS.

Property 3
Let s be an ICS at time t_0. For a given control input ϕ, let t_c denote the time at which a collision occurs. Then $\forall t \in [t_0, t_c], s(t) = \phi(s, t)$ is also an ICS.

Proof: suppose that a state $s_i = \phi(s, t_i)$, with $t_i \in [0, t_c]$, is not an ICS. By definition, $\exists \phi^j$ that yields no collision when applied to s_i. Let ϕ^i denote the part of ϕ defined over $[t_0, t_i]$. Clearly, the combination of ϕ^i and ϕ^j also yields no collision when applied to $s \rightsquigarrow$ contradiction. $\diamond$

We can now state the reduction property for a partial trajectory that states that provided a trajectory is collision free, if the last state of the trajectory is not an inevitable collision state then none of the states of the trajectory are inevitable collision states.

Property 4 (Reduction Property)
Given a trajectory defined over $[t_0, t_f]$, if
(H1) *the trajectory is collision-free and*
(H2) *$s(t_f)$ is not an ICS*
then $\forall t \in [t_0, t_f]$, $s(t)$ is not an ICS.

Proof: Suppose that $\exists t_i \in [t_0, t_f]$ such that $s_i = s(t_i)$ is an ICS. Then, by definition, $\forall \phi, \exists t_c$ such that $\phi(s_i, t_c)$ is a collision state. If $t_0 \leq t_c \leq t_f$ then collision occurs before $t_f \rightsquigarrow$ contradiction with H1. Now, if $t_c > t_f$ then by previous property P1, we must have $s(t_f)$ is an ICS $\rightsquigarrow$ contradiction with H2. $\diamond$

This property is important since it proves a trajectory is continuously safe, *ie* ICS-free, if its final state is ICS-free. Therefore, it permits a practical computation of a safe trajectory.

3 Discussion

If we compare our approach to the ones described in previous chapter and for which main features are summarized in Table III.2, we can make several observations:

- The diffusion technique allows PMP to use a transition function that accounts explicitly for kinematic constraints without relying to crude approximations as in several reactive techniques (*eg* Curvature Method (CM), Nearness Diagram (ND), Velocity Obstacles (VO), Vector Field Histogram (VFH)).
- This approach enables also to account explicitly for the dynamics of the system whereas several approaches account for maximal longitudinal acceleration only (*eg* Dynamic Window Approach (DWA)). A few recent approaches have adopted a similar approach based on the use of a transition function (Fast Rapidly-Exploring Random Trees of [BV02] (FRRT), Fast Probabilistic Roadmap planners of [HKLR00] (FPRM), or the work of [Fra01]). As a result, PMP generates high quality feasible trajectories that faithfully represent the motion of the system at execution.
- PMP uses a diffusion technique within $\mathcal{ST}$ which enables to account explicitly for dynamic environment as opposed to the methods that consider obstacles without their future motion (*eg* Elastic Band of [BK00b] (EB), star-Vector Field Histogram of [UB00] (VFH*)) .
- Finally, PMP is the only approach to our knowledge (beside the work of [Fra01]) that handles the hard real time decision constraint explicitly. The main difference with the method proposed in [Fra01], is the environment, which for our target application is assumed to be partially predictable.

Table III.2: Overview of PMP features compared to other methods. (+/-/0 respectively means strength/weakness/specific cases only).

	Kinematic constraints	Dynamic constraints	Dynamic Environment	Real-Time constraint	Safety	Real world application
PMP	+	+	+	+	+	+
DWA [Sim96]	+	0	0	0 (1step)	0 (braking traj.)	+
GDWA [BK99]	+	0	0	−	0 (braking traj.)	+
EDWA [OM05]	+	0	+	−	0 (braking traj.)	+
CM [Sim96]	+	−	0	0 (1step)	−	+
ND [MM00]	0	−	0	0 (1step)	−	+
GND [MMSA01]	0	−	0	0	−	+
VFH [BK91]	0	0	0	0 (1step)	−	+
VFH* [UB00]	0	0	0	0	−	+
VO [FS98]	−	0	+	0 (1step)	0	+
NLVO [LSSL02]	0	0	+	+	0	0
EB [Kha96]	−	−	−	0	−	+
EB [BK00b]	+	−	0	0	−	0
FRRT [BV02]	+	+	+	0 (no guarantee)	0 (escape traj.)	+
DP[LFG+05]	−	−	−	0	−	0
FPRM [HKLR00]	+	+	+	0 (no guarantee)	0 (escape traj.)	+
FRA[Fra01]	+	+	+	+	0 (braking traj.)	−

(Notre approche)

Placé dans un environnement dynamique, un robot a un temps limité afin de prendre une décision sur son futur mouvement. Cette contrainte temps réel imposée par la dynamique de l'environnement doit être prise en compte de manière explicite par la technique de planification. Parmi les récentes techniques de planification, les méthodes probabilistes ont montré récemment des résultats impressionnants. Cependant, aucune garantie ne peut être apportée en terme de temps de calcul. Ainsi, il est probable que dans un environnement dynamique arbitraire ces méthodes ne remplissent pas les contraintes imposées. En fait il est de notre avis que le problème de planification de mouvement en environnement dynamique sous contrainte temps réel est un problème impossible à résoudre en général étant donné la complexité intrinsèque du problème de planification de mouvement.

La Planification de Mouvement Partiel

(PMP) est la réponse que nous proposons dans ce travail au problème de navigation en environnement dynamique. PMP est spécialement conçu pour prendre en compte de manière explicite la contrainte de temps imposée par l'environnement. L'algorithme PMP consiste en une séquence de cycle PMP. Chacun des cycles prend en compte de manière explicite la contrainte de temps de décision. Un cycle peut être décrit de la manière suivante :

1. Le modèle de l'environnement est obtenu à un instant t_i.

2. Une trajectoire sûre est planifiée de l'état de la trajectoire nominale courante, à l'instant t_{i+1} du début du prochain cycle, en direction de l'état but.

3. A l'instant t_{i+1}, le temps de calcul disponible est écoulé. La meilleure trajectoire partielle, celle qui optimise une fonction de coût donnée est sélectionnée Cette trajectoire a une durée spécifique. Cette trajectoire devient le nouveau plan à exécuter dans le monde réel par le robot durant le prochain cycle.

Lors de la planification, une technique de diffusion par exploration incrémentale de l'espace à explorer, cad. l'espace des états, est utilisée. Cette technique s'appuie sur la construction d'un arbre. L'expansion de cet arbre se base sur des techniques d'échantillonnage probabiliste, ce qui évite la construction explicite de l'espace exploré. PMP explore l'espace des états temps en construisant un arbre vers le but dans un temps limité. Dans le cas ou une trajectoire complète vers le but n'a pas été déterminée, un plan partiel revient au robot. Si nous nous concentrons sur le dernier état de la trajectoire partielle, dans le cas ou cet état se trouve à proximité d'un obstacle, il est possible qu'il n'ait pas la capacité physique d'éviter cet obstacle durant le prochain cycle. Ainsi, il est important de noter que quand bien même le système est sans collision, il n'en demeure par moins qu'il est en danger, ce que nous illustrons dans le travail. Cette observation nous oblige donc à redéfinir la notion de sureté et ne plus considérer uniquement la notion d'état sans collision. En fait, PMP fait face au même problème de sureté que les méthodes réactives. La sûreté de notre système passe dans notre travail par l'utilisation des états de collisions inévitables (ICS), formalisme récemment introduit et sur lequel nous nous basons pour apporter la réponse théorique à notre problème de sureté. Un ICS peut être défini comme un état pour lequel, quel que soit sa future trajectoire à suivre, une collision avec un obstacle aura lieu.

Si nous comparons notre approche théorique avec les approches présentées lors de l'état de l'art, nous pouvons faire les observations suivantes :

- *La technique de diffusion permet à PMP d'utiliser une fonction de transition (déduite du modèle du système) qui permet la prise en compte explicite des différentes contraintes cinématiques sans devoir faire des approximations, comme le font la plupart*

des approches réactives.

- *Cette approche permet, pour la même raison, la prise en compte explicite de la dynamique du système alors que de nombreuses approches ne prennent en compte que la contrainte en accélération longitudinale maximale. Certaines techniques récentes utilisent également une fonction de transition. De cette façon, PMP génère des trajectoires de qualité, faisables, qui représentent fidèlement le mouvement du système lors de l'exécution.*

- *PMP utilise une technique de diffusion dans l'espace des états qui permet la prise en compte explicite de la dynamique de l'environnement, en opposition aux méthodes qui considèrent les obstacles environnants sans prendre en compte leur mouvement futur.*

- *Finalement, PMP est la seule approche à notre connaissance qui permet la prise en compte explicite de la contrainte de temps réel imposée par l'environnement.*

Chapter IV

Case study of a Car-like System

1 Introduction

Partial Motion Planning (PMP) has been introduced in this work as the theoretical answer to the problem of navigation within a dynamic environment. In this chapter, we propose to discuss a practical implementation. In particular, our interest focuses on the navigation of autonomous cars within urban environments. We propose therefore to study the use of PMP for the case of a car-like vehicle evolving within a dynamic environment. At first, we present the model of vehicle that is suitable for this study. At second we discuss the construction of a model for the environment, as motion planning consists in calculating a plan within a space for which a model exists. At third, we discuss how practically, the notion of safety and particularly inevitable collision states (ICS) can be integrated within the Partial Motion Planner. Finally, an adaptation of the exploration scheme for this case study is described.

2 Model of the Vehicle

There are several models that can be used to describe a car. However, the more constraints are taken into account, the more complex becomes the model and the more elaborate the planning strategy must be. Therefore, in this section we will present the most famous models used in the literature before to introduce the one chosen for this work. Our goal is to bring to the fore the differences between the models so as to clearly present the limitations of the two first models and better motivate our choice for the third and more complex one, the *dynamic* car model. This will contribute to show, in the remaining of

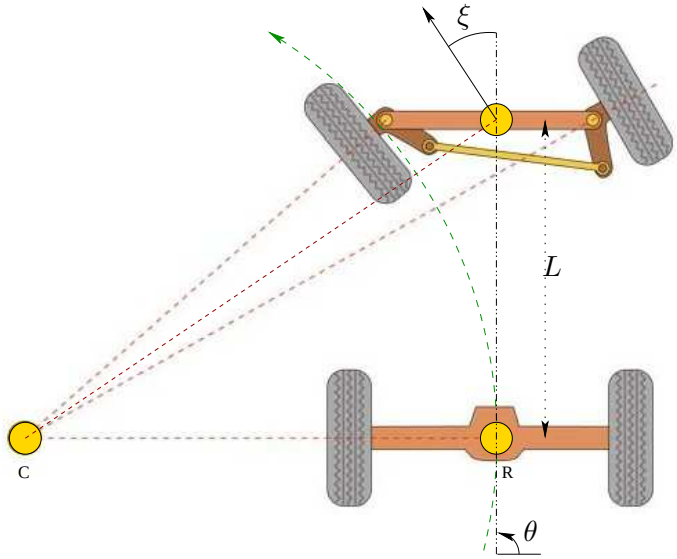


Figure IV.1: Car-like system.

this chapter, the strong ability of PMP to handle such complex models.

2.1 General Assumptions

In our case study we consider the following :

1. the car-like robot $\mathcal{A}$ evolves on a planar workspace $\mathcal{W} \equiv \mathbb{R}^2$

2. the front and rear wheels of the robot obey the pure rolling and no slipping conditions (*ie* the velocity vector is null at the point of contact between the wheel and the road). These constraints keep the car from moving sideways. Practically, they make parallel parking a challenging task!

2.2 Simple Car Model

We recall from chapter II, section 3.3 the simple car model notation. Let θ be the car heading orientation, ξ be the steering angle and L be the distance between the front wheels and rear axles (fig. IV.1). If the steering angle ξ is arbitrarily fixed, the car will describe a circle of radius ρ. At first, in a small time interval Δt, the car moves in the direction imposed by the rear wheels direction. As this time interval tends to zero, this implies $dy/dx = \tan\theta$. Since $dy/dx = \dot{y}/\dot{x}$ and $\tan\theta = \sin\theta/\cos\theta$ we obtain the implicit nonholonomic constraint II.1 introduced in chapter II, section 3.3.1 that we recall here :

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \tag{IV.1}$$

Suppose the speed v of the vehicle is completely specified by a control u_v, this constraint can be written in the following parametric form :

$$\begin{aligned} \dot{x} &= u_v \cos\theta \\ \dot{y} &= u_v \sin\theta \end{aligned} \tag{IV.2}$$

Furthermore, let w the distance traveled by the car and ρ represents the radius of a circle that will be traversed by the center of the rear axle, if the steering angle is fixed. We note that $dw = \rho d\theta$. From trigonometry, $\rho = L/\tan\xi$. We obtain by combining both equations : $d\theta = \frac{\tan\xi}{L} dw$. By dividing both sides by dt we obtain the second nonholonomic constraint in $\dot{\theta}$:

$$\dot{\theta} = \frac{v}{L}\tan\xi \tag{IV.3}$$

In addition to the action variable u_v, we suppose the steering angle is completely defined by the action variable u_ξ, we obtain the following parametric equation :

$$\dot{\theta} = \frac{u_v}{L}\tan u_\xi \tag{IV.4}$$

Equations IV.2 and IV.4 describe a system of three configurations of our system. We refer to the model that this system describes describe as the *simple car model* of a car-like system.

2.3 Smooth Car Model

For the simple car model, planar paths are made up of line segments connected with tangential circular arcs of minimum radius (see [Dub57, RS90]). The curvature of this type of path is discontinuous. Discontinuities occur at the transition between segments and arcs and between arcs with opposite direction of rotation. The curvature is related to the front wheels' orientation (see section 2.2), therefore when a real car is to track precisely such a path it has to stop at each curvature discontinuity so as to reorient its front wheels, since such reorientation can not be physically instantaneous. In case the car does not stop, there will be a deviation between the planned path and the executed one which might be quite large specially at high vehicle speed. Indeed, in reality, a vehicle will describe a clothoid during such a transition. The shape of the clothoid depends upon the

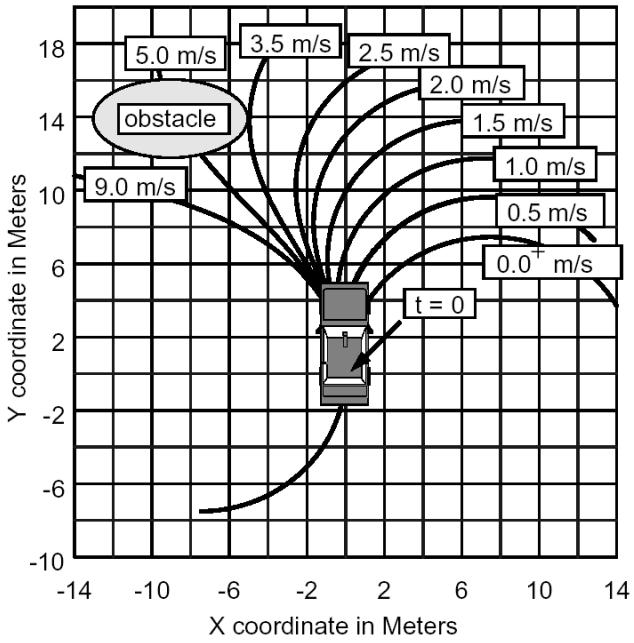


Figure IV.2: Path shapes while turning at different vehicle speeds.

speed of the vehicle as illustrated in fig. IV.2. In case such transition is not considered at the planning stage, the system will deviate from its original plan at execution and might be in danger. Curvature continuity is therefore a desirable property. Besides, since the derivative of the curvature is related to the steering velocity of the car, it is also desirable that the derivative of the curvature be upper bounded so as to ensure that such paths can be followed at a given speed. One way to achieve this is to consider the action variable u_ψ where $\psi = \dot{\xi}$ instead of u_ξ. The parametric form of the second nonholonomic constraint becomes :

$$\begin{aligned} \dot{\theta} &= \tfrac{u_v}{L} \tan \xi \\ \dot{\xi} &= u_\psi \end{aligned} \tag{IV.5}$$

Equations IV.2 and IV.5 form the system that describe the *smooth car model*.

2.4 Dynamic Car Model

In addition to the kinematics it is important to also consider the dynamic constraints. The constraints that impact most the vehicle's motion are the following :

1. The **maximum speed constraint**, which is maximal possible speed of the vehicle,

is expressed as follows:

$$0 \leq \dot{x}^2 + \dot{y}^2 \leq v_{max}^2 \tag{IV.6}$$

2. The **maximum acceleration and deceleration constraint,** which is the maximal possible acceleration (resp. deceleration) resulting from the mechanical internal capabilities of the system in terms of engine/electrical power (resp. braking performance) is described by :

$$\alpha_{min} \leq \alpha \leq \alpha_{max} \tag{IV.7}$$

In order to be able to account for these constraints, it is necessary to disregard the speed v as action variable and consider the vehicle's longitudinal acceleration u_γ, where $\gamma = \dot{v}$.

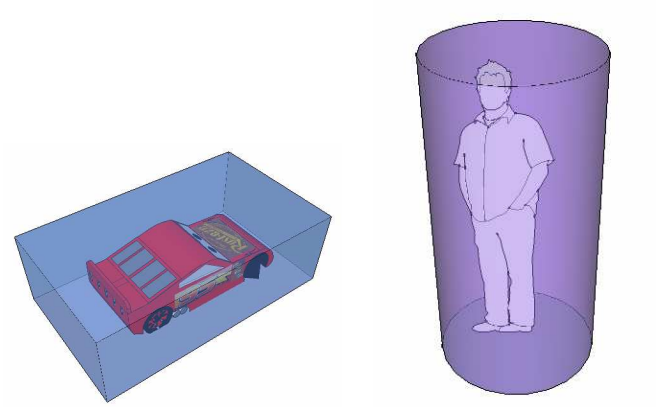
Of course there are several other dynamic constraints that could be worth considering. However, as the complexity of the problem increases with its dimension (Cf. 3.4.2), we do not consider these higher order dynamic constraints as relevant and critical in our work.

Therefore, to account for the first two dynamic constraints, we select at first, the action variable u_γ, *ie* the throttle or accelerator pedal of a car-like system, in order to control its longitudinal acceleration . At second, we choose the angular speed of the steering wheel u_ψ as second control variable. Thus, we built the *dynamic car model* as follows :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\xi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ v\frac{\tan\xi}{L} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_\psi + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_\gamma \tag{IV.8}$$

where γ is the acceleration of the rear wheels and ψ the angular steering speed of front wheels.

In our work we consider this model of the car like system. This model has a drift term (the first term on the right hand side) which is a deviation accounting for the inertia of the system. The equation IV.8 is a transition function of the form $\dot{s} = f(s, u)$ where $s \in \mathcal{S}$ is the state of the system, $\dot{s}$ its time derivative and $u \in \mathcal{U}$ a control. $\mathcal{S}$ is the state space and $\mathcal{U}$ the control space of $\mathcal{A}$. A state of $\mathcal{A}$is defined by the 5-tuple $s = (x, y, \theta, \xi, v)$ where (x, y) are the coordinates of the rear wheel, θ is the main orientation of $\mathcal{A}$, ξ is the orientation of the front wheels and v is the linear velocity of the rear wheels. A control of $\mathcal{A}$ is defined by the couple (γ, ψ) where γ is the rear wheel linear acceleration. and ψ the steering velocity. with $\gamma \in [\gamma_{min}, \gamma_{max}]$ (acceleration bounds), $\psi \in [\psi_{min}, \psi_{max}]$ (steering velocity bounds), and $|\xi| \leq \xi_{max}$ (steering angle bounds).



(a) rectangular box for vehicles (b) cylindric box for pedestrians

Figure IV.3: bounding boxes used for geometric collision detection.

3 Model of the World

In order to be able to construct a complete model of the world, several information on all surrounding obstacles must be provided.

At first, information on the **shape** of all $\mathcal{B}$ must be informed. It is common to approximate obstacles using bounding boxes or cylinders so as to avoid complicated geometric shapes description and optimize geometric collision detection calculations. In our work we consider both rectangular bounding boxes, defined by the width and depth (*eg* surrounding cars in a city) and cylindrical bounding box, defined by the radius (*eg* surrounding pedestrians) as illustrated in fig. IV.3.

At second it is needed to know **the future trajectory** of each obstacle. The future trajectory can be provided as a sequence of configurations, velocities or accelerations so as to allow a complete reconstruction of the obstacles $\mathcal{B}(t)$ within $\mathcal{ST}$. In our work we consider sequences of velocities for each obstacle, to describe its future motion. Fig. IV.4 illustrates this idea. Thus, a model consists of a set of obstacles, the shape of which is modeled as described above and a their related sequence of future velocities. Each velocity holds for an arbitrary time slice which allows to calculate and predict the position of each obstacle.

4 ICS Computation

In chapter III, section 2.2, the concept of Inevitable Collision States (ICS) was introduced. From a theoretical point of view, it is very appealing. First, it allows a formal definition

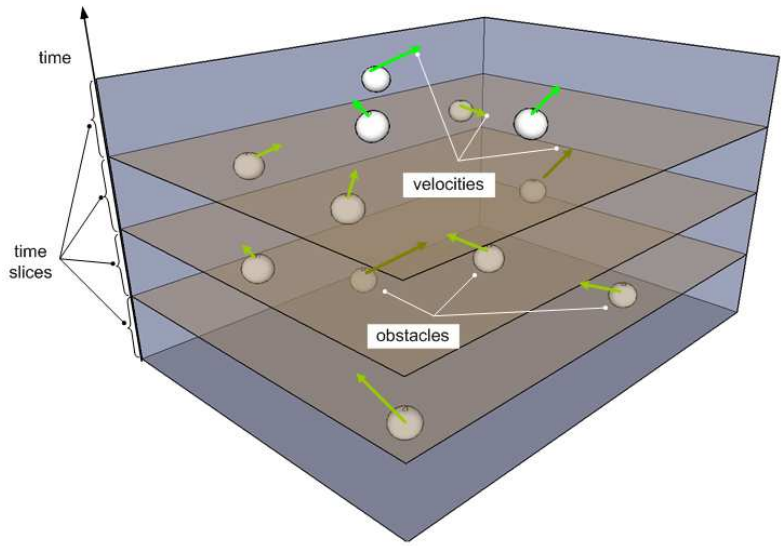
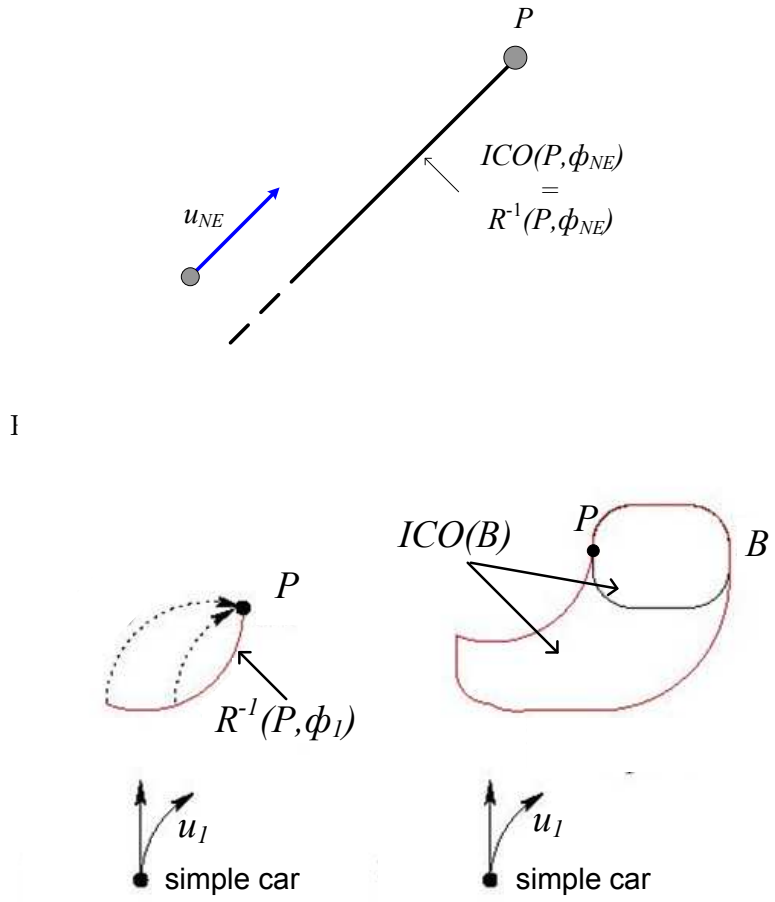


Figure IV.4: Model of the environment.

of the safety problem in general. It provides a tool to analyze and compare the different safety levels offered by the existing navigation techniques. Second, it opens the door to safe navigation schemes, for which the safety problem addressed from the concept of ICS has an explicit meaning. In fact, the Partial Motion Planner (PMP) proposed in this work is the first navigation strategy that is formally exploring the safety issue from this perspective. From a practical point of view however, the problem becomes the characterization of the ICS for a system in a given environment. At first, we will discuss specific systems for which an analytical representation can be done. For complex systems however, one will have to rely on numerical approaches to characterize it. We discuss this aspect before to finally introduce and motivate our approach, based on implicit ICS representation for this case-study.

4.1 Analytic Representation

We propose to get a better understanding of the $ICO(\mathcal{B})$ characterization by analyzing the reachable set of states for the system $\mathcal{A}$. Indeed, it might be possible for all states $s \in \mathcal{ST}$ to be visited by $\mathcal{A}$ but in general, from an arbitrary state s_0, some states may not be reached, depending upon the differential constraints applied on the system. Formally, given a control input ϕ and a state s_0, a state s is reachable from s_0 by ϕ if and only if $\exists t, \phi(s_0, t) = s$. Let $\mathcal{R}(s_o, \phi)$ denote the set of states reachable from s_0 by ϕ. We recall that Φ is the set of all possible control inputs for $\mathcal{A}$. Consequently we note $\mathcal{R}(s_0, \Phi)$ the reachable set from s_0 which is the set of all states that are visited by any trajectories that



(a) determination of the set ICS for a single control u_1 case of point obstacle

(b) determination of the set of ICS for a single control u_1 in case of square shape obstacle

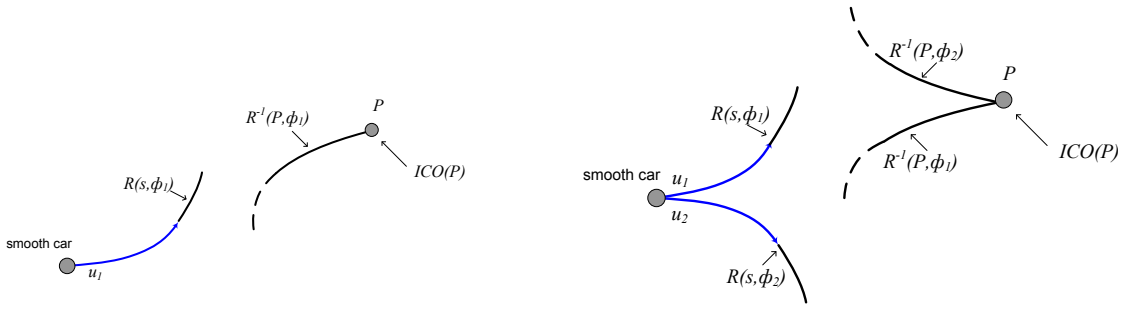
Figure IV.6: Inevitable Collision Obstacle (ICO) of a static obstacle for a *simple car*.

start at s_0. Formally,

Definition 9 (Reachable set)
$\mathcal{R}(s_0, \Phi) = \{s \in \mathcal{ST} | \exists \phi \in \Phi \ \ and \ \exists t \in [0, \infty) such \ that \ s = s_0(\phi, t)\}$

Now the definition of reachable set has been introduced, we can look at ICS from another perspective. Given a point obstacle P, for a given control input ϕ, $ICO(P, \phi)$ is trivially equal to $\mathcal{R}^{-1}(P, \phi)$. Fig. IV.5 illustrates the relation between the set of ICS for a point obstacle P and the set of reachable states given a control input of the system.

Let consider a simple car. Its motion describes arc of circles (and straight segments, but not important here). From a given state s_0 , the reachable set for this system is therefore an arc of circle. This arc of circle has radius $\rho = L/\sin\xi$. In such a case, an analytic



(a) set of points for which the obstacle is reached given a single control input

(b) set of points for which the obstacle is reached given two control inputs

Figure IV.7: ICO for a point obstacle P and a *smooth car*.

construction is possible. Fig. IV.6 illustrates such an analytical construction of $ICO(\mathcal{B})$ for simple car in front of an obstacle of a squared shape, considering one control input u_1. In fig. IV.6(a), a single point obstacle is considered and $\mathcal{R}^{-1}(P, \phi_1)$ is illustrated, which logically represents an arc of circle as weel, of same radius as the one describe by the simple car under the same control. By extension, it is possible to analytically calculate the complete $ICO(\mathcal{B}, \phi_1)$, illustrated in fig. IV.6(b).

Furthermore, we know that computing the ICS for a given system requires to consider the set of all of its possible future trajectories. In chapter III section 2.2.2, the approximation property 2 is used as a mean to represent explicitly a conservative approximation of $ICO(\mathcal{B})$. For every obstacles $\mathcal{B}$, only a limited finite set of control inputs for the system can be considered. Hence, for the case of the simple car, the calculation can be performed given a finite subset of trajectories. For instance, the velocity u_v of the vehicle can be fixed and three different steering angle values chosen, namely $u_\xi \in \left\{-\frac{\pi}{6}, 0, \frac{\pi}{6}\right\}$. Thus we apply the approximation property for the system over a subset of all possible trajectories consisting in a set of three control inputs of 2-tuple (u_v,u_ξ) defined as $\mathcal{I} = \{(u_v, u_\xi) | u_v = \text{const.}, u_\xi \in \left\{-\frac{\pi}{6}, 0, \frac{\pi}{6}\right\}\}$. Then, for this model, an analytical representation of $ICO(\mathcal{B})$ which is conservative thanks to the approximation property can be explicitly calculated.

4.2 Explicit Numerical Representation

Unfortunately, there are systems, for which differential constraints greatly complicates the analytical expression of the path they describe. Even though a nice geometric representation can be obtained for simple systems, in general however, this might not be as simple. Let consider the *smooth car* model. The action variables of such a system are the vehicle speed u_v as in the previous example and the steering rate u_ψ. Under a fixed control input of the 2-tuple (u_v,u_ψ) the car will describe a clothoid (depending upon the initial steer-

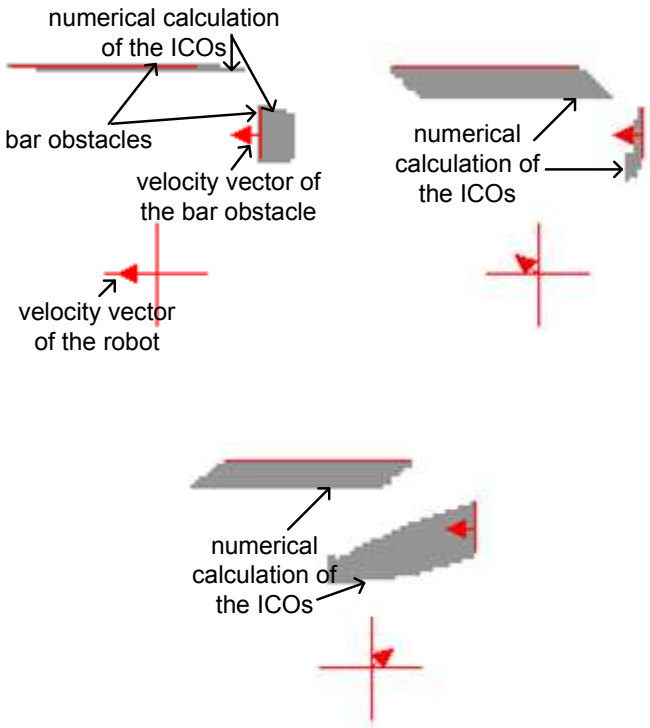
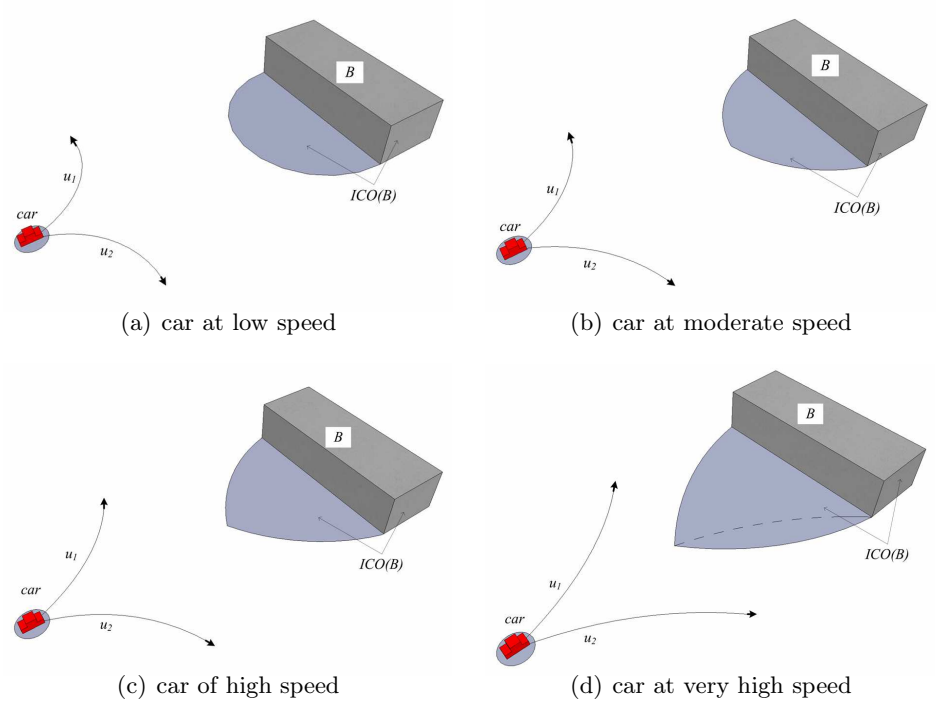


Figure IV.8: Numerical calculation Inevitable Collision Obstacles (ICO) for a static and dynamic obstacle.

ing angle). In order to characterize $ICO(\mathcal{B})$ of a given obstacle for this system, we first consider a point obstacle P and a smooth car moving at a fixed speed v_1 with a steering speed ξ_1, *ie* the control input $\phi_1 = (v_1, \xi_1)$. Fig. IV.7(a) shows the motion of the car and the corresponding reachable set for the point obstacle P, $\mathcal{R}^{-1}(P, \phi_1)$. Fig. IV.7(b) illustrates the case where the car evolves under two control inputs, namely $\phi_1 = (v_1, \xi_1)$ and $\phi_2 = (v_1, -\xi_1)$. The analytical representation of this path becomes much more involved than circles described by simple cars under a fixed control input. In general, the analytic computation of $ICO(\mathcal{B})$ becomes eventually so complex that one might need to resort to a numerical characterization of $ICO(\mathcal{B})$ instead.

The numerical calculation consists in calculating numerically the complete representation of $ICO(\mathcal{B})$. Fig. IV.8 shows the numerical calculation of $ICO(\mathcal{B})$ of 2 different obstacles (a static one and a moving one) for car-like robot modeled using the smooth car model. Such explicit numerical calculation is feasible. However, It is important to note that this representation is however impacted by several parameters. For the smooth car, for instance, the path described by the system depends upon its initial speed. In fig. IV.9, several illustration of $ICO(\mathcal{B})$ are depicted with different initial vehicle speed v. The



(a) car at low speed

(b) car at moderate speed

(c) car of high speed

(d) car at very high speed

Figure IV.9: Characterization of an Inevitable Collision Obstacle (ICO) at different vehicle speed.

shape of $ICO(\mathcal{B})$ clearly changes. Therefore, in this example, the $ICO(\mathcal{B})$ calculation will have to be done as the vehicle speed changes. Generally speaking, $ICO(\mathcal{B})$ calculation must be performed for each obstacles $\mathcal{B}$ and for every possible state of the system $\mathcal{A}$ which is rapidly a tedious task, extremely expensive in terms of computational ressources.

4.3 Implicit Numerical Representation

In the two previous sections, we have seen that there are cases for which it is possible to calculate explicitly $ICO(\mathcal{B})$. Once such a set is characterized, the planning scheme uses the calculated models of $ICO(\mathcal{B})$ instead of the conventional $\mathcal{B}$, to perform collision detection. The main problem of characterizing numerically all complete $ICO(\mathcal{B})$ for every $\mathcal{B}$ is a high computational cost. Our approach consists in implicitly characterizing $ICO(\mathcal{B})$ and avoid the complete explicit numerical calculation of $ICO(\mathcal{B})$ for every $\mathcal{B}$ in order to reduce the computation burden. Practically, this means that instead of attempting to define the area of forbidden states that the system will have to avoid, we check that a given state, *ie* a candidate state for the planned trajectory, does not belong to such a forbidden area. Such a state is individually tested to be an ICS or not. Indeed, even though a complete characterization of $ICO(\mathcal{B})$ for every $\mathcal{B}$ is not explicitly known by the

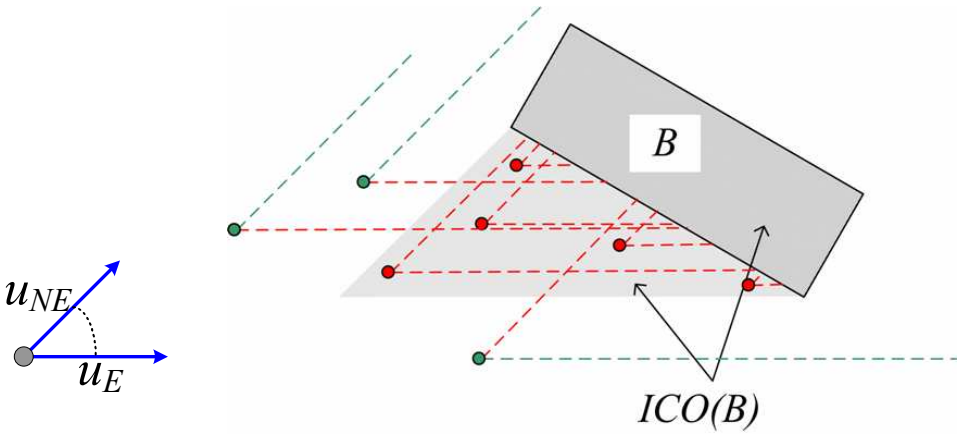
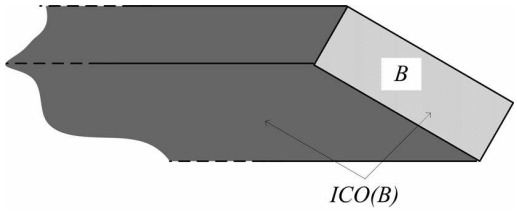


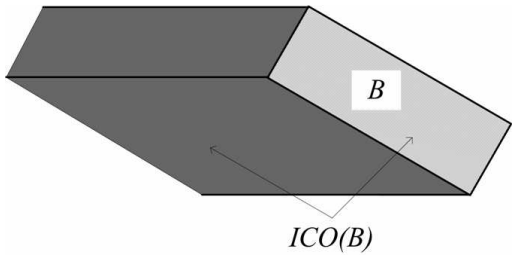
Figure IV.10: Implicit ICO representation consists in finding inevitable collision states instead of the complete ICO characterization.

motion planner, a safe trajectory can be computed provided each state of this trajectory is verified to be ICS-free with respect to the surrounding obstacles. We recall that a state s is not an ICS provided there exists a control input ϕ for which for all t, $\phi(s,t)$ is collision-free. The idea therefore lies in defining a set of different ¨evasive maneuvers¨, *ie* trajectories avoiding collision (*eg* braking trajectories). Then the technique consists in verifying that the system in a given state, moving under each control input is not colliding with its environment. This set of trajectory is the set $\mathcal{I}$ of feasible control inputs for the system. By the approximation property (chapter III, section 2.2, property 2) the safety guarantee remains conservative. Therefore instead of calculating the complete set of ICS for all obstacle $\mathcal{B}$ and a given system, we consider only one state at a time and check whether it is an ICS or not, *ie* whether this state belongs to $ICO(\mathcal{B})$ of $\mathcal{B}$ which is much more efficient than previous characterization (see fig. IV.10).

We make here an additional comment that concerns the practical implementation of such numerical calculation methods. This comment applies to numerical methods described earlier as well, however it seems more relevant here. The numerical computation of inevitable collision states, requires a discretization of the state time space $\mathcal{ST}$. As time is infinite, the discretization in the time dimension must be bounded for practical reasons. Indeed, the computation over an infinite time horizon is purely impossible. The concept of inevitable collision states provides a strong safety guarantee as a system in an ICS-free state is guaranteed to always have a possibility to avoid a collision in the future. In case the safety exploration is limited in time, the arbitrary approximation might impact the ICO characterization. For instance, for an East system, an ICO that is theoretically infinite (see fig. IV.11(a)) is reduced in case safety exploration is limited in time (see fig. IV.11(a)). Consequently, such a incomplete characterization might result in a safety



(a) Inevitable Collision Obstacle for the East system



(b) Time approximated of the inevitable Collision Obstalce

Figure IV.11: Inevitable Collision Obstacle (ICO) under arbitrary temporal approximation for the East system.

problem. However, we argue here that an infinite time exploration is not necessary. Indeed, for a system which moves under several distinct controls, we know by definition that $ICO(\mathcal{B}) = \bigcap_i ICO(\mathcal{B}, \phi_i)$. Therefore, $ICO(\mathcal{B})$ is the intersection of several $ICO(\mathcal{B})$ and is therefore a finite set for which complete representation does not require an exploration over a infinite time, as it is finite. The proper time upper bound might be however difficult to determine and this issue is not addressed in our work.

5 Partial Motion Planning (PMP) Algorithm

In this section we detail how the PMP algorithm, is adapted for our case study. The diffusion technique lies at the heart of the PMP algorithm. Indeed, as we explained in the previous chapter, $\mathcal{ST}$ exploration and sampling are done simultaneously during this phase. In the following, we focus on this particular aspect and provide details for our case study. In the first section, we present the local planning method used. The second section details how the ICS computation technique is used within the diffusion technique. The aspect of distance metric is discussed in the third section. The advantage of the use of a nonholonomic metric is discussed. Finally, the complete tree construction technique adapted for our case study is presented in the fourth section.

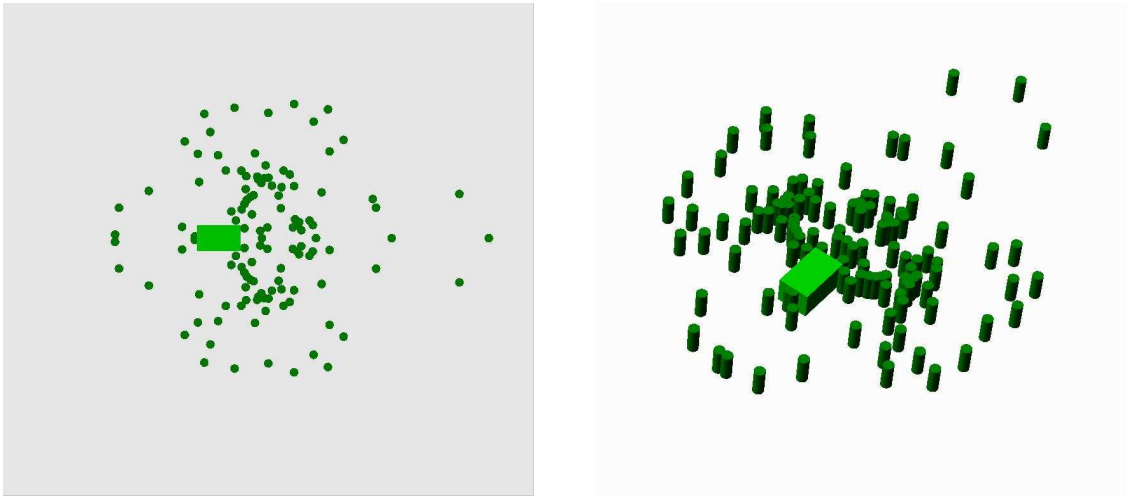


Figure IV.12: Representation of the tree of reachable states for the car-like robot (100 nodes).

5.1 Tree Expansion

A car-like robot exhibits differential constrains. The differential equation of the dynamic car model, presented earlier in this chapter, section 1, is a transition function, of the form $\dot{s} = f(s, u)$. The discrete time model introduced in the previous chapter, section **??** requires to partition the time interval T into intervals of length Δt and to chose a finite subset $\mathcal{U}_d$ of the control space $\mathcal{U}$. Given the dynamic car model used in this case study, we define $\mathcal{U}_d = (\gamma, \dot{\xi}) =$ with $\gamma \in [\gamma_{min}, 0, \gamma_{max}]$ and $\dot{\xi} \in \left[\dot{\xi}_{max}, 0, \dot{\xi}_{min}\right]$. Therefore, instead of directly sampling the state-time space $\mathcal{ST}$, sampled states are determined by integrating over the time interval Δt the transition function $\dot{s} = f(s, u)$ with a given control $u_d \in \mathcal{U}_d$ constant over the time interval Δt. This integration is done numerically using a numerical integration method, *eg* Runge-Kutta method.

During the tree construction, a control can be chosen randomly or deterministically. We note that if each control $u_d \in \mathcal{U}_d$ is successively applied over the time interval Δt, repeatedly several consecutive times, a reachability tree is built (see fig. IV.12). It is a discrete subset of the reachable set. For some nonholonomic systems, a careful discretization can be chosen so that the states become trapped on a grid or lattice [PPSB04].

5.2 Safe Trajectories Generation

Traditional sampling techniques rely on geometric collision detection so as to identify whether a state is collision-free or not. As we discussed, such an approach does not provide sufficient safety guarantees. Thus, the key idea in our PMP algorithm resides in

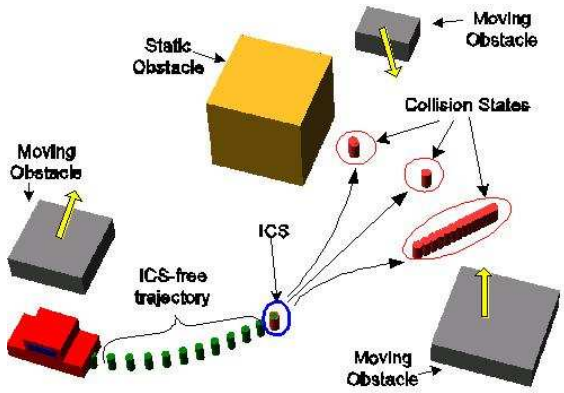


Figure IV.13: Inevitable Collision States within the PMP framework. Each state of the planned partial trajectory is verified to be an ICS with respect to the surrounding dynamic environment.

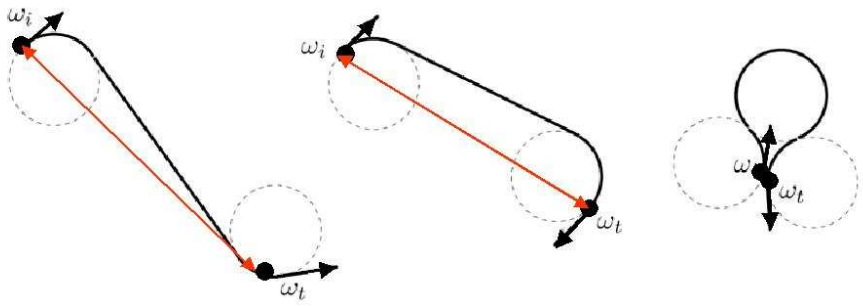
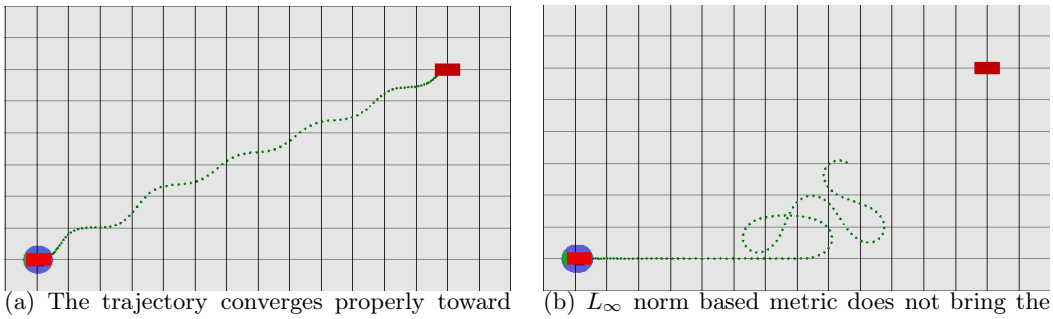


Figure IV.14: Notion of distance for non-holonomic systems. The double arrow represents the Euclidean distance whereas the nonholonomic distance is the length of the represented path composed of arc of circles and straight segments.

generating ICS-free trajectories instead of simply collision-free. For the dynamic car model that we use in our case-study, introduced earlier in 2.4, the finite set $\mathcal{I}$ must therefore be chosen with respect to our model of the robot $\mathcal{A}$. At first we use evasive maneuvers of the system defined by the steering speed control u_ψ so as explore a wide space. As for the longitudinal acceleration we use the deceleration control input. Thus $\mathcal{I}$ is defined by the following set of control inputs $\mathcal{I} = \left\{ \left(\alpha_{min}, \dot{\xi}_{max}\right), (\alpha_{min}, 0), \left(\alpha_{min}, \dot{\xi}_{min}\right) \right\}$. Fig. IV.13 illustrates how a given state, proposed by the local planner, in this case the final state of a planned trajectory, is using this approach to determine whether it is ICS-free. In this case, the state is in collision in the future with respect to all control inputs of $\mathcal{I}$. Therefore this state is an ICS and therefore not safe. As the local planner is iteratively called, a sequence of discrete safe states is built.



(a) The trajectory converges properly toward the goal thanks to the use of the Continuous Curvature metric.

(b) L_∞ norm based metric does not bring the generated trajectory towards the goal.

Figure IV.15: Influence of metric on trajectory generation.

5.3 Metric

As the diffusion technique of PMP is incremental, it requires a metric to conduct the tree toward the goal and ensure the overall scheme convergence. A metric is a cost function that quantifies a distance between two states. One difficulty when performing motion planning using an incremental approach is the choice of such a metric used to select and expand the nodes in order to build the tree. This parameter has a large influence on the trajectory quality specially when dealing with non-holonomic systems. A Euclidean metric could be used to calculate the distance between two states of systems like a car-like robot, however due do its non-holonomy, the Euclidean distance does not reflects at all the distance that should move the system to go from a given state to another one. The most significant example is described by the third picture of fig. IV.14, where the two states are at the same position but with an opposite orientation. When the Euclidean distance is null we however easily understand that in fact, a car will have to move significantly to go from one state to the other one. The first non-holonomic metric that have been developed and addresses this issue is the Dubbin's metric [Dub57]. In this case study, due to the nonholonomic nature of the system, a continuous curvature (CC) metric derived from the work presented in [SF96], which is an extension of the Dubins metric. This metric has been developed so as to consider the fact that car-like robot can not change instantaneously the wheels orientation. While moving and steering, the car describes a clothoid shaped path. Basically, instead of measuring a path made of arc of circles and straight path, as for the Dubins metric, the CC metric connects, at the curvature discontinuity point, the straight path and arc of circle with a clothoid.

Some authors have proposed other metrics based on the L_∞ norm [LFE04]. This type of metric have poor goal orientation yielding trajectories of poor quality for complex systems like cars, which is not suitable for real application. On one hand, fig. IV.15(a) shows a

trajectory generated for the dynamic car model during a large PMP cycle. The tree expansion converges to the goal by optimizing a nonholonomic metric. As this metric is natural for the system, the trajectory properly reaches its goal. As a remark, the oscillations of the trajectory result from the discretization choice of the control space. Fig. IV.15(b) on the other hand shows a trajectory generated under the same conditions as in previous case, with different metric based on L_∞ norm. We see that the tree expansion is greatly impacted by this choice and that the notion of distance is not natural for the system. This figure clearly illustrates how the choice of the metric greatly influences the performances of the tee expansion.

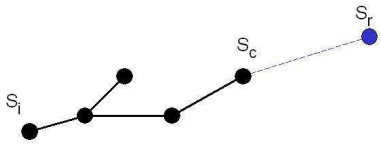
5.4 Tree Construction

We can now detail the complete diffusion technique developed for this case study. The tree builds incrementally within $\mathcal{ST}$ as follows (see fig. IV.16) :

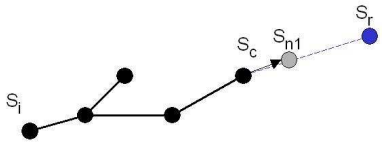
1. The tree is initialized by the initial state s_I
2. A milestone s_r is generated. This milestones can either be generated randomly, be set as the goal state or generated randomly with a probability p to be the goal.
3. The state in the tree s_c, closest to s_r, is selected.
4. A control from $\mathcal{U}_d$ is applied to the system during a fixed time period, the integration step. In case the new state s_n is safe (*ie* ICS-free), this control is considered as valid. The operation is repeated over all control inputs.
5. Finally, the distance between all safe states that have been kept and s_r is calculated and the one that is the closest to s_r is finally selected and added to the tree.

The process repeats until the allotted time for such a calculation has elapsed. Such a trajectory construction allows therefore to be interrupted anytime while insuring the safety and quality of the generated trajectories [PF05b, LPV$^+$06].

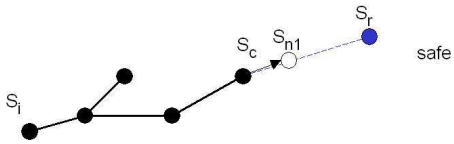
In fig. IV.17(a), a partial trajectory generated using random milestones s_r generation is shown and fig. IV.17(b) shows a different trajectories where the milestone s_r corresponds to the goal for each iteration. This variant is more efficient and realistic for highly dynamic environment in which local minima are unlikely. As examples, this approach is particularly well suited for general urban like areas as illustrated in fig. IV.18, as well as for areas highly cluttered with concave obstacles as illustrated in fig. IV.19. These figures depict the car-like robot (rectangle) evolving within an environment cluttered with static (polygons) and



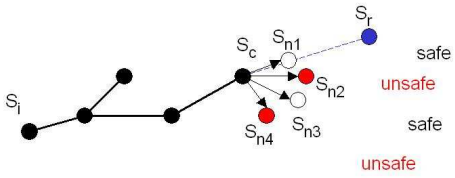
(a) Incrementation from initial tree



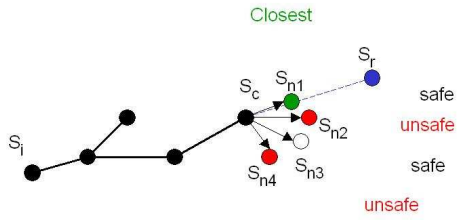
(b) New node contruction



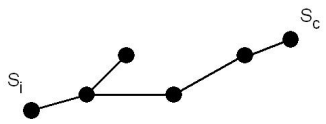
(c) Safety validation of constructed new node



(d) Repeat over all control input set

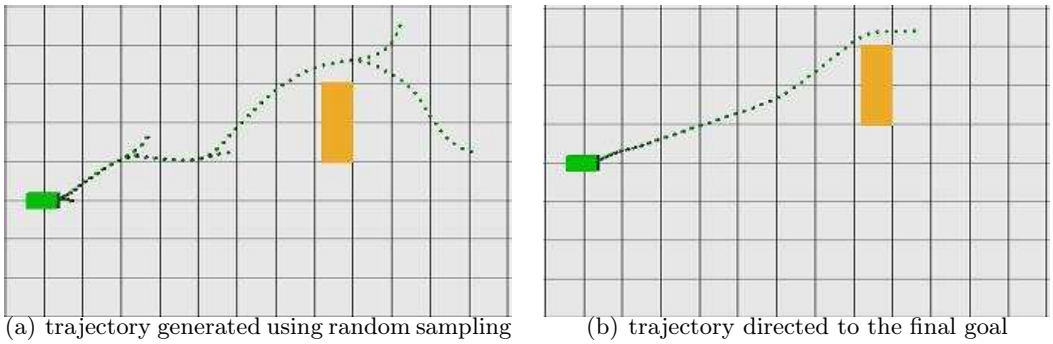


(e) Choose the best safe node



(f) New incremented tree

Figure IV.16: Tree construction.



(a) trajectory generated using random sampling (b) trajectory directed to the final goal

Figure IV.17: Examples of trajectories generated by the PMP.

dynamic obstacles (cylinders). At each cycle, PMP generates a new trajectory (depicted at front of the system) while executing the previously planned, whose trace is represented by the thick trajectory behind the system. Both fig. IV.18 and fig. IV.19 represent a sequence of different consecutives PMP cycles, where each cycle lasts $1s$. For these plans, the physical constraints are a maximum velocity of $2.0m/s$, maximum acceleration of $0.25m/s^2$, a maximum turning angle of $\pi/3rad$ and a maximum steering speed of $\pi/6rad/s$. As for the safety check and the related calculation of ICS, we choose $\mathcal{I} = \left\{\left(\alpha_{min}, \dot{\xi}_{max}\right), (\alpha_{min}, 0), \left(\alpha_{min}, \dot{\xi}_{min}\right)\right\}$.

These examples illustrate how PMP is able to generate trajectories of high quality at each cycle (the trace left behind the system represents the executed trajectory) with a long lookahead (trace in front of the system). Besides, one can note that safety is insured without being too conservative which keeps sufficient free space for the planner to be explored. Therefore, PMP is capable to generate efficiently high quality trajectories that allow the system to reach its goal while accounting for both dynamics of the system and the environment.

In some practical situations, the system might get stuck within a local minimum formed by a concave shape obstacle. Indeed, in case the trajectory lookahead is too limited and the system has to brake to remain safe, we modified the step 4 so as to add a penalty (weight for the cost function) to this new calculated state. Thus, other states of the tree are favored to further expand the tree during the step 3 of the next cycle. This technique allows the scheme to converge to the goal while avoiding getting trapped in local minima. Fig. IV.20 illustrates how PMP efficiently generates trajectories for the dynamic car that move toward the goal while avoiding a static U-shape obstacle.

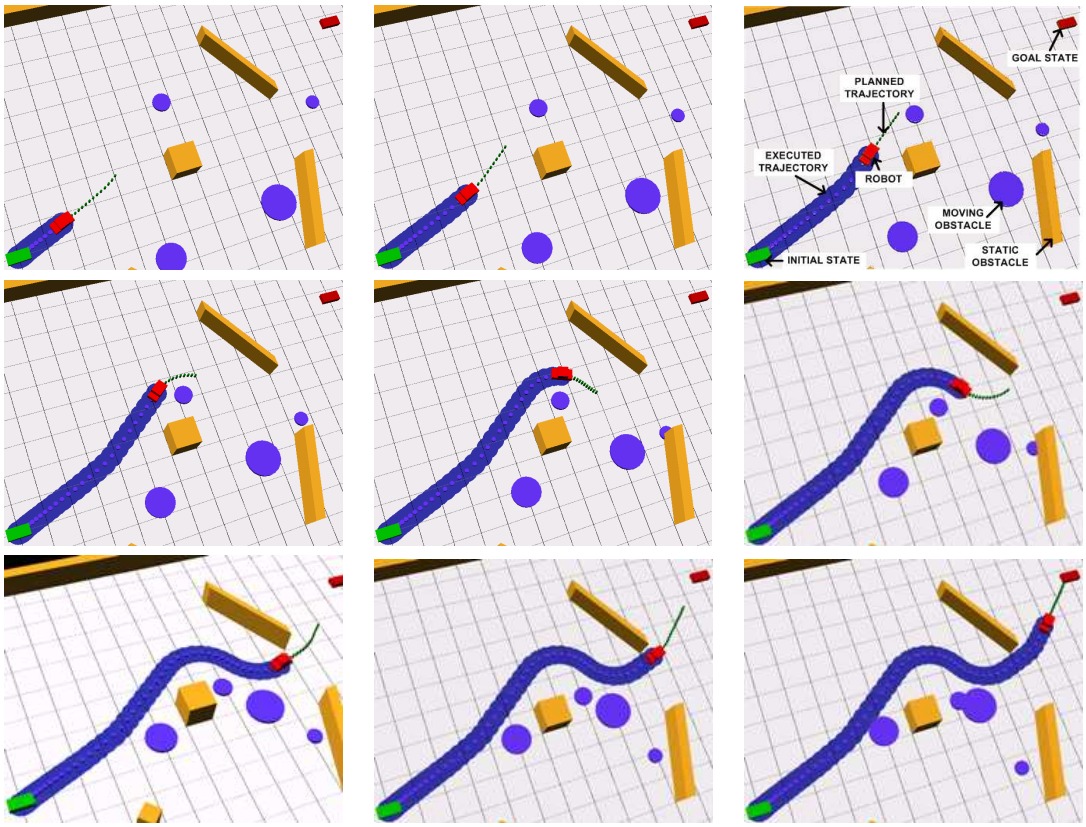


Figure IV.18: Sequence of PMP cycles in a general dynamic urban environment.

5.5 Performances

At first, we focus on the lookahead of the generated trajectories and its relation with two factors, the number of surrounding obstacles and the cycle duration. Fig. IV.21 shows the diminution of the lookahead of the planned trajectories during a PMP cycle as the number of obstacles increases. We observe at first that the impact of rectangular moving obstacles is much higher than the others, due to the fact that collision check with moving obstacles require more calculation. Indeed, these obstacles have to be placed in the correct time slice before to proceed to collision detection. The impact is not as important with moving circular obstacles. Indeed, a circular bounding box of the car is used to perform the collision checks with circular moving obstacles and such a calculation is extremely efficient. Generally speaking, we see that despite the presence of surrounding obstacles, in a reasonably cluttered environment (50 obstacles), the lookahead remains largely greater than the cycle duration, up to more than 35 times greater than the cycle duration (Pentium 4 1.6GHz). As for the cycle duration, fig. IV.22 illustrates the trend of the increase in lookahead duration with respect to the duration of a PMP cycle. We observe a global linear behavior within a given environment, which allows a consistant and predictable

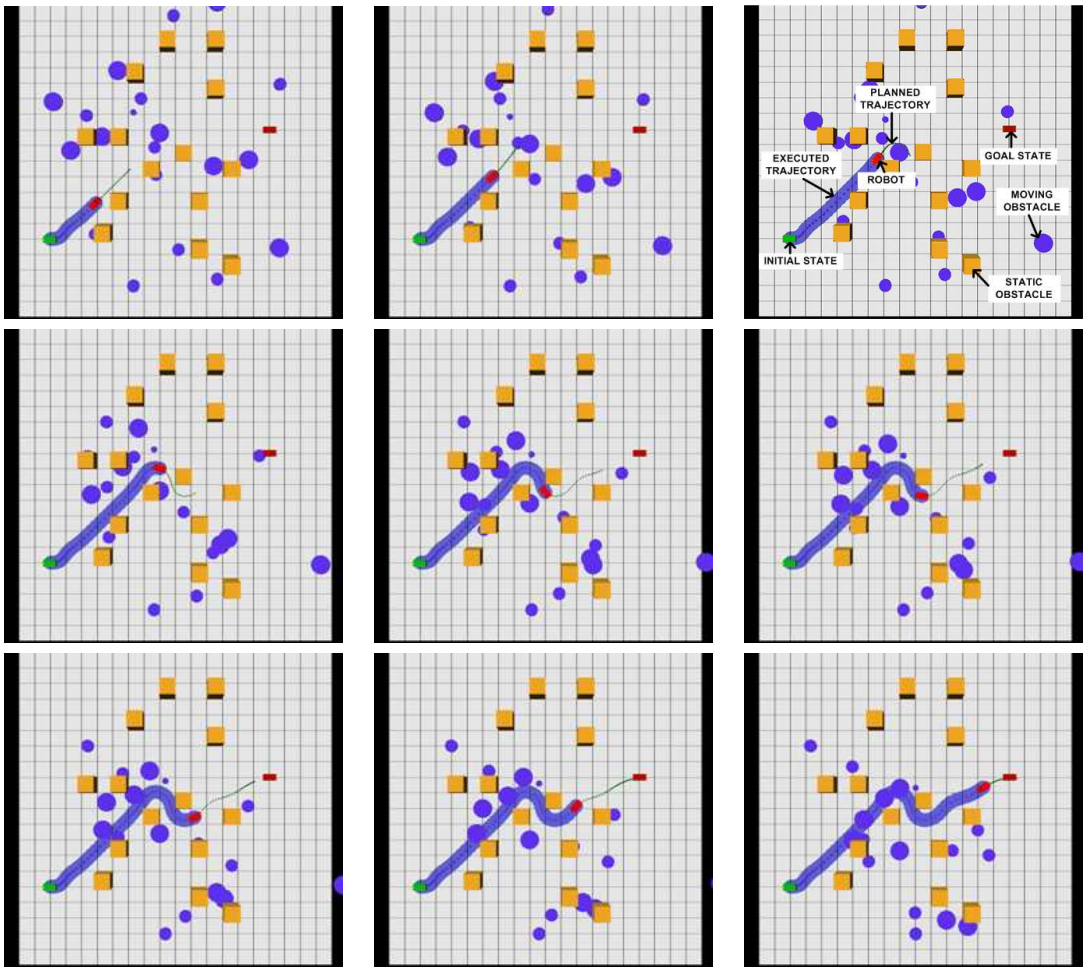


Figure IV.19: Sequence of PMP cycles in an environment highly cluttered with static and dynamic obstacles.

behavior.

The second point of interest is the impact of the mechanical constraints on the shapes and quality of the trajectories. We focus on the influence of the maximum steering speed constraint. In fig. IV.23 and fig. IV.24 we see trajectories calculated within one PMP cycle of $2s$ for different maximum steering angle and steering speed values. The remaining physical characteristics remain same as in the previous examples. At first, we observe that the higher the maximum steering angle value is, the better the robot is maneuverable. Indeed, we observe that all trajectories of fig. IV.24 seem to be more flexible than the ones in fig. IV.23. At second, we can see that the steering speed mainly affects the turning capability of the robot. In fig. IV.23(a), the trajectory keeps turning, whereas with a higher steering speed value, in IV.23(b), the robot is capable to straighten up its direction to the goal. We can observe a similar behavior between fig. IV.24(a) and IV.24(b).

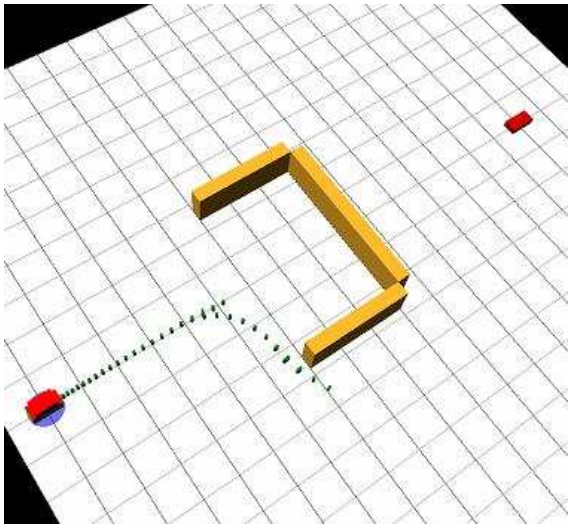


Figure IV.20: PMP generates large lookahead trajectories that can avoid U-shaped obstacles.

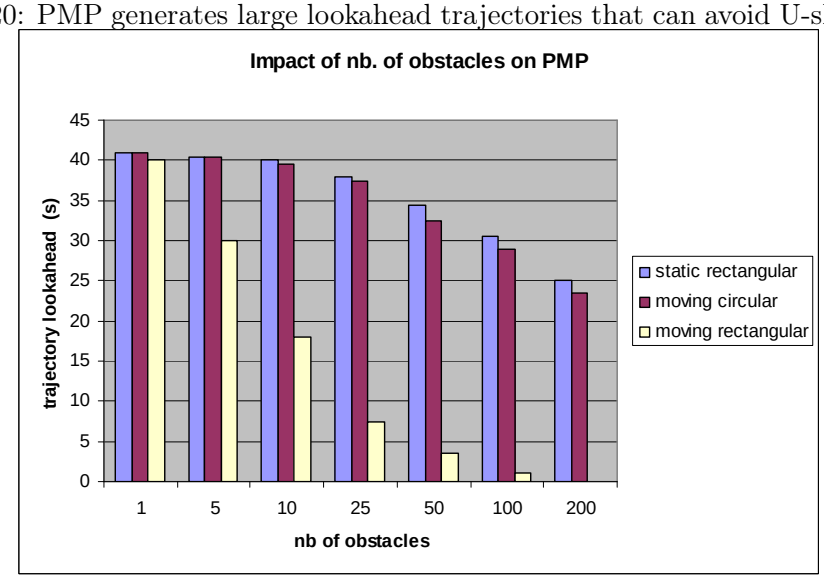
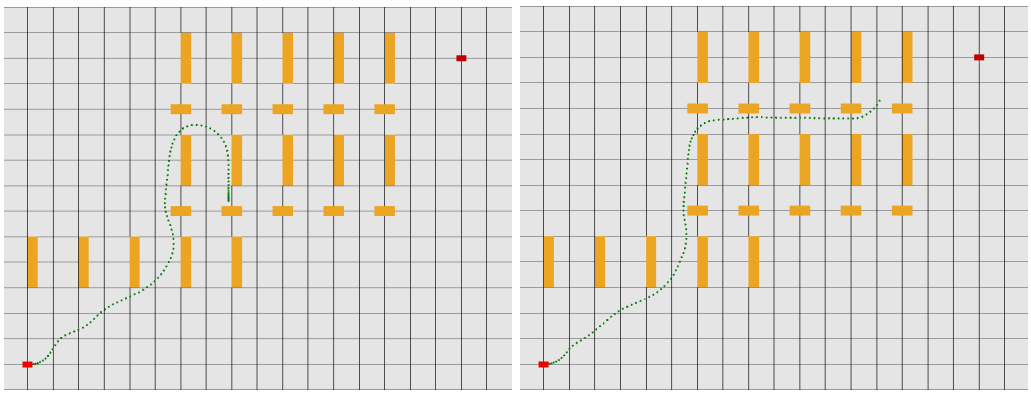


Figure IV.21: Trajectory lookahead with respect to the number of surrounding obstacles (data for 1 PMP cycle of 1 second).



(a) trajectory with max steering speed of 0.3 rad/s

(b) trajectory with max steering speed of 0.4 rad/s

Figure IV.23: Impact of steering speed constraint on trajectories generated during one PMP cycle of 2s, with a maximum steering angle of 0.5 radians.

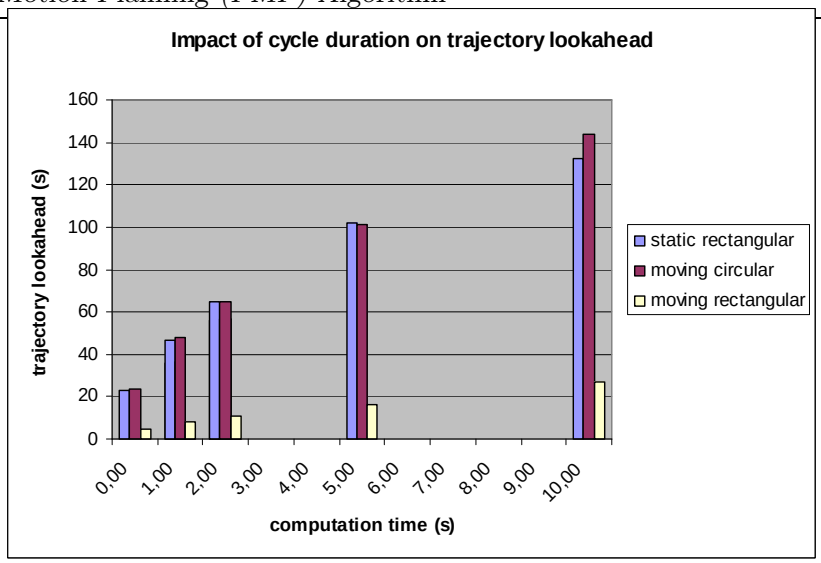
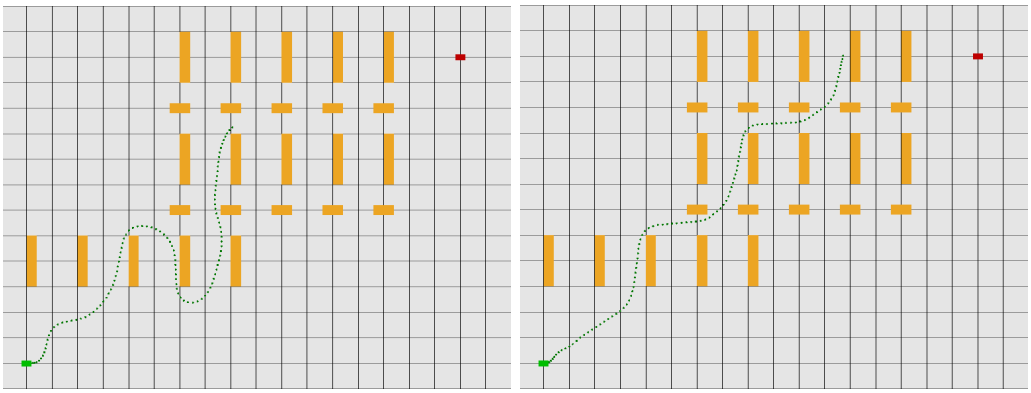


Figure IV.22: Trajectory lookahead with respect to a cycle duration (data generated for an environment cluttered with 25 obstacles in each cases).



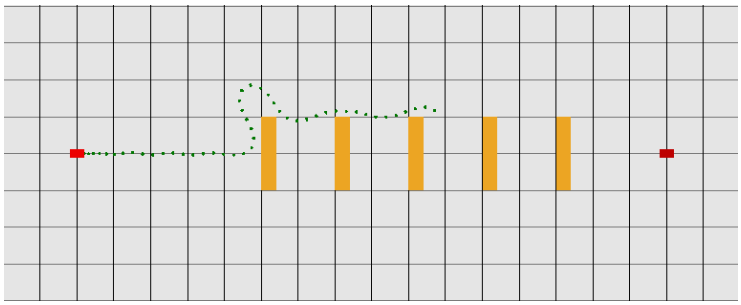
(a) trajectory with max steering speed of 0.3 rad/s

(b) trajectory with max steering speed of 0.6 rad/s

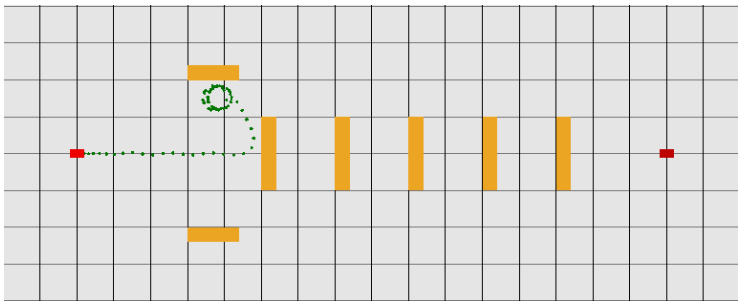
Figure IV.24: Impact of steering speed constraint on trajectories generated during one PMP cycle of 2s, with a maximum steering angle of 1.0 radians.

Finally, we focus on the choice of the trajectories of the $\mathcal{I}$ and their impact on the quality of the trajectories generated by PMP. In fig. IV.25(a), a the safety check is performed with one single trajectory. With a single control input, the approximation of the ICOs of the surrounding obstacles is very conservative. As a result, when an obstacle is added, the robot cannot find its safe way through (see fig. IV.25(b)). A similar comment can be made using the opposite control (see IV.25(d)). However, as soon as the number of trajectories of $\mathcal{I}$ increases, the approximation of ICOs becomes much finer, which allows the system to find a safe trajectory through the obstacles (see fig. IV.25(c)). This clearly illustrates the impact of too conservative safety approximations and the meaning of the

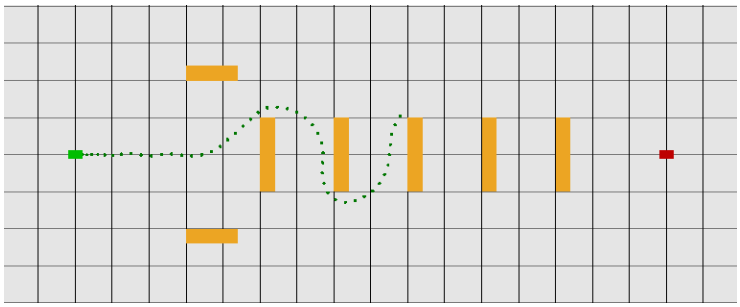
intersection property.



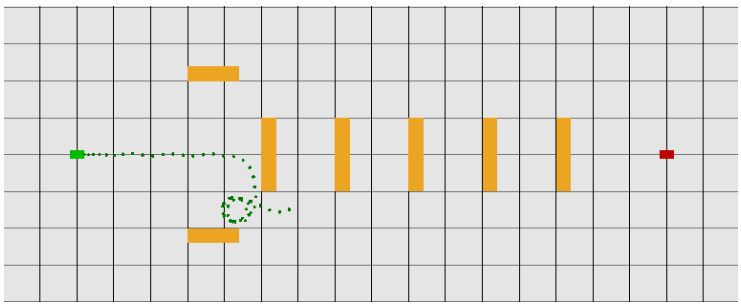
(a) $\mathcal{I} = (\alpha_{min}, \dot{\xi}_{max})$



(b) $\mathcal{I} = (\alpha_{min}, \dot{\xi}_{max})$



(c) $\mathcal{I} = (\alpha_{min}, \dot{\xi}_{max}), (\alpha_{min}, 0), (\alpha_{min}, \dot{\xi}_{min})$



(d) $\mathcal{I} = (\alpha_{min}, \dot{\xi}_{min})$

Figure IV.25: Impact of the choice of $\mathcal{I}$ on the generated trajectories.

(Etude de cas: le robot type voiture)

Dans ce chapitre, nous présentons une instanciation pratique de l'algorithme PMP dans le cas particulier d'un robot type voiture. Dans un premier temps, les différents types de modèles de véhicules sont discutés, le modèle simple, le modèle à courbure continu et le modèle dynamique. Nous motivons le choix de ce dernier par le fait qu'il prend en compte à la fois les principales contraintes cinématiques (les contraintes non holonomes) et les contraintes dynamiques (la borne sur la vitesse et accélération). La commande en accélération longitudinale a pour actionneur typique la pédale d'accélérateur. La deuxième commande de ce système est la vitesse angulaire de l'angle de direction. Quant au modèle de l'environnement, le concept d'enveloppe est utilisé afin de faciliter les calculs de détection de collisions. Ces enveloppes sont de type boites rectangulaires ou cylindriques, suivant la représentation. Enfin, le problème de la sureté des trajectoires, est abordé d'un point de vue des états de collisions inévitables (ICS).

L'implantation s'appuie sur des techniques récentes de planification pour systèmes non-holonomes basées sur la diffusion d'arbres aléatoires pour les adapter au schéma PMP et produire des trajectoires à courbure continue et à dérivée bornée, avec également prise en compte de bornes sur les vitesses et accélérations du véhicule. La caractérisation des ICS se fait par le biais d'approches numériques. En effet, il existe des représentations exactes mais pour des systèmes simples uniquement. Pour le système étudié, ce n'est pas le cas. L'inconvénient majeur de ce type de représentation est la difficulté des calculs à mettre en oeuvre. La spécificité de notre approche, consiste à éviter de calculer de manière explicite les ICS pour chaque obstacle, mais plutôt de passer par le calcul implicite qui consiste à vérifier chacun des états de la trajectoire comme étant un ICS ou non. Cette approche est non seulement beaucoup plus légère, en termes de calcul, à mettre en oeuvre, mais aussi s'intègre très naturellement dans la technique de diffusion par construction d'arbre. Le fait d'utiliser un modèle implique de ne plus échantillonner l'espace des états temps directement, mais l'espace des commandes. En effet, ce modèle est pris en compte à travers une fonction de transition qui est intégrée numériquement pour chacune des commandes échantillonnées afin de déterminer le nouvel état candidat à faire croître l'arbre. Enfin, la notion de métrique est également adaptée et une métrique non holonome est utilisée afin de garantir la meilleure convergence possible pour notre système.

L'algorithme de construction d'arbre ainsi présenté permet de calculer rapidement des trajectoires sûres et de qualité, qui prennent en compte les contraintes cinématiques et dynamiques du modèle, et qui convergent vers l'état final grâce à l'utilisation d'une métrique non holonome adaptée à ce véhicule. Les performances du planificateur sont discutées et plus particulièrement l'influence du nombre des obstacles environnants, des trajectoires

choisies pour la vérification de la sureté ainsi que le temps de calcul d'un cycle, sur la durée et qualité des trajectoires générées.

Chapter V

Experimentations

1 Introduction

In this work we have introduced Partial Motion Planning (PMP) as a new approach toward autonomous navigation within dynamic environments. In this chapter we present how this technique is implemented and integrated within a real robotic platform. The objective is to verify the overall behaviour and performance of PMP in real conditions, integrated on a real robot. The demonstrator we use is the Cycab, a car-like robot developed by INRIA. At first, we describe this Cycab platform and detail its control architecture. At second, we explain how PMP fits in the Cycab architecture. Then, we present the results of our experiments, separated into three main show cases that we use as proof of concept. Finally, we discuss some of the issues raised by this experiments.

Ce chapitre presente les resultats d'experimentations qui procedent de l'integration de l'algorithme PMP au sein d'une architecture et plateforme robotique reelle, le Cycab. Le Cycab est un petit vehicule electrique presente comme une alternative au vehicule prive pour les villes ou l'utilisation du vehicule particulier ne sera plus permise.

2 The Cycab Platform

The platform chosen for the experiments is a car-like robot, the Cycab (Fig. V.1). Designed in 1997 by INRIA during the LaRa project[1], the Cycab is a Cybercar (Fig. V.2), *ie* an electric vehicle specifically presented as an alternative to private vehicles in car-free cities. In its original design, the Cycab can move two people with luggage. It

[1] LaRa is a French National Research project on Automated Road conducted from 1998 to 2001

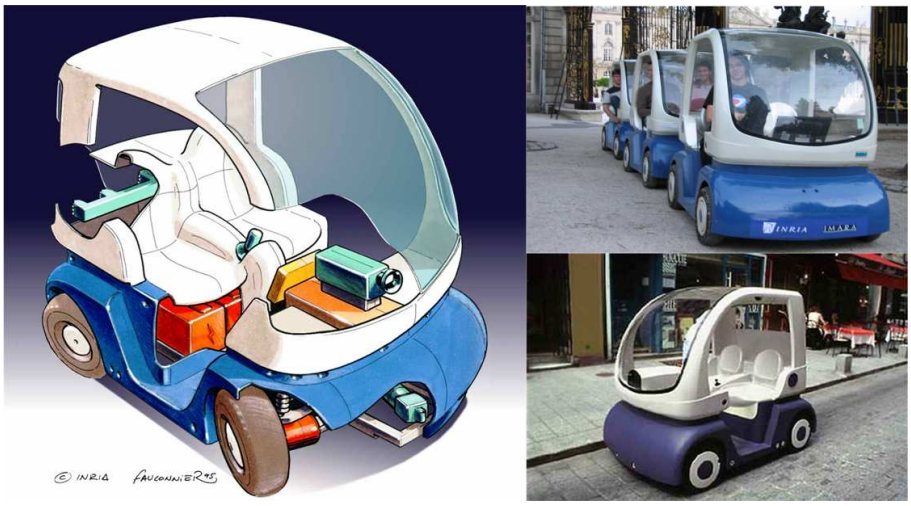


Figure V.1: The Cycab.

Figure V.2: Several types of Cybercars.

should be available on a self-service basis to the largest population possible, including youngsters, elderly people and handicapped. This concept is aimed to bring two advantages : a net reduction in the number of cars (and therefore parking spaces) in a given area, and a reduction in air, street and noise pollution. The system should offer a good service for short local trips and every convenient access to mass transportation.

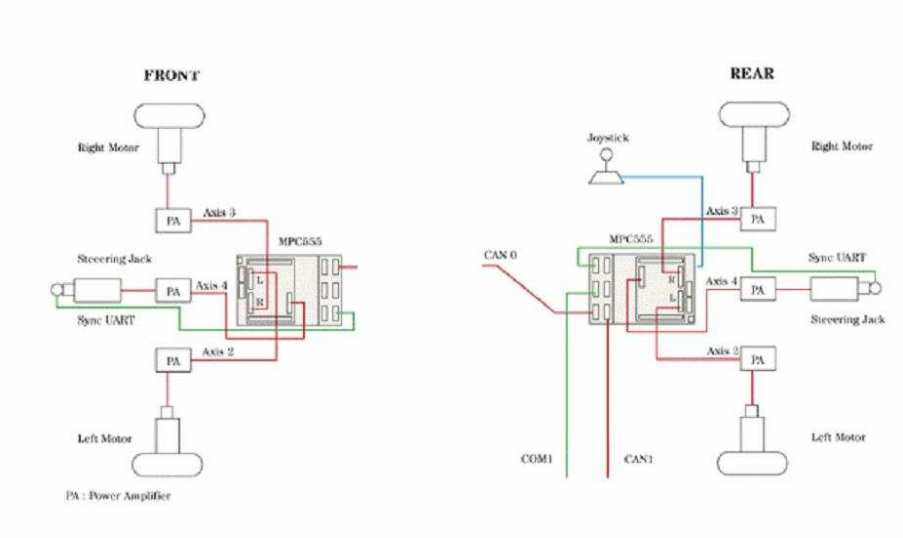


Figure V.3: Cycab hardware architecture.

2.1 Hardware Architecture

The Cycab is an innovative vehicle, entirely under computer control. It has a complete X-by-wire architecture, *ie* conventional mechanical and hydraulic links from commands (steering, accelerating and braking) to actuators have been replaced by electrical signal. It can be driven manually with a joystick or automatically under various modes. It weights around 430kg and can reach 30km/h. It includes four electric motors of 1kW, two for front and rear acceleration and two for the front and rear steering. The computing units are based on micro-controllers, embedded within the platform and used to control in real time and "by-wire" the different actuators. The Cycab contains two inboard units based on an MPC555 card to control respectively the front and rear actuators. All the units are linked via a Controller Area Network (CAN) bus. The CAN is a serial bus developed by the automotive supplier Bosch and is one of the most popular and used within current commercialized cars (Fig. V.3).

2.2 Software Architecture

2.2.1 High Level Architecture

There exists several system's architecture aimed at achieving motion autonomy. The benefits to include a planning strategy within the navigation scheme of an autonomous systems has been largely motivated throughout this dissertation. Such a strategy naturally fits within the traditional deliberative architecture, paradigm coming from Artificial

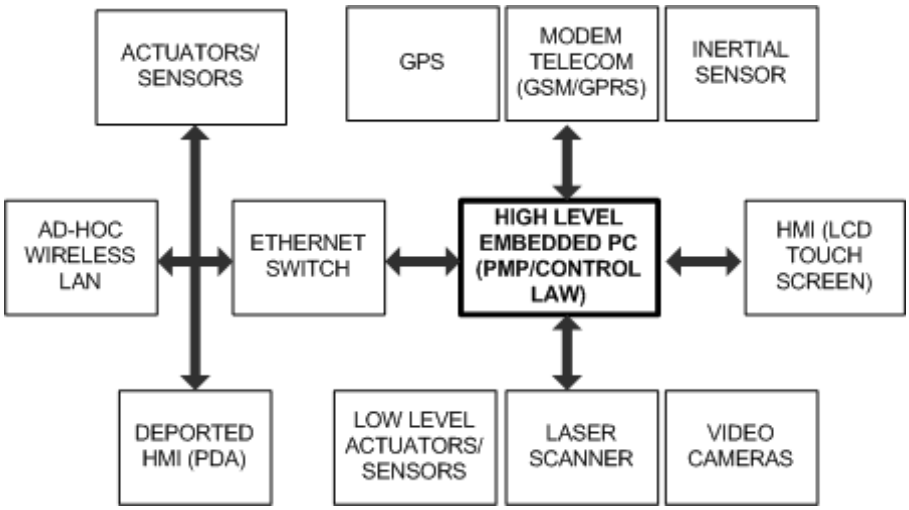


Figure V.4: Cycab high level architecture.

Intelligence (AI). It tries to implement a simplified view of Human reasoning and is often referred as the "Sense-Model-Plan-Act" scheme. In practice, this concept is implemented into robotic control systems using a hierarchical architecture made of three main components: perception (which includes sensing and modeling functions), decision, and action.

- Perception is the first stage of the control process. The main purpose of this function is to properly model the world by extracting high level relevant information from raw sensory data and priori knowledge. The difficulty of this task has become through years a complete active research domain. The Cycab is equipped with several sensors. Our experiments are performed with a 3D optical sensor, a laser scanner from IBEO Gmbh. This sensor provides extremely reliable and precise information about its environment.
- A decision (deliberative) phase consists in "reasoning" about the task model and the environment model, in order to decide what is the more appropriate sequence of actions to execute. For navigation purposes, this reasoning mainly consists in planning the trajectory to be executed. This stage is addressed by PMP.
- The last processing phase of a deliberative architecture is the controller, aimed at executing the plan via the robot's actuators.

One of the most famous and first robot using an architecture based on this approach is *Shakey* ([Nil84]) that could find an object within a room by mean of a video camera and push it to a given point. This could take however a lot of time as each motion would require more than a hour of external computing, with a strong probability of failure at

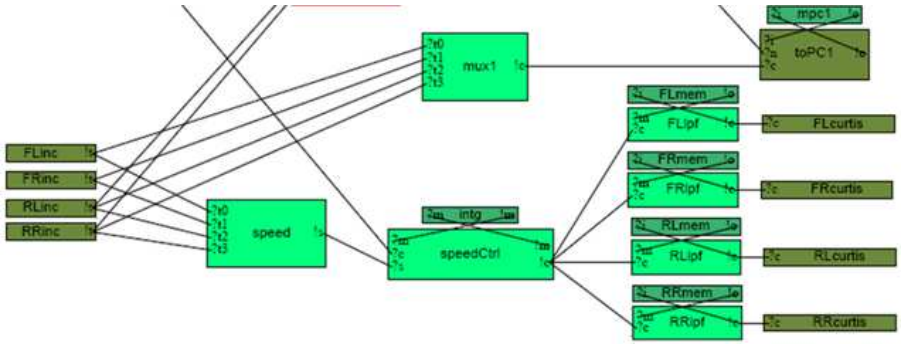


Figure V.5: Illustration of a partial Cycab low-lever architecture based on Syndex.

execution time. In fact, the main drawback of the ¨Sense-Model-Plan-Act¨ architecture lies in the almost incapacity to cope with unpredicted changes within the environment. As a consequence, historically, dynamic obstacles were almost impossible to handle at execution. PMP algorithm however is specifically designed to address this issue. For this reason, we implement this control architecture in the Cycab for our experiments. All three high level modules, Perception, PMP and feedback controller are implemented on a high level embedded PC, a Pentium 4 1.6GHz, using a software architecture developed by INRIA, a multi-threaded C++ framework for developing robotic application using different sensors and actuators. Fig. V.4 depicts the system architecture of the Cycab.

2.2.2 Low-Level Architecture

X-by-wire architecture has brought in recent years a tremendous attention from researchers and industrial. Indeed, the design of a distributed network of real-time embedded systems is a great challenge in the automotive. In the Cycab, the AAA approach (Adequation Algorithm Architecture) is implemented. It allows for instance to simplify the design process by decoupling the software design with the operating system design dealing with the synchronization of the units and the different processes repartition. A 486 PC is used to supervise the overall architecture that runs under the real time software Syndex developed by INRIA, working on the AAA principle (see fig. V.5).

3 Control Layer

Once motion planning has been performed and thus its a priori trajectory calculated, the wheeled mobile robot has to realize it. The motion is therefore executed by applying proper controls on the robot's different actuators. One might directly apply the pre-calculated

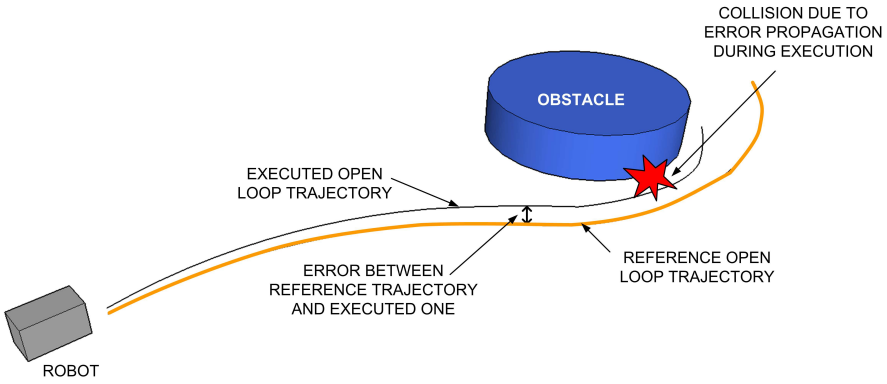


Figure V.6: Execution of a reference trajectory in open loop.

open loop controls of the planned trajectory provided by PMP to the actuators, but it is known in automatism that this will not yield to satisfying results. Open loop control schemes do not guarantee sufficiently robustness properties needed for most applications and the errors that will accumulate during execution might ultimately yield to a wrong destination or even to a collision, even though collision free trajectory was planned, as illustrated in figure V.6).

The best way to handle the execution is to base the control on error regulation, to close the control loop by feeding back the observed state of the robot. The feedback schemes presuppose a capability of measuring variables used in the control loop in order to feed it back, in an accurate and stable way. Such closed loop or feedback control schemes exhibit an intrinsic degree of robustness, the study of which might be unnecessary as long as linear control theory is used, unavoidable otherwise. In a word, a closed loop control scheme is aimed to convert ideal plans into real motion execution.

3.1 General Control Problems

In this section we explain the control law that has been chosen to execute the trajectories generated by the PMP. We clarify first the main strategies in order to motivate our choice.

3.1.1 Point-To-Point Motion

In point-to-point motion, the robot must reach a desired goal configuration starting from a given initial configuration. Using a more control-oriented terminology, the point-to-point motion task is a stabilization problem at a point of the robot state space. The search for a feedback solution to the point stabilization problem is complicated by the general theoretical obstruction of the necessary condition for stabilizability via smooth time-invariant feedback, known as the Brockett's theorem [Bro83]. This means that the class of stabilizing controllers should be suitably enlarged so as to include non smooth and/or time-varying feedback control laws. Thus, researchers have offered both non-smooth feedback laws [CdWS91] and time-varying feedback laws [Sam93] for stabilizing simple mobile robots to a point. A general approach remains however an open problem.

3.1.2 Path Following

The path following motion consists in following a geometric path in the Cartesian space starting from a given initial configuration (on or off the path). To perform this task, the controller is given a geometric description of the assigned Cartesian path. This information is usually available in a parametrized form expressing the desired motion in terms of a path parameter, which may be in particular the arc length along the path. For path following, time dependence is not relevant because one is concerned only with the geometric displacement between the robot and the path. In this context, the time evolution of the path parameter is usually free and, accordingly, the control inputs can be arbitrarily scaled with respect to time without changing the resulting robot path. It is then customary to set the robot forward velocity (one of the two inputs) to an arbitrary constant or time-varying value, leaving the second input available for control.

3.1.3 Trajectory Tracking

This motion is realized by following a trajectory starting from a given initial state (on or off the trajectory). In the trajectory tracking task, the robot must follow the desired Cartesian path with a specific timing law. Equivalently, it must track a moving reference robot. Although the trajectory can be split into a parametrized geometric path and a timing law for the parameter, such separation is not strictly necessary. Often, it is simpler to specify the workspace trajectory as the desired evolution for the position of some representative point of the robot. The trajectory tracking problem consists then in the stabilization to zero of the two-dimensional Cartesian error e using both control inputs.

3.2 The Cycab Control Law

The amount of information that should be provided by a high-level motion planner varies for each control task. In our work, we are addressing the trajectory tracking problem. This problem requires the planner to provide a path which is kinematically feasible, namely, which complies with the nonholonomic constraints of the specific vehicle, along with a timing law so as to allow its perfect execution in nominal conditions. Given this desired (or reference) trajectory, the control law consists in stabilizing the system to this trajectory, and allow the system to execute robustly the planned trajectory. Practically, this control scheme consists in stabilizing to zero the error between the reference state and the actual robot state. To this mean, one need to equip the robot with suitable sensors so as to correctly observe the necessary system's states. The observation is done by means of proprioceptive or exteroceptive sensors. For our experiments with the Cycab, we consider the two controls (u_1, u_2) that respectively control to the forward velocity and the steering wheels' angle of the Cycab. These two controls can adjust four configuration variables, namely the two Cartesian coordinates (x, y) characterizing the position of a reference point on the vehicle, its orientation θ, and the steering wheels' angle ξ. We thus define the system state at a given time by the following parameters: (x, y, θ, ξ, v) with the two control inputs (u_1, u_2). The state parameters of the car-like robot come along with the reference parameters of the planned trajectory: $(x_{ref}, y_{ref}, \theta_{ref}, \xi_{ref}, v_{ref})$. In the same way, we also have the reference control inputs: $u_1^{ref} = v_{ref}$ and $u_2^{ref} = \xi_{ref}$. The feedback trajectory tracking control law is a linear control law based on the work of [Mor04]. It makes the hypothesis that the Cycab rolls without slipping on plane roads. In case the high level planner provides feasible trajectories, the control law is made simple by defining the error matrix $\tilde{g}$ as follows:

$$\tilde{g} = \begin{pmatrix} \tilde{g_1} \\ \tilde{g_2} \\ \tilde{\theta} \end{pmatrix} = \begin{pmatrix} \Re(-\theta_{ref}) \begin{pmatrix} x - x_{ref} \\ y - y_{ref} \end{pmatrix} \\ \theta - \theta_{ref} \end{pmatrix} \tag{V.1}$$

In this expression, $\Re$ is the rotation matrix of the angle ξ. The relation between and the different state and reference coordinates is this one:

Now, we can give the law, able to calculate for a given state the controls to apply to tend to reach a reference state:

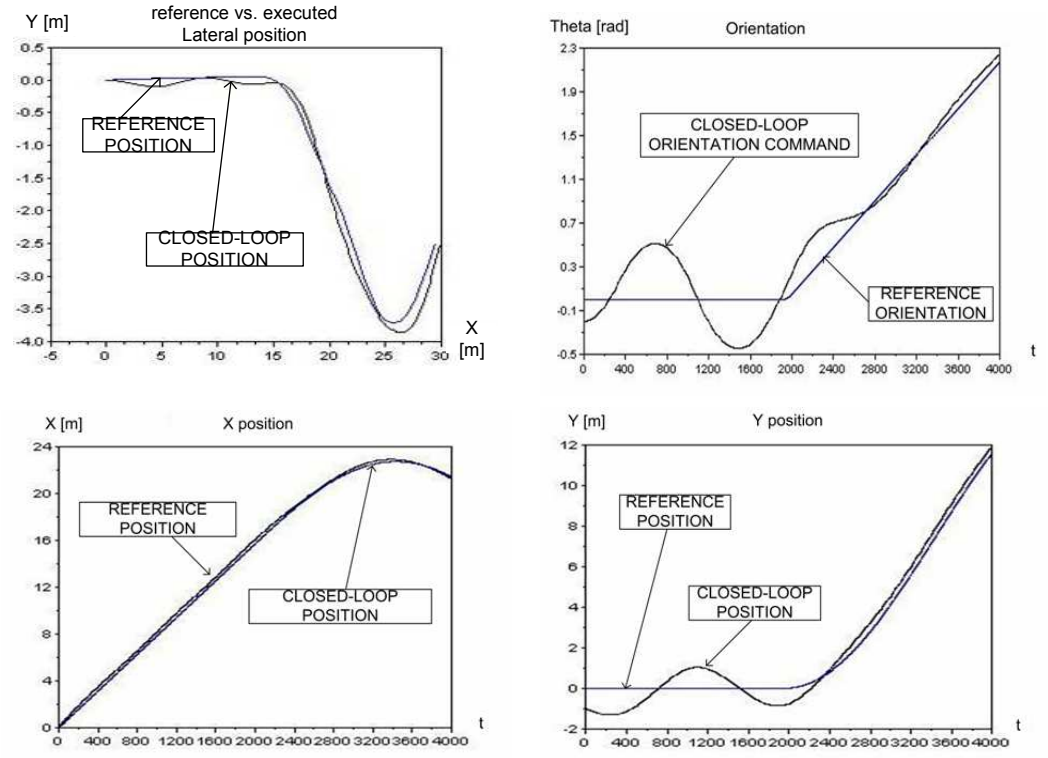


Figure V.7: Simulation of the tracking law (V.2) on Scilab software.

$$\begin{cases} u_1 = u_1^{ref} - k_1 \left| u_!^{ref} \right| g_1 - \frac{k_1}{2k_2} \frac{\tan \tilde{\phi}_{ref}}{L} \left| u_1^{ref} \right| \tilde{\theta} \\ u_2 = u_2^{ref} + 2k_2 \left| u_1^{ref} \right| \frac{\tan \tilde{\theta}_{ref}}{L} g_1 - 2k_2 k_4 \left| u_1^{ref} \right| - u_1^{ref} (2k_2 + \frac{k_3}{2}) \tilde{\theta} - k_4 \left| u_1^{ref} \right| \frac{\tan \xi - \tan \xi_{ref}}{L} \end{cases} \quad \text{(V.2)}$$

where k_1, k_2, k_3 and k_4 are gain constants that have to be adjusted on experiment.

This law is particularly suitable as its simplicity is based on the fact that the provided open loop trajectories already satisfies all constraints in terms of feasibility and collision avoidance. It allows therefore a nice coupling with trajectories generated by PMP algorithm that satisfy already all these constraints. In order to properly identify the coefficients k_i we ran several simulations using the Scilab software developed by INRIA (Fig. V.7) and in a second step tested it on the Cycab platform, tracking an apriori generated trajectory (Fig. V.8).

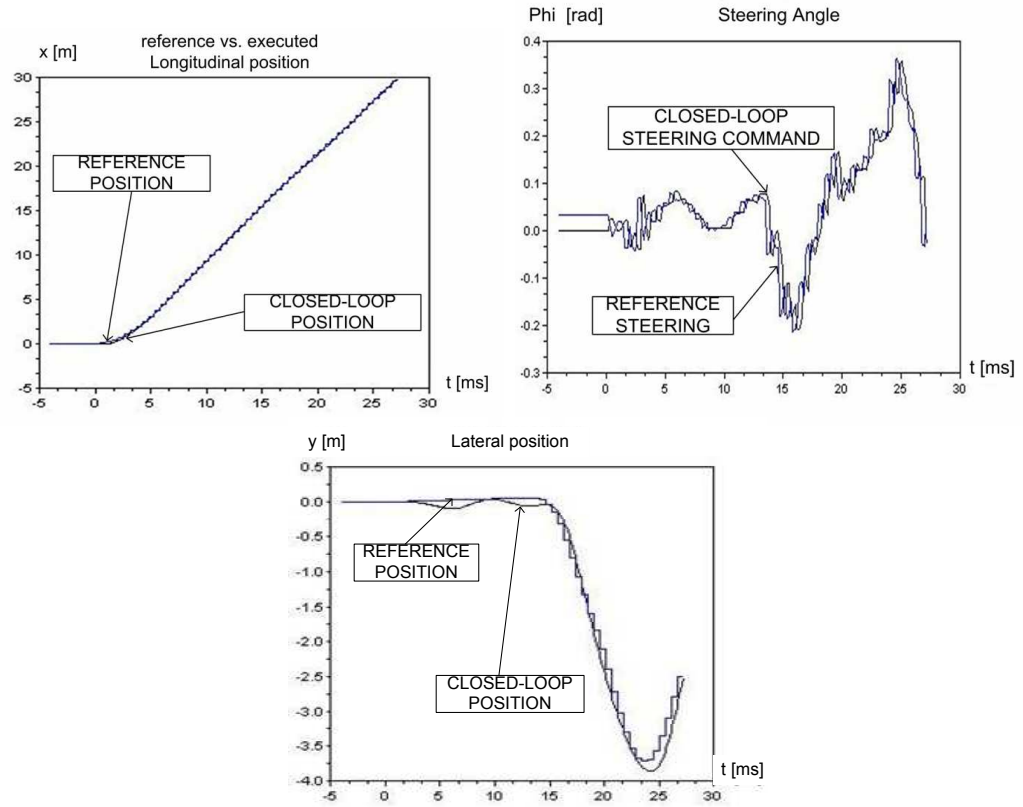


Figure V.8: Error measurements of a trajectory tracking by the Cycab.

4 PMP Software Architecture

PMP is composed of a Solver in charge of the planning and a deported graphical interface (fig. V.4). The two blocks communicate via a TCP/IP protocol in order to allow distant monitoring and control. Both kernel and GUI are implemented in C++.

4.1 PMP Solver

The kernel (fig. V.9) consists in several objects, used in most existing motion planning software. A first module integrates all the *problem* data. A second module describes the *geometry* of the world. The main purpose of this module is to transform the environment information (static or dynamic, remotely acquired or stored) into geometric objects (rectangular boxes or cylinders) that can be used for the collision detection implemented within the *collision checker* module. A specific module encompasses the *model* of the system which is used, *eg* the car-like robot in our case. The *PMP scheduler* module supervises the trajectory generation and handles the calls to the *PMP tree expansion* module,

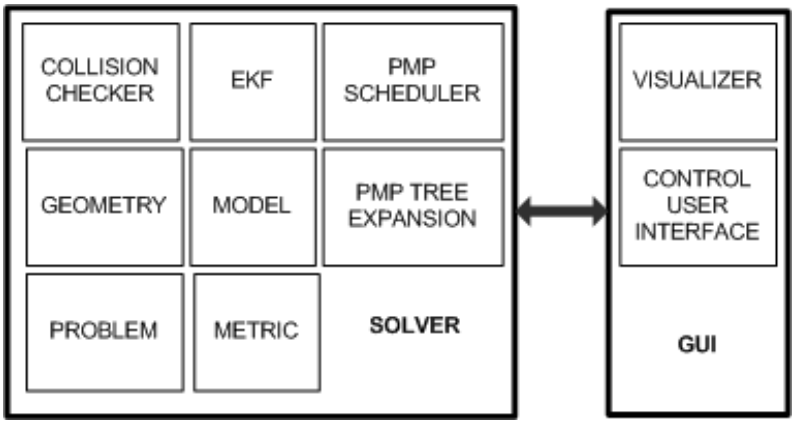


Figure V.9: PMP software architecture.

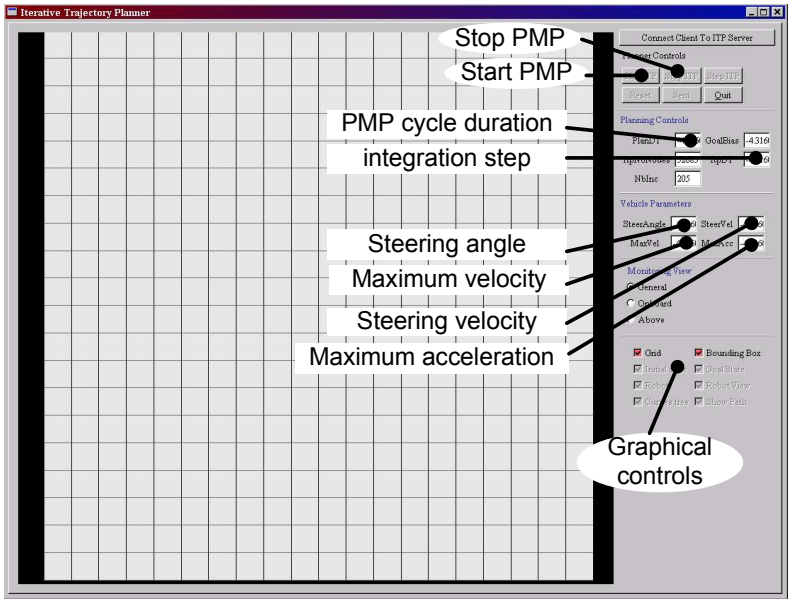


Figure V.10: PMP User Interface.

in charge of the incremental construction of the tree. The safety check for each new generated state is performed by the *collision checker* module. Finally, a *metric* module is used to define the proper metric, *eg* the continuous curvature (CC) metric in our work. This module is used to choose the best vertex out of the constructed tree. Finally, the solver calls the *communication* module used to send information to the GUI and the robot, *eg* TCP/IP module in our work.

4.2 PMP Graphical User Interface

The graphical user interface (GUI) (fig. V.10) is deported mainly to provide a visual feedback as well as control mean of the PMP running on real platform. The visual feedback consists in the generated trajectory represented by the sequence of calculated states, displayed within a 3D window handled by OpenGL as well as the obstacles known or perceived by the robot. The communication module is handling input/output communication from and to the robot and is based on TCP/IP communication in our work. There exists a light remote user control interface aiming at starting the planning process as well as modifying parameters from the car physics (*eg* max velocity, acceleration, steering angle and speed) or the PMP algorithm (*eg* the PMP cycle time, integration step).

5 Scenario 1 : Obstacle Avoidance

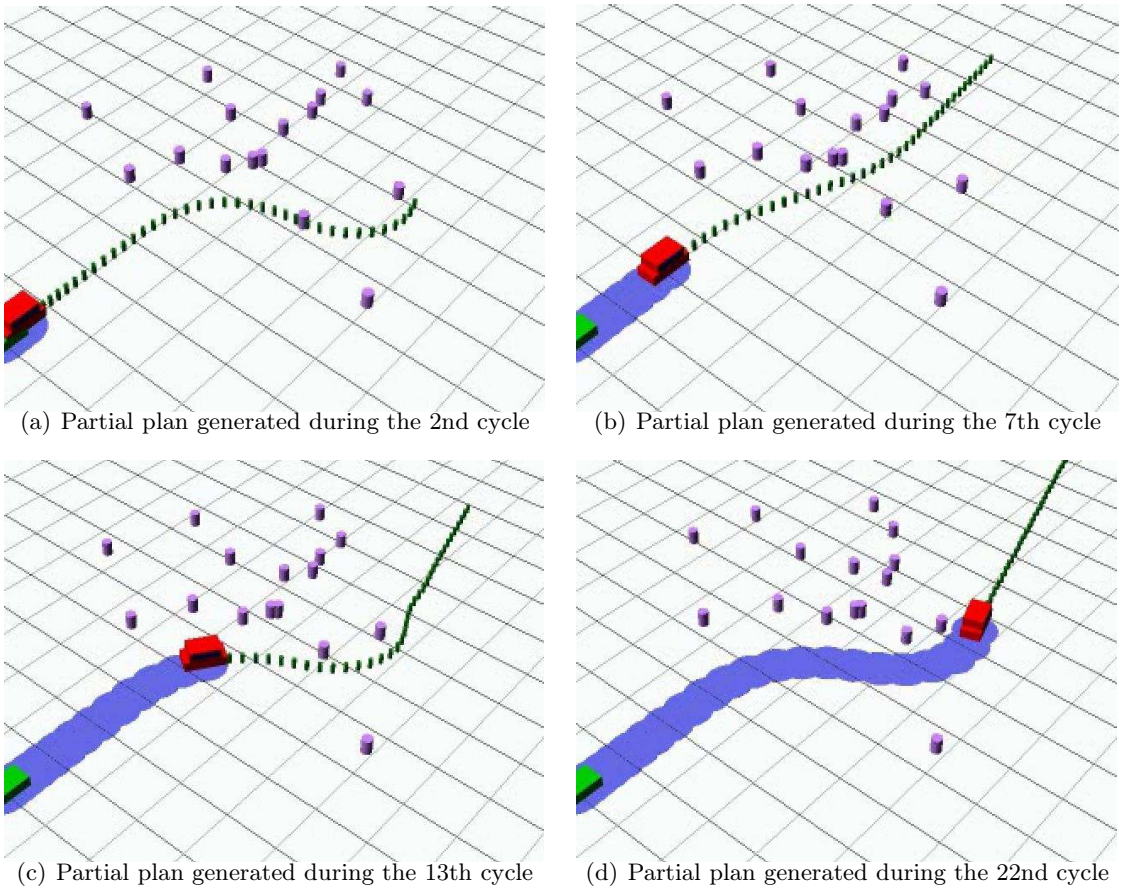
5.1 Motivation

Conventional Cybercars move autonomously along a given path that is predetermined and usually physically materialized by a wire, magnets or transponders buried in the ground. For these applications the main objective lies in the obstacle detection mainly, while the decision trivially consists in braking to stop the vehicle in front of it. However, more subtle trajectories might be required in some situations for which braking might not produce a desirable effect and for which avoidance would be preferred. In such a case, the PMP could be used as an alternative to generate such an avoidance trajectory. In the following of this section we will present simulation results aimed at illustrating how PMP can address this issue. Then we will describe a practical test, for which we first explain the settings and the present the results.

5.2 Simulation Results

The simulation results show safe trajectories generated by PMP avoiding virtual obstacles. In case the obstacles are moving, their future motion is known. PMP can produce therefore trajectories with large lookahead even in environments cluttered with a large number of moving obstacles, the motion of which needs to be updated. Figure V.11 illustrates several simulated cases for which one observe that a trajectory is generated to safely avoid the obstacles, pedestrians in this case. When the model is updated, a new trajectory is generated. PMP generates periodically safe trajectories, allowing the system to avoid obstacles, while accounting properly for the inherent kinematic and dynamic system's

constraints.



(a) Partial plan generated during the 2nd cycle

(b) Partial plan generated during the 7th cycle

(c) Partial plan generated during the 13th cycle

(d) Partial plan generated during the 22nd cycle

Figure V.11: Simulation of pedestrians avoidance with PMP (partially known environment).

5.3 Real Test Settings

5.3.1 Perception of the Surrounding Obstacles

The perception is achieved through the use of the IBEO laser scanner (Fig. V.12). The laser functions in 2 different modes. One mode, the "object mode" returns the position, relative to the car, of three points representing a detected obstacle (Fig. V.13). PMP uses this information to construct from this information a geometric bounding cylinder. These objects define the geometry of the obstacles with which PMP will perform the collision detection. The motion prediction of these moving obstacles, is calculated out of the velocity vector of the tracked objects, provided by the Laser scanner as well.

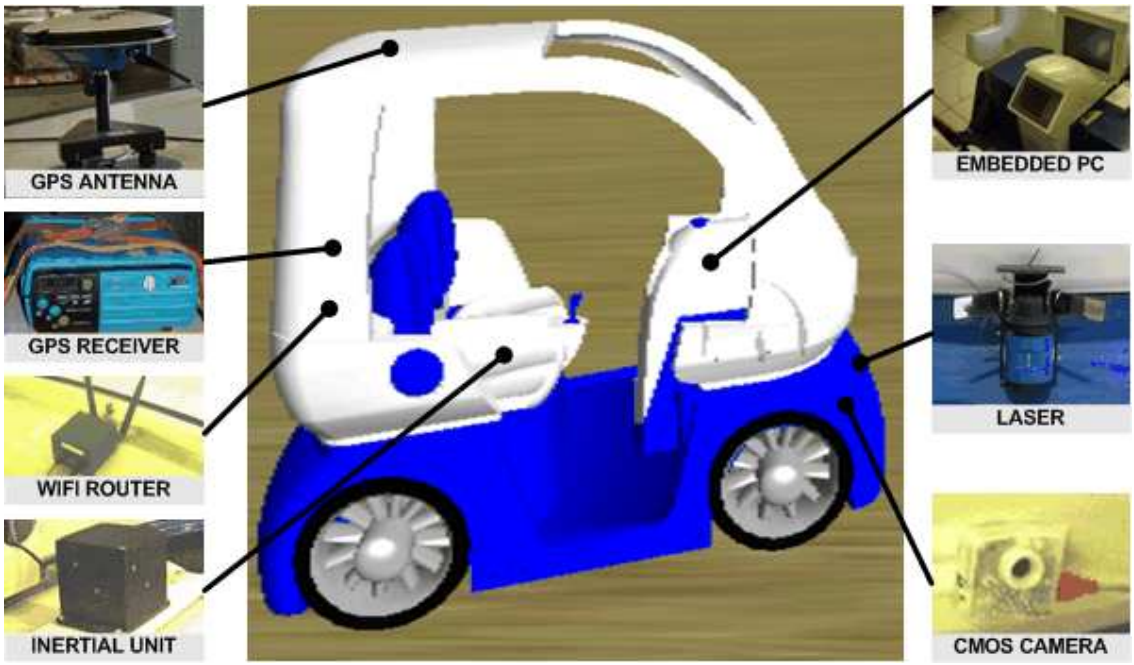


Figure V.12: Sensors used on the Cycab for perception and localization.

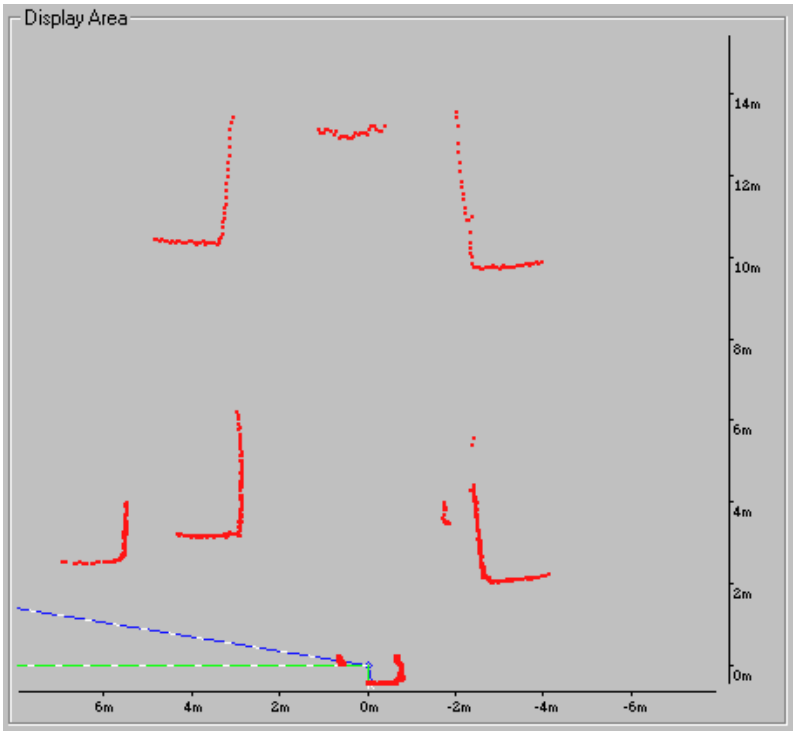


Figure V.13: Obstacle object sensed by the Laser Scanner IBEO.

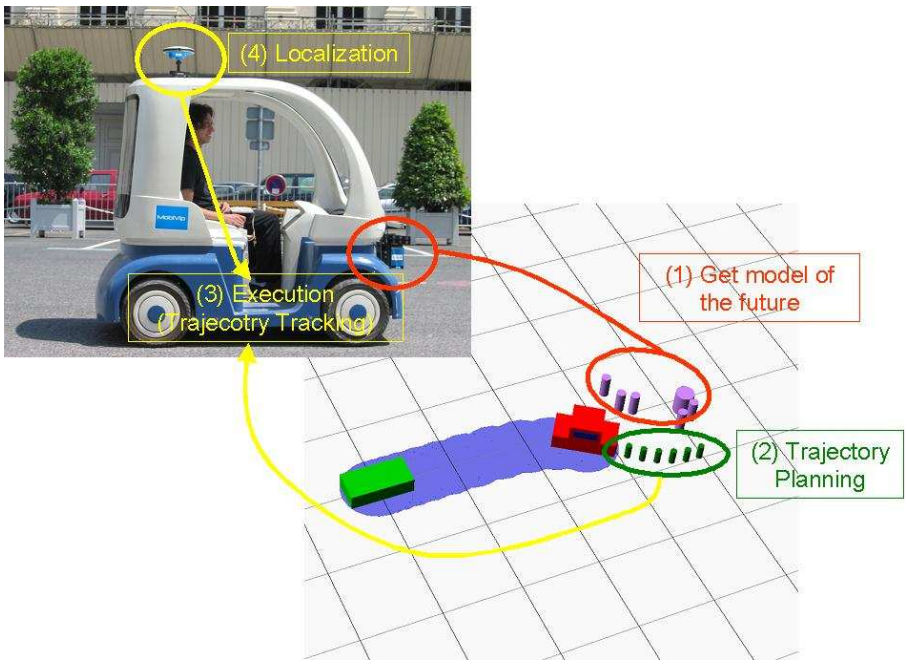


Figure V.14: Cycab avoiding pedestrians.

5.3.2 Localization

As for localization, a GPS Real Time Kinematics (RTK) sensor was originally used (Fig. V.12). This sensor has an accuracy of less than 5cm which allows to close the control loop. The GPS information is fused with inertial unit measurements so as to improve the estimate of the observation state.

5.4 Experiments

In this test scenario a user ¨calls¨ the Cycab, ie launches remotely the PMP program from a remote mobile device (*eg* smartphone or PDA) to pick him up. The Cycab final destination, which can be sent or hardcoded is set into the PMP program and PMP computes trajectories so as to simply move the Cycab straight along a large open road. The complete knowledge of the surrounding static obstacles is therefore not necessary in this case. As soon as a pedestrian crosses the road, his shape and motion (velocity) detected by the laser are sent to PMP. A new trajectory is then generated by PMP in order to safely avoid the pedestrian (see fig. V.14). As the Cycab performs its task, the model is updated and new trajectories are performed until the goal position is reached.

6 Scenario 2 : Intelligent Crossing

In a urban environment, the crossing is one of the major point of interest in terms of safety. Several research projects [2] have been funded to increase the flow of vehicles within a crossing while improving the safety. Motivated by this observation, we address in this experiment the problem of automating vehicles on one road of the crossing, for a safe intersection [BPL+06]. The main goal is therefore to use the longitudinal control inputs generated by PMP allowing safe intersection crossing. In the following, we first show simulation results for this scenario and then describe the practical test we performed.

6.1 Simulation Results

In our simulation, we present a vehicle that intends to move straight across an intersection. The knowledge of the vehicles moving on the perpendicular road allows to built a model of the world within which safe trajectories are generated. In this example, the safe control inputs require the Cycab to brake at the right time as it is the only possibility to remain safe with respect to the vehicle on the other road. When this latter has passed, the Cycab can continue its trip toward its goal. This example demonstrates how PMP can be useful even in its most simple form, *ie* used only to generate safe longitudinal control inputs.

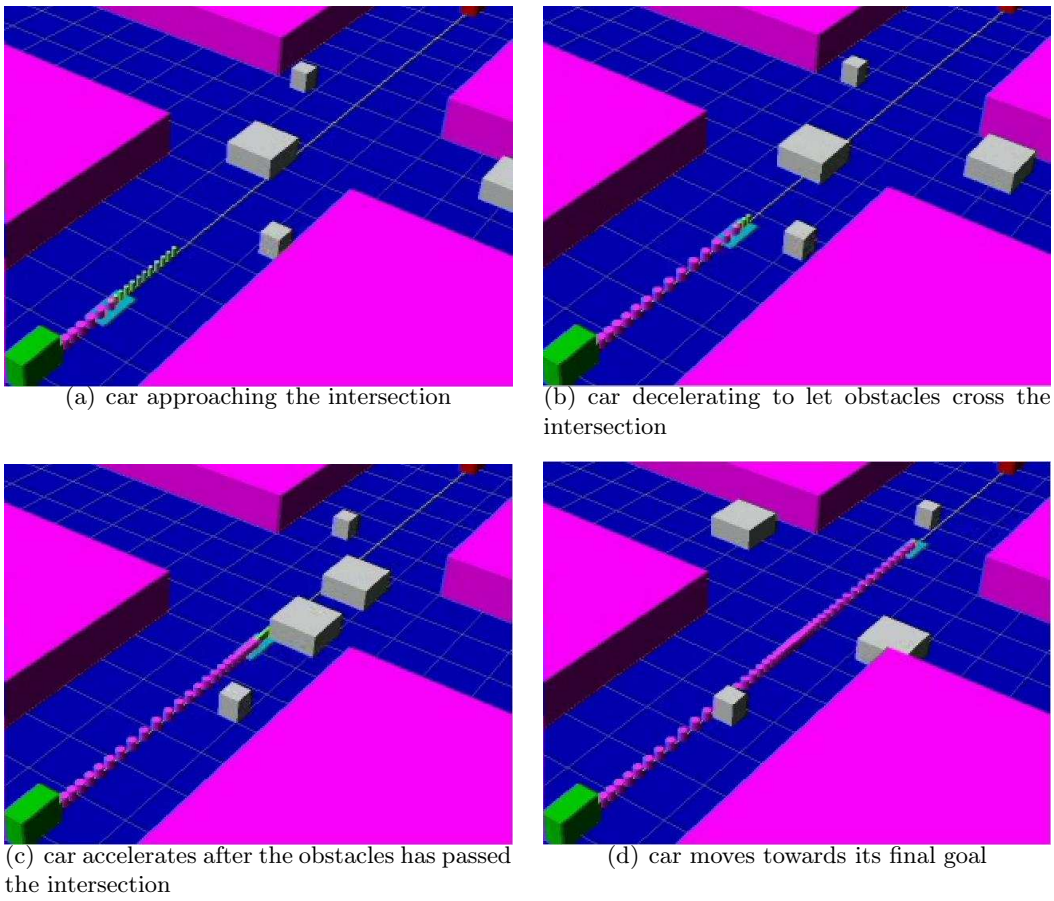
6.2 Real Test Settings

6.2.1 Configuration

Our architecture is composed of three components. The PMP block and its visualization module, a communication block which receives information of new neighboring vehicles and broadcasts its own vehicle information (via a Wifi router, see fig. V.12) and the Cycab application block which coordinates the different information flows. In this scenario, the only possible moving obstacles are other vehicles that are detected by the reception of a set of information forwarded by the communication block. The set of information contains useful data used by PMP to built a model of the environment used for motion planning. This data could be a description of the vehicle like its GPS information, its speed, its dimensions, or its planned itinerary (such as turning left, or right in the next crossing).

For our experiment we have used 2 Cybercars (One Cycab and one AGV, a Yamaha vehicle based on an electric golf car). Both vehicles were crossing at the same moment an intersection. Only the Cycab was running PMP and thus fully automated. The lateral

[2] INTERSAFE project, a subproject of the EU funded Prevent project (2002-2006)



(a) car approaching the intersection

(b) car decelerating to let obstacles cross the intersection

(c) car accelerates after the obstacles has passed the intersection

(d) car moves towards its final goal

Figure V.15: Intelligent Crossing simulation.

control inputs is determined thanks to a computer vision processing module which detects the road sides (Fig. V.12). The vision algorithm is based on the Poppet (Position of Pivot Point Estimating Trajectory) algorithm [WD99]. These inputs controls the steering of the Cycab and keeps it on track. On the other side, PMP generates safe trajectories from which longitudinal control inputs only are fed to the low level controller so as to set the longitudinal speed of the Cycab.

6.2.2 Car to Car Communication

During the experiment, each car is broadcasting periodically its GPS information to the mesh network. A static wireless mesh cube has been added at the intersection in order to enable communication between vehicles by relaying the forwarded messages, when they were out of reach in terms of radio. In fact, vehicles equipped with wireless medium in a crossing may form a mobile ad hoc network, and then may communicate and exchange their

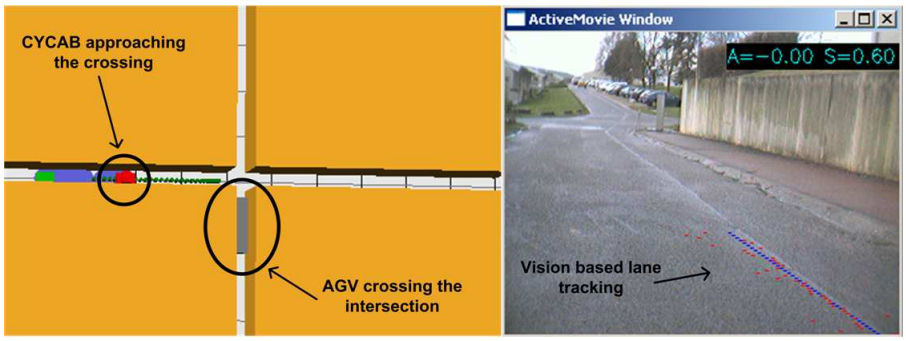
(a) the two cars at the intersection

Figure V.16: Automated cars at an intersection.

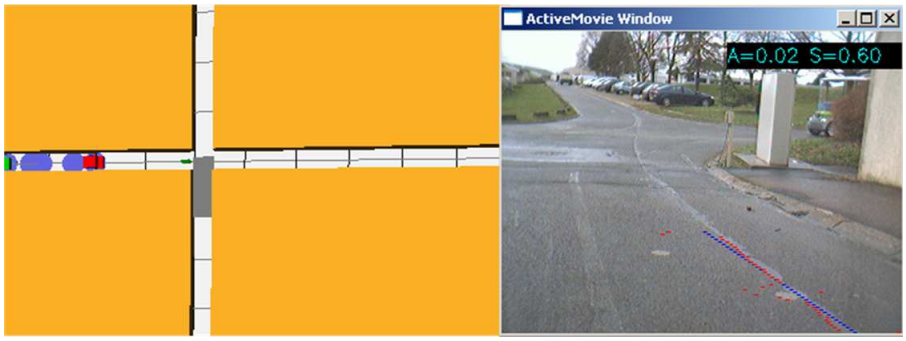
information. Several protocols were designed to enable communication for wireless ad hoc networks. This work uses the OLSR protocol. OLSR is an optimization of a pure link state routing protocol. It is based on the concept of *multipoint relays* (*MPRs*) [QLV02]. First, using multipoint relays reduces the size of the control messages: rather than declaring all links, a node declares only the set of links with its neighbors that are its "*multipoint relay selectors*". The use of MPRs also minimizes flooding of control traffic. Indeed only multipoint relays forward control messages. This technique significantly reduces the number of retransmissions of broadcast control messages [QLV02, Jac03]. The entire communication task is embedded in a small MIPS Linux Box, the 4G System Cube placed in each vehicle (Fig. V.12).

6.3 Experiments

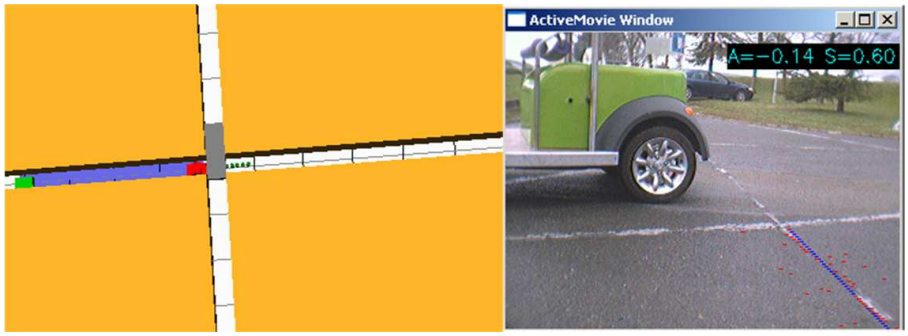
Each vehicle must broadcast periodically the set of information to its neighbors in order to refresh its characteristics, because a vehicle could change its direction or slow down its speed in any moment. As the vehicles approach the intersection, the wireless links between the three nodes (the Cycab, the AGV, and the static mesh cube) is established. Hence, the two vehicles can communicate their GPS information. This information is initially relayed by the static node when the cars are out of reach. This is done by the MPR relaying technique of OLSR. When the two cars are in radio range, they exchange their information without any relaying. Each received GPS information is processed by Taxi and sent to PMP. PMP interprets this information so as to recover obstacle



(a) Cycab approaching the intersection



(b) Cycab approaching the intersection



(c) Cycab approaching the intersection

Figure V.17: Intelligent crossing experiment.

information. Each time a new information is received, PMP generates an updated safe trajectory on the basis of each new obstacle information. Figure V.17 illustrates how PMP generated trajectories that decelerate the Cycab as the only possible way in this situation to remain safe. The AGV continues its way and passes the crossing. The Cycab can then safely cross in its turn, the crossing. Regarding the bandwidth of the communication between the cars, we noticed that we can guarantee around 500Kbits when the vehicle is communicating through the infrastructure node (the static mesh cube) and around 3Mbits with a direct connection.

7 Scenario 3 : Autonomous Navigation

7.1 Motivation

This test consists in demonstrating a scenario of autonomous navigation, in outdoor urban conditions. The main interest of this test is to demonstrate motion autonomy for a real car-like system, considering not only the differential constraints of this complex system but also the dynamics of the environment. Indeed, as opposed to most of practical implementations that usually rely on strong geometric assumptions for the map construction, and disregard the moving obstacles, this test consists of an original coupling of the PMP with a SLAMMOT technique that is capable to model the environment, both static and dynamic. We first present simulation results for our scenario. Then we describe the system architecture and present the practical test results of a Cycab moving autonomously outdoor among moving obstacles [BPFP06, BPFP07].

7.2 Simulation Results

In our simulation, a car has the objective to cross a urban area to reach its goal. Simulation results demonstrate the capability of PMP to generate trajectories within a urban context that are safe with respect to the known environment and with a significant lookahead.

7.3 Real Test Settings

7.3.1 Architecture

The approach proposed to address the problem of autonomous navigation lies in the coupling between perception and planning capabilities. The perception relies on a Simultaneously Localization and Mapping algorithm (SLAM) extended for moving objects detection and tracking so as to build a world model including static obstacles as well as a short term prediction of the moving obstacles motions. The deliberative scheme uses these models to generate trajectories that explicitly account for the dynamic constraints stemming from the environment and the system. The approach which is used rely on a deliberative strategy that interleaves planning with execution. It consists in incrementally and iteratively calculating a safe trajectory to the goal in order to provide motion autonomy to the system. Perception is the process of transforming measures of the world into an internal model. The kind of model (and the choice of the sensors) depends on the application. For autonomous navigation, the world model needs to integrate at least four elements: the target to attain, the position of the static obstacles, the current and future

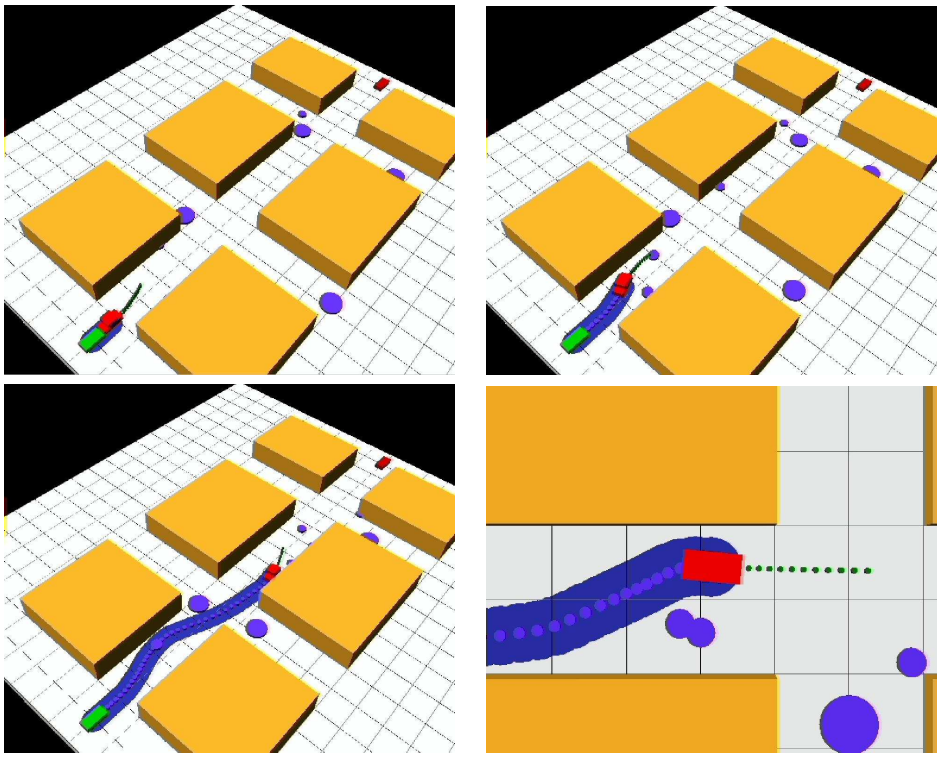


Figure V.18: Simulation results of an autonomous navigation within a city.

position of moving obstacles and the current state (position, speed, etc...) of the vehicle.

7.3.2 Perception and Localization

Environment Observability The work presented so far assumed fully observable environment, which in practice is very difficult to achieve due to the limited field of view of embedded sensors and the occlusion problem that further limits the observability of the world. In case the environment in which a plan is calculated is partially observed, our approach is to conservatively plan a trajectory considering the limit of the observed area as a potential danger.

Long Term Prediction Our experiments take place in a world not known in advance. Thus one has to resort to prediction in order to estimate the future position of the obstacles and built a model of the environment. long term prediction (many steps ahead) are much more involved given the complexity of the other cars or pedestrians movement behavior. In [VF04], the method assumes that humans mostly do not just move around but with the intention of reaching a specific location. The approach is based on the definition of the so-called ¨hot points¨ or goals in the environment where people would have interest

in visiting them. Once the points of interest of an environment are defined, then the long-term prediction refers to the prediction of which hot point moving obstacles is going to approach. At each time step t, the tangent vector of the obstacle's positions at times t-1, t and the predicted position at time t+1 is taken. This tangent vector essentially determines the global direction of the obstacle's motion trajectory, termed as the Global Direction of Obstacle (GDO). This direction is employed to determine which hot point a moving obstacle is going to approach. In order to find these points, a field of view is established and potential points are reachable according to a probability defined by a Gaussian probability distribution centered at the GDO with a standard deviation proportional to the angular extent of the field of view

Short Term Prediction Short term prediction over the next step provides satisfactory results. The solution of a sequential decision problem in a completely observable environment where the robot always knows its state is called a Markov Decision Process. Due of occlusion and limited field of view the robot can not observe the entire world at each measurement. When there is uncertainty or not enough information to determine the state of the robot, the problem is called, a Partially Observable Markov Decision Process. Navigation combining prediction and uncertainties have been proposed using Bayesian framework in order to propose a solution. POMDP are computationally inefficient and rely on a coarse discretization of the state space of about one squared meter for most of the work. The state of actions is discretized as well. In [FT02] the use of a hierarchical POMDP model is proposed that integrates the localization and collision avoidance with the use of future motion prediction modules for autonomous robot navigation. Integrating successive observations into a consistent map of forward obstacles is required to create an effective planning. It is well know that it exist a duality between creating consistent maps and localizing the robot, such duality has been extensively studied as the Simultaneous Localization And Mapping (SLAM) problem [Thr02]. Unfortunately most of the works in SLAM suppose that the environment is static. The presence of moving obstacles will contaminate the map and perturb the data association between two observations. For the planning purpose we require to explicitly identify the moving obstacles and estimate they current state in order to predict they future position.

7.3.3 The SLAMMOT Approach

We can see that for autonomous navigation, as a strict minimum the robot requires to solve the Simultaneous Localization, Mapping and Moving Objects Tracking (SLAMMOT) problem [Wan04]. The key point to create a correct maps (and thus correctly localize the robot) is to successfully do data association between current and past measures. Data

association methods have a limited "attraction region", if the initial guess is outside this region the association will produce an erroneous result. The attraction region depends of the existing map, the initial estimate, the current measure and the method employed. When the robot successfully recognize a previously visited place the SLAM algorithms will allow it reduce the uncertainty of his pose uncertainty helping thus in the data association process. The Incremental Maximum Likelihood method [Thr02] is a simple approach for small scale map construction. The incurred error is acceptable when the robot does not close a loop and the drift inside the map is under the desired bound. The incremental construction of the map eliminates the need to store the previous measures or to recompute online the map. A set of small scale maps can be used as building blocks for a larger map. In outdoor mobile robotics, the sensors commonly employed to observe the surrounds are video cameras, radars and laser scans [TCD+01]. We choose the last one due of his larger range (more than 180° and 40 meters) and high precision (±1° and ±0.1 meters) (Fig. V.12). Notice that the laser scanner measures provide information about the presence of obstacles and the existence of free space.

7.4 Experiments

The SLAMMOT algorithm has been implemented in C++. It has been integrated in the Cycab combined together with PMP into one single application. Both PMP and SLAMMOT algorithms are designed to incrementally and iteratively construct a solution which enables an efficient and simple interweaving. The tracking of the generated trajectories is insured by a non-linear closed loop controller detailed in earlier [SBA05]. The integrated system is able to autonomously drive in real world environments toward goals lying within about a hundred meters, while avoiding static and moving obstacles. The complete software runs at 10 [Hz] on a standard 3.3 [GHz] PC. Currently the only input data used is one layer of an IbeoML laser scanner.

In figure V.19 we present the result of the experiment. The top pictures are snapshots of the world model constructed during a single experiment. The black areas represent higher occupancy probability of static obstacles, whereas the gray areas depicts the part of the world that have not been observed yet. In this implementation PMP does not consider these areas as obstacles. The choice might be more conservative in other circumstances, and these areas might be seen as obstacles instead. As for the moving obstacles, they are represented by a circle. Current results do not include the estimation of unobserved obstacles. The rectangle at the bottom of the picture describes the current vehicle pose.The bold line represent the trajectory that has already been executed in previous cycles. The dotted line represents the trajectory planned within one single PMP cycle.The bottom

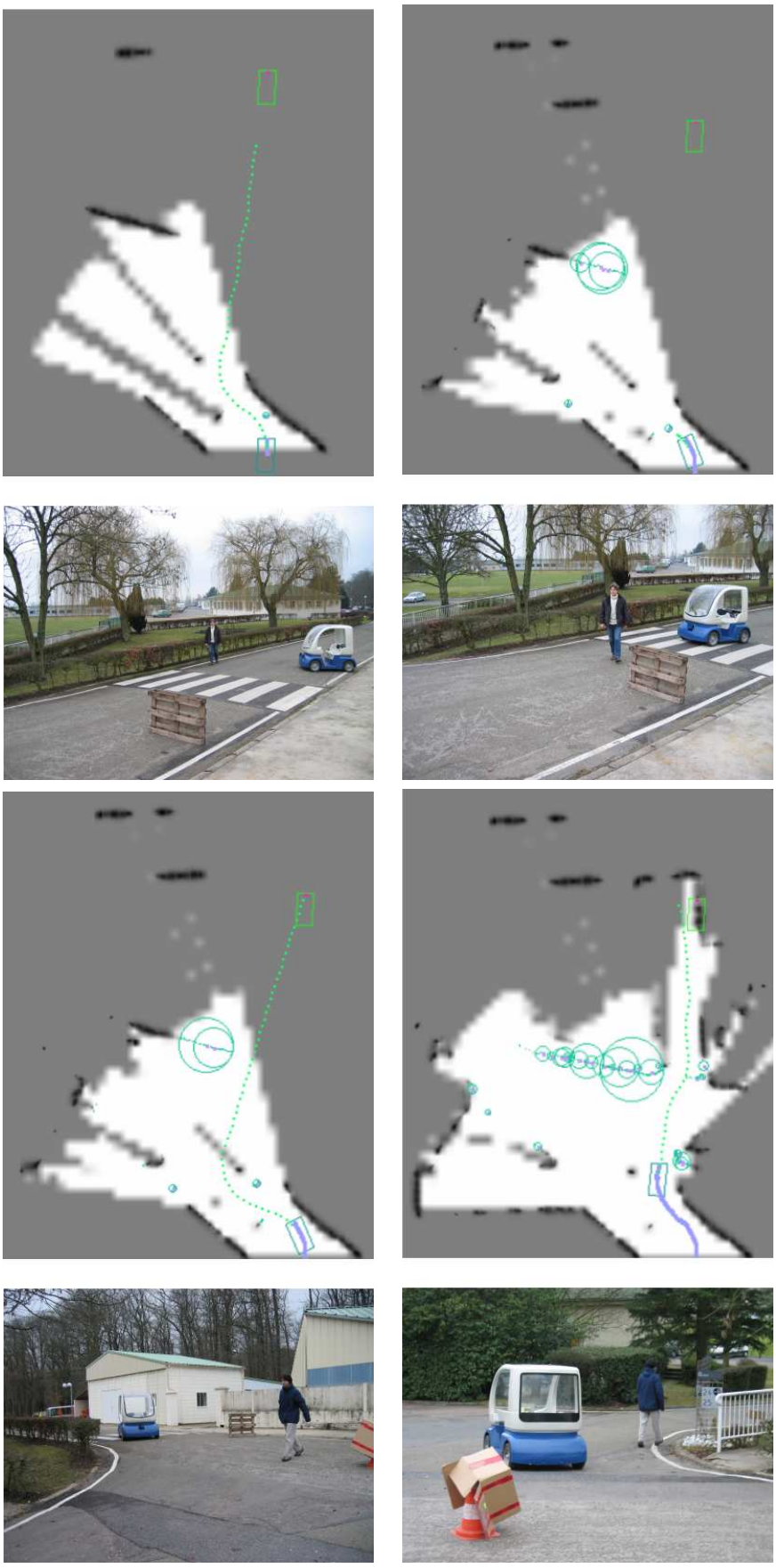


Figure V.19: Autonomous Navigation.

pictures show the corresponding scenes in the real world. During initial validation, the maximum speed of the Cycab is limited to low speeds (1.5 [m/s]), full speed experiments at higher speed (4 [m/s]) will be done in the future. A PMP cycle duration is $1s$. First results indicate that this new architecture is functional and provides the expected behavior. Morevoer, we can notice that for each cycle PMP has actually enough time to compute a complete safe trajectory to the goal, which shows the efficiency of the calculation.

8 Return of experience

The integration of PMP within the Cycab architecture has raised many challenges. At first, concerning the platform architecture, the choice to connect PMP as a node communicating with the other nodes using TCP/IP was very appealing for its interoperability within different platforms and its ease of integration. However, this protocol is not deterministic which leads to some uncontrolled latency. Or this time error yields to space error when it propagates to the low level controller. This observation motivated us to completely integrate the PMP and SLAMMOT application within the Cycab framework to avoid such latency for the autonomous vehicle experiment. This integration proved to be able to remove most of these delays. The internal CAN based control architecture granularity of 10Hz brought in turn some problems and in fact every data from every sensor, even though synchronized, brought their part of error, as once again, time error causes space errors. The concept of synchronization requires much more care and requires to be improved. The key is to control be able to precisely time-stamp any sensor observation, so as to recover later on the real information in case the information is used with some delay. The concept of time-stamping however, over a distributed architecture is difficult. Furthermore, for the positionning, the difficult handling of the GPS RTK motivated us to locate relatively to the obstacles and not absolutely on the earth. In fact there are several applications for which relative positioning might be sufficient, *eg* collision avoidance. However, as soon as one perform quite a longer motion, one will need to be absolutely locating or at least located on a local map that encompasses the complete motion. This is the reason why we integrated PMP with a SLAMMOT technique, which, even though requires some refinement and improvement, paves the way from our prospective, to a promissing approach toward driverless vehicles.

(Expérimentations)

Ce chapitre présente les résultats d'expérimentations qui procèdent de l'intégration de l'algorithme PMP au sein d'une architecture et plateforme robotique réelle, le Cycab. Le Cycab est un petit véhicule électrique présenté comme une alternative au véhicule privé pour les villes ou l'utilisation du véhicule particulier ne sera plus permise. Après une présentation de l'architecture véhicule, la couche commande bas niveau est discutée. En effet, il est à noter qu'une fois que la planification de mouvement calcule une trajectoire, le Cycab doit l'exécuter. Il est bien connu en automatique, que l'exécution de cette trajectoire planifiée en boucle ouverte ne donnerait pas de résultats satisfaisants. Pour ce faire l'utilisation d'une loi de commande de suivi de trajectoire a été implantée afin de garantir l'exécution du plan correct.

Trois expériences ont été menées à bien sous formes de trois scenarii différents. Tout d'abord, dans le cas d'évitement d'obstacle pour lequel l'architecture utilisée comprend des capteurs permettant un positionnement précis, GPS-RTK et centrale inertielle, et un laser scanner pour la détection des obstacles. Le second cas consiste en en croisement au sein duquel une voiture est gérée de manière autonome en utilisant la génération de trajectoire effectuée par PMP. Enfin le dernier scenario est celui de navigation complètement autonome d'un Cycab. L'architecture utilisée originale comprend cette fois le couplage du planificateur avec un algorithme complexe de localisation simultanée et construction de cartes, avec tracking d'obstacles mobiles (SLAMMOT) développé au sein de l'équipe. Ce couplage innovant permet de planifier des trajectoires dans un monde modélisé et prédit suivant cet algorithme. Différents tests montrent la faisabilité d'intégration du planificateur au sein de l'architecture de la plateforme réelle, le Cycab, en environnement réel et valide le bien fondé de l'approche proposée dans ce travail.

Chapter VI

Conclusions and perspectives

1 Conclusion

For more than ten years, in the United States, in Europe or in Japan, we observe an increasing effort in research and development in road transport. In response to the problems of congestion, pollution and safety raised by the increasing usage of personal cars in urban areas, future transport modes have been proposed. They rely on the use of individual vehicles circulating in urban site (Praxitèle in France, ICVS of Honda in Japan), or on dedicated sites (ParkShuttle in the Netherlands, Serpentine in Switzerland, IMTS of Toyota in Japan). In the long term, it is envisaged that this type of vehicles is equipped with full autonomous control capability. However, the development of fully autonomous cars can be done only by taking into account the particular nature of the environments: one face here environments cluttered with many mobile obstacles (other vehicles, pedestrians, etc.), being able to evolve and move at high speeds and whose future behavior is a priori unknown. To date, autonomous navigation in environments of this type remains largely an open problem.

Our work addressed the problem of autonomous navigation within partially known dynamic environment. This problem has attracted a lot of work within last ten years, and the most relevant work were presented and discussed in the first chapter. Nevertheless, we observed that several crucial aspects related to dynamic environments have been neglected, to start with, the limited time that a system placed in a real environment has to take a decision and execute a motion. This real time constraint lied at the heart of our approach.

- Thus, the first contribution of this work was to present a planning scheme that

accounts explicitly for the real-time constraint imposed by the dynamic environment. This approach, lies between the well known reactive and deliberative paradigm, the Partial Motion Planning (PMP). PMP is a motion planning scheme with an anytime flavor : when the time available to compute a new trajectory is over, PMP returns the best partial motion to the goal computed so far. As opposed to reactive methods that calculate the next time step at a time, and deliberative methods that calculate the complete sequence to the goal, PMP consists in planning as many steps as possible within a fixed available time. Compare to other approaches, PMP exhibits several benefits combining advantages of deliberative approaches, (long lookahead, dynamics of both the system and the environment), and reactive approaches (able to cope changing environments), while meeting the time requirements. In fact, to our prospective, PMP approach is the best answer to the problem that we observed, namely the incompatibility between motion planning in a dynamic environment (MPDE) and the real time constraint (RT).

- Furthermore, PMP faces a safety issue, that we addressed from a novel concept of Inevitable Collision State (ICS), concept that encompasses all existing approaches. The second main contribution of this work was to incorporate the ICS framework within PMP and propose a technique that insures strong safety guarantees for our system.

- Finally, PMP is designed for real world application and therefore accounts explicitly for both dynamics of the system and the environment, which enables high quality, feasible and safe trajectory generation for systems as complex as car-like systems evolving within dynamic environment as for instance a urban environment. The third main contribution of this work was certainly to demonstrate the effectiveness of the PMP approach for quite complex real world applications. Thus, we have integrated PMP within a real platform, the Cycab, car-like robot platform designed by INRIA. We successfully demonstrated the effectiveness of PMP through two complex scenarii. At first we used PMP to control a car on an intersection. Intersections are dangerous zones, where a system that is designed to take control of the car (or at least warn the driver) could prevent collision. In our experiment, the Cycab was automatically guided and used a car to car communication unit to obtain relevant information about the surrounding moving vehicles. At second, PMP was used as the planning core of a full autonomous car application. PMP has been combined with an innovative SLAMMOT algorithm developed at INRIA to model and predict the environment. Then both modules have been integrated within the Cycab framework developed at INRIA. The Cycab moved toward the goal while successfully avoiding surrounding moving obstacles. For both of these applications, PMP

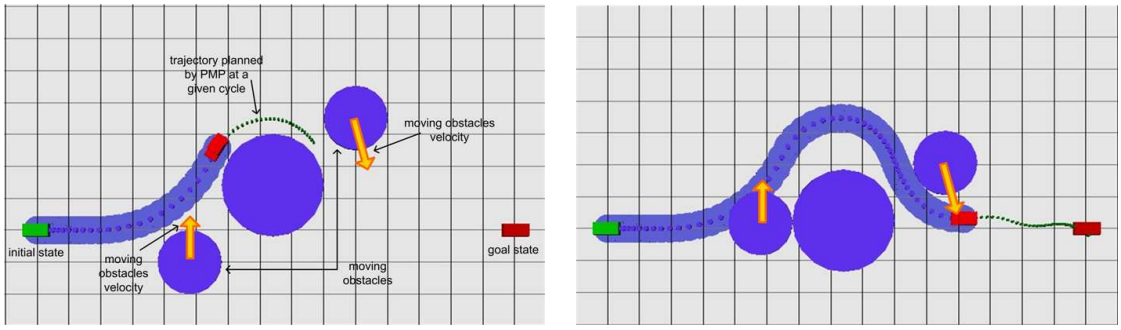


Figure VI.1: PMP planning without uncertainty among moving obstacles.

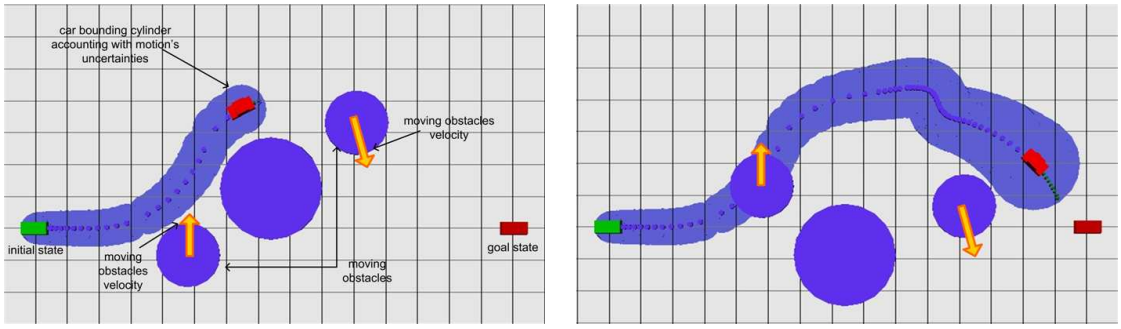


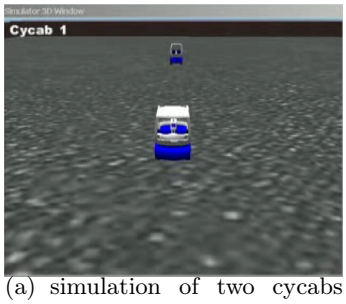
Figure VI.2: PMP planning with uncertainty among moving obstacles.

appeared extremely well adapted to provide in the required time constraint, high quality, feasible and safe trajectories, allowing the Cycab, to reach its goal safely.

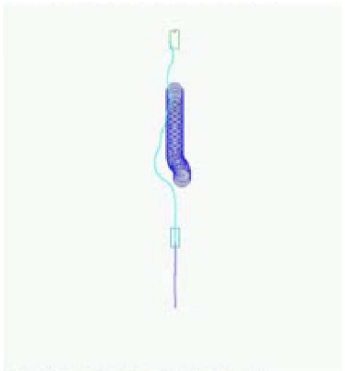
2 Perspectives

As future works, the tree expansion technique at first will be optimized so as to generate longer trajectories within a shorter computation time. Our implementation provided good results sufficient in our experimentation, however, the observations made could not provide robust models over a long period and in order to cope with it, PMP might need to run at higher frequency while generating still long trajectories, specially at higher vehicle speed. At second, the convergence aspect of PMP shall be studied in a formal and general way. In this work, the choice of the metric in our case study has been driven by the need to improve the scheme convergence. A general approach to this problem remains however an open issue.

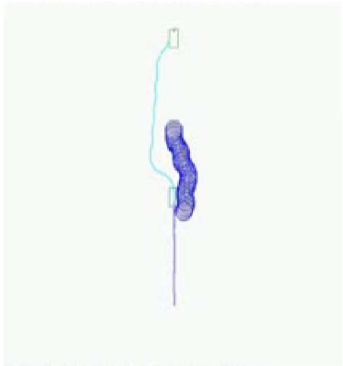
There are several extensions to this work that we believe could be promising. The first interesting aspects concerns the consideration of uncertainty while planning. Indeed, basic motion planning problems assume that the current state of the robot is known at



(a) simulation of two cycabs moving against eachother



(b) snapshot of the two cycabs generated collaborative trajectories



(c) snaphot of the two Cycab's trajectories avoiding eachother

Figure VI.3: Two autonomous Cycab's using PMP and adapting to each other's motion to collaboratively avoid collision .

execution such that the plan can be perfectly executed. However, in real world application, uncertainty exists. A solution to this problem is to design a scheme that explicitly takes this uncertainty into account and guarantee that the goal will be reached. Our preliminary investigation on the subject [PF05a] are promising as they demonstrate how to extending the partial motion planning (PMP) algorithm so as to account for the real robot's motion, using a probabilistic representation of the errors that appear at execution in the controls. The first results are very promising as we can see that the trajectories planned without accounting for uncertainty (see fig. VII.4) the ones and accounting for uncertainty (see fig. VII.5) at the planning stage are different. The trajectory planned to account for uncertainties behaves more conservatively.

The second major extension we foresee relies on the concept of cooperative planning. Indeed, it is certainly of a great interest to better understand how several autonomous systems moving using PMP would react to each other and how there task can be properly performed while adapting to others motions. A first investigation performed at INRIA on this subject has produced interesting results. In fig. VI.3 the case of two cars moving one against the other has been studied. Each one of them uses PMP to calculate its trajectory. We can see that both vehicles correctly and collaboratively avoid each other and share the avoidance motion. This work points to the direction of the motion planning problem for a set of collaborative vehicles, which certainly will become a fascinating problem in the near future.

Conclusion

Depuis plus de dix ans aux USA, en Europe ou au Japon, nous observons un effort grandissant dans la recherche et le développement, dans le domaine du transport. En réponse aux problèmes de congestion, pollution et sûreté, soulevés par l'utilisation croissante des voitures en milieu urbain, de nouveaux modes de transport ont été proposés. Ils s'appuient sur l'utilisation de véhicules individuels circulant sur site urbain (projet Praxitèle en France, ICVS de Honda au Japon) ou sur site dédié (Parkshuttle aux Pays-Bas, Serpentine en Suisse, IMTS de Toyota au Japon). Dans le long terme, il est envisagé d'équiper ce type de véhicules afin de les rendre complètement autonomes. Cependant, le développement de voitures complètement autonomes ne peut être réalisé qu'en prenant en compte la nature particulière de l'environnement: nous sommes confrontés ici à des environnements peuplés d'obstacles mobiles (autres voitures ou piétons), capables de se mouvoir à des vitesses élevées et dont le mouvement futur est à priori inconnu. A ce jour, le problème de navigation autonome dans ce type d'environnement reste largement ouvert.

Notre travail aborde le problème de navigation autonome en milieu dynamique partiellement connu. Ce problème a généré de nombreux travaux durant les dix dernières années, dont les travaux majeurs ont été présentés dans le premier chapitre. Néanmoins, nous avons observé que certains aspect fondamentaux liés à la nature dynamique de l'environnement ont été oubliés, en commençant par la limite de temps qu'un système placé dans un environnement dynamique a pour prendre une décision et exécuter son mouvement. Cette contrainte temps réel a été placée au coeur de notre travail.

- *Ainsi la première contribution de ce travail, a été de présenter une technique de planification qui prend en compte de manière explicite la contrainte temps réel imposée par l'environnement dynamique. Cette approche se situe entre les paradigmes connus des approches délibératives et réactives. La planification de mouvement partiel (PMP) est une technique de planification qui réagit à tout moment; dès que le temps imparti pour calculer une trajectoire est écoulé, la meilleure trajectoire calculée disponible est transmise. Au contraire des méthodes réactives qui calculent sur un seul pas de temps à la fois et des méthodes délibératives qui calculent la séquence complète jusqu'au but, PMP consiste à planifier dans un temps limité, une trajectoire qui eventuellement n'est que partielle. Comparé aux autres approches, PMP montre plusieurs avantages, combinaison des approches délibératives (longue visibilité et prise en compte de la dynamique du système et de l'environnement) et réactives (capacité à prendre en compte les changements dans l'environnement), tout en respectant cette contrainte de temps. En fait, de notre point de vue, l'approche PMP est la meilleure approche possible au problème que nous avons observé, à savoir*

l'incompatibilité intrinsèque entre la planification en environnement dynamique et la contrainte temps réel.

- *De plus, PMP est confronté à un problème de sûreté, que nous considérons d'un point de vue des états de collisions inévitables (ICS), concept récent qui englobe toutes les approches existantes. La seconde contribution de ce travail a donc été d'intégrer le cadre des ICS à PMP et de proposer une technique qui donne de fortes garanties de sûreté pour notre système.*

- *Enfin, PMP est conçu pour des applications réelles et prend en compte de ce fait la dynamique du système et de l'environnement de manière explicite, ce qui permet de générer des trajectoires faisables de qualité et sûres pour des systèmes complexes comme des voitures évoluant dans des environnements dynamiques tel que l'environnement urbain. La troisième contribution de ce travail est certainement de démontrer l'efficacité de notre approche PMP pour des environnements relativement complexes. Ainsi PMP a été intégré sur une plateforme réelle, le robot type voiture Cycab, véhicule conçu par l'INRIA. Nous avons prouvé avec succès l'efficacité de l'approche à travers différents cas d'études. Tout d'abord nous avons utilisé PMP pour contrôler un véhicule à travers une intersection. Les intersections sont des zones dangereuses, où des systèmes automatisés pourraient éviter de nombreuses collisions. Dans notre expérience, le Cycab était automatiquement guidé et obtenait par le biais d'un module de communication véhicule-véhicule l'information sur les voitures environnantes. Ensuite, PMP a été utilisé comme planificateur pour véhicule complètement autonome. PMP a été combiné à un algorithme de SLAM-MOT développé à l'INRIA destiné à modéliser et prédire l'environnement. Les deux modules ont été intégrés au sein de l'architecture du Cycab développée également à l'INRIA. Le Cycab a pu se déplacer vers son objectif tout en évitant les obstacles mobiles de l'environnement. Pour ces deux applications, PMP est apparu comme une technique très bien adaptée permettant de générer dans un temps limité de longues trajectoires faisables, sûres et de qualité qui ont amenées le Cycab vers son objectif.*

Perspectives

Comme travail futur, la technique de diffusion par construction d'arbre peut être optimisée afin de générer des trajectoires de durée supérieure, calculées plus rapidement. L'implantation actuelle suffisait largement, cependant la difficulté de modéliser l'évolution de l'environnement de manière robuste sur une longue période, nécessiterait une mise à jour du modèle plus rapide ce qui impliquerait au module de planification de travailler à

une fréquence plus élevée, spécialement pour des applications évoluant à des vitesses plus grandes. Ensuite, le problème de la convergence de PMP doit être étudié de manière plus formelle et générale. Dans ce travail, le choix de la métrique a en fait été guidé par ce souci d'amélioration de la convergence. Une approche générale à ce problème reste cependant ouverte.

Il y a plusieurs extensions à ce travail qui nous paraissent intéressantes. La première concerne la prise en compte des incertitudes au niveau de PMP même. En effet, la plupart des problèmes de planification de trajectoire considèrent que l'état courant du système est parfaitement connu lors de l'exécution et que l'exécution du plan se passe parfaitement. Cependant, dans le monde réel, les incertitudes existent. Une solution à ce problème pourrait passer par la prise en compte au niveau même du planificateur des incertitudes. Nos premières investigations en ce sens sont très prometteuses et montrent que PMP peut très facilement s'adapter pour intégrer ces erreurs.

La deuxième extension que nous entrevoyons à ce travail, se base sur le concept de véhicules collaboratifs. En effet, il serait extrêmement intéressant de mieux comprendre comment plusieurs systèmes, suivant des trajectoires générées par PMP réagiraient l'un envers l'autre. Une première investigation sur le sujet a fourni des premiers résultats intéressants dans le cas de deux véhicules se déplaçant l'un contre l'autre. Chacun calcule sa trajectoire grâce à PMP. Nous observons que chacun des véhicules s'adapte et l'évitement se fait de manière collaborative. Ce travail montre la direction du problème de planification pour un ensemble de véhicules collaboratifs, qui est fascinante.

Chapter VII

Annexes 1 : Partial Motion Planning under Uncertainty

1 Partial Motion Planning and Uncertainty

1.1 Planning under Uncertainty

Basic motion planning problems assume that the current state of the robot is known at execution such that the plan can be perfectly executed. However, in real world application, uncertainty exists. At first, there is uncertainty in the model of the environment (geometry, current and predicted state) as well as the model of the robot (model of the physics, geometry, state) that will affect the plan. Secondly, it is important to consider the case where the state cannot be known. In this case, information regarding the state is obtained from sensors during the execution of the plan. Sensor errors as well as control errors will further affect the execution of a motion. It is not possible to eliminate these errors and in case these imperfections are not small relative to the tolerance of the task being performed, it is important to generate plans robust to these errors. A solution to this problem is to design a scheme that explicitly takes this uncertainty into account and guarantee that the goal will be reached. It might be possible thus, to extend the partial motion planning (PMP) algorithm so as to accounting in the real robot's motion, using a probabilistic representation of the errors that appear at execution in the controls. As a model based control method, the PMP is highly suitable for such an extension.

First work on the subject ([Tay76, Bro82]) considered bounds or worst case on uncertainty within motion planning. A first plan was generated with no uncertainty and

then the plan was analyzed and modified in order to produce a robust plan. In these approaches, uncertainty is represented as a set of equiprobable possible values. The preimage backchaining approach is pioneered in [LMT84]. It was extended later [AS94] and used in simple cases for mobile robots [LL92]. A comprehensive state of the art of preimage backchaining is presented in [Lat91]. A preimage for a given motion command and a given goal region in configuration space is a set of free configurations from which the command can be started with the guarantee that the robot will reach the goal. In [TFL94] the concept of sensory uncertainty field (SUF) is introduced. A SUF represents for each configuration its estimation error computed by a sensor based localization. A planner using SUF can generate a path that minimizes expected errors by traversing workspace areas where visible environment features yield low sensory uncertainty. More specific approaches are presented in [BSA95, KJCL97, FM98]for which non-holonomic constraints are added to the problem.

1.2 Error Propagation

In this work the uncertainty stemming from the actuation imperfections is considered at the planning stage. Indeed, since the model for which motion planning is performed is known in advance, it is possible to establish an error propagation model for this error. The model of error propagation is analyzed using the predictive step of the Extended Kalman Filter (EKF) using the linearized form of the model. The linearized model is of the form

$$X = A(t)\dot{X} + B(t)U \tag{VII.1}$$

with $A(t) = \frac{\delta f}{\delta X}(t, X(T), U(t))$ and $B(t) = \frac{\delta}{\delta U}(t, X(T), U(T))$.

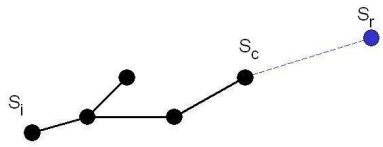
The error prediction is given by the covariance matrix

$$P(t_{k+1}|t_k) = A_k P(t_k|t_k) A_k^t + R_{sys} + B_k R_{com} B_k^t \tag{VII.2}$$

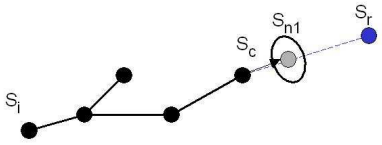
with R_{sys} the noise on the model and R_{com} the noise on the command represented using Gaussian probabilistic density functions.

1.3 Partial Motion Planning under Uncertainty

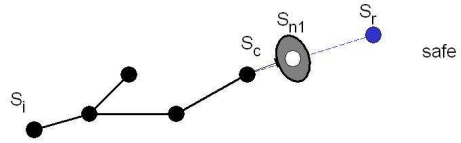
The covariance matrix informs about the propagation errors of the system. These errors appear as uncertainty of the robot configuration during exploration. Since the exploration tree method is a sample-based method, it relies on a geometric collision checker. The collision detection is performed over a circular bounding box of our system. Therefore,



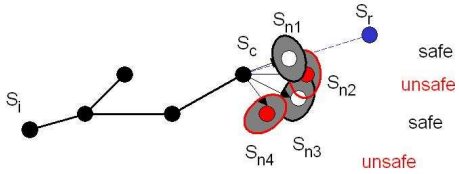
(a) Incrementation from initial tree



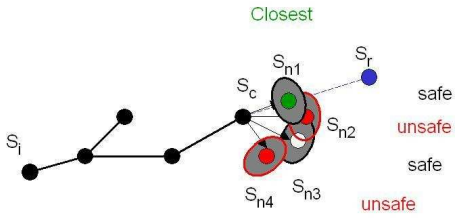
(b) New node contruction



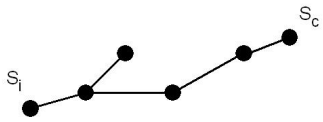
(c) Safety validation of constructed new node



(d) Repeat over all control input set



(e) Choose the best safe node



(f) New incremented tree

Figure VII.1: Tree construction for planning under uncertainty.

the maximum calculated component of the estimated position error is added to the radius of the circular bounding box of our system in order to provide safe planning with respect to the system's actuator errors. The tree construction is therefore similar to the previous one, with an additional step consisting in calculating together with the new state, the associated covariance matrix providing uncertainty information (fig. VII.1.

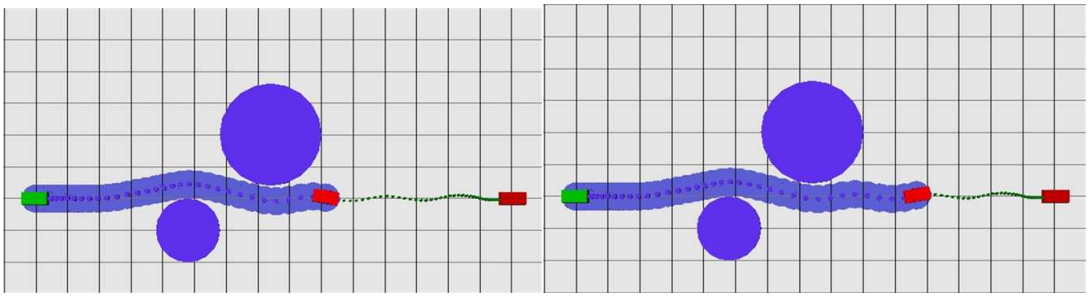
1.4 Towards Information Feedback Plans

The planning problem under uncertainty lies on the assumption that limited information only on the state of the system can be sensed. Thus, instead of estimating the state and pretend that there is no longer any uncertainty, the uncertainty is modeled within the planning scheme. In fact, such a planning problem is expressed in terms of an information space whose elements represent accumulated information about a system [BF95]. In our case, we suppose that there are no sensors and therefore no observations. In this case the future states are predictable. In the real world, observations from various sensors can be provided and incorporated in the presented method within the EKF framework in the update phase. Furthermore, this method can be seen as a step toward information feedback planning in the probabilistic information space. This space is derived from the information space, where each history information state is converted into a probability distribution over the state space and a Markov probabilistic model is assumed.

The results [PF05a] show the trajectories of a car after a few iterations where the model of the future of the environment is known a priori. In fig. VII.2, PMP generates a amidst static obstacles, without considering the system's uncertainty. In fig. VII.3 PMP generates a trajectory considering the uncertainty of the system. We can see that the trajectory between the two obstacles is not safe in case uncertainty is considered and therefore not preferred. This example perfectly illustrate the usefulness of integrating uncertainty as early as at the planning stage.

Figures VII.4 and VII.5 significantly illustrate the result of a planned motion for a car evolving within a dynamic environment. The trajectory planned with uncertainty (fig. VII.5) bypasses the obstacles and is therefore safer than the plan which does not account for uncertainty (fig.VII.4) which travels through the obstacles.

In these examples, we assume that no observations on the system's state are performed. The predictive phase of the EKF is used to calculate the error propagation in the robot's (speed and steering) controls. The largest error in position is added to the radius of the bounded circle of the robot used for collision detection. Thus, the PMP generates robust trajectories accounting for the drift in the controls. The safety of the trip is therefore increased as illustrated by simulation results. Depending on the task to be performed, it might not be necessary to perform observation during the trip and still reach the goal. A future work would consist in gathering observation during execution and update the state of the system in the update phase of the EKF.



(a)

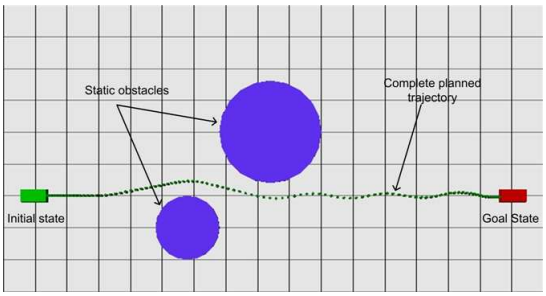
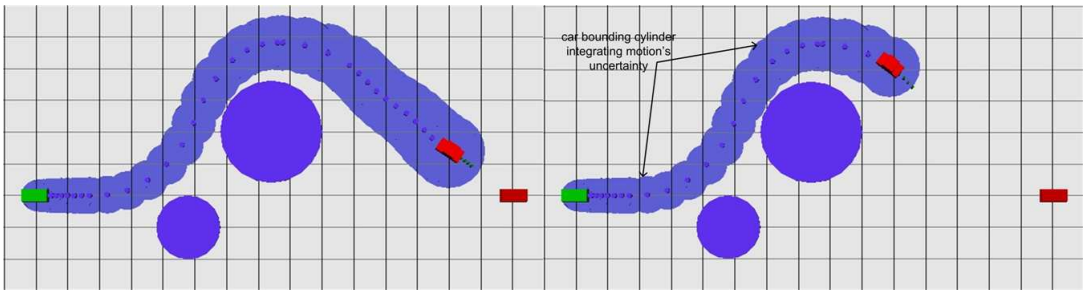


Figure VII.2: In case motion uncertainty are not considered, the size of the cylinder bounding the car does not increase. In this case PMP plans a trajectory between the two static obstacles.



(a)

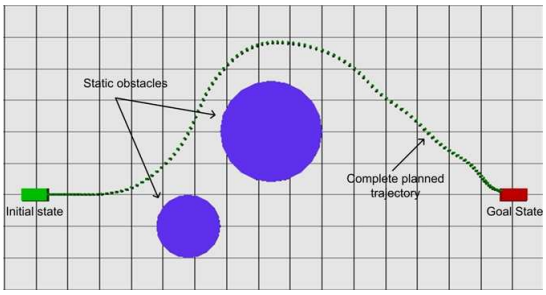


Figure VII.3: In case motion uncertainty is considered, the size of the cylinder bounding the car increases. In this case PMP plans a trajectory that bypasses the two static obstacles.

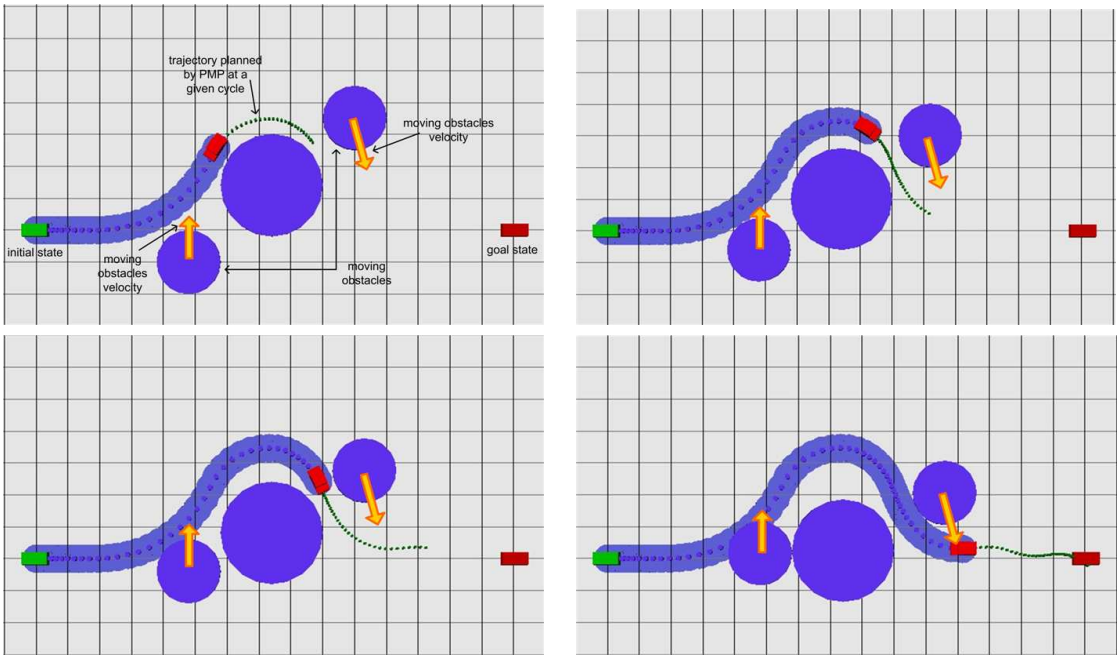


Figure VII.4: PMP planning without uncertainty among moving obstacles.

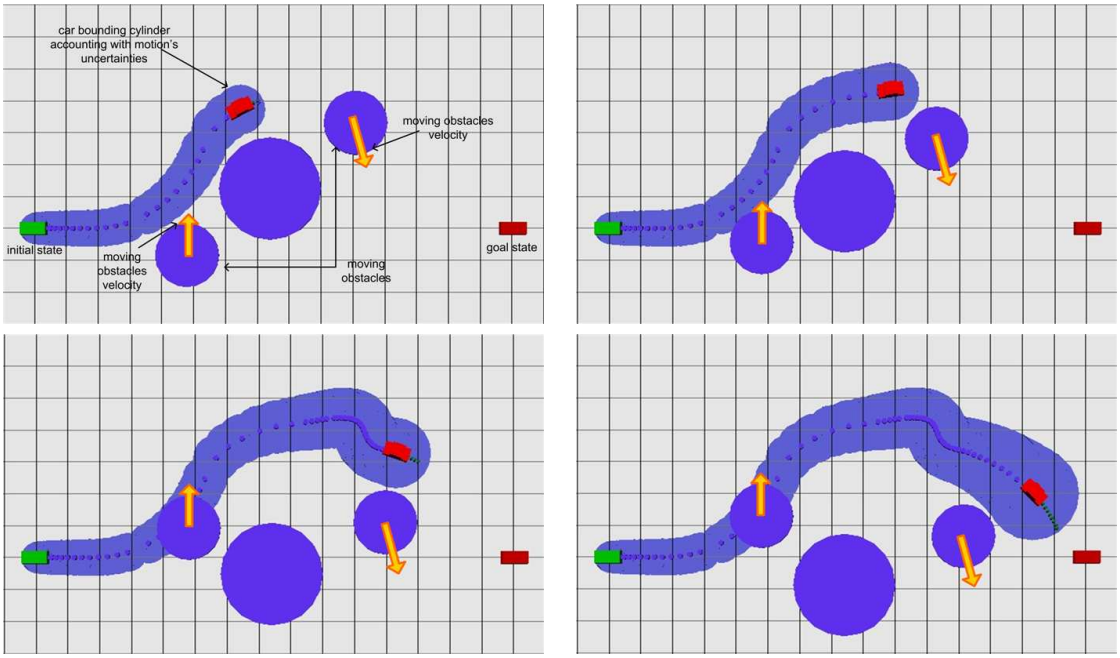


Figure VII.5: PMP planning with uncertainty among moving obstacles.

Bibliography

[ABD+98] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo. *Robotics: The algorithmic perspective*, chapter OBPRM: An obstacle-based PRM for 3D workspaces, pages 155–168. A.K. Peters, Natick, 1998.

[ABF88] F. Avnaim, J. D. Boissonat, and B. Faverjon. A practical exact planning algorithm for polygonal objects amidst polygonal obstacles. In *IEEE Int. Conf. Robot. and Autom*, pages 1656–1660, 1988.

[AS94] R. Alami and Th. Siméon. Planning robust motion strategies for a mobile robot. In *Int. Conf. in Robotics and Automation*, volume 2, pages 1312–1318, 1994.

[ASMK02] R. Alami, T. Simeon, and K. Madhava Krishna. On the influence of sensor capacities and environment dynamics onto collision-free motion plans. In *IEE/RSJ Int. Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.

[BDG85a] J. Bobrow, S. Dubowsky, and J. Gibson. Time optimal control of robot manipulators. *Int. Jour. Robotics Research*, 4(3), 1985.

[BDG85b] J. E. Bobrow, S. Dubowsky, and J. S. Gibson. Time-optimal control of robotic manipulators along specified paths. 4(3):3–17, Fall 1985.

[BF95] J. Barraquand and P. Ferbach. Motion planning with uncertainty: the information space approaches. In *Int. Conf. on Robotics and Automation*, 1995.

[BK91] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, June 1991.

[BK99] O. Brock and O. Khatib. High-speed navigation using the global dynamic

window approach. In *Proceedings of International Conference on Robotics and Automation*, Detroit (US), May 1999.

[BK00a] R. Bohlin and L.E. Kavraki. Path planning using lazy prm. In *Int. Conf. on Robotics and Automation*, pages 521–528, 2000.

[BK00b] O. Brock and O. Khatib. Real time replanning in high-dimensional configuration spaces using sets of homotopic paths. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, May 2000.

[BL93] J. Barraquand and J.C. Latombe. Nonholonomic multobody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10(2-3-4):121–155, 1993.

[BLL92] J. Barraquand, B. Langlois, and J.C Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2):224–241, 1992.

[BOvdS99] V. Boor, M.H. Overmars, and A.F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *IEEE Int. Conf. on Robotics and Automation*, pages 1018–1023, 1999.

[BPFP06] R. Benenson, S. Petti, Th. Fraichard, and M. Parent. Integrating perception and planning for autonomous navigation of urban vehicles. In *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing (CN), October 2006.

[BPFP07] R. Benenson, S. Petti, Th. Fraichard, and M. Parent. Toward urban driverless vehicles. *to appear in Int. Journal of Vehicle Autonomous Systems*, 2007.

[BPL+06] L. Bouraoui, S. Petti, A. Laouiti, Th. Fraichard, and M. Parent. Cybercar cooperation for safe intersections. In *IEEE Int. Conf. on Intelligent Transportation Systems*, Toronto, ON (CA), October 2006.

[Bro82] R.A. Brooks. Symbolic error analysis and robot planning. *Int. Journal of Robotics Research*, 1, 1982.

[Bro83] R.W. Brockett. *Differential Geometric Control Theory*, chapter Asymptotic Stability and Feedback Stabilization, pages 181–191. Birkkauser, 1983.

[BSA95] B. Bouilly, T. Siméon, and R. Alami. A numerical technique for planning motion strategies of a mobile robot in presence of uncertainty. volume 2, pages 1327–1332, 1995.

[BV02] J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.

[Can87] J. Canny. A new algebraic method for robot moitn planning and real geometry. In *28th IEEE Symp. on the Foundations of Computer Science*, pages 39–48, Los Angeles, CA (US), october 1987.

[Can88] J.F. Canny. *The complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[Cap20] Karel Capek. R. u. r. (rossum's universal robots), 1920.

[CBDN96] G. Campion, G. Bastin, and B. D'Andréa-Novel. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 12(1):47–62, 1996.

[CdWS91] C. Canudas de Wit and O.J. Sordalen. Exponential stabilization of mobile robots with nonholonomic constraints. In *IEEE Conference on decision and Control*, 1991.

[CR87] J. Canny and J. H. Reif. New lower bound techniques for robot motion planning problems. In *IEEE Symp. on the Foundations of Computer Science*, pages 49–60, Los Angeles, CA (US), October 1987.

[DSY87] C. Ó.Dúnlaing, M. Sharir, and C. K . Yap. *Planning, Geometry, and Complexity of Robot Motion,*, chapter Retraction: A new approach to motion planning., pages 193–213. Ablex Publishing Corporation,, Norwood, NJ, 1987.

[Dub57] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. 79:497–517, 1957.

[DXCR93] B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *Journal of the ACM*, 40(5):1048–1066, 1993.

[Elf89] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, June 1989.

[ET87] M. Erdmann and Lozano-Perez T. On multiple moving objects. 2:477–521, 1987.

[FA04] Th. Fraichard and H. Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.

[FBL94] W. Feiten, R. Bauer, and G. Lawitzky. Robust obstacle avoidance in unknown and cramped environments. *IEE International Conference on Robotics and Automation*, 3(2):2412–2417, may 1994.

[FBT95] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. Technical Report IAI-TR-95-13, 1 1995.

[FBT96] D. Fox, W. Burgard, and S. Thrun. Controlling synchro-drive robots with the dynamic window approach to collision avoidance. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[FBT97] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE journal of Robotics and Automation*, 4(1), 1997.

[FL91] T. Fraichard and C. Laugier. On line reactive planning for a non-holonomic mobile in a dynamic world. In *IEEE Int. Conf. on Robotics and Automation*, pages 432–437, 1991.

[FL92] T. Fraichard and C. Laugier. Kinodynamic planning in a structured and time-varying 2D workspace. In *Proceedings IEEE International Conference on Robotics and Automation*, Nice, (FR), May 1992.

[FM98] Th. Fraichard and R. Mermond. Path planning with uncertainty for carlike vehicles. Technical report, Inst. Nat. de Recherche en Informatique et en Automatique, Montbonnot (FR), 1998.

[Fra99] Th. Fraichard. Trajectory planning in a dynamic workspace: a state-time approach. *Advanced Robotics*, 13(1):75–94, 1999.

[Fra01] E. Frazzoli. *Robust Hybrid Control for Autonomous Vehicle Motion Planning.* PhD thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2001.

[FS89] K. Fujimura and H. Samet. A hierarchihcal strategy for path planning among moving obstacles. *IEEE Trans. Robotics and Automation*, 5(1):61–69, 1989.

[FS98] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772, July 1998.

[FT02] A. Foka and P. Trahanias. Predictive autonomous robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, Sep. 30 - Oct. 4, 2002.

[HJW84] John E. Hopcroft, Deborah Joseph, and Sue Whitesides. Movement problems for 2-dimensional linkages. *SIAM J. Comput.*, 13(3):610–629, 1984.

[HKL+98] D. Hsu, L. Kavraki, J.C Latombe, R. Motwani, and S. Sorkin. *Robotics: The algorithmic perspective*, chapter On finding narrow passages with probabilistic roadmap planners, pages 141–154. A.K. Peters, Natick, 1998.

[HKLR00] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Kinodynamic motion planning amidst moving obstacles. In *IEEE International Conference on Robotics and Automation*, San Francisco (US), April 2000.

[HKLR02] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, March 2002.

[Hol83] J.M. Hollerbach. Dynamic scaling of manipulator trajectories. In *American Control Conference*, pages 752–756, San Francisco (US), June 1983.

[HW86] John E. Hopcroft and Gordon T. Wilfong. Reducing multiple object motion planning to graph searching. *SIAM J. Comput.*, 15(3):768–785, 1986.

[Jac03] D. Jacquet. Détection de collision et action automatique pour transport routier. Master's thesis, Institut National des Sciences Appliquées de Lyon, 2003.

[Jai05] L. Jaillet. *Méthodes Probabilistes pour la Planification Réactive de Mouvements*. PhD thesis, University of Paul Sabatier, 2005.

[JC89] P. Jacobs and J. Canny. Planning smooth paths for mobile robots. In *Int. Conf. on Robotics and Automation*, pages 2–7, 1989.

[JP85] D. A. Joseph and W.H. Plantiga. On the complexity of reachability and motion planning questions. In *1st ACM Symposium on Computational Geometry*, pages 62–66, Baltimore (US), 1985.

[JS04] L. Jaillet and T. Siméon. A prm-based motion planner for dynamically changing environments. In *IEEE Int. Conf. on Int. Robots and Systems*, 2004.

[Kha86] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[Kha96] M Khatib. *Sensor-based motion control for mobile robots*. PhD thesis, LAAS-CNRS December, 1996, 1996.

[KJCL97] M. Khatib, H. Jaouni, R. Chatila, and J.P. Laumond. Dynamic path modification for car-like nonholonomic mobile robots. In *Int. Conf. on Robotics and Automation*, Albuquerque, USA, 1997.

[KL02] S. Koenig and M. Likhachev. Improved fast replanning for robot navigation in unknown terrain. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002.

[KS98] N.Y. Ko and R. Simmons. The lane-curvature method for local obstacle avoidance. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria (Canada), October 1998.

[KSLO96] L. Kavraki, P. Svestka, J-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional conguration spaces. *IEEE Trans. on Robotics and Automation*, 12:566–580, 1996.

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

[Lat91] J.-C. Latombe. *Robot motion planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[Lau86] J.-P. Laumond. Feasible trajectories for mobile robots with kinematic and environment constraints. In *Int. Conf. in Intelligent Autonomous Systems*, pages 346–354, 1986.

[LaV06] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[LB02] F. Lamiraux and D. Bonnafous. Reactive trajectory deformation for nonholonomic systems: Application to mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, DC, May 2002.

[LFE04] F. Lamiraux, E. Ferre, and Vallee E. Kinodynamic motion planning : Connecting exploration trees using trajectory optimization methods. In *IEEE International Conference on Robotics and Automation*, New Orleans (US), April 2004.

[LFG+05] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun. Anytime dynamic A*: An anytime, replanning algorithm. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, June 2005.

[LK01a] S. Lavalle and J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

[LK01b] Steve Lavalle and James Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

[LL92] A. Lazanas and J.-C. Latombe. Landmark-based robot navigation. 1992.

[LMT84] T. LozanoPerez, M.T. Mason, and R.H. Taylor. Automatic synthesis of fine motion strategies for robots. *Int. Journal of Robotics Research*, 3(1):3–24, 1984.

[LP81] T. Lozano-Perez. Automatic path planning of manipulator transfer movements. 11(10):681–698, 1981.

[LP91] R.C. Luo and T.J. Pan. On dynamic motion planning problems. In *IEEE Int. Conf. on Robotics and Automation*, pages 1071–1078, 1991.

[LPV+06] Ch. Laugier, S. Petti, D. Vasquez, M. Yguel, Th. Fraichard, and O. Aycard. *Autonomous Navigation in Dynamic Environments: Models and Algorithms*, chapter Autonomous Navigation in Dynamic Environments: Models and Algorithms. Springer, 2006.

[LS85] D. Leven and M. Sharir. Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagrams. Technical Report 34/85, The Eskenasy Inst., Tel-Aviv Univ. (Israël), 1985.

[LSSL02] F. Large, S. Sekhavat, Z. Shiller, and Ch. Laugier. Towards real-time global motion planning in a dynamic environment using the NLVO concept. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, October 2002.

[MM00] J. Minguez and L. Montano. Nearness diagram navigation (ND): A new real time collision avoidance approach for holonomic and no holonomic mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, November 2000.

[MMK02] J. Minguez, L. Montano, and O. Khatib. The ego-dynamic space (EDS): Dynamics of the vehicle. In *In Proceedings of the Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, 2002.

[MMSA01] J. Minguez, L. Montano, Th. Simeon, and R. Alami. Global nearness diagram navigation (GND). In *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001.

[MMSV02] J. Minguez, L. Montano, and J. Santos-Victor. Reactive navigation for nonholonomic robots using the ego kinematic space. In *Proceedings IEEE International Conference on Robotics and Automation*, Washington (US), May 2002.

[Mor88] H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988.

[Mor04] P. Morin. Feedback control of nonholonomic mobile robots. In *5th Summer School on Image and Robotics*, 2004.

[Nil84] N. J. Nilsson. Shakey the robot. Technical note 323, AI Center, SRI International, Menlo Park, CA (US), April 1984.

[O'D87] C. O'Dunlaing. Motion planning with inertial constraints. *Algorithmica*, 2(4):431–475, 1987.

[OM05] Eduardo Owen and Luis Montano. Motion planning in dynamic environments using the velocity space. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 997–1002., Edmonton, Alberta, Canada, August 2005.

[OS96] M.H. Overmars and P. Svestka. The probabilistic path planner: A general approach to robot motion planning. In Domek S. Banka, S. and Z. Emirsajlow, editors, *Methods and Models in Automation and Robotics*, pages 909–916, Miedzyzdroje, Poland., 1996.

[OY82] C. O'Dunlaing and C. K. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6:104–11, 1982.

[Par97] M. Parent. Automated public vehicles : A first step towards the automated highway. In *4th World Congress on Intelligent Transport Systems*, October 1997.

[PF05a] S. Petti and S. Fraichard. Reactive planning under uncertainty among moving obstacles. In *Int. Symp. on Robotics*, Tokyo (JP), November 2005.

[PF05b] S. Petti and Th. Fraichard. Safe motion planning in dynamic environments. In *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Edmonton, AB (CA), August 2005.

[PKB03] J.M. Phillips, L.E. Kavraki, and N. Bedrossian. Spacecraft rendez-vous and docking with real-time, randomized optimization. *AIAA Guidance, Navigation, and Control*, 2003.

[PNIV01] L. Podsedkowski, J. Nowakowski, M. Idzikowski, and I. Vizvary. A new solution for path planning in partially known or unknown environment for nonholonomic mobile robots. *Robotics and Autonomous Systems*, (34):145–152, 2001.

[PPSB04] S. Pancanti, L. Pallottino, D. Salvadorini, and A. Bicchi. Motion planning through symbols and lattices. In *IEEE Int. Conf. Robot. & Autom.*, pages 3914–3919, 2004.

[Pri99] A.R. Pritchett. Pilot performance at collision avoidance during closely spaced parallel approaches. *Air Traffic Control Quartely*, 7(1):47–75, 1999.

[QK92] S. Quinlan and O. Khatib. Towards real-time execution of motion tasks. In R. Chatila and G. Hirzinger, editors, *Experimental Robotics 2*. Springer-Verlag, Berlin Heidelberg (1992), 1992.

[QK93] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 802–807, 1993.

[QLV02] A. Qayyum, A. Laouiti, and L. Viennot. Multipoint relaying technique for flooding broadcast messages in mobile wireless networks. In *HICSS: Hawai Int. Conference on System Sciences*, 2002.

[Qui94] S. Quinlan. Real-time modication of collisionfree paths. Master's thesis, 1994.

[Rei79] J.H. Reif. Complexity of the mover's problem and generalizations. In *20th IEEE Symposium on the Foundations of Computer Science*, pages 421–427, 1979.

[RK88] E. Rimon and D.E. Koditschek. Exact robot navigation using cost functions: the case of distinct spherical boundaries in E. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 1791–1796, Philadelphia, PA, USA, 1988.

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

[RS90] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. 145(2):367–393, 1990.

[RW98] J. H. Reif and H Wang. The complexity of the two dimensional curvature-constrained shortest-path problem. pages 49–57, 1998.

[Sam93] C. Samson. Time-varying feedback stabilization of car-like wheeled mobile robots. *Int. Journal of Robotic Research*, 12(1):55–64, 1993.

[SBA05] P. Rives S. Benhimane, E. Malis and J. R. Azinheira. Vision-based control for car platooning using homography decomposition. In *IEEE International Conference on Robotics and Automation*, pages 2173–2178, Barcelona, Spain, April 2005.

[SD85] Z. Shiller and S. Dubowsky. On the optimal control of robotic manipulators with actuator and end effector constraints. In *IEEE Int. Conf. on Robotics and Automation*, pages 614–620, St Louis, MI (USA), 1985.

[SD88] Z. Shiller and S.. Dubowsky. Global time optimal motions of robotic manipulators in the presence of obstacles. In *IEEE Int. Conf. On Robotics and Automation*, Philadelphia, 1988.

[SD89] Z. Shiller and S. Dubowsky. Robot path planning with obstacles, actuator, gripper and payload constraints. *Int. Journal of Robotics Research*, 8(6):3–18, 1989.

[SF96] A. Scheuer and T. Fraichard. Planning continuous-curvature paths for car-like robots. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1304– 1311, Osaka, Japan, November 1996.

[SH85] G. Sahar and J. Hollerbach. Planning of minimumtime trajectories for robot arms. In *IEEE Int. Conf. On Robotics and Automation*, Saint Louis, (MI) USA, 1985.

[Sim96] R. Simmons. The curvature velocity method for local obstacle avoidance. In *Proceedings of the International Conference on Robotics and Automation*, pages 3375–3382, Minneapolis (USA), april 1996.

[SLN00] T. Siméon, J.P. Laumond, and C. Nissoux. Visibility based probabilistic roadmaps for motion planning. *Advanced Robotics Journal*, 14(6), 2000.

[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, 1985.

[SO97] P. Svestka and M.H. Overmars. Motion planning for car-like robots, a probabilistic learning approach. *Int. Journal of Robotics Research*, 16:119–143, 1997.

[SS83] J. Schwartz and M. Sharir. On the piano movers problem ii. general techniques for computing topological properties of real algebraic manifolds. *Advanced Applied Mathematics*, (4):298–351, 1983.

[Ste94] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3310–3317, San Diego, CA, 1994.

[Ste95] A. Stentz. The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659, Montreal, Quebec, 1995.

[Ste02] A Stentz. Constrained D*. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 2002.

[Tay76] R. Taylor. *Synthesis of manipulator control programs from tasklevel specifications.* PhD thesis, Dept. of Computer Science, Stanford University, CA (US), 1976.

[TCD+01] C. Thorpe, O. Clatz, D. Duggins, J. Gowdy, R. MacLachlan, J. R. Miller, C. Mertz, M. Siegel, C.C. Wang, and T. Yata. Dependable perception for robots. In *Int. Advanced Robotics Programme IEEE*, Seoul, Korea, May 2001. Robotics and Automation Society.

[TFL94] H. Takeda, C. Facchinetti, and J.-C. Latombe. Planning the motions of a mobile robot in a sensory uncertainty field. 16(10):1002–1017, October 1994.

[Thr02] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium.* Morgan Kaufmann, 2002.

[TKA95] T. Tsubouchi, S. Kuramochi, and S. Arimoto. Iterated forecast and planning algorithm to steer and drive a mobile robot in the presence pf multiple moving objects. In *IEE/RSJ In t. Conf. on Intelligent Robot and Systems*, pages 33–38, 1995.

[TT03] R. Teo and C. Tomlin. Computing danger zones for provably safe closely spaced parallel approaches. *Journal of Guidance, Dynamics and Control*, 26(3):434–443, May 2003.

[UB98] L. Ulrich and J. Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1572–1577, Leuven, Belgium, May 1998.

[UB00] L. Ulrich and J. Borenstein. VFH*: Local obstacle avoidance with look-ahead verification. In *IEEE International Conference on Robotics and Automation*, pages 2505–2511, San Fransisco (US), April 2000.

[vdB07] Jur van den Berg. *Path Planning in Dynamic Environments.* PhD thesis, Utrecht University, The Netherlands., 2007.

[vdBFK06] Jur van den Berg, Dave Ferguson, and James Kuffner. Anytime path planning and replanning in dynamic environments. In *IEEE Int. Conf. in Robotics and Automation*, 2006.

[VF04] A. D. Vasquez and Th. Fraichard. Motion prediction for moving objects: a statistical approach. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3931–3936, New Orleans, LA (US), April 2004.

[Wan04] Chieh-Chih Wang. *Simultaneous Localization, Mapping and Moving Object Tracking.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2004.

[WAS99] S.A. Wilmarth, N. Amato, and P.F. Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *IEEE Int. Conf. on Robotics and Automation*, 1999.

[WBN93] T. S. Wikman, M. S. Branicky, and W. S. Newman. Reflexive collision avoidance: a generalized approach. In *Proceedings IEEE International Conference on Robotics and Automation*, volume 3, pages 31–36, Atlanta (US), May 1993.

[WD99] M.B. Wilson and S. Dickson. Poppet: A robust road boundary detection ans tracking algorithm. In *British machine vision conference 1999*, June 1999.

[WTT03] Chieh-Chih Wang, Charles Thorpe, and Sebastian Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan, September 2003.

NAVIGATION SÛRE EN ENVIRONNEMENT DYNAMIQUE : UNE APPROCHE PAR PLANIFICATION DE MOUVEMENT PARTIEL

Enjeux :

Depuis plus de dix ans aux USA, en Europe ou au Japon, nous observons un effort grandissant dans la recherche et le développement, dans le domaine du transport. En réponse aux problèmes de congestion, pollution et sûreté, soulevés par l'utilisation croissante des voitures en milieu urbain, de nouveaux modes de transport ont été proposés. Ils s'appuient sur l'utilisation de véhicules individuels circulant sur site urbain (projet Praxitèle en France, ICVS de Honda au Japon) ou sur site dédié (Parkshuttle aux Pays-Bas, Serpentine en Suisse, IMTS de Toyota au Japon). Dans le long terme, il est envisagé d'équiper ce type de véhicules afin de les rendre complètement autonomes. Cependant, le développement de voitures complètement autonomes ne peut être réalisé qu'en prenant en compte la nature particulière de l'environnement: nous sommes confrontés ici à des environnements peuplés d'obstacles mobiles (autres voitures ou piétons), capable de se mouvoir à des vitesses élevées et dont le mouvement future est à priori inconnu. A ce jour, le problème de navigation autonome dans ce type d'environnement reste largement ouvert.

Positionnement du sujet :

Notre travail aborde le problème de navigation autonome en milieu dynamique partiellement connu. Ce problème a généré de nombreux travaux durant les dix dernières années. Néanmoins, nous avons observé que certains aspect fondamentaux liés à la nature dynamique de l'environnement ont été oubliés, en commençant par la limite de temps qu'un système placé dans un environnement dynamique a, pour prendre une décision et exécuter son mouvement. Bien que cette contrainte soit d'importance cruciale, il y a paradoxalement peu de travaux dans la littérature la prennent en compte. Cette contrainte temps réel a été placée au cœur de notre travail.

Résultats :

Dans ce travail nous présentons une technique de planification qui prend en compte de manière explicite la contrainte temps réel imposée par l'environnement dynamique. La planification de mouvement partiel (PMP) est une technique originale de planification qui réagit à tout moment; dès que le temps imparti pour calculer une trajectoire est écoulé, la meilleure trajectoire calculée disponible est transmise. En fait, de notre point de vue, l'approche PMP est la meilleure approche possible au problème que nous avons observé, à savoir l'incompatibilité intrinsèque entre la planification en environnement dynamique et la contrainte temps réelle. PMP intègre également le problème de sûreté, que nous considérons d'un point de vue des états de collisions inévitables (ICS), concept récent qui englobe toutes les approches existantes et de proposer une technique qui donne de fortes garanties de sûreté pour notre système. Enfin, dans de ce travail, nous démontrons l'efficacité de notre approche PMP pour des environnements relativement complexes. Plusieurs expérience ont été mises en œuvre où PMP a ainsi été intégré sur une plate forme réelle, le robot type voiture Cycab, véhicule conçu par l'INRIA.

Transfert des résultats vers l'industrie :

Dans un contexte d'automatisation des véhicules à des fins d'assistance à la conduite ou de confort, la technique developpée dans ce travail pourrait tout à fait servir de base à des stratégies d'automatisation de certaines manœuvres de véhicules.

Mots clés :

Planification de trajectoire, Planification de mouvement partiel, Navigation autonome, environnement dynamique, Véhicule intelligent, Robotique,