UNIVERSITE BLIDA 01

Faculté de Technologie
Département d'Electronique

THESE DE DOCTORAT

Spécialité : électronique

CONTRIBUTION A LA PLANIFICATION DE MOUVEMENTS EN ENVIRONNEMENTS DYNAMIQUES POUR DES ROBOTS MOBILES DE TYPE VOITURE : CAS DU ROBUCAR

Par

Sara BOURAINE

Devant le jury composé de :

A. GUESSOUM	Professeur, U. de Blida	Président
R. TOUMI	Professeur, USTHB	Examinateur
F. BOUDJEMA	Professeur, ENP	Examinateur
N. BELKHAMZA	Maître de conférences A, U. de Blida	Examinateur
H. SALHI	Professeur, U. de Blida	Directeur de thèse
O. AZOUAOUI	Directeur de recherche, CDTA	Co- Directeur de thèse

Blida, 2016

RESUME

Ce travail traite le problème de la navigation pour un robot mobile avec un champ de vision limité et placé dans un environnement dynamique inconnu. Dans une telle situation, la sûreté de mouvement absolue (dans un sens où aucune collision n'aura jamais lieu peu importe ce qui se passe dans l'environnement) est impossible à garantir en général. C'est pourquoi, nous proposons un niveau plus faible de sûreté de mouvement, appelé *sûreté de mouvement passive* : elle garantit que, si une collision a lieu, le robot sera à l'arrêt. La première contribution principale de ce travail est le concept des *états de collision inévitable de freinage* (ICS de freinage). Les ICS de freinage sont définis tel que, quelle que soit la trajectoire de freinage future suivie par le robot, une collision se produit avant qu'il ne s'arrête. La sûreté de mouvement passive est obtenue en évitant les ICS de freinage à tout moment. Pour déterminer si un état est un ICS de freinage ou non, ICS^{b}-CHECK; un algorithme de vérification des ICS de freinage est développé. Il est intégré dans un système de navigation réactif appelé PASSAVOID pour valider le concept ICS de freinage et démontrer son utilité. Il est formellement établi que PASSAVOID est passivement sûr, dans un sens où le robot va toujours rester loin des ICS de freinage peu importe ce qui se passe dans l'environnement. La seconde principale contribution de ce travail est PASSPMP, un planificateur de mouvement partiel qui garantit la sûreté de mouvement passive. PASSPMP calcule périodiquement une trajectoire partielle passivement sûre, générée pour conduire le robot vers l'état but. Les résultats de simulation démontrent comment les algorithmes développés agissent et manipulent des champs de vision limités, des régions occultées et des obstacles mobiles avec un comportement futur inconnu.

Mots clés Robots mobiles – Environnements dynamiques – Navigation autonome – Sûreté de mouvement – Evitement d'obstacles – Etats de collision inévitable – Planification de mouvement.

ABSTRACT

This work addresses the problem of navigating in a provably safe manner a mobile robot with a limited field-of-view placed in an unknown dynamic environment. In such situation, absolute motion safety (in the sense that no collision will ever take place whatever happens in the environment) is impossible to guarantee in general. It is therefore settled for a weaker level of motion safety dubbed *passive motion safety*: it guarantees that, if a collision takes place, the robot will be at rest. The first main contribution of this work is the concept of *Braking Inevitable Collision States* (braking ICS). Braking ICS are defined as states such that, whatever the future braking trajectory followed by the robot, a collision occurs before it is at rest. Passive motion safety is obtained by avoiding Braking ICS at all times. To determine whether a given state is a Braking ICS or not, ICS^{b}-CHECK; an efficient Braking ICS-Checking algorithm is designed. It is integrated in a reactive navigation scheme called PASSAVOID to validate the Braking ICS concept and demonstrate its usefulness. It is formally established that PASSAVOID is *provably passively safe* in the sense that it is guaranteed that the robot will always stay away from Braking ICS no matter what happens in the environment. The second main contribution of this work is PASSPMP, a partial motion planner enforcing passive motion safety. PASSPMP periodically computes a passively safe partial trajectory designed to drive the robot towards its goal state. Simulation results demonstrate how the developed algorithms operate and handle limited sensory field-of-views, occlusions and moving obstacles with unknown future behaviors.

Keywords Mobile robots – Dynamic environments – Autonomous navigation – Motion safety – Collision avoidance – Inevitable collision states – Motion Planning.

ملخص

يتناول هذا العمل مشكلة الملاحة بطريقة آمنة لروبوت متحرك مع مجال رؤية محدود وضع في بيئة ديناميكية غير معروفة. في مثل هذه الحالة، ضمان السلامة المطلقة للحركة (بمعنى أنه لن ينتج اي اصطدام بغض النظر عما يحدث في البيئة) مستحيل بشكل عام. ولذلك، فإننا نقترح مستوى أضعف من سلامة الحركة يطلق عليه اسم السلامة الكامنة للحركة : إنها تضمن أنه، في حالة حدوث اصطدام، الروبوت سيكون متوقفا. المساهمة الرئيسية الأولى لهذا العمل هي مفهوم مواضع الاصطدام المحتوم للفرملة. يتم تعيين مواضع الاصطدام المحتوم للفرملة بحيث، أيا كان مسار الفرملة المستقبلي الذي يتبعه الروبوت ، يحدث الاصطدام قبل أن يتوقف. تتحقق السلامة المطلقة للحركة عن طريق تجنب مواضع الاصطدام المحتوم للفرملة في كل وقت. لتحديد ما إذا كان موضعا ما، هو موضع اصطدام محتوم للفرملة أم لا، تم تصميم برنامج فعال للتحقق من مواضع الاصطدام المحتوم للفرملة. يتم دمج هذا البرنامج في نظام ملاحة تفاعلي للتحقق من صحة مفهوم مواضع الاصطدام المحتوم للفرملة وإثبات فائدته. و قد بُرهن ان هذا النظام يضمن السلامة الكامنة للحركة بمعنى أن الروبوت سوف يبقى دائما بعيدا عن مواضع الاصطدام المحتوم للفرملة بغض النظر عما يحدث في البيئة. المساهمة الرئيسية الثانية لهذا العمل هي مخطِط الحركة الجزئية الذي يضمن السلامة الكامنة للحركة. هذا المخطِط يحسب دوريا مسار جزئي آمن مصمم لقيادة الروبوت نحو هدفه. وتُظهر نتائج المحاكاة كيفية عمل البرامج المصممة و كذا كيفية تعاملها مع مجالات رؤية محدودة، الانسداد و العقبات المتحركة بسلوك مجهول في المستقبل.

الكلمات الرئيسية الروبوتات المتحركة - البيئات الديناميكية - الملاحة المستقلة - سلامة الحركة - تجنب الاصطدام - مواضع الاصطدام المحتوم - تخطيط الحركة.

REMERCIEMENTS

Bien que le travail de thèse ne puisse être achevé que grâce aux efforts et à la persévérance du thésard, c'est loin d'être suffisant. En effet, ce travail n'aurait pas pu être réalisé sans le soutien et l'aide de nombreuses personnes.

Tout d'abord, j'adresse ma reconnaissance à mon directeur de thèse Monsieur H. Salhi pour m'avoir encadré, pour ses encouragements, son soutien et sa gentillesse.

J'exprime ma gratitude à ma co- directrice de thèse Madame O. Azouaoui de m'avoir épaulé, d'avoir mis à ma disposition les moyens possibles, ainsi que pour ses directives, ses conseils et sa confiance.

J'exprime ma reconnaissance envers Mr Th. Fraichard pour sa grande contribution dans mon encadrement, pour le temps qu'il m'a consacré durant mon séjour à l'INRIA et durant les séances de travail à distance. Je lui suis redevable quant aux connaissances que j'ai acquises dans le domaine de la robotique mobile particulièrement ou la recherche scientifique en général.

Je tiens à exprimer mes remerciements pour l'ensemble des membres du jury pour avoir accepté d'examiner mon travail et pour m'avoir fait l'honneur de l'évaluer.

J'ai à cœur de remercier également l'équipe e-Motion, INRIA pour leur convivialité et leur esprit d'équipe.

Un très grand merci à mon époux Riad pour son soutien, sa générosité et pour avoir toujours été à l'écoute, à ma fille Malak pour le temps que je lui ai subtilisé et pour sa gentillesse, et par-dessus tout, à mes parents pour tous leurs sacrifices, que sans eux je n'aurai pas pu arriver là où j'en suis. Je remercie également toute ma famille, mes amis et toute personne ayant pu m'encourager.

TABLE DES MATIERES

LISTE DES ILLUSTRATIONS GRAPHIQUES ET TABLEAUX

INTRODUCTION

De tout temps, l'homme n'a cessé de rêver de concevoir des machines ayant une capacité de décision qui approche le raisonnement humain et qui visent à accomplir des tâches spécifiques, soit pour assister l'être humain ou pour le remplacer. Les films de science-fiction illustrent bien l'étendue des défis à relever, où les robots sont partout ; à la maison, au travail, dans le transport, dans les espaces de loisir, etc. Désormais, cela ne relève plus de l'imagination ; au cours des dernières décennies, différentes recherches en robotique ont été élaborées pour réaliser des systèmes qui s'étendent dans le monde humain. Plusieurs secteurs d'application ont été explorés comme l'industrie manufacturière, le secteur médical (chirurgie de précision, robots infirmiers), le secteur militaire (sauvetage, surveillance, robots démineurs, drônes), l'exploration spatiale et sous-marine, le transport, etc. Ainsi, afin de mener à bien sa tâche, un robot doit être capable de naviguer vers son but sans rentrer en aucun cas en collision avec les obstacles qui l'entourent, et en utilisant uniquement l'information issue de ses capteurs embarqués. Ce problème a été un défi pour les chercheurs, mais aujourd'hui l'avancement rapide vers le développement de nouvelles techniques de perception, contrôle et planification de mouvement montre la capacité de ces dernières à résoudre divers problèmes.

L'une de ces problématiques est les systèmes de transport intelligents où les voitures sans conducteur sont devenues une réalité. Le succès du *DARPA challenge*[1] (Defence Advanced Research Project Agency) est un exemple concret d'une telle capacité, démontrant que ce type de véhicules peut parcourir des distances importantes dans des environnements tous terrains, naviguer dans des pistes désertes et se déplacer à grandes vitesses. Lors des deux premières années du challenge (2004 et 2005), les véhicules devaient faire un circuit dans le désert de Mojave (USA). Bien que ces expériences aient montré que ces

[1] www.darpa.mil/grandchallenge.

véhicules pouvaient se déplacer de manière autonome dans de telles conditions, leur navigation dans des environnements plus dynamiques et souvent encombrés d'obstacles mobiles (humains) reste un problème ouvert. C'est pourquoi, pour le troisième challenge ; *Urban DARPA challenge 2007*, le niveau de difficulté a augmenté et le défi a été relevé pour des environnements urbains caractérisés par une circulation routière. A cet effet, les véhicules devaient parcourir 97km en moins de six heures à travers un environnement urbain, en intégrant la sécurité routière, tout en respectant les règles de la route et en évitant les obstacles qui peuvent se présenter devant le véhicule (figure i.a). Le DARPA challenge est l'un des plus grands exploits réalisés dans le domaine de la navigation des robots mobiles. D'autres travaux ont été réalisés par la suite en plaçant la barre de plus en plus haut ; en 2010, le *VisLab Intercontinental Autonomous Challenge*[2] eut lieu. C'est un défi comme jamais essayé auparavant, des véhicules électriques sans conducteur ont parcouru une distance de 13000km, en démarrant de Parme, Italie et sont arrivés à Shanghai, Chine après trois mois (figure i.b). Ces challenges sont très prometteurs quant aux progrès dans différents domaines de la robotique, en particulier celui du transport (figure i.c).

(a) (b) (c)

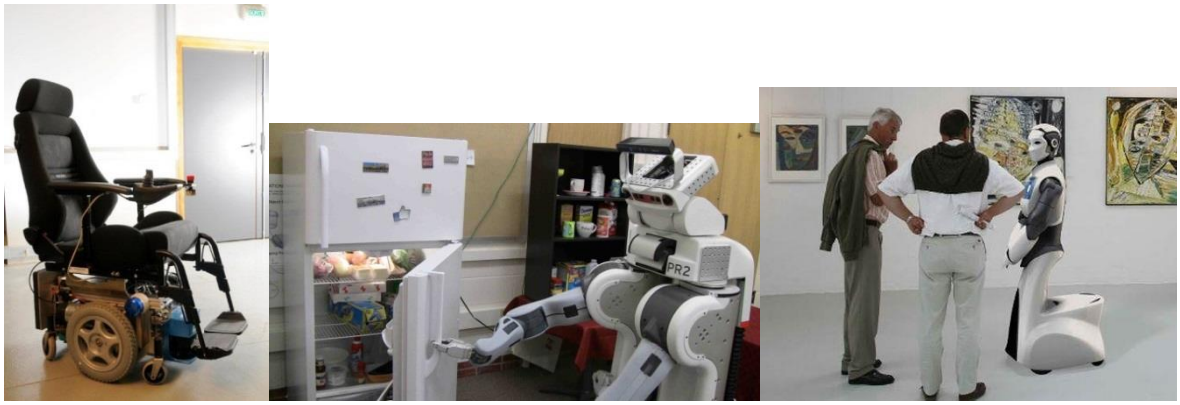
Figure i : Des véhicules sans conducteur. (a) Véhicule *Boss*, vainqueur du DARPA Urban Challenge 2007, (b) véhicules du VISLAB Intercontinental Autonomous Challenge (2010), (c) Cycab, Inria.

L'objectif de telles recherches qui visent à développer des voitures sans conducteur est d'aider à prévenir les accidents de la route, libérer le temps des gens (le conducteur peut faire des activités plus productives telles que travailler, lire, manger, etc.), améliorer la circulation, réduire les émissions de carbone (i.e.

[2] www.IntercontinentalChallenge.eu.

réduire la pollution) en changeant fondamentalement le concept d'utilisation de la voiture. Les voitures sans conducteur ont donc le potentiel de transformer l'industrie du transport en éliminant les accidents et palier ses effets indésirables.

Des exploits comme les challenges DARPA et VisLab démontrent la capacité d'un système robotique à parcourir des distances importantes, à grande vitesse dans des environnements complexes et réalistes. Cependant, de tels systèmes restent sujets à des accidents (voir [1]). Thrun [2], l'un des membres de l'équipe *Junior* ayant remporté le second prix dans le DARPA challenge (2007) et l'un des pionniers dans ce domaine, a rapporté que leurs voitures ne sont jamais sans conducteur ; qu'il y a toujours un conducteur formé derrière le volant pour prendre le contrôle en cas de problèmes, ainsi qu'un opérateur dans le siège passager pour contrôler la partie logicielle.



(a) La chaise roulante intelligente, INRIA. (b) PR2, Willow Garage. (c) Le robot REEM-H2, PAL Robotics.

Figure ii : Robots de service.

Un autre domaine d'application pour lequel le robot doit faire passer en premier le fait que ses mouvements ne soient pas dangereux par rapport à ce qui l'entoure est la robotique de service. Les robots de service comme leur nom l'indique sont généralement destinés à côtoyer l'être humain ; assister les handicapés (comme la chaise roulante intelligente (voir la figure ii.a) qui peut être utilisée aussi bien en environnement d'intérieur ou d'extérieur), accomplir les différentes tâches ménagères (comme le PR2 qui utilise le réfrigérateur dans la figure ii.b), les robots guides dans les musées par exemple (voir la figure ii.c), etc.

Ces robots sont de plus en plus utilisés vu les bienfaits qu'ils apportent à toute catégorie de gens, et le plus souvent ils sont amenés à se déplacer dans des environnements encombrés d'obstacles mobiles ; à savoir des personnes. Dans ces conditions, il est primordial de garantir qu'un robot tel que la chaise roulante ou le PR2 ne cause pas de dégâts ou qu'il rentre en collision avec une personne.

Problématique

Les roboticiens étaient conscients, depuis bien longtemps, du problème de concevoir un système de navigation sûr (i.e. la sûreté de mouvement est garantie); les systèmes d'évitement d'obstacles ont été largement traités dans la littérature, cependant, la notion de la sûreté de mouvement a été souvent mal définie, voire même jamais formellement analysée (voir [3]). Prouver qu'un robot est capable d'éviter les collisions sur un ensemble fini d'expériences n'est pas suffisant. Si les robots sont amenés à être déployés à grande échelle parmi les êtres humains, il est nécessaire de concevoir des systèmes d'évitement d'obstacles et de navigation pour lesquels la sûreté de mouvement doit être caractérisée voire même garantie. Des travaux de recherches récents [4, 5, 6, 7, 8, 9] montrent un intérêt croissant pour la résolution de ce problème. Cependant, la sûreté de mouvement dans le monde réel demeure un problème ouvert, où le terme monde réel implique que :

(1) L'environnement comporte des objets fixes et mobiles dont le comportement futur est inconnu.
(2) Le robot a uniquement une connaissance partielle de son entourage, en raison de ses limitations sensorielles.

Afin de mener à bien sa tâche dans de telles conditions, un robot mobile autonome doit être capable d'y naviguer tout en garantissant la sûreté de son mouvement et celle de son environnement. Les contraintes imposées à la navigation en environnement dynamique peuvent être résumées comme suit [10]:

Dans un environnement dynamique, il existe un temps limité pour prendre une décision, il faut raisonner sur l'évolution future de l'environnement et le faire avec un horizon temporel[3] approprié.

Le but de ce travail de thèse est précisément de traiter de tels problèmes ; à savoir la navigation de robots mobiles autonomes équipés de capteurs ayant un champ de vision limité dans des environnements inconnus peuplés d'objets mobiles dont le comportement futur est inconnu (voir la figure iii). En fait, la contribution principale de ce travail est d'étudier si dans de telles situations, il est possible d'obtenir des garanties strictes de la sûreté de mouvement, stricte dans un sens où elles peuvent être établies formellement.

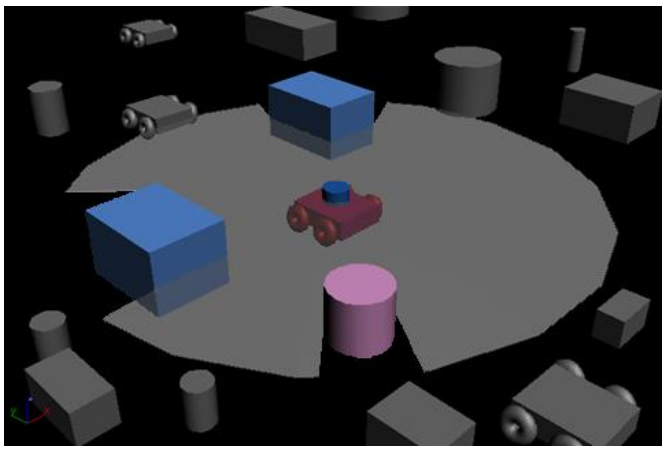


Figure iii : Un robot mobile ayant un champ de vision limité naviguant dans un environnement dynamique inconnu (l'environnement contient des objets fixes (ex. les cubes) et des objets mobiles (les cylindres et d'autres robots)). La région perçue est représentée en gris, et la région non perçue est représentée en noir.

En résumé, notre travail consiste donc à résoudre un certain nombre de problèmes :

- Concevoir un modèle du futur qui représente le comportement futur des objets de l'environnement en se basant sur l'information du système sensoriel du robot. Ce modèle doit prendre en considération aussi bien le comportement des objets perçus (se situant à l'intérieur du champ de

[3] L'horizon temporel représente jusqu'à quel point (ou la limite) dans le futur, le raisonnement est fait.

vision) que celui des objets non perçus (se plaçant sur la limite et en dehors de la limite du champ de vision). En effet, les objets non perçus peuvent également représenter un danger et un risque de collision avec le robot qui peut être inévitable.

- Répondre à la question : comment garantir la sûreté de mouvement où le robot doit toujours éviter les états de collision, en tenant compte de ses contraintes kinodynamiques, et de la dynamique de son environnement ainsi que son comportement futur ? Le système robotique ne suppose aucune connaissance a priori de l'environnement, et utilise uniquement ses capteurs embarqués pour la perception.
- Développer un système de navigation capable de conduire le robot vers son but tout en garantissant la sûreté du mouvement. Ce système doit prendre en considération les contraintes évoquées dans les deux points précédents, et il doit également impliquer le modèle du futur régulièrement mis à jour afin de mener à bien sa mission.

Contributions

Etant donné que la sûreté du mouvement concerne le fait d'éviter les états de collision (soit dans le présent ou plus tard dans le futur), ce travail traite le problème de la sûreté de mouvement selon le concept d'*état de collision inévitable* (Inevitable Collision State) (ICS) (développé dans [11]). Un ICS est un état pour lequel quel que soit la trajectoire future du robot, une collision finit par se produire. Les ICS sont définis avec une perspective de sûreté de mouvement absolue (absolue dans un sens où aucune collision ne va jamais se produire peu importe ce qui se passe dans l'environnement). En théorie, la sûreté de mouvement absolue nécessite une connaissance complète du futur jusqu'à l'infini. Nous pouvons alors affirmer, qu'en général, la sûreté absolue est impossible à garantir [12], à moins d'établir certaines hypothèses (qui restent contestables) concernant le robot et son environnement. Par exemple, imposer que la vitesse du robot soit un multiple de la vitesse maximale des objets [13], ou que les objets mobiles doivent apparaître au-delà d'une distance, qui est en fonction de leur nombre, tailles et vitesses [14]. Dans une situation telle que celle de la figure iii où le comportement futur de l'environnement est inconnu, la sûreté absolue est

impossible à garantir. Pour résoudre ce problème, la solution proposée est : *mieux vaut garantir moins que garantir rien*. Pour ce faire, nous introduisons dans cette thèse un niveau plus faible de sûreté de mouvement, qui garantit que si une collision a lieu, le robot sera à l'arrêt. Ce niveau de sûreté est appelé *sûreté de mouvement passive*.

La première principale contribution de notre travail est le concept *ICS de freinage* (Braking ICS), i.e. une version du concept ICS correspondant à la sûreté de mouvement passive. Les ICS de freinage sont définis comme des états pour lesquels, quel que soit la trajectoire de freinage future suivie par le robot, une collision se produit avant qu'il ne soit à l'arrêt. La sûreté de mouvement passive est obtenue en évitant les ICS de freinage à tout moment. Pour ce faire, l'algorithme ICS^{b}-CHECK a été développé ; il permet de vérifier si un état donné est passivement sûr (non ICS de freinage). Le concept ICS de freinage peut être implémenté pour n'importe quel système de navigation grâce à cet algorithme.

Notre deuxième principale contribution est PASSAVOID ; un schéma de navigation réactif dont la sûreté passive est garantie. Il est appliqué pour un robot mobile ayant un champ de vision limité et placé dans un environnement dynamique inconnu. C'est un système qui agit durant un pas de temps donné, et son but est de calculer le contrôle à appliquer au robot pour le pas de temps suivant en se basant sur le concept ICS de freinage. Il est formellement établi que PASSAVOID est passivement sûr, dans le sens où le robot va toujours éviter les ICS de freinage peu importe ce qui se produit dans l'environnement.

La troisième principale contribution de ce travail de thèse est de développer un système de planification de mouvement qui remplit les contraintes imposées par les environnements dynamiques (temps de décision limité), et tente de combler les lacunes des approches réactives (qui calculent le contrôle à appliquer au pas de temps suivant et par conséquent manquent de convergence vers le but) et des approches délibératives (qui cherchent à calculer un chemin complet vers le but, mais en revanche nécessitent le plus souvent une connaissance a priori de l'environnement et sont peu adaptées pour des environnements dynamiques). Ce système opère selon le principe de la planification de mouvement partiel (PMP) introduit dans [15], et il est appelé PASSPMP. A chaque cycle, PASSPMP étend un

arbre de recherche dont la racine est l'état actuel du robot et lorsque le temps disponible est écoulé, PASSPMP extrait de l'arbre le meilleur mouvement partiel calculé jusqu'ici. Le processus est répété jusqu'à ce que le but soit atteint. L'expansion de l'arbre est réalisée grâce à l'approche p-safe RRT, inspirée de l'approche Rapidly exploring Random Tree (RRT) [16]. Comme pour PASSAvoid, la sûreté passive de PASSPMP est formellement prouvée. Cependant, les améliorations apportées dans PASSPMP résident dans le processus de planification lui-même et la qualité de la trajectoire planifiée, et dans la convergence vers le but.

Plan du document

Le document est organisé comme suit. Un état de l'art sur les méthodes de navigation existantes est présenté dans le chapitre 1. Ces méthodes sont évaluées par rapport à certains critères imposés pour notre travail, et ainsi le situer par rapport à la communauté de recherche. Le chapitre 2 discute le problème de la sûreté de mouvement dans un environnement dynamique inconnu, et propose les solutions pour notre cas. Le modèle du futur adopté dans ce travail est également étudié. Le cœur de notre travail est présenté dans le chapitre 3. Le concept ICS de freinage est introduit et formellement démontré, ainsi que ses différentes propriétés. L'algorithme ICS^b-CHECK y est illustré, il vérifie si un état est ICS de freinage ou non. Dans le chapitre 4, l'approche PASSAVOID développée pour valider le concept ICS de freinage, est présentée en premier. C'est une approche d'évitement d'obstacles qui garantit la sûreté de mouvement passive, mais sans se préoccuper de la destination à atteindre. Par la suite, le problème de planification de mouvement est discuté, où l'approche développée PASSPMP est illustrée en détail. Les résultats de simulation de l'implémentation des approches développées dans ce travail sont présentés dans le chapitre 5. Enfin, pour conclure, un résumé des résultats obtenus et les perspectives envisagées sont discutés à la fin du document.

CHAPITRE 1

NAVIGATION D'UN ROBOT MOBILE DANS UN ENVIRONNEMENT DYNAMIQUE: TRAVAUX CONNEXES.

Parmi les travaux de recherche sur la robotique mobile, un grand nombre d'approches a été proposé pour résoudre le problème de la navigation. Une méthode de navigation consiste à générer et exécuter une stratégie de mouvement qui conduit le robot vers un but prédéfini, tout en évitant les obstacles présents dans l'environnement. Ce problème peut être décomposé en plusieurs sous-problèmes ; la localisation, la modélisation de l'environnement, le contrôle du mouvement, l'évitement d'obstacles, la planification de mouvement, etc. Le défi ne réside pas uniquement dans ces sous problèmes, mais également dans le fait qu'ils doivent agir ensembles pour atteindre le résultat voulu. Dans ce travail de thèse, nous nous intéressons aux deux derniers problèmes. Notre objectif consiste à développer un système de navigation capable de conduire le robot vers son but dans un environnement dynamique inconnu, en utilisant uniquement les données sensorielles, tout en évitant les collisions. En plus de la dynamicité de l'environnement, ce système doit prendre en considération le comportement futur de tous les obstacles mobiles de l'environnement ; qu'ils soient perçus ou non perçus, ainsi que les contraintes kinodynamiques du robot. Ce système ne se contente pas d'éviter les états de collisions du robot, mais il garantit également la sûreté du mouvement. Ce chapitre présente un état de l'art sur les travaux les plus pertinents dans ce domaine ; à savoir ceux ayant pu nous inspirer ou nous ont servi comme outils mathématiques ou méthodologiques. Relativement à notre travail, nous évaluons ces méthodes par rapport à plusieurs critères, entre autres leur applicabilité à des environnements

statiques ou dynamiques, leur considération ou non des contraintes du robot, etc. et également par rapport au critère de la sûreté du mouvement.

1.1 Préliminaires

Avant d'étudier les différentes approches de navigation existantes, il convient de présenter certaines notions nécessaires pour l'élaboration de ces approches.

1.1.1 L'espace de configuration

Afin de créer des plans de mouvement pour un système robotique, il est primordial de préciser sa position, ainsi que la position de chaque point de son corps. Car, aucun point du robot ne doit toucher les obstacles de l'environnement, d'où l'introduction des définitions suivantes. La *configuration* d'un système robotique est une spécification complète de la position de chaque point de ce système. Elle est décrite par un certain nombre de paramètres. Pour un robot mobile, c'est sa position (x, y) et son orientation θ (avec $\theta \in [-\pi, \pi]$). L'*espace de configuration* noté *espace-C* (ou *C*) est l'espace de toutes les configurations possibles du système. La dimension de C correspond au nombre de degrés de liberté du système. Cette notion d'espace de configuration a été introduite par Lozano-Pérez [17, 18]. Généralement, l'espace de configuration du robot est construit en utilisant les différentes informations concernant la géométrie du robot et de l'environnement dans lequel il se déplace ; nommé *espace de travail* (*workspace*) noté WS. Le robot A peut être alors décrit comme un sous-ensemble de l'espace de travail occupé par A à la configuration q et ainsi il est désigné par $A(q)$. Quant aux différents obstacles de l'environnement $B_i, i = 1, n$ où n est le nombre d'obstacles; ils sont décrits par des sous-ensembles fermés de l'espace de travail.

Les obstacles présents dans l'espace de travail font que certaines configurations sont interdites. Par exemple, une configuration q est interdite si le robot occupant la position q entre en collision avec l'un des obstacles de l'espace de

travail, i.e. $CB_i = \{q \in C \backslash A(q) \cap B_i \neq \emptyset\}$. Cette région est appelée *obstacle-C* (ou CB)[19]. Par opposition, l'espace libre (C_{libre}) est l'ensemble de toutes les configurations libres de collision dans C (pour lesquelles le robot n'intersecte aucun obstacle). C_{libre} représente le complément de CB (où $CB = \bigcup_{i=1}^{n} B_i$), i.e. $C_{libre} = C \backslash CB$.

L'idée de base de l'espace de configuration est de représenter le robot par un point et d'agrandir les obstacles. Cette représentation transforme le problème de planification de mouvement pour un objet dimensionné en un problème de planification de mouvement pour un point. Cette transformation ne change pas le problème d'aucune manière, mais au contraire, il est souvent plus simple de traiter le mouvement d'un point que celui d'une forme géométrique complexe du robot. D'un point de vue mathématique, le problème peut être formulé plus précisément, en particulier quand d'autres contraintes sont considérées (ex. la vitesse linéaire du robot, son accélération, etc.). En supposant que le robot est planaire[1], C peut être calculé en utilisant la *différence de Minkowski*[2] entre le robot et les obstacles. Le robot est alors réduit à un point et les obstacles sont agrandis avec la forme du robot. En général, le robot mobile ne se déplace pas uniquement en faisant des translations mais aussi des rotations. Dans ce cas, il suffit de calculer des tranches de C pour diverses orientations fixes du robot, puis de les empiler ensemble [20]. Il est aussi possible de calculer C par d'autres outils, mais cela peut être plus complexe [21] (une étude poussée sur l'espace de configuration est disponible dans [22]).

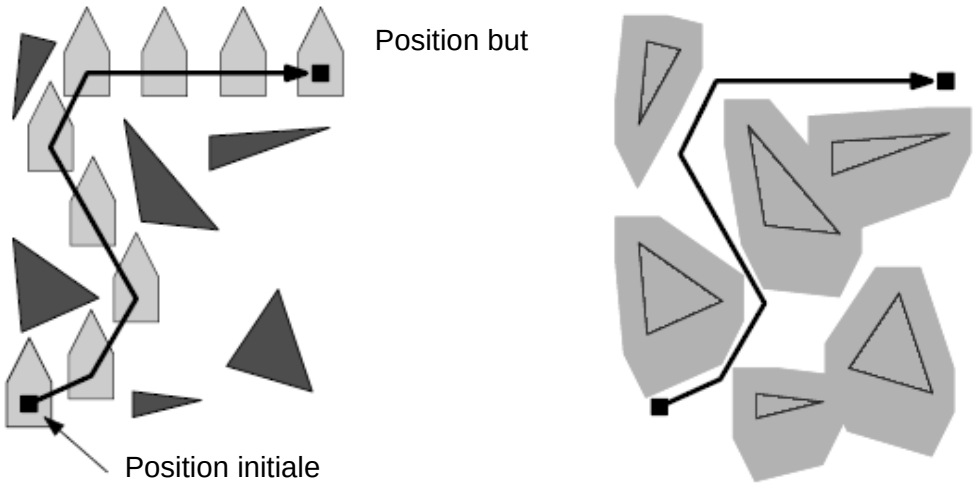
Dans l'espace de travail, un chemin à partir d'une configuration de départ q_{init} et une configuration but q_{but} est défini par une fonction continue $\pi : [0, L] \rightarrow C$, avec L, la longueur du chemin, $\pi(0) = q_{init}$ et $\pi(L) = q_{but}$. Un chemin libre de collision entre deux configurations libres q_{init} et q_{but} est une fonction continue $\pi : [0, L] \rightarrow C_{libre}$. Si un tel chemin n'existe pas, alors l'échec doit être rapporté. Un chemin semi-libre de collision est une fonction continue : $\pi : [0, L] \rightarrow cl(C_{libre})$, où $cl(C_{libre})$ désigne la clôture de C_{libre}. Ainsi, quand le robot se déplace le long d'un tel chemin, il touche

[1] Il peut uniquement se translater dans le plan.

[2] La différence de Minkowski de deux ensembles A et B dans WS est la différence entre chaque vecteur de A et chaque vecteur de B, i.e. $A \ominus B = \{a - b \backslash a \in A, b \in B\}$.

les obstacles. Par conséquent, un chemin libre ne permet pas le contact entre le robot et les obstacles de l'environnement alors qu'un chemin semi-libre le permet.

La Figure 1.1 montre l'exemple d'un robot planaire de forme polygonale qui peut se mouvoir en translation librement dans l'espace de travail. Dans la figure 1.1.a, l'espace de travail est représenté par un chemin libre de collision à partir de la position initiale jusqu'à la position but du robot. L'espace de configuration correspondant est illustré dans la figure 1.1.b, où la région interdite (CB) est représentée en gris et la région libre (C_{libre}) est représentée en blanc (l'empreinte des obstacles est représentée uniquement à titre indicatif). Le chemin (libre de collision) du robot est également indiqué.



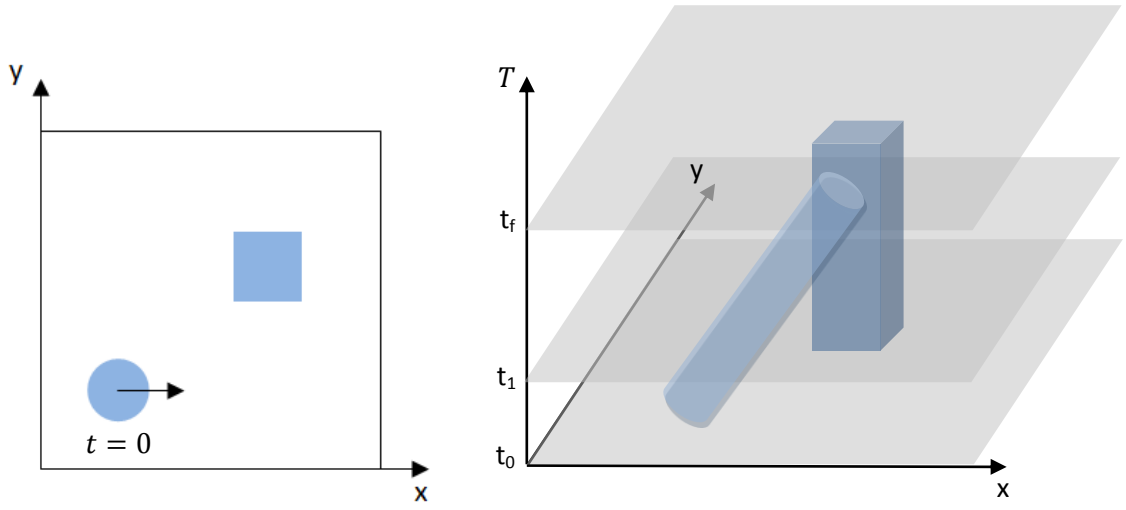
(a) (b)

Figure 1.1 : La translation d'un robot de forme polygonale dans un plan 2D encombré d'obstacles statiques. (a) L'espace de travail, le robot est représenté en gris clair et les obstacles sont représentés en gris foncé, (b) l'espace de configuration.

1.1.2 L'espace-temps des configurations

L'espace de configuration présenté dans le paragraphe précédent est dédié pour la résolution de problèmes de planification de mouvement dans des environnements statiques. Cependant, dans un environnement dynamique

caractérisé par des objets mobiles dont la position évolue dans le temps, une simple planification de chemin n'est plus suffisante. Le chemin du robot doit être alors paramétré par le facteur temps, c'est pourquoi il est plus judicieux d'utiliser la notion de *trajectoire*. Ainsi, la notion de temps est introduite dans l'espace de configuration comme dimension supplémentaire. L'espace résultant est appelé *espace-temps des configurations*, et il est noté CT. Soit $T = [0, t_{max}]$ l'intervalle de temps d'intérêt, avec 0 le temps initial et t_{max} le temps final (qui peut être fini ou infini). L'espace-temps des configurations est alors désigné par $C \times T$. Il se compose de paires (q, t), où q est un élément de C désignant la configuration du robot et t un scalaire de T désignant le temps. Dans de nombreux cas, la notion d'espace-temps des configurations est liée au temps réel du monde. Dans la figure 1.2.a, un espace de configuration contenant deux objets ; un objet fixe et un objet mobile est représenté. L'espace-temps des configurations correspondant est illustré dans la figure 1.2.b (où la dimension temps est ajoutée).



(a) (b)

Figure 1.2 : Un exemple illustratif (a) un espace de configuration avec un objet mobile (le disque avec le vecteur de vitesse) et un objet fixe (le carré), (b) l'espace-temps des configurations correspondant.

Une configuration q est interdite si elle entre en collision avec n'importe quel objet mobile ou statique à un instant donné t. Dans ce cas, la région d'obstacle est $CTB_i = \{(q,t) \in CT \backslash A(q) \cap B_i(t) \neq \emptyset\}$, où $B_i(t)$ représente l'obstacle à l'instant t [22]. La trajectoire est définie par une fonction continue : $\pi : T \rightarrow C$. Trouver une trajectoire libre de collision entre une configuration de départ q_{init} et une configuration but q_{but} revient en terme d'espace-temps des configurations à trouver une trajectoire π, tel que $\pi : [0, t_f] \rightarrow CT_{libre}$, où $\pi(0) = q_{init}$, $\pi(t_f) = q_{but}$ et t_f la durée de la trajectoire. Notons que dans ce cas, un modèle du futur est nécessaire pour déterminer la région d'obstacle.

1.1.3 L'espace d'états-temps

Bien que l'espace-temps des configurations permet de considérer les contraintes imposées par les obstacles mobiles (ces contraintes sont représentées par un ensemble de points interdits dans l'espace $q \times t$; à savoir CTB_i), il ne permet pas d'étudier les différents aspects de la planification de trajectoire dans un environnement dynamique. Par exemple, le robot ne peut pas se déplacer plus rapidement ou freiner pour éviter les obstacles. C'est pourquoi, il faut calculer une trajectoire qui prend en considération les contraintes dynamiques du robot (ex. vitesse, accélération, force, couple, inertie, etc.). L'espace considéré dans ce cas est appelé ***espace d'états***, noté S. C'est l'espace des paramètres de configuration et de leurs dérivées, il s'exprime par $(q, \dot{q})$. La région interdite est un ensemble de points de l'espace $q \times \dot{q}$ pour lesquels les contraintes du robot ne sont pas vérifiées (par exemple contraintes de vitesse). Cette région s'exprime alors par : $SV = \{(q, \dot{q}) \backslash \dot{q} < 0 \; ou \; \dot{q} > \dot{q}_{max}\}$, sachant que la vitesse $\dot{q}$ est contrainte par l'expression : $0 \leq \dot{q} \leq \dot{q}_{max}$, avec $\dot{q}_{max}$ la vitesse maximale admissible.

Pour prendre en compte à la fois les contraintes imposées par les obstacles mobiles et la dynamique du robot, une modélisation unifiée s'impose, d'où l'utilisation du concept ***espace d'états-temps***, noté ST, introduit dans [23]. ST est le résultat de la fusion des deux concepts espace-temps des configurations et espace d'états, i.e.

l'espace d'états-temps est l'espace d'état augmenté de la dimension temporelle, ainsi, il est exprimé par $(q, q^{\cdot}, t)$. Les contraintes imposées par les obstacles mobiles et la dynamique du robot peuvent être représentées par des régions interdites statiques de l'espace d'états-temps.

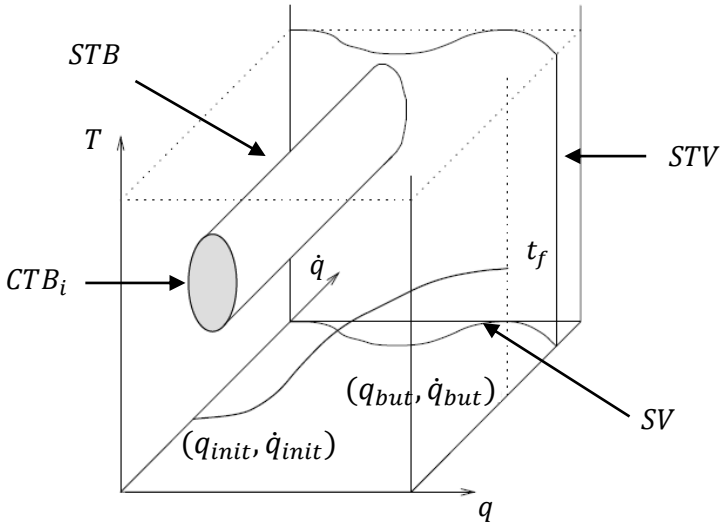


Figure 1.3 : L'espace d'état-temps de A (voir [24] pour plus de détail).

Un état (qui représente à la fois la configuration du robot à un instant donné ainsi que sa vitesse instantanée) est admissible s'il n'appartient pas à la région d'obstacle et il respecte les contraintes de vitesse. La région d'obstacle (noté STB) représente les états pour lesquels le robot entre en collision avec un obstacle de l'environnement, tel que, $STB = \{(q, \dot{q}, t) \backslash A(q) \cap B_i(t) \neq \emptyset\}$. De la même manière, nous définissons STV ; l'ensemble des états qui ne respectent pas les contraintes de vitesse, où, $STV = \{(q, \dot{q}, t) \backslash \dot{q} < 0 \; ou \; \dot{q} > \dot{q}_{max}\}$. Par conséquent, un état q est admissible si et seulement si $q \in ST \backslash (STB \cup STV)$, i.e. q appartient à l'espace d'états-temps admissible (l'ensemble de tous les états admissibles). Planifier une trajectoire π dans un environnement dynamique revient donc à calculer une courbe dans l'espace d'états-temps, i.e. une séquence d'états entre l'état initial $(q_{init}, \dot{q}_{init}, 0)$ et l'état but $(q_{but}, \dot{q}_{but}, t_f)$ du robot, qui évite les régions interdites ($\pi \in ST \backslash (STB \cup STV)$),

et son profil d'accélération respecte la condition $0 \leq \ddot{q} \leq \ddot{q}_{max}$ (avec $\ddot{q}_{max}$ l'accélération maximale). Une étude plus détaillée est à consulter dans [24].

Un exemple d'espace d'états-temps pour un robot A est illustré dans la figure 1.3. Une trajectoire sans collision entre $(q_{init}, \dot{q}_{init})$ et $(q_{but}, \dot{q}_{but})$ est représentée.

1.2 Approches de navigation

Plusieurs approches ont été proposées dans la littérature pour résoudre le problème de la navigation d'un robot mobile dans des environnements statiques ou dynamiques. Ce problème peut être traité à partir de plusieurs points de vue, mais généralement, nous distinguons deux grandes catégories d'approches : *les approches délibératives* qui représentent les approches de planification de mouvement et *les approches réactives* qui représentent les approches d'évitement d'obstacles.

1.2.1 Approches délibératives

Les approches délibératives ou globales consistent à résoudre un problème de planification de mouvement, connu également comme "le problème du déménageur de piano". Cela revient à calculer un chemin complet (une séquence de configurations) à partir d'une configuration initiale jusqu'à la configuration du but à atteindre, en se basant sur la représentation de l'environnement d'évolution (qui est généralement a priori connue ou construite pendant une première phase d'exploration). Le problème se résume donc à trouver un chemin sans collision entre une configuration initiale q_{init} et une configuration but q_{but} donnée. Ce problème est l'un des premiers problèmes traités en robotique mobile, mais qui reste toujours ouvert. Il existe une grande variété de méthodes qui visent à proposer des solutions diverses. Elles peuvent être principalement regroupées en deux catégories :

- Méthodes par graphes.
- Méthodes par arbres.

Un bref aperçu sur ces méthodes est présenté dans ce qui suit. Plus de détails sont à consulter dans [19, 20, 22, 25, 26, 27]. Bien que la majorité des approches de planification de mouvement ont été conçues pour des environnements statiques, certaines d'entre elles ont été étendues pour être appliquées dans des environnements dynamiques, ainsi, nous discutons également dans ce paragraphe leurs éventuelles extensions ou leurs dérivées.

1.2.1.1 Les méthodes par graphes

Le principe de base des méthodes par graphes est de capturer la connectivité de C_{libre} (l'espace libre du robot) en construisant un graphe qui représente un réseau de courbes ou lignes de dimension 1 (les nœuds du graphe correspondent à des lieux spécifiques et les arêtes à des chemins) [28, 29]. Cette représentation simplifie le problème et le réduit à une recherche dans un graphe. Ces méthodes procèdent en deux étapes ; la première étape revient à construire le graphe dans l'espace de recherche. Quand le système robotique n'a aucune connaissance a priori de son environnement, il utilise ses capteurs pour collecter les différentes informations et de manière incrémentale construire ce graphe. La deuxième étape consiste à parcourir ce graphe afin de déterminer un chemin qui relie la position initiale q_{init} et la position but q_{but}. Un algorithme de recherche graphique comme A* [30] ou Dijkstra [31] est utilisé pour trouver le chemin qui connecte ces deux positions. L'utilisation de ces algorithmes évite une exploration complète de l'espace de recherche.

La plupart des méthodes agissent de la sorte, ex. la décomposition cellulaire, les cartes de routes probabilistes.

La décomposition cellulaire. Le principe de cette approche consiste tout d'abord à diviser l'espace libre du robot en régions disjointes (généralement convexes) appelées *cellules*. Puis, *le graphe de connectivité* (appelé aussi *graphe d'adjacence*) qui représente la relation d'adjacence entre les cellules est construit où un nœud correspond à une cellule, et une arête connecte les nœuds correspondant

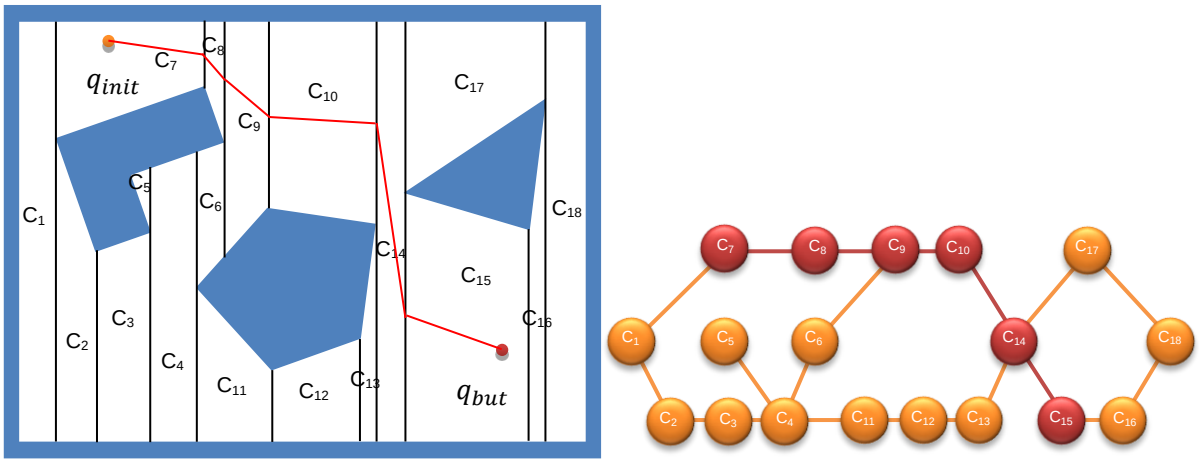
aux cellules adjacentes. La planification de chemin est réalisée en deux étapes ; déterminer les deux cellules contenant respectivement la configuration initiale et la configuration but, et chercher un chemin qui les relie dans le graphe de connectivité (voir la figure 1.4). Cette méthode garantit de trouver un chemin en un temps fini quand il existe, ou sinon elle rapporte qu'il n'existe aucun, ce qui fait d'elle une méthode complète.

Selon la façon de décomposer l'espace libre du robot, il existe deux types de décomposition cellulaire : *exacte* (ex. *la décomposition trapézoïdale* [32] (aussi appelée décomposition verticale [33]), *la décomposition par triangulation* [34], les méthodes des *courbes critiques* [35], *algébrique cylindrique* [36], *balles connectées* [37, 38] et *décomposition morse* [39]) et *approximative* [40, 41, 42]. La différence entre ces méthodes est que la méthode exacte décompose de manière exacte C_{libre} en un ensemble de cellules, où l'union des cellules correspond à C_{libre}. L'efficacité de calcul du planificateur dépend alors de la densité et la complexité des objets de l'environnement (le stockage et le traitement des cellules croient avec leur nombre). Par contre, la méthode approximative utilise une approximation conservative de C_{libre}. C_{libre} est représenté par des cellules de forme simple, comme des rectangles (le principe de la représentation par grille). Cette méthode est moins précise mais plus simple, mais comme elle utilise une représentation approximative de C_{libre}, elle peut échouer à trouver un chemin libre de collision même s'il existe. Une solution possible pour les deux cas est d'utiliser une représentation hiérarchique[3] (ex. utilisation de quadtree ou octree) [19].

Les méthodes de décomposition cellulaire sont dédiées pour des environnements statiques. Une adaptation potentielle pour un environnement dynamique nécessite l'introduction de la dimension temps, ce qui va augmenter la complexité du problème (à partir d'une certaine dimension, cette méthode ne peut être appliquée [19]). Il est aussi possible de re-planifier le chemin obstrué par les obstacles lors de la mise à jour de l'environnement en calculant un nouveau plan à partir du graphe de connectivité préalablement construit. Cependant, comme ce type

[3] Elle consiste à générer une décomposition initiale grossière, puis de l'affiner localement.

de représentations se limite à des espaces de faible dimension, une représentation dans un espace $C \times T$ n'est pas possible (i.e. uniquement une représentation dans C est possible) [19]. Le comportement futur des obstacles mobiles ne peut pas être alors considéré du moment que le raisonnement dans le futur est impossible.



(a) (b)

Figure 1.4 : Calcul du chemin entre q_{init} et q_{but} (représenté en rouge) par la méthode de décomposition cellulaire (exacte) : (a) la décomposition de c_{libre} en trapèzes et triangles, (b) le graphe de connectivité.

Cartes de routes probabilistes. Quand l'espace de recherche est de grande dimension, la taille du graphe devient encombrante et donc sa construction devient très coûteuse. Une solution pour diminuer la taille du graphe revient à construire une *carte de routes probabiliste* (Probabilistic Roadmap) (PRM) [43]. La carte de routes probabiliste est construite en générant des configurations aléatoires (nœuds) dans l'espace de recherche du robot. Un nœud est ajouté au graphe s'il est sans collision avec les obstacles de l'environnement, et il est relié à d'autres nœuds de la carte par des arêtes (en utilisant un planificateur local). Deux nœuds sont reliés s'ils sont suffisamment proches, et qu'il existe un chemin sans collision entre eux (voir la figure 1.5). PRM est aussi connue comme une méthode de planification multi-requêtes, car, elle pré-calcule une carte de routes puis elle l'utilise pour traiter de multiples requêtes [44].

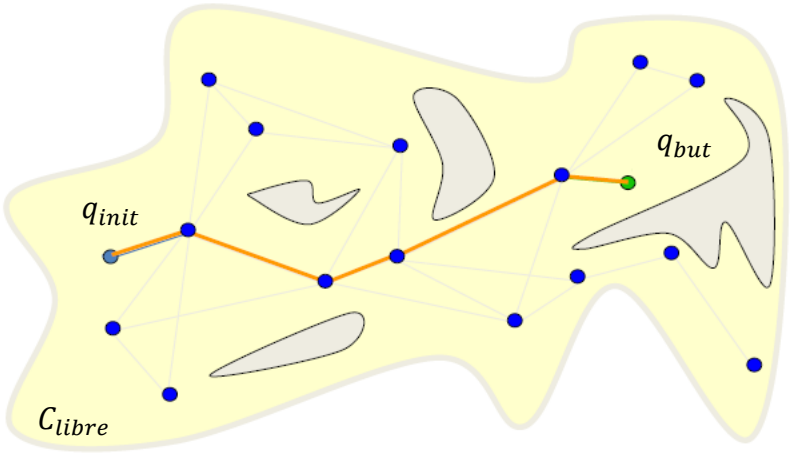


Figure 1.5 : La carte de routes probabiliste utilisée pour le calcul d'un chemin entre les deux configurations q_{init} et q_{but}.

Bien que, cette méthode a été développée à la base pour planifier des chemins géométriques sans collision dans un environnement statique, en raison de sa performance dans des espaces de configuration de grande dimension, elle est adaptée pour opérer dans des environnements dynamiques. Dans [45], la carte de routes probabiliste est en premier construite pour un environnement statique. Puis, elle est mise à jour en présence d'obstacles mobiles ; les nœuds et les arrêtes affectés par le mouvement des obstacles sont désactivés ou réactivés selon les situations. Ainsi, une reconstruction de zéro de la carte est évitée. Le planificateur génère une trajectoire sans collision dans l'espace d'états-temps qui obéit aux contraintes kinodynamiques du robot. Cependant, cette approche, en plus d'être hors-ligne, suppose également que le comportement des obstacles mobiles est a priori connu.

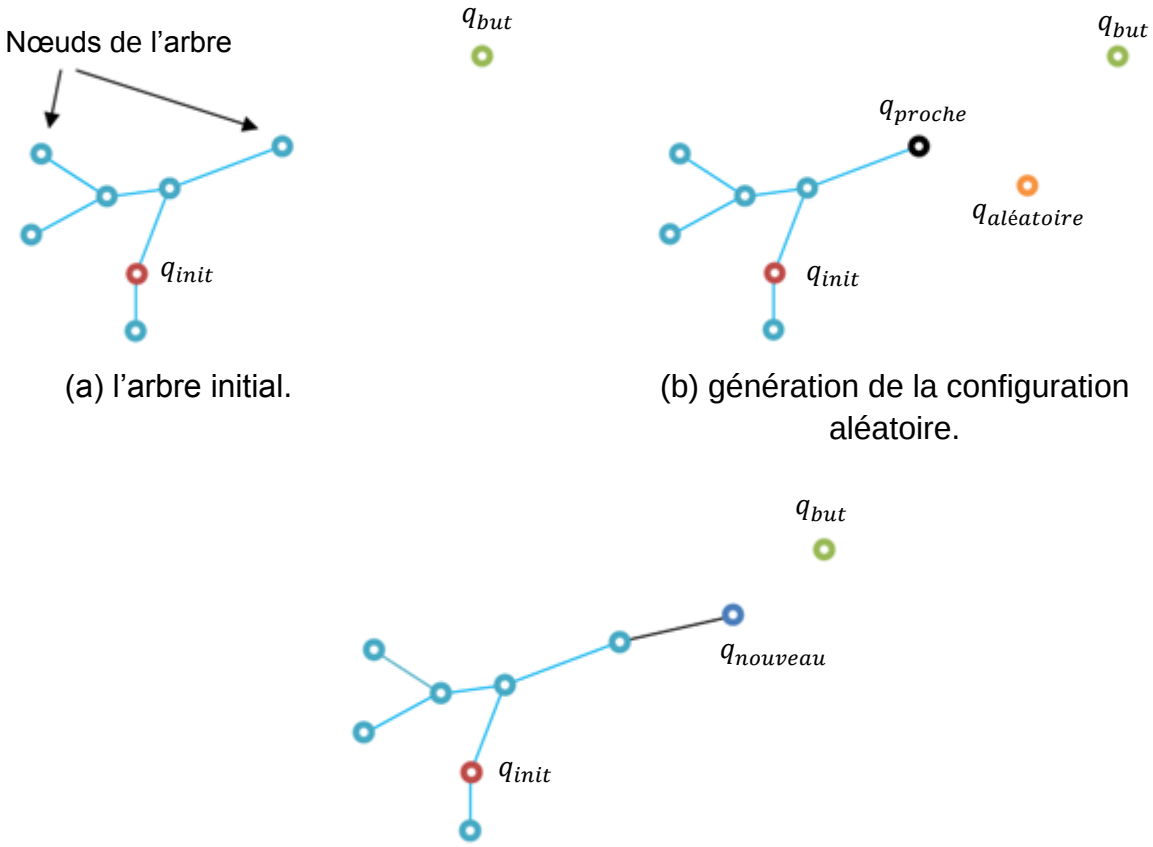
Ces planificateurs ont une caractéristique importante ; ils peuvent présenter une certaine forme de complétude qui est plus faible mais toujours intéressante : comme la plupart de ces planificateurs sont basés sur un échantillonnage aléatoire, cette forme de complétude est appelée *complétude probabiliste*, i.e. la probabilité de rapporter si une solution existe ou non tend vers un quand le nombre d'échantillons générés tend vers l'infini. Si les échantillons sont déterministes (par exemple

échantillonnage sur une grille), c'est alors *une complétude en résolution*, i.e. le planificateur rapporte si une solution existe ou non à une résolution donnée. Si aucune solution n'est trouvée, une autre peut exister à une résolution plus fine. PRM satisfait une complétude probabiliste.

1.2.1.2 Les méthodes par arbres

Au lieu d'explorer l'espace de recherche en utilisant un graphe, un arbre est étendu. Cet arbre est construit à partir de la configuration initiale du système, et il se développe en couvrant l'espace. Ce qui fait que ces méthodes sont très adaptées pour des espaces de recherche à forte dimensionnalité.

Rapidly-exploring Random Trees (RRT). RRT fait partie des méthodes les plus répandues à l'heure actuelle. C'est une technique de diffusion introduite dans [46, 16]. Elle est basée sur une construction incrémentale d'arbres aléatoires de manière à réduire rapidement la distance entre une configuration choisie aléatoirement et l'arbre. La construction est réalisée en démarrant d'un arbre initial (voir la figure 1.6.a). Puis, une configuration aléatoire $q_{aléatoire}$ (de l'espace de recherche) est choisie. À partir de l'arbre, la configuration q_{proche} la plus proche de cette configuration aléatoire est sélectionnée (voir la figure 1.6.b). Une nouvelle configuration $q_{nouveau}$ est alors calculée dans la direction liant la configuration aléatoire à la configuration voisine la plus proche dans l'arbre (voir la figure 1.6.c). Le processus est répété jusqu'à ce qu'un chemin qui lie q_{init} et q_{but} est trouvé. A l'inverse de la planification multi-requête, RRT est une méthode de planification unique-requête [47]. Ce type de méthodes calcule une nouvelle carte de routes pour chaque requête. Comme PRM, RRT satisfait une complétude probabiliste car elle est basée sur un échantillonnage probabiliste de l'espace de recherche.



(a) l'arbre initial.

(b) génération de la configuration aléatoire.

(c) insertion d'une nouvelle configuration.

Figure 1.6 : Principe de la diffusion de l'arbre dans la technique RRT.

Dans les premières approches [46, 16], l'arbre est développé dans toutes les directions autour du robot dans l'espace de recherche, jusqu'à ce qu'il soit entièrement couvert. Un exemple d'une telle extension est donné dans la figure 1.7. Une telle extension garantit de trouver une trajectoire entre q_{init} et q_{but}, cependant,elle est très coûteuse en temps de calcul. Des travaux plus récents ont proposé des solutions ; par exemple de combiner une méthode multi-requêtes avec une méthode unique-requête, et ainsi intégrer les propriétés d'échantillonnage global des planificateurs multi-requêtes avec les propriétés d'échantillonnage local des planificateurs unique-requête. Dans [48, 49], la méthode *Probabilistic Roadmap of Trees* (PRT) combine la méthode RRT et la méthode PRM, car dans ce cas, le coût

de construction d'une carte de routes est inférieur à celui de construire un arbre. Notons que la méthode PRM peut être remplacée par n'importe quelle autre stratégie d'échantillonnage, c'est pourquoi ces méthodes sont plus connues comme *Sampling-Based Roadmap of Trees* (SRT) [49]. Dans [50], le problème est résolu en utilisant une fenêtre de recherche ; au lieu d'effectuer la recherche dans tout l'espace, elle est effectuée dans les limites de cette fenêtre. Cette technique présente plusieurs avantages ; en plus d'être très efficace, elle prend explicitement en considération les contraintes cinématiques et dynamiques du système robotique.

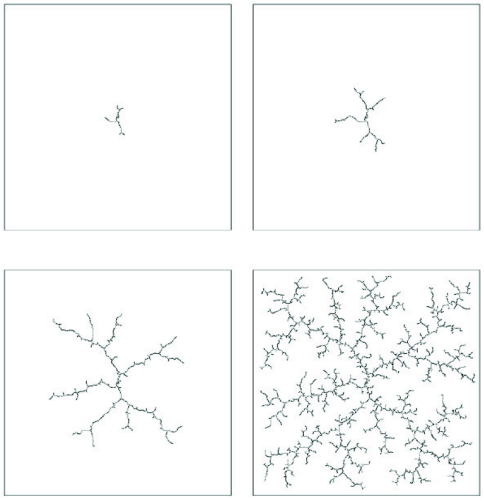


Figure 1.7 : Un exemple de la construction de l'arbre de recherche dans l'espace de configuration du robot en utilisant RRT [46].

Dans le cas où le système évolue dans un environnement dynamique, plusieurs extensions de RRT ont été proposées. Dans le travail de [51], RRT a été étendu à ERRT (*RRT étendu*), où la phase de planification et exécution se chevauchent, et ainsi les changements de l'environnement sont pris en considération. Une version "*anytime RRT*" est proposée dans [50] ; l'arbre est mis à jour progressivement et un contrôle est généré à chaque pas de temps pour conduire le robot à une nouvelle position de manière à se rapprocher de plus en plus vers le but. Une autre variante de RRT est la technique "*Multipartite RRT*" qui combine ERRT et anytime RRT [52]. Elle est adaptée pour des environnements inconnus contenant des

obstacles mobiles dont la trajectoire est prédéterminée. Il existe également d'autres extensions de RRT dans des environnements dynamiques (ex. [53, 54]). Toutefois, dans la majorité des cas, le comportement futur des obstacles mobiles n'est pas considéré, sinon, il est supposé a priori connu.

Il existe d'autres méthodes moins utilisées de la planification par expansion d'un arbre, comme ; *le fil d'Ariane* (en anglais *Ariadne's Clew*) [55, 56, 57], la planification expansive sur l'espace de configuration (en anglais Expansive-Spaces Tree planner (EST)) [58, 59]. Malheureusement, ces méthodes ne sont pas dédiées pour des environnements dynamiques.

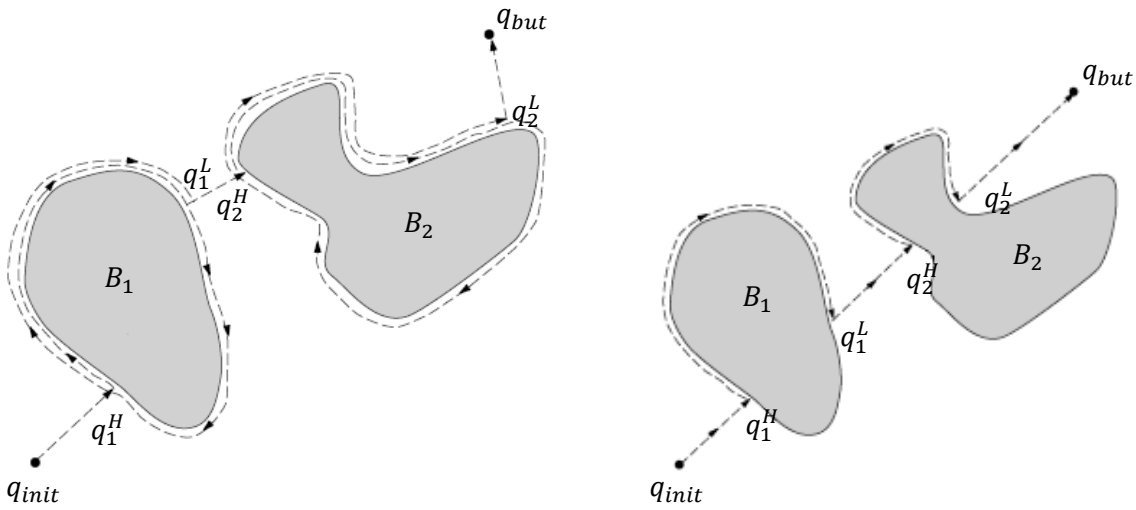
1.2.2 Les approches réactives

A l'inverse des approches délibératives qui calculent un chemin complet vers le but (généralement dans un environnement a priori connu ou construit dans une première phase d'exploration), les approches réactives (connues comme approches d'évitement d'obstacles) utilisent les informations des capteurs embarqués afin de calculer à chaque pas de temps le contrôle à appliquer sur les actionneurs. Ce qui fait d'elles des approches très adaptées pour une exécution en temps réel. Une présentation non exhaustive de ces méthodes est donnée dans les paragraphes suivants.

1.2.2.1 Les algorithmes Bug

Les algorithmes *Bug1* et *Bug2* font partie des méthodes les plus anciennes et les plus simples [60]. Ces algorithmes supposent que le robot est un point qui opère dans un plan 2D et qui utilise un capteur tactile pour détecter les obstacles. Ces algorithmes raisonnent selon deux comportements : "déplacement vers le but" et "suivi d'une limite". Pour l'algorithme Bug1, durant le comportement "déplacement vers le but", le robot se déplace tout droit vers le but (q_{but}). Quand il rencontre un obstacle en un point donné (q_1^H), le comportement "suivi d'une limite" est exploité, où

le robot fait le tour de l'obstacle jusqu'à ce qu'il retourne au même point. Dans ce périmètre, il détermine le point le plus proche vers le but (q_1^L) qu'il va atteindre. À partir de ce point, le robot reprend son chemin vers le but (en rappelant le comportement "déplacement vers le but"), et le processus est répété jusqu'à ce que le robot atteigne son but (voir la figure 1.8.a).



(a) (b)

Figure 1.8 : Calcul du chemin du robot par les algorithmes Bug : (a) Bug1, (b) Bug2. Plus de détails sont disponibles dans [20].

A l'inverse de Bug 1 qui utilise une recherche exhaustive, bug 2 utilise une recherche gloutonne (greedy), i.e. il opte pour la première solution prometteuse qu'il trouve. Durant le comportement "déplacement vers le but", comme pour Bug1, le robot se déplace selon des lignes droites, mais dans Bug2, elles sont toujours alignées de manière à connecter la position initiale du robot q_{init} et la position but q_{but}. Quand un obstacle est rencontré (en q_1^H), Bug2 fait appel au comportement "suivi d'une limite", mais celui-ci est différent du comportement de Bug1. Le robot fait le tour de l'obstacle, jusqu'à ce qu'il atteigne un nouveau point (q_1^L) qui appartient à la

ligne droite qui relie q_{init} et q_{but} et qui soit le plus proche de q_{but} (voir la figure 1.8.b). Le même processus est répété jusqu'à ce que le robot atteigne le but.

A première vue, il semble que l'algorithme Bug 2 est plus efficace que Bug1. Quand l'obstacle est simple, l'approche gloutonne de Bug 2 fournit effectivement une solution plus optimale. Cependant, quand l'obstacle est complexe, l'approche conservative de Bug 1 donne une meilleure performance. En revanche, ces algorithmes restent loin de fournir une solution optimale vu que le comportement du robot est restreint à un déplacement sur les contours de l'obstacle. Parmi les améliorations de l'algorithme Bug 2 pour remédier à ces limitations est *l'algorithme Bug Tangent* [61, 62], qui ajoute un capteur avec un champ de vision de 360° et une résolution infinie (au lieu d'un capteur tactile). Désormais, le robot dispose de plus d'information sur l'espace libre et peut donc mieux optimiser son chemin.

Bien que ces algorithmes sont simples, aucune contrainte sur le robot n'est considérée (qu'elle soit géométrique, cinématique ou dynamique). Mais leur plus grand inconvénient est que, la sûreté du mouvement ne peut être garantie car le robot doit se déplacer à proximité des obstacles. En plus ils se limitent à des espaces de configuration de dimension 2.

Un autre inconvénient de ces algorithmes est qu'ils sont destinés pour des environnements statiques. Récemment, de nouveaux algorithmes bug traitant des environnements contenant des obstacles mobiles ont été développés [63, 64], même si le comportement futur des obstacles n'est pas considéré. D'un point de vue sûreté de mouvement, l'algorithme a été également amélioré ; dans [63], le rayon du robot ainsi qu'une distance de sécurité ont été ajoutés au rayon de l'obstacle pour éviter que le robot passe trop près des obstacles.

1.2.2.2 Approche des champs de potentiel (PF)

L'approche des champs de potentiel a été initialement proposée par Khatib [65] pour le calcul du mouvement de bras manipulateurs. Cette approche construit dans l'espace de configuration une fonction de potentiel artificiel (qui peut être vue

comme une énergie) dont le gradient (qui est une force) définit un champ de vecteur pour conduire le robot vers le but. Cette approche traite le robot comme un point qui suit les champs de potentiel ; le but (un minimum dans l'espace) agit comme une force attractive alors que les obstacles agissent comme des forces répulsives (voir figure 1.9). La combinaison de ces forces guide le robot à partir de sa position initiale vers la position but, tout en évitant les obstacles a priori connus. Notons que les champs répulsifs doivent être très forts quand le robot est proche de l'obstacle, mais d'autre part, ils ne doivent pas influencer le mouvement du robot quand il est loin de l'obstacle.

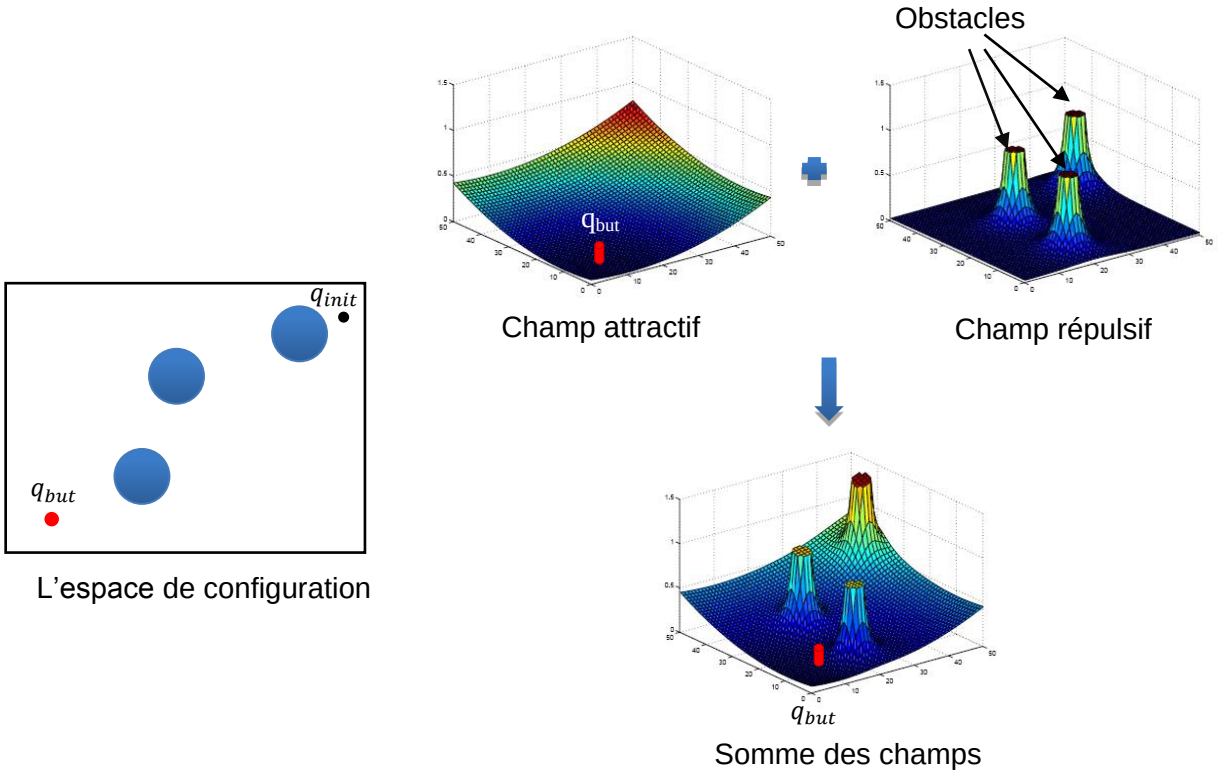


Figure 1.9 : Champs de potentiel pour un environnement contenant trois obstacles.

Un avantage important de cette approche est le fait qu'elle s'applique à des espaces de configuration multidimensionnels, et de ce fait, elle produit une plus grande variété de chemins. En contrepartie, elle ne prend en aucun cas en compte la cinématique et la dynamique du robot, qui par conséquent, aura du mal à suivre le chemin calculé. D'autre part, cette approche soufre du problème des minima locaux

(qui dépend de la forme et de la taille de l'obstacle), et donc la convergence vers le but ne peut pas être assurée. Cette approche est également sujette à d'autres problèmes qui sont mis en évidence par Koren et Borenstein [66]. De nombreuses variantes et améliorations de cette approche ont été proposées et mises en œuvre [67, 68, 69, 48, 70]. Dans la plupart des cas, ces variantes visent à améliorer le comportement des champs de potentiel quant au problème des minima locaux et de réduire les oscillations quand le robot doit se déplacer dans des espaces étroits (ex. couloirs, portes).

Malgré ces améliorations, toutes ces approches considèrent que l'environnement est statique. Cependant, des travaux récents ont étendu les champs de potentiel pour des environnements dynamiques ; dans [71,72], la vitesse des objets mobiles est prise en considération pour le calcul des champs répulsifs. D'autre part, dans [73], une nouvelle méthode appelée "algorithme d'évasion" a été développée. Elle prédit le mouvement des obstacles mobiles par le filtre de Kalman et le combine avec la méthode des champs de potentiel. Cette méthode introduit une force modificatrice en plus des forces attractives et répulsives, où la vitesse et la position des obstacles mobiles sont estimées par le filtre de Kalman et les résultats obtenus sont utilisés pour définir cette force modificatrice.

1.2.2.3 La fenêtre dynamique (DW)

L'approche de la fenêtre dynamique a été proposée par Burgardand et Thrun [74]. Elle opère dans l'espace des vitesses du robot, permettant de prendre en considération les contraintes cinématiques et dynamiques en limitant l'espace de recherche uniquement à un espace de vitesses atteignables. A partir d'une perception locale de l'environnement, DW vise à sélectionner un couple (v, w) représentant la vitesse linéaire et angulaire du robot qui permet d'éviter les obstacles perçus. Quand ce couple de contrôles est appliqué au robot, un chemin circulaire est généré, pour lequel les différentes contraintes (kinodynamiques) sont évaluées. A partir des différents couples possibles, DW sélectionne le couple de vitesses le plus pertinent. Ces vitesses sont appelées *vitesses admissibles*. La zone de recherche est

limitée par une fenêtre dynamique qui désigne les limitations dynamiques du robot et les vitesses admissibles qui peuvent être atteintes durant un intervalle de temps donné (voir la figure 1.10). Comme toutes les approches locales, DW soufre du problème des minima locaux. C'est pourquoi la méthode *DW globale* a été proposée dans [75] et ainsi le robot peut éviter de telles situations.

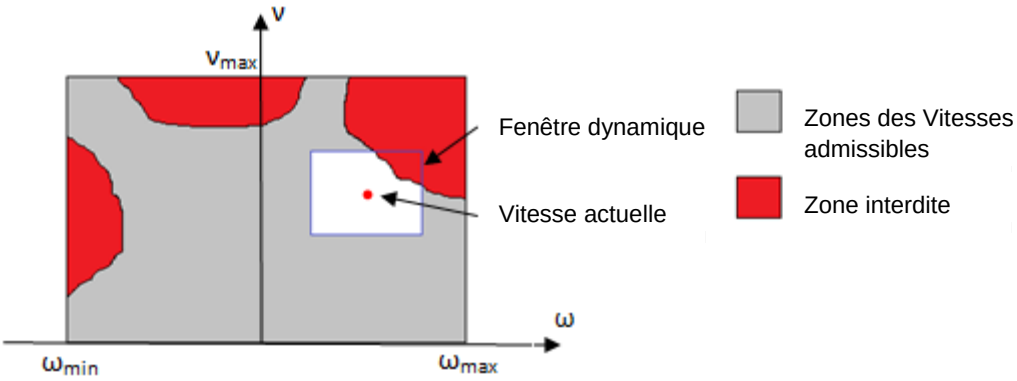


Figure 1.10 : L'approche de la fenêtre dynamique. L'espace des vitesses est divisé en régions admissibles et interdites. DW est représentée par le rectangle bleu contenant les vitesses atteignables par le robot durant un intervalle de temps spécifique.

Toutefois, DW et DW globale sont toutes les deux adaptées pour des environnements statiques ou à la rigueur changeants (par exemple un obstacle est ajouté à l'environnement ou le cas des portes). Dans [76], DW a été étendue à *la fenêtre dynamique variant dans le temps* (Time-Varying Dynamic Window) (TVDW) pour traiter les obstacles mobiles. A chaque instant, TVDW calcule un ensemble de trajectoires probablement suivies par les obstacles dans le futur pour effectuer une vérification de collision à court terme.

1.2.2.4 Représentation des obstacles dans l'espace des vitesses (VO)

L'approche "velocity obstacles" (VO) a été proposée par Fiorini et Shiller [77] afin de prendre en compte la dynamique de l'environnement et celle du robot. De

plus, le comportement futur des obstacles mobiles est considéré en supposant que l'obstacle maintient une vitesse linéaire constante. L'information de vitesse est directement utilisée pour déterminer une collision potentielle. VO représente l'ensemble des vitesses qui mènent le robot vers une collision avec les obstacles voisins. Les obstacles statiques et dynamiques sont modélisés dans l'espace des vitesses du robot, i.e. la VO d'un obstacle circulaire planaire B qui se déplace avec une vitesse constante est un cône dans l'espace des vitesses de A, où A est réduit à un point et B est élargi du rayon de A (comme illustré dans la figure 1.11). Chaque point de VO représente un vecteur de vitesse dont l'origine est le point correspondant à A. Toute vitesse de A qui pénètre dans VO est une vitesse qui va conduire le robot vers une collision avec B à un certain point dans le futur. Avec une telle représentation, une manœuvre d'évitement peut être facilement calculée en sélectionnant une vitesse à l'extérieur de VO.

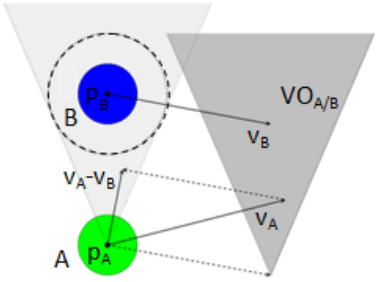
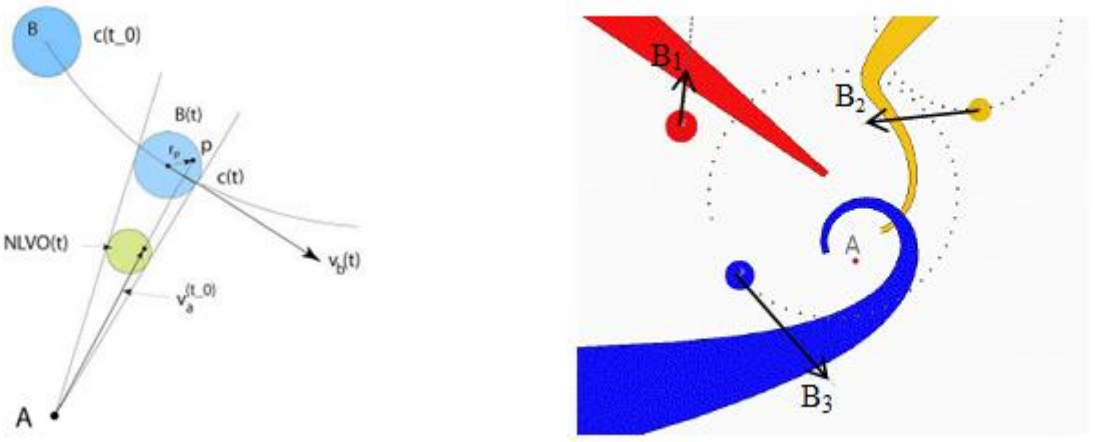


Figure 1.11 : L'approche VO : $VO_{A/B}$ pour A induit par B.

La méthode VO a certaines limitations ; dans un environnement fermé, chaque vitesse est interdite du moment qu'elle conduit le robot vers une collision. Cette méthode pose également l'hypothèse que le robot et l'obstacle doivent être de forme circulaire (disques). De plus, dans le cas où l'obstacle suit une trajectoire arbitraire (avec des vitesses variables), VO ne peut plus être appliquée. C'est pour remédier à ce dernier problème que la méthode VO non linéaire (NLVO) (non linear velocity obstacles) a été développée [78]. NLVO considère une trajectoire non linéaire de l'obstacle, mais suppose qu'elle est a priori connue. NLVO contient toutes les vitesses de A à un instant t_0 qui conduira le robot vers une collision avec B à

n'importe quel instant $t > t_0$. Sélectionner une vitesse à l'instant $t = t_0$ qui est en dehors de NLVO garantit que A peut éviter la collision à tout instant (voir la figure 1.12). NLVO est construite comme l'union de ses éléments temporels NLVO(t), qui représentent l'ensemble de toutes les vitesses absolues de A qui conduisent A vers une collision à un instant spécifique t.



(a) (b)

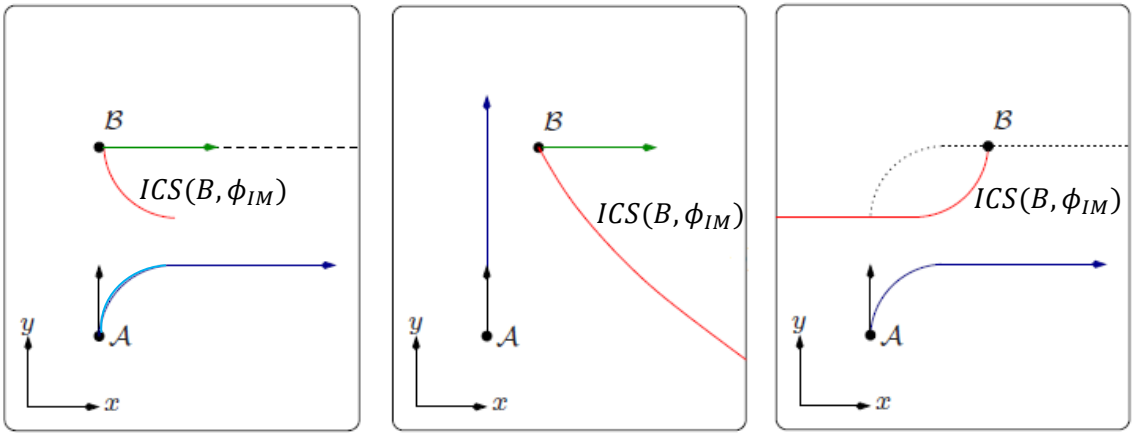
Figure 1.12 : L'approche NLVO : (a) $NLVO_{A/B}$ pour A induit par B, à l'instant t_0, A tente d'éviter B qui suit une trajectoire arbitraire connue $c(t)$ (où à t_0, sa position est $c(t_0)$). (b) représentation de NLVO pour trois obstacles mobiles [78].

D'autres extensions de VO ont été proposées par la suite, par exemple "Generalized Velocity Obstacles" (GVO) qui prend en compte les contraintes de non holonomie [79], ce qui est nécessaire pour bien raisonner sur le comportement futur du système. Il existe également l'approche RVO (*Reciprocal Velocity Obstacles*) [80] qui a été développée pour garantir plus de sûreté de navigation. RVO suppose que les autres obstacles (robots) utilisent le même protocole d'évitement d'obstacles. Ce qui implique que chaque robot prend en considération la vitesse des autres robots afin de les éviter, et sélectionne sa propre vitesse à partir de l'espace des vitesses qui ne fait pas partie des régions interdites (causé par la présence des autres robots). Ainsi, le robot prend uniquement 50% de la responsabilité d'éviter la collision. Dans [81], RVO a été améliorée en HRVO (*Hybrid Reciprocal Velocity Obstacles*), car, la

RVO présente certaines limitations, comme *la danse réciproque* (i.e. oscillations) où les robots ne réussissent pas à trouver un accord concernant la trajectoire à prendre (de quel côté passer). Pour ce faire, HRVO combine RVO et VO pour remédier au problème. De plus elle prend en considération les incertitudes de la perception.

1.2.2.5 Navigation basée sur les états de collision inévitable (ICS)

Bien que la majorité des approches présentées précédemment puissent être appliquées en environnement dynamique, dans un milieu urbain encombré de piétons, voitures ou autres robots, il ne peut pas être garanti qu'aucune collision n'aura jamais lieu. Hors, dans de tels milieux, la sûreté du système robotique ainsi que ce qui l'entoure est primordiale. C'est pour faire face à ce genre de problèmes que le *concept d'états de collision inévitable* a été proposé dans [11]. Son principe est de déterminer les états pour lesquels, quelle que soit la trajectoire du robot, à un instant t une collision aura lieu entre le robot et les obstacles. Ainsi, la sûreté est garantie tout simplement en évitant ces états. En effet, il faut aller au-delà d'éviter les états de collision (le cas de la majorité des approches d'évitement d'obstacles), il faut éviter les ICS. Le concept ICS prend en compte non seulement la dynamique de l'environnement, mais également le comportement futur des obstacles mobiles. Déterminer si un état est ICS nécessite de vérifier la collision pour toutes les trajectoires futures possibles de durée infinie que le robot peut suivre à partir de cet état. Parthasarathi et Fraichard [82] ont proposé une approche basée sur une approximation de l'espace de contrôle en considérant uniquement un sous-ensemble (un ensemble fini) de l'ensemble des trajectoires futures possibles. Une approche récente [83] évite l'approximation en utilisant *l'analyse d'accessibilité* (reachability analysis), et ainsi l'espace de contrôle est entièrement exploité (i.e. toutes les trajectoires futures possibles du robot). Cependant, cette méthode n'est applicable qu'en environnement statique. Une extension aux travaux de Parthasarathi et Fraichard est l'approche développée dans [84] : c'est une approche d'évitement d'obstacle qui conduit le robot d'un état non ICS vers un autre.



(a) (b) (c)

Figure 1.13 : Calcul de ICS pour (a) un objet mobile B et une trajectoire d'imitation (ϕ_{IM}) (A imite le comportement de B). La trajectoire ϕ_{IM} comprend deux partie ; la partie "rattrapage" (la partie cyan) à la fin de laquelle A atteint une vitesse relative nulle avec B, et la partie "suivi" (la partie bleue) durant laquelle A duplique le comportement de B. (b) un objet mobile et une trajectoire d'imitation arbitraire (i.e. imitant le comportement d'un autre objet). (c) un objet fixe et une trajectoire d'imitation arbitraire [82].

La majorité des approches basées sur les ICS [82, 84, 85] utilisent le concept de manœuvres d'imitation (imitating manoeuvres) qui consiste à dupliquer le comportement des obstacles mobiles afin que le robot puisse atteindre et maintenir une vitesse relative nulle avec l'obstacle mobile (voir la figure 1.13). Ainsi, la collision peut être évitée à l'infini mais à condition d'avoir une connaissance à l'infini du comportement futur des obstacles.

Malheureusement, en pratique cette dernière hypothèse n'est pas possible, c'est pourquoi, le plus souvent, le comportement futur des obstacles est supposé a priori connu ou dans certains cas une prédiction à court-terme est utilisée. Dans le cas où le robot utilise uniquement ses capteurs embarqués et le comportement futur des obstacles est inconnu, la sûreté du mouvement n'est plus garantie. Toutefois, à l'heure actuelle, ces approches semblent fournir la meilleure solution pour le problème de la sûreté du mouvement.

En plus des approches présentées ci-dessus, d'autres méthodes réactives peuvent être notées : parmi celles-ci nous trouvons des méthodes qui ont été conçues pour être appliquées dans des environnements statiques mais par la suite ont été étendues pour des environnements dynamiques comme par exemple : l'histogramme de champs de vecteurs (en anglais : Vector Field Histogram (VFH)) [86, 87, 88, 89, 90], la méthode de navigation courbure-vélocité (en anglais Curvature-Velocity Method (CVM)) [91, 92, 93], Navigation par diagrammes de proximité (en anglais Nearness Diagram Navigation (ND)) [94, 95, 96, 97, 98] (elle a été étendue pour des environnements changeants uniquement) et l'approche par cône de collision (en anglais : collision cone approach) [99, 100, 101], etc. Nous trouvons également d'autres approches d'évitement d'obstacles qui ont été développées récemment pour prendre en compte la dynamique de l'environnement (évitement des obstacles mobiles), comme par exemple : "Dynamic Velocity Space" (DVS) [102], "Trajectory Parameter Space" [103], navigation réactive versatile basée sur la sélection de manœuvres [104], l'algorithme "subtargets" combiné avec "Cubic B-spline" [105], la méthode de prédiction de collision [106], etc.

1.2.3 Planification réactive

Les deux classes de méthodes délibératives et réactives (présentées ci-dessus) ont des avantages et des inconvénients. Par exemple les méthodes délibératives sont les plus adaptées pour trouver un chemin complet vers le but, cependant elles nécessitent généralement une connaissance a priori de l'environnement et elles s'adaptent difficilement, sinon jamais, aux changements de l'environnement (environnements dynamiques). Par contre, les méthodes réactives sont très adaptées aux applications en temps réel, mais en contrepartie, elles se contentent de raisonner sur le pas de temps suivant. C'est donc pour pallier les inconvénients de chaque classe et pour bénéficier des avantages de chacune que les méthodes de planification réactive ont été proposées. Ces méthodes sont très pertinentes pour la navigation en environnements dynamiques inconnus.

1.2.3.1 La planification de mouvement partiel (PMP)

C'est pour prendre en compte explicitement la contrainte temps réel que la méthode de planification de mouvement partiel (en anglais : Partial Motion Planning) (PMP) a été proposée [15, 107, 108]. Elle doit agir et rendre une décision pendant un temps limité (temps de décision) qui dépend de l'état actuel de l'environnement. C'est pourquoi elle est très adaptée pour faire naviguer le robot dans des environnements dynamiques. Cette méthode est classée comme méthode de planification réactive car elle agit pendant un cycle de temps limité, durant lequel elle calcule une trajectoire qui se rapproche le plus possible du but. L'algorithme PMP opère selon trois étapes, répétées périodiquement pour chaque cycle PMP :

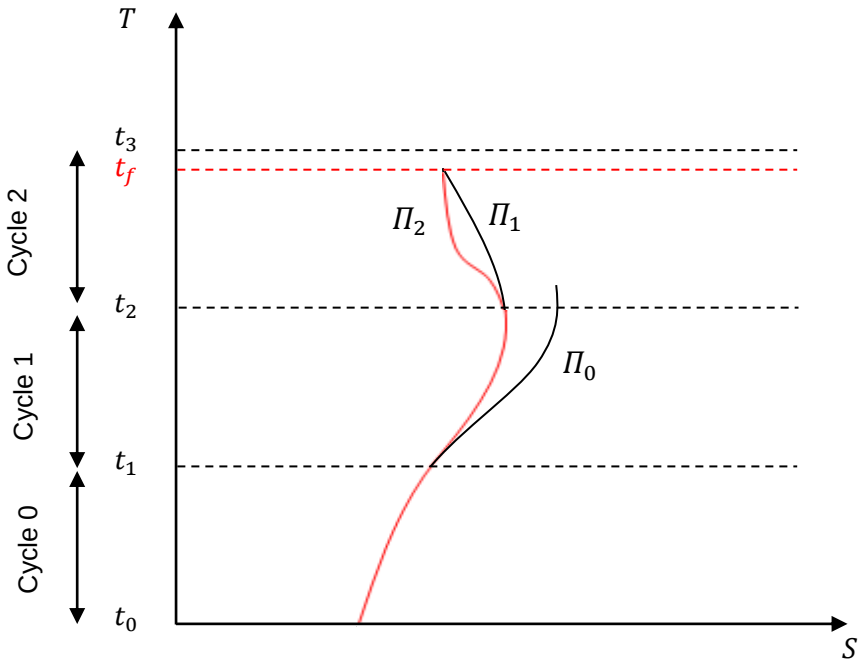


Figure 1.14 : Planification de mouvement partiel : la trajectoire finale (i.e. ensemble de trajectoires partielles) exécutée par le robot est représentée en rouge. Durant le cycle 0, une partie de la trajectoire partielle Π_0 est exécutée et la trajectoire partielle Π_1 est planifiée. Durant le cycle 1, une partie de la trajectoire partielle Π_1 est exécutée et la trajectoire partielle Π_2 est planifiée. Enfin, durant le cycle 3, la trajectoire partielle Π_2 est exécutée, et le robot atteint son but (à l'instant t_f).

- Mise à jour du modèle de l'environnement à partir de l'information issue des capteurs embarqués du robot.
- Calcul d'une trajectoire partielle sans collision vers le but à exécuter pendant le cycle de temps suivant.
- A la fin du cycle (le temps disponible est écoulé), la meilleure trajectoire partielle, i.e. celle qui optimise une fonction de coût donnée, est choisie. Elle est exécutée pendant le cycle suivant. Notons que cette trajectoire n'atteint pas forcément le but.

L'ensemble des trajectoires partielles calculées durant les différents cycles PMP conduit le robot vers le but (voir la figure 1.14). Cette trajectoire doit être faisable (prend en compte la dynamique du robot) et sans collision (une approche d'évitement d'obstacle peut être par exemple intégrée).

Le processus PMP peut être différent selon la nature de l'environnement, i.e. selon le comportement futur des obstacles mobiles (modèle du futur). Deux cas ont été traités jusqu'ici [15, 109] :

- *Planification dans un environnement connu* : comme l'environnement est a priori connu, le modèle du futur reste valide jusqu'à l'infini. Dans ce cas, dès que la trajectoire partielle planifiée atteint le but, la planification est arrêtée, cette trajectoire est exécutée, et ainsi le processus PMP se termine. Dans ce cas la durée d'un cycle PMP n'est pas contrainte par une période de temps donnée, sauf le temps de décision pour le premier cycle. La durée du cycle correspond alors à la durée de la trajectoire partielle en cours d'exécution. Par conséquent PMP est apériodique (du moment que la durée des différentes trajectoires partielles est généralement différente).
- *Planification dans un environnement partiellement prédit* : dans des situations réelles, l'environnement ne peut pas être connu sur une période de temps illimitée. Le modèle du futur peut être prédit sur une période de temps limitée (en utilisant la prédiction à court-terme). C'est pourquoi un tel environnement est appelé partiellement prédit. Le modèle du futur dans ce cas est valable

pour une période de temps limitée. Par conséquent, pour garantir que le modèle du futur reste valable, la planification d'une trajectoire partielle et son exécution doivent avoir lieu dans la limite de cette période de validité. Si cette période de validité est supposée constante, alors le cycle PMP est également constant. Dans ce cas PMP est périodique. Durant un cycle PMP, la trajectoire partielle calculée peut ne pas atteindre le but, comme elle peut aussi l'atteindre. Cependant, contrairement à la planification dans un environnement connu, même si la trajectoire partielle planifiée atteint le but, lors de son exécution, si le but n'est pas atteint avant la fin du cycle en cours (i.e. la durée de la trajectoire est supérieure à celle du cycle PMP), une nouvelle trajectoire partielle est exécutée au cycle suivant.

L'inconvénient majeur des travaux basés sur PMP est que le comportement futur des obstacles mobiles est généralement supposé connu (ex. [15, 107]) ou prédit à court-terme (ex. [109]). Dans ce cas, la sûreté de mouvement ne peut être garantie car aucune garantie sur le futur ne peut être fournie.

1.2.3.2 Les techniques par échantillonnage

Les techniques par échantillonnage sont le plus souvent utilisées dans une navigation hiérarchique[4] (appropriée quand le but est hors de la portée des capteurs) comme planificateurs locaux afin de permettre une planification en temps réel dans des environnements partiellement connus ou inconnus.

Ces techniques sont généralement basées sur la génération et le tri de mouvements candidats. Leur défi fondamental est de déterminer le meilleur moyen d'échantillonner efficacement l'espace de mouvements faisables. Ce paragraphe présente deux types de techniques ; les techniques par échantillonnage de l'espace

[4] Elle est basée sur la combinaison d'un planificateur local (génération de trajectoires faisables) et un planificateur global (guider le robot vers le but).

d'entrées (ISS) (en anglais : input space sampling) et les techniques par échantillonnage de l'espace d'états (SSS) (en anglais : state space sampling).

Échantillonnage de l'espace d'entrées (ISS). Au contraire des anciennes approches de planification de mouvement qui échantillonnent dans l'espace de chemins géométriques, cette technique échantillonne dans l'espace d'entrées (l'espace des contrôles) et génère des mouvements faisables. Tous ces mouvements sont intrinsèquement exécutables, car chaque entrée est simulée par le modèle du système robotique. Parmi les premiers travaux basés sur l'échantillonnage de l'espace d'entrées viennent les travaux de Kelly et Stentz [110]. Leur technique a été appliquée dans de nombreux domaines de la robotique mobile ; les AGV[5] [111, 112] et le domaine spatial [113, 114].

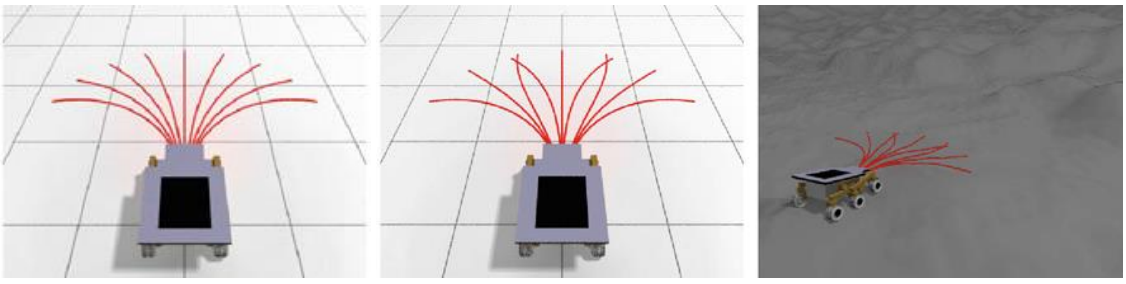
Le principe de l'échantillonnage de l'espace d'entrées est simple ; à partir d'un état initial, le modèle prédictif du système est utilisé pour générer un ensemble de trajectoires de contrôle (leur forme dépend donc du modèle du système et de son état initial (courbures, vitesses)). Ces trajectoires peuvent être triées par rapport à une fonction de coût donnée. De plus, elles peuvent être testées par rapport à la non collision avec les obstacles en utilisant par exemple l'une des techniques citées dans le paragraphe 1.2.2. L'exemple d'un ensemble de trajectoires généré en utilisant la technique par échantillonnage de l'espace d'entrées est illustré dans la figure 1.15. La figure 1.15.a montre un espace de recherche produit par la simulation de neuf entrées de courbure constante. Cette technique peut être également appliquée pour une recherche dans des espaces de contrôles qui sont plus expressifs que des arcs de cercles ; dans la figure 1.15.b, un espace de recherche produit par l'échantillonnage de clothoïdes est représenté. Dans ce cas, l'espace des directions atteignables par le robot est mieux représenté que celui de la figure 1.15.a. D'autre part, les ensembles de trajectoires calculés par cette technique peuvent être aussi adaptés pour une utilisation sur terrain accidenté, en simulant pour chaque contrôle,

[5] Autonomous ground vehicles.

l'interaction entre le châssis du véhicule et le terrain. Ces informations sont considérées dans un modèle de mouvement prédictif complexe qui est appliqué pour générer l'ensemble des trajectoires. L'espace de recherche résultant est représenté dans la figure 1.15.c.

Cette technique reste un sujet de recherche actif ; plusieurs améliorations concernant l'optimisation de l'algorithme (en réduisant le nombre de trajectoires testées pour non collision) [115] et le modèle prédictif utilisé [116] ont été apportées. Cependant, tous les travaux développés autour de cette technique n'ont traité que les obstacles statiques (ex. des roches dans le cas de l'exploration spatiale).

L'échantillonnage dans l'espace des contrôles est une méthode bien adaptée pour assurer une planification locale de plans faisables. Cependant, dans des environnements complexes, cette méthode cesse d'être efficace et un échantillonnage dans l'espace d'états est plus approprié.



(a) (b) (c)

Figure 1.15 : La technique par échantillonnage de l'espace d'entrées. (a) un ensemble de trajectoires généré par l'échantillonnage de l'espace d'entrées, ces trajectoires sont de courbure et vitesse linéaire constante. (b) Echantillonnage dans l'espace d'entrées de clothoïdes avec courbure initiale nulle. (c) Echantillonnage dans l'espace d'entrées de clothoïdes avec courbure initiale nulle sur un terrain accidenté [10].

Échantillonnage de l'espace d'états (SSS). Cette technique consiste à faire un échantillonnage dans l'espace d'états (au lieu de l'espace des contrôles comme

pour la technique ci-dessus), en spécifiant les conditions aux limites de chaque mouvement de l'ensemble des trajectoires générées [117, 10]. Mais, le compromis de cette technique est que la faisabilité du mouvement n'est pas garantie. Une solution est d'utiliser une méthode de génération de trajectoire inverse [117, 118] ; i.e. pour un robot mobile à un état initial donné, un contrôle qui satisfait le modèle du mouvement et les contraintes limites de l'état (positions, directions, courbure, vitesses, etc.) doit être trouvé (s'il existe).

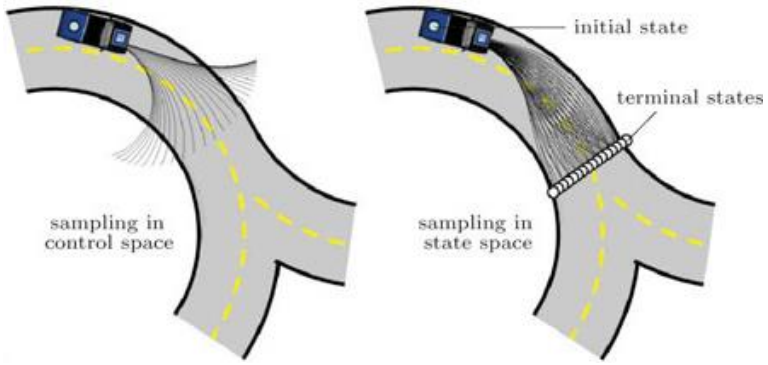


Figure 1.16 : Illustration d'espaces de recherche générés par échantillonnage dans l'espace d'états vs. l'espace des contrôles qui sont fortement contraints (cas des réseaux routiers) (source [117]).

Quand les contraintes environnementales limitent sévèrement l'espace de mouvements acceptables ou quand le planificateur global exprime de forte exigence (par rapport au but par exemple), cette technique est très efficace. Dans un réseau routier par exemple, il est plus bénéfique d'échantillonner uniquement à l'intérieur des voies, ainsi, le temps de calcul peut être optimisé en évitant de tester la non-collision des trajectoires qui sortent des voies. La figure 1.16 montre un exemple de trajectoires générées en échantillonnant l'espace d'états qui est comparé à un échantillonnage dans l'espace des contrôles (effectué dans les mêmes conditions de test). La majorité des trajectoires générées par l'échantillonnage dans l'espace des contrôles quittent la voie de circulation ou sont mal orientées. Alors que, celles générées par l'échantillonnage dans l'espace d'états restent à l'intérieur de la voie, où

les contraintes limites se restreignent à la position et la direction de l'état final. En effet, les contraintes environnementales sont facilement satisfaites quand les trajectoires planifiées sont exprimées dans l'espace d'états mais leur faisabilité dynamique ne peut pas être facilement garantie. Inversement, quand les trajectoires sont exprimées dans l'espace d'entrées, les contraintes de faisabilité peuvent être trivialement satisfaites, mais les contraintes environnementales ne peuvent pas être facilement satisfaites. Un autre avantage de la technique par échantillonnage de l'espace d'états est le fait que les trajectoires planifiées peuvent être biaisées par le planificateur global, le problème des minima locaux peut être alors minimisé et même parmi l'ensemble des trajectoires générées, le nombre de trajectoires qui se croise avec la région d'obstacle (i.e. le système est en collision avec l'obstacle) est réduit.

Cependant, cette méthode présente quelques inconvénients. Mise à part la complexité de cette technique par rapport à la technique SSI, vu que le temps de re-planification est contraint, le nombre de trajectoires évaluées est limité pour trouver la solution optimale. D'autre part, le comportement futur des obstacles n'est pas considéré, donc la sûreté du mouvement ne peut pas être considérée. Dans [117], l'approche a été testée dans une route contenant un obstacle mobile (un véhicule) dont la vitesse est connue.

Les techniques par échantillonnage (ISS et SSS) sont des techniques très adaptées pour des environnements inconnus où le robot utilise uniquement ses capteurs embarqués pour la perception, même si dans de nombreux cas le planificateur global nécessite une certaine connaissance a priori de l'environnement (ex. carte de route dans le cas d'un réseau routier) pour pouvoir guider le robot. Cependant, malgré que les contraintes kinodynamiques du système soient considérées, la dynamique de l'environnement est rarement prise en compte et le comportement futur des obstacles n'est pas traité. Toutefois, ces techniques peuvent être adaptées pour résoudre ces problèmes.

1.2.3.3 Les techniques de planification discrète

Afin de prendre en compte les changements ou la dynamique de l'environnement, l'une des premières approches était d'adapter ***des méthodes de programmation dynamique*** (ex. Dijkstra [31], A*[30], etc.). Ce sont des techniques de recherche efficaces qui discrétisent l'espace de recherche afin d'obtenir une grille ou un graphe qui est utilisé par la suite pour trouver une solution optimale. Principalement, la méthode A* (qui s'applique uniquement dans des environnements statiques) a été adaptée pour trouver le chemin optimal dans des environnements inconnus et dynamiques : plusieurs techniques ont été développées ; comme D* (A* dynamique) [119, 120] ou plus récentes "D* Lite" [121]. Ces techniques permettent dans un premier temps de planifier un chemin complet vers le but (comme pour les méthodes délibératives) en utilisant les informations connues initialement, puis de planifier un nouveau plan à chaque fois que de nouveaux changements apparaissent dans l'environnement introduisant ainsi la réactivité. Ainsi, chaque fois que le plan est invalide à cause d'un nouvel obstacle, le plan est réparé localement où la partie du plan en question est re-planifiée. Ce qui est plus efficace qu'une re-planification à zéro (du chemin en entier). Cependant, l'un des inconvénients de ces approches est que l'environnement traité est le plus souvent considéré comme changeant (apparition de nouveaux obstacles ou présence d'obstacles mobiles avec des vitesses très faibles (ex. les portes)) où le comportement futur des obstacles n'est pas pris en compte. De plus, Malgré la rapidité de ces approches pour fournir une solution, le problème qui se pose est que, durant la re-planification, le robot doit choisir entre attendre qu'un nouveau chemin soit calculé ou se déplacer dans une mauvaise direction (le temps d'avoir le nouveau chemin). Malheureusement, aucune indication sur le temps ou ses limites ne sont fournis. Dans [122], une variante de D* est proposée. L'algorithme développé est supposé agir en temps réel où le temps de re-planification est inférieur à une seconde. Toutefois, il n'y a aucune garantie que cette limite de temps soit respectée ou qu'elle soit toujours suffisante pour que l'algorithme agisse en temps réel. Un autre inconvénient de ces approches est que les contraintes kinodynamiques du robot ne sont pas considérées lors de la

planification du chemin. Récemment, une nouvelle approche "A* 3D" [123, 124] a été proposée pour remédier à ce problème, de plus elle traite des environnements plus complexes peuplés d'obstacles mobiles.

Quand le problème de planification est complexe et que le temps de planification disponible est limité, il faut se contenter de la meilleure solution qui peut être générée pendent le temps de calcul disponible. C'est pour fournir une telle solution que ***les algorithmes "Anytime"*** ont été proposé [125, 126, 127, 128]. Dans [129], l'algorithme développé ; "Anytime dynamic A*" (AD*) aborde explicitement la contrainte temps réel. A chaque pas de temps, une nouvelle recherche A* est effectuée. Au lieu d'attendre donc que le plan soit invalide pour re-planifier (ce qui peut se faire en plusieurs pas de temps), la re-planification est systématiquement réalisée à chaque pas de temps. Généralement, une première solution est calculée rapidement, puis elle est améliorée en calculant à chaque pas de temps une nouvelle solution qui soit plus optimale que la précédente. Cependant, comme pour l'approche D*, cette technique ne prend pas en considération la dynamique du système, ni le mouvement des obstacles mobiles (i.e. planifie dans un environnement statique puis re-planifie quand des changements sont observés ou suppose que la dynamique de l'environnement est a priori connu). De plus, elle nécessite qu'un plan initial soit disponible, donc une connaissance a priori de l'environnement doit être considérée initialement. Dans [130], une carte de route probabiliste est construite en premier à partir d'une connaissance a priori de l'environnement, puis AD* est utilisé pour la planification et la re-planification dans l'espace d'états-temps. En plus de la prise en compte de la dynamique du système, cette approche prend en considération le comportement futur des obstacles mobiles lors de la planification mais suppose qu'ils se déplacent selon un mouvement linéaire.

1.2.3.4 Déformation de mouvement

La première méthode de déformation a été proposée par Quinlan et Khatib [131] (nommée bande élastique). Cette méthode opère en deux phases ; en premier

lieu, comme une méthode globale, elle utilise un algorithme de planification de mouvement pour générer un chemin sans collision entre la position initiale du robot et le but en se basant sur une connaissance a priori de l'environnement. Cette étape est généralement réalisée en hors ligne. La deuxième phase de cette méthode est réalisée durant l'exécution (en temps réel) ; où le chemin du robot est déformé continuellement en fonction de la mise à jour de l'information sur l'environnement (i.e. quand un nouveau obstacle est rencontré) (un exemple est donné dans la figure 1.17). La déformation résulte en général de deux types de contraintes : celles en provenance des obstacles (contraintes externes) qui le poussent loin de ces derniers, et celles qui maintiennent la faisabilité (par rapport à la dynamique du robot) et la connectivité du chemin (contraintes internes). Ainsi la convergence vers le but est assurée. Les premiers travaux concernaient la déformation de chemin [132, 133], où le robot était supposé être holonome. Par la suite, la méthode a été étendue pour des robots non holonomes présentant plus de contraintes [134, 135].

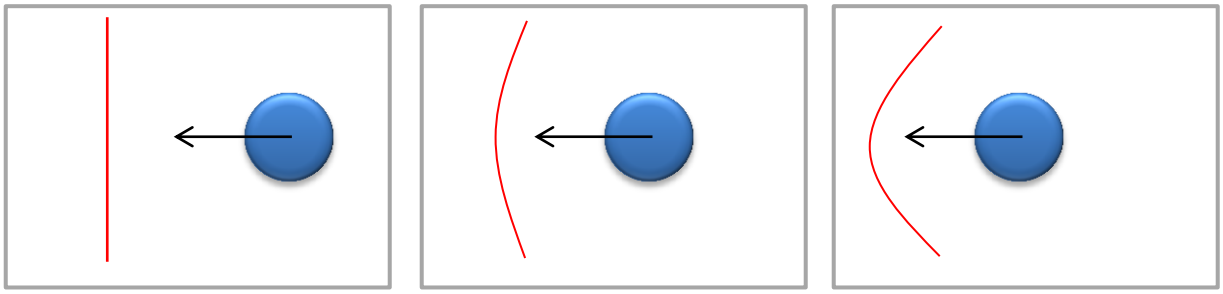


Figure 1.17 : La déformation de chemin ; le chemin (en rouge) est continuellement déformé en réponse à l'approche d'un objet mobile.

Bien que cette méthode offre une solution intéressante quant à sa réaction aux obstacles mobiles, elle a des limitations : en réponse à l'approche d'un obstacle mobile qui coupe le chemin du robot, celui-ci est déformé jusqu'à perdre sa connectivité. D'autre part, la déformation du chemin peut être bloquée par un autre obstacle statique, et donc le robot ne peut pas contourner l'obstacle mobile. C'est pourquoi, dans [136], cette méthode a été adaptée pour faire face à ce type de problèmes. Un grand inconvénient de la déformation de chemin qui persiste, est que

la dimension temps n'est pas considérée. Dans certains cas, la déformation continue ne fait pas l'affaire ; par exemple, une voiture se trouvant dans une intersection, où une autre voiture se présente avec une grande vitesse. Dans cette situation, il est plus approprié de s'arrêter que de déformer son chemin. C'est pourquoi, afin de traiter ce genre de situations en plus de la prise en compte des obstacles mobiles que la déformation de trajectoire a été introduite. Cette dernière opère dans un espace CT au lieu d'opérer uniquement dans C. Elle a été développée pour des robots holonomes, comme dans [137] ou non holonome, comme dans [138]. Dans [137], la trajectoire des obstacles mobiles est connue. Alors que, dans [138], une prédiction à court-terme est utilisée (par la méthode SLAMMOT[6]) : à chaque pas de temps la vitesse de l'obstacle est estimée, puis il est supposé que cette vitesse sera maintenu à l'infini (le mouvement est linéaire).

1.2.3.5 La commande prédictive (MPC)

Comme les méthodes précédentes, ex. PMP et les techniques par échantillonnage, la commande prédictive (Model prédictive control) (MPC) [139, 140, 141, 142] est une méthode qui offre un compromis entre les méthodes réactives et les méthodes délibératives en raisonnant sur k pas de temps. Cette méthode est capable de gérer explicitement les contraintes et l'incertitude. À partir de sa première version [143], MPC était devenue une technique très populaire dans le domaine de l'industrie, traitant des systèmes complexes (ex. [144, 145, 146]). Par la suite, elle a été utilisée comme outil très intéressant pour résoudre des problèmes de planification de mouvement et générer des trajectoires sans collision pour différents systèmes ; des robots mobiles (ex. [147, 148, 149, 150), des AGVs (ex. [151, 152, 153, 154, 148]) souvent déployés dans des environnements plus encombrés, des UAVs (ex. [155, 156]), etc.

Le principe de MPC est de raisonner sur plusieurs pas de temps mais d'effectuer qu'un seul pas. I.e. à tout instant t, étant donné le modèle de

[6] Simultaneous localization, mapping and moving object tracking.

l'environnement, le mouvement du robot est planifié sur un temps horizon fini (pour un nombre de pas N donné). Une séquence de contrôles optimaux est donc obtenue, mais uniquement le premier contrôle est appliqué. Le modèle de l'environnent est par la suite mis à jour et le processus est répété. Notons qu'il est possible d'appliquer uniquement un contrôle, comme il est possible d'appliquer un ensembles de contrôles (les premiers contrôles) pour une durée de temps inférieure au temps horizon (pour un nombre de contrôles $M < N$) [157, 150]. Pour obtenir la séquence de contrôles optimaux, MPC doit avoir recours à d'autres techniques. Dans [142] par exemple, les deux techniques "Exhaustive Expansion Tree Search" (EETS) et "Sequential Quadratic Programming" (SQP) sont utilisées.

MPC est une méthode très efficace qui prend en compte les contraintes kinodynamiques du robot, et qui est adaptée pour des environnements inconnus. Elle utilise l'information sensorielle pour régulièrement mettre à jour le modèle de l'environnement. Cependant, sa performance dépend de la portée du champ de vision. D'autre part, malgré que la majorité des travaux basés sur MPC suppose que l'environnement est statique, cette méthode peut être adaptée pour traiter les obstacles mobiles. Par exemple, une prédiction à court-terme peut être utilisée pour déterminer le comportement des obstacles mobiles, mais l'algorithme d'estimation risque de diverger en perdant la trace des obstacles mobiles. Dans [158], l'approche développée traite un environnement statique mais considère des cibles mobiles et génère une trajectoire dans un espace d'états-temps, où une caméra fixe est utilisée. Dans [148], MPC est combinée avec la technique RRT pour traiter des environnements contenant des obstacles mobiles dont le comportement futur est connu. Toutefois, ce qui reste un grand inconvénient de cette méthode est que pour pouvoir planifier sur un certain temps horizon, il est supposé qu'aucun nouvel obstacle n'est détecté pendant cette période de temps. En contrepartie, aucune garantie que les informations considérées sur l'environnement restent valables pendant ce temps. De plus, aucune contrainte ou limite par rapport au temps de planification ou la valeur du temps horizon choisi n'est considérée. C'est pourquoi, la

sûreté de mouvement ne peut pas être garantie dans ce cas, et le chemin planifié peut être obstrué par un obstacle non détecté au début de la planification.

1.2.4 Discussion

Comme expliqué dans la partie problématique, l'objectif de notre travail est de développer un système de navigation pour un robot mobile ayant un champ de vision limité, amené à se déplacer dans des environnements dynamiques inconnus (en se basant uniquement sur l'information du système sensoriel du robot). Ce système de navigation doit prendre en considération le comportement futur des obstacles mobiles (perçus et non perçus) et les contraintes kinodynamiques du robot. Il doit être capable de conduire le robot vers son but tout en garantissant la sûreté de mouvement.

Le résultat de nos recherches dans la littérature concernant les travaux élaborés dans le domaine de la navigation a été présenté dans ce chapitre. Les principales caractéristiques des différentes méthodes sont résumées dans les tableaux 1.1, 1.2 et 1.3. Les méthodes délibératives (méthode de planification de mouvement) permettent de calculer un chemin complet et libre de collision vers le but, mais elles sont principalement dédiées pour des environnements statiques, comme les méthodes de décomposition cellulaire. Ces méthodes peuvent être adaptées pour des environnements dynamiques, mais en ajoutant la dimension temporelle, la complexité du problème augmente. Par contre, d'autres méthodes comme PRM et RRT présentent de meilleures performances dans des espaces de configuration de grandes dimensions, et ainsi elles sont adaptées pour agir dans des environnements dynamiques. Ce qui résout uniquement une partie du problème, car par exemple la méthode PRM doit disposer d'une carte initiale de l'environnement construite en hors ligne (ou a priori connue), et donc l'environnement ne peut être réellement considéré comme inconnu. La méthode RRT quant à elle a l'avantage de prendre explicitement les contraintes kinodynamiques du robot, ce qui n'est pas toujours le cas pour d'autres méthodes (ex. décomposition cellulaire). En plus, la

construction incrémentale de l'arbre prend mieux en compte les forts changements de l'environnement comparée à PRM. Toujours est-il, le comportement futur des objets mobiles est supposé a priori connu comme pour la méthode PRM.

De façon générale, l'inconvénient majeur des méthodes délibératives est leur manque de réactivité par rapport aux changements de l'environnement pour pouvoir agir en temps réel. C'est là le point fort des méthodes d'évitement d'obstacles (méthodes réactives). Le robot utilise ses capteurs embarqués pour mettre à jour l'information sur l'environnement. Cependant, certaines méthodes traitent uniquement les environnements statiques, comme DW. D'autres méthodes ont été développées ou améliorées pour agir dans des environnements dynamiques, tel que, TVDW, VO, les algorithmes Bug et PF. Toutefois, malgré que ces méthodes puissent s'appliquer dans des environnements dynamiques, elles ne considèrent pas toujours le comportement futur des obstacles mobiles. Sinon, elles supposent qu'ils se déplacent avec une vitesse linéaire constante (ex. VO). Ce qui n'est pas le cas en pratique, car généralement les obstacles mobiles suivent des trajectoires arbitraires. C'est pourquoi, des méthodes comme NLVO et ICS considèrent des obstacles mobiles avec des trajectoires arbitraires. Cependant, ces trajectoires sont a priori connues, ce qui pose problème pour des applications dans des environnements inconnus. D'autres méthodes quant à elles (ex. TVDW, PF [73]), utilisent une prédiction à court-terme pour estimer le comportement futur des obstacles. Ce qui est très adaptée dans des environnements inconnus, mais la sûreté de mouvement ne peut pas être garantie car le raisonnement dans le futur est très court pour pouvoir éviter une éventuelle collision. Parmi toutes les méthodes réactives, c'est la méthode ICS qui fournit la meilleure solution pour le problème de la sûreté de mouvement ; elle prend en considération les contraintes kinodynamiques du robot et la dynamique de l'environnement ainsi que le comportement futur des obstacles mobiles. De plus, la sûreté de mouvement peut être garantie en évitant les états qui conduiront à une collision dans le futur. Cependant, Dans le cas où le robot utilise uniquement ses capteurs embarqués et le comportement futur des obstacles est inconnu, la sûreté de mouvement n'est plus garantie.

L’aspect réactif des méthodes d’évitement d’obstacles permet de réagir par rapport aux changements de l’environnement, et de générer un contrôle pour un seul pas de temps. Ces méthodes sont alors loin d’être les plus adaptées pour conduire le robot vers son but en garantissant sa sûreté.

En raison des causes citées plus haut, aucune des méthodes délibératives ni réactives n’offrent de solutions à nos problèmes. Ainsi, une méthode alternative qui combine les deux classes de méthodes peut être la solution ; à savoir, les méthodes de planification réactive. Dans un environnement dynamique, la méthode de déformation de trajectoire déforme le chemin du robot à l’encontre d’un obstacle mobile, alors que les méthodes de planification discrète planifient un nouveau plan à chaque fois que de nouveaux changements apparaissent dans l’environnement (ex. D*) ou de façon anytime (ex. AD*). Bien que ces méthodes réagissent quant aux changements qui apparaissent dans l’environnement, le plus souvent, le comportement futur des obstacles mobiles n’est pas pris en compte. Sinon, il est considéré comme connu ou prédit à court-terme. De plus, ces méthodes nécessitent qu’un plan initial soit disponible et donc une connaissance a priori de l’environnement est initialement considérée. Parmi les méthodes de planification réactive, les méthodes par échantillonnage (ISS et SSS) sont les plus adaptées pour des environnements inconnus où le robot utilise uniquement ses capteurs embarqués pour la perception. Elles sont très appliquées dans les réseaux routiers, les terrains accidentés et l’exploration spatiale. Ces méthodes sont utilisées comme planificateurs locaux, ainsi elles gagnent en réactivité. D’autre part, les trajectoires planifiées peuvent être biaisées par un planificateur global, ainsi le robot peut facilement être guidé vers le but. Toutefois, ces méthodes ne prennent pas en considération le comportement futur des obstacles et elles ont été très peu appliquées pour des environnements dynamiques. De plus, la méthode SSI est peu efficace dans des environnements complexes alors que la méthode SSS nécessite l’utilisation d’une méthode de génération inverse pour pouvoir planifier des trajectoires faisables. D’autres méthodes quant à elles planifient pour une certaine période de temps, où l’information sur l’environnement est mise à jour grâce aux

données sensorielles. Par exemple, les méthodes MPC et PMP. MPC suppose que le modèle de l'environnement reste inchangé sur toute la période de planification, en n'ayant aucune garantie que les informations considérées sur l'environnement restent valables pendant ce temps. De plus, aucune contrainte ou limite par rapport au temps de planification ou la valeur du temps horizon choisi n'est considérée. Dans ce cas, la sûreté du mouvement ne peut pas être garantie. Au contraire, la méthode PMP planifie pendant un cycle de temps donné la trajectoire à exécuter durant le cycle suivant en tenant compte du modèle du futur de l'environnement, supposé valable pour la planification et l'exécution de la trajectoire. PMP doit également rendre une décision pendant un temps limité (temps de décision) qui dépend de l'état actuel de l'environnement. C'est pourquoi elle est très adaptée pour faire naviguer le robot dans des environnements dynamiques. Dans ce cas, la non collision peut être garantie mais uniquement pour un modèle du futur à priori connu ou prédit à court-terme.

Pour toutes les raisons citées ci-dessus, nous avons opté pour une méthode de planification réactive. La méthode que nous avons développé est une extension de la méthode de *planification de mouvement partiel*, qui prend explicitement la contrainte temps réel et dont le principe est de calculer des trajectoires partielles jusqu'à atteindre le but. Selon notre opinion, c'est la méthode la plus adaptée pour répondre à nos besoins (i.e. traiter des environnements inconnus et dynamiques, prise en compte du comportement futur des obstacles mobiles et des contraintes kinodynamiques du robot, etc.). Cette méthode prend avantage des méthodes globales par rapport à la convergence vers le but et des méthodes réactives en mettant à jour l'information sur l'environnement lors de la planification de chaque trajectoire partielle. La méthode RRT a été exploitée pour l'exploration de l'espace de recherche, car elle est bien adaptée aux environnements dynamiques inconnus. De plus, elle peut être facilement intégrée au concept de planification partiel qui planifie une trajectoire pour un temps limité.

La plupart des approches étudiées dans ce chapitre ne considèrent pas le comportement futur des obstacles mobiles ou supposent qu'il est a priori connu, et donc la sûreté du mouvement ne peut être garantie. Dans notre travail, nous élaborons un modèle du futur qui permet non seulement de prendre en compte le comportement du futur des obstacles perçus par le robot mais également ceux qui se trouvent hors de son champ de vision et qui peuvent le placer dans des situations dangereuses dans le futur. Ce modèle a été utilisé dans la conception d'un planificateur de mouvement partiel qui garantit la sûreté du mouvement. Pour garantir la sûreté, l'approche développée est basée sur une extension du concept ICS.

Tableau 1.1 : Comparaison des méthodes délibératives ((-) : la caractéristique est vérifiée dans certains cas, (x) : la caractéristique peut être vérifiée sous certaines conditions).

Méthodes délibératives	Contraintes cinématiques	Contraintes dynamiques	Utilisation d'information sensorielle	Environnement inconnu	Prise en compte des objets non perçus	Environnement dynamique	Raisonnement futur (obstacles mobiles)	Sûreté
Décomposition Cellulaire	Non	Non	- (en hors ligne)	- (exploration hors ligne)	Non	Non	Non	Non
PRM	Oui ([45])	Oui ([45])	- (mise à jour)	- (carte initiale connue)	Non	Oui ([45])	Oui ([45]) (connu)	x
RRT	Oui	Oui	Oui ([52])	Oui ([52])	Non	Oui ([51])	Oui ([52]) (connu)	x

Tableau 1.2 : Comparaison des méthodes réactives.

Méthodes réactives	Contraintes cinématiques	Contraintes dynamiques	Utilisation d'information sensorielle	Environnement inconnu	Prise en compte des objets non perçus	Environnement dynamique	Raisonnement futur (obstacles mobiles)	Sûreté
Algs. Bug	Non	Non	Oui	Oui	Non	Oui ([63])	Non	Non
PF	Non	Non	Oui	Oui	Non	Oui ([71])	Oui ([73]) (prédiction à court-terme)	Non
DW	Oui	Oui	Oui	Oui	Non	Oui ([76])	Oui ([76]) (prédiction à court-terme)	x
VO	Oui ([79])	Non	Oui	Oui	Non	Oui	Oui (vitesse constante ou trajectoire connue)	x
ICS	Oui	Oui	Oui	Oui	Non	Oui	Oui (connu)	x

Tableau 1.3 Comparaison des méthodes de planification réactive.

Méthodes de planification réactive	Contraintes cinématiques	Contraintes dynamiques	Utilisation d'information sensorielle	Environnement inconnu	Prise en compte des objets non perçus	Environnement dynamique	Raisonnement futur (obstacles mobiles)	Sûreté
déformation de mouvement	Oui [134]	Oui [138]	- (pendant la déformation)	- (carte initiale connue)	Non	Oui	Oui ([138]) (prédiction à court-terme)	Non
PMP	Oui	Oui	Oui	Oui	Non	Oui	Oui (prédiction à court-terme ou connu)	x
SSI	Oui	Oui	Oui	oui	Non	Non	Non	Non
SSS	Oui	Oui	Oui	Oui	Non	Oui	Non	Non
D*	Oui ([123])	Oui ([123])	- (pendant la re-planification)	- (carte initiale connue)	Non	- (changeant)	Non	Non
AD*	Oui ([130])	Oui ([130])	- (pendant la re-planification)	- (carte initiale connue)	Non	Oui	Oui ([130]) (vitesse constante)	Non
MPC	Oui	Oui	Oui	Oui	Non	Oui	Oui	Non

						([148])	([148]) (connu)	

1.3 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur les méthodes de navigation existantes; les méthodes délibératives calculent un chemin complet entre une configuration initiale du robot et une configuration but, alors que les méthodes réactives calculent le mouvement à appliquer au pas de temps suivant. Notre travail vise à développer un système de navigation qui opère dans un environnement dynamique inconnu et qui prend en compte le comportement futur des obstacles mobiles. Dans ce cas, une méthode de planification de mouvement qui est le plus souvent adaptée pour des environnements statiques ou qui nécessite une connaissance a priori de l'environnement et une méthode d'évitement d'obstacle qui manque de convergence vers le but ne peuvent correspondre à nos attentes. Afin de tirer profit des avantages de chacune des deux catégories de méthodes et pour palier leurs inconvénients, nous avons opté pour une méthode de planification réactive ; *la planification de mouvement partiel*. Le calcul des trajectoires partielles permet d'une part la convergence vers le but (i.e. l'ensemble des trajectoires partielles conduit le robot vers le but), et d'autre part de mettre à jour l'information sur l'environnement. Cependant, ce qui reste un problème posé est la garantie de la sûreté de mouvement. Nos travaux se sont principalement concentrés sur ce problème. La notion de sûreté de mouvement ainsi que les approches proposées sont présentées dans les chapitres suivants.

CHAPITRE 2

LA SURETE DE MOUVEMENT

Doter un robot mobile de capacités humaines a toujours été un challenge pour tous les chercheurs dans le domaine de la robotique [20, 159]. Plusieurs travaux ont été développés pour permettre aux robots de naviguer de manière entièrement autonome dans des environnements encombrés d'objets fixes et mobiles. Cependant, un problème crucial reste posé dans ce type d'environnements et qui est de garantir que ces robots soient sans danger aux autres agents de l'environnement peu importe ce qui peut se passer dans le futur. Dans ce chapitre, nous nous intéressons précisément à ce problème à savoir *la sûreté de mouvement*. Le problème de la sûreté de mouvement dans un environnement dynamique et partiellement observable est tout d'abord posé de manière plus rigoureuse; les travaux les plus pertinents dans ce volet sont cités afin de positionner notre travail. Par la suite, une étude détaillée est présentée ; les critères de sûreté de mouvement sont définis. Ces critères sont d'une grande importance car ils déterminent les limites et les conditions sous lesquelles une méthode de navigation peut être utilisée. D'autre part, étant donné que la sûreté de mouvement est traitée dans des environnements dynamiques partiellement observables, le modèle du futur utilisé est alors décrit. Enfin, la solution que nous proposons pour garantir la sûreté de mouvement dans de tels environnements est dégagée.

2.1 La sûreté dans un environnement dynamique et partiellement observable

Quand des robots mobiles sont amenés à être déployés à grande échelle entre les êtres humains, il y a alors un besoin primordial de concevoir des approches d'évitement d'obstacles et de navigation pour lesquelles la sûreté de mouvement peut

être garantie. Il existe une riche littérature sur l'évitement d'obstacles et la navigation sans collision, cependant, pendant longtemps la notion de sûreté de mouvement a été ignorée ou mal définie. La sûreté de mouvement dans un monde réel (ex. un milieu urbain) reste un problème ouvert, où le terme monde réel implique que :

1- L'environnement traite des objets fixes et mobiles dont le comportement futur est inconnu.
2- Le robot a uniquement une connaissance partielle de son entourage en raison de ses limitations sensorielles.

La première contrainte d'un monde réel concerne sa dynamique où des objets mobiles ayant des trajectoires arbitraires évoluent. En fait, non seulement la position de l'objet doit être prise en compte mais aussi son comportement futur, car même si le robot peut éviter l'objet au temps présent, cela pourrait ne plus être le cas dans un temps futur. Parmi le peu de travaux de recherches traitant le problème de sûreté de mouvement, plusieurs le font uniquement pour des environnements statiques et ne considèrent pas les obstacles mobiles [160, 161, 162]. D'autres travaux résolvent ce problème en coordonnant le mouvement d'un ensemble de robots (i.e. pour un robot donné, le reste des robots représente des obstacles mobiles). Plusieurs approches pour lesquelles l'évitement de collision est garanti ont été proposées (ex. [4, 5, 6, 7]). Cependant, cette garantie est perdue si l'environnement contient des obstacles mobiles incontrôlables (ce qui est le cas du monde réel). Pour les travaux qui s'intéressent au cas dynamique, les travaux [163, 164, 76] garantissent que le robot peut toujours être dans un état où une trajectoire d'évitement peut être exécutée dans un environnement comportant des objets statiques et mobiles. Dans [165, 166, 167], les auteurs ont minimisé la collision de manière probabiliste. Ces approches sont intéressantes mais elles ne fournissent aucune garantie stricte de la sûreté de mouvement, du moment que des modèles probabilistes sont utilisés.

La deuxième contrainte d'un monde réel est liée aux capacités perceptuelles du robot. Quand le robot utilise ses capteurs embarqués, il est alors fondamentalement contraint à exploiter uniquement une représentation incomplète du monde. Cela signifie que certaines régions de l'environnement du robot seront perçues alors que d'autres seront inaperçues, en raison du champ de vision limité et

les occlusions (i.e. les régions cachées par d'autres objets). Il y a peu de travaux qui prennent en considération les limitations sensorielles. Par exemple, le problème des occlusions est abordé de manière grossière dans [168] et d'une manière plus poussée dans [169].

Le but du présent travail est précisément de traiter un tel problème, i.e. celui de la navigation des robots mobiles autonomes ayant un champ de vision limité dans des environnements inconnus. Ces environnements sont peuplés d'obstacles fixes et mobiles ayant une vitesse maximale et un comportement futur inconnu. Une de nos motivations premières est d'étudier si des garanties strictes de la sûreté de mouvement peuvent être obtenues ; strictes dans le sens où elles peuvent être établies de manière formelles. Les solutions aux problèmes posés ainsi que les approches proposées sont données dans le reste du chapitre et dans les chapitres suivants.

2.2 Critères de la sûreté de mouvement

Afin d'analyser le problème de sûreté, trois critères doivent être considérés [3, 170]; de manière à ce que si l'un des critères n'est pas considéré par la méthode de navigation en question, le système sera dans une situation dangereuse et sera exposé à un risque de collision à un certain point dans le futur. Ces trois critères sont présentés dans ce qui suit.

2.2.1 Critère 1: Contraintes liées au mouvement du robot

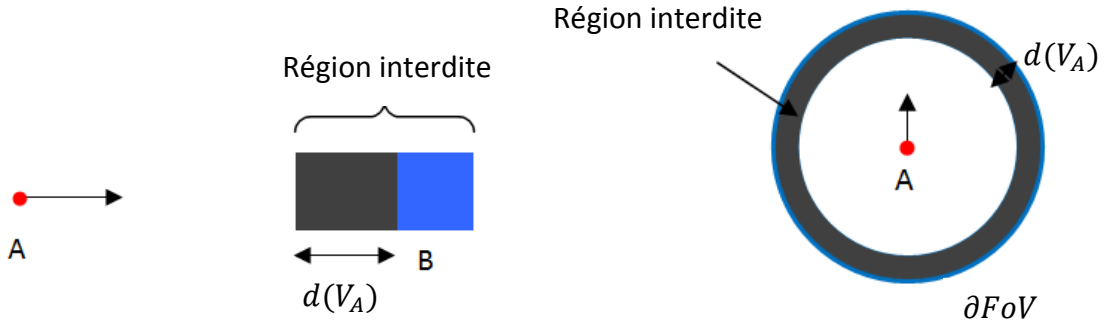
Le premier critère de sûreté indique que les limitations du mouvement du robot doivent être prises en considération, à savoir la dynamique du robot (accélération/décélération maximale, vitesse maximale, etc.). Prenons un exemple illustratif d'un robot A (considéré comme une masse ponctuelle dans l'espace 1D) caractérisé par sa position p_A, sa vitesse $|V_A| \leq V_{A_{max}}$ et son accélération $|\alpha| \leq \alpha_{max}$ et un obstacle statique B, caractérisé par sa position p_B. A cause de sa dynamique, A prend un temps minimal V_A/α_{max} pour freiner et s'arrêter. La distance d correspondante est en fonction de l'accélération maximale α_{max} , où :

$$d(V_A) = \frac{V_A^2}{2\alpha_{max}} \qquad (2.1)$$

Si la distance entre A et B est inférieure à cette valeur, peu importe ce qui se passe, une collision aura lieu dans le futur (figure 2.1.a). Par conséquent, afin de garantir la sûreté de mouvement, la condition suivante doit être vérifiée :

$$d_{A/B} > d(V_A) \qquad (2.2)$$

Où la distance $d(V_A)$ correspond à la région interdite (la zone grise dans la figure 2.1.a), alors que, $d_{A/B}$ représente la distance qui sépare le robot de la limite géométrique de l'obstacle. Par exemple, si l'obstacle B est de dimension $\rho \times \rho$, alors, $d_{A/B} = |p_B - p_A| - \frac{\rho}{2}$.



(a) Cas d'un seul objet.

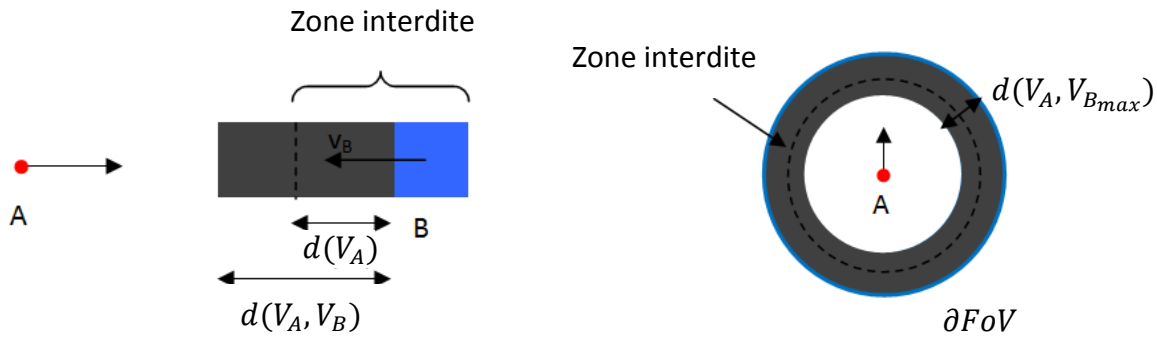
(b) cas d'un ensemble d'objets mobiles inattendus (limite du champ de vision de A).

Figure.2.1 : Un exemple de l'influence de la dynamique d'un robot (une masse ponctuelle) sur sa sûreté.

Dans un autre cas plus proche de nos hypothèses, si nous considérons que A à un champ de vision limité de forme circulaire (∂FoV) et que chaque point de ∂FoV est un obstacle potentiel (étant donné qu'aucune information n'est disponible sur la possiblité de présence ou non d'un obstacle sur ∂FoV), alors en appliquant la condition (2.2) la region interdite devient la bande autour de ∂FoV (voir figure 2.1.b).

2.2.2 Critère 2: raisonnement sur le futur

Dans l'exemple de la figure 2.1.a, l'obstacle B est fixe. Dans le cas où l'obstacle B est mobile, il ne suffit pas de s'assurer que A ne soit pas dans la zone interdite (distance nécessaire pour permettre à A de freiner) pour garantir qu'il n'y ait pas de collision; i.e. la condition (2.2) n'est plus vérifiée. Une représentation de l'evolution future de l'environnement (modèle du futur) est alors nécessaire afin de garantir la sûreté, ce qui représente le deuxième critère de sûreté de mouvement.



(a) cas d'un seul objet mobile.

(b) cas d'un ensemble d'objets mobiles inattendus (limite du champ de vision de A).

Figure 2.2 : L'influence de la dynamique d'un robot et celle des obstacles mobiles sur la sûreté de mouvement d'un robot.

Prenons l'exemple précédent mais cette fois-ci avec B un obstacle mobile de vitesse constante $V_B \leq V_{B_{max}}$. Dans ce cas, la dynamique de A et celle de B doivent être toutes les deux prises en compte. En effet, la distance séparant A et B doit être suffisante pour que A freine et s'arrête avant de rencontrer l'obstacle mobile. La sûreté est alors garantie en vérifiant la condition suivante (figure 2.2.a):

$$d_{A/B} > d(V_A, V_B) \qquad (2.3)$$

Avec :

$$d(V_A, V_B) = \frac{V_A^2}{2\alpha_{max}} + V_B \frac{V_A}{\alpha_{max}} \qquad (2.4)$$

Cette distance est la somme de la distance minimale nécessaire pour que A freine et la distance parcourue par B pendant le temps de freinage (V_A/α_{max}). Ce qui correspond à la région interdite.

Quand le robot à un champ de vision limité et que l'environnement est inconnu (comme dans notre cas), tous les objets inattendus doivent être pris en considération afin de garantir la sûreté de mouvement. Nous supposons donc que chaque point de la limite de visibilité ∂FoV est considéré comme un obstacle potentiel. De plus, aucune information sur le comportement futur de ces obstacles n'est disponible, la seule information qui peut être considérée est leur vitesse maximale $V_{B_{max}}$. La région interdite devient la bande le long de ∂FoV qui correspond à :

$$d\left(V_A, V_{B_{max}}\right) = \frac{V_A^2}{2\alpha_{max}} + V_{B_{max}} \frac{V_A}{\alpha_{max}} \qquad (2.5)$$

Si la distance entre A et ∂FoV est inférieure à cette valeur, peu importe ce qui se passe une collision aura lieu dans le futur (figure 2.2.b). Par conséquent, la condition suivante doit être vérifiée :

$$d_{A/B} > d(V_A, V_{B_{max}}) \quad (2.6)$$

2.2.3 Critère 3: un horizon temporel approprié

Les deux exemples précédents ont révélé que la garantie de sûreté de mouvement ne revient pas simplement à maintenir le robot loin des états de collision, mais de le garder loin des états qui le conduiraient à une collision à un certain point dans le futur. Pour ce faire, un résonnement sur le futur est requis. D'où le troisième critère de sûreté de mouvement. Pour garantir la sûreté, le robot doit être prévenu à l'avance, étant donné sa dynamique (critère 1) et la dynamique des objets voisins présents dans son environnement (critère 2), afin d'avoir assez de temps pour éviter la collision. La question qui se pose est : jusqu'à quel point dans le futur le raisonnement (à savoir la modélisation) doit avoir lieu? La réponse se traduit par *l'horizon* temporel *approprié*. En général, il est supposé qu'une connaissance a priori du comportement futur de l'environnement est disponible. Dans ce cas, pour garantir que le robot n'atteindra jamais un état de collision inévitable, les risques de collision

sont vérifiés à travers un horizon temporel infini [11, 171, 80]. Cependant, quand la connaissance de l'environnement est limitée (ex. la trajectoire de l'obstacle est connue sur un temps fini, ou le robot a un champ de vision limité), l'hypothèse précédente n'est plus valable. Ainsi, la sûreté doit être garantie sur un horizon temporel fini. Cela revient à assurer qu'aucune collision ne va avoir lieu pendant cette limite de temps, sans se préoccuper de ce qui peut se produire après cette limite (i.e. même si une collision aura lieu, la sûreté reste garantie). D'autre part, dans certaines situations, il n'est pas possible de considérer un horizon temporel infini, car cela peut être très restrictif. Par exemple, à partir d'un certain moment dans le futur, l'environnement devient entièrement interdit (ce point sera discuté par la suite). C'est pourquoi, il faut considérer un temps horizon fini. Le choix de ce temps est défini dans le chapitre suivant.

En résumé, la sûreté de mouvement revient à trois règles principales :

1- Contraintes liées au mouvement du robot : prise en compte de la dynamique du robot.

2- Raisonnement sur le futur : un modèle du futur est requis.

3- Un horizon temporel approprié : un temps fini qui définit jusqu'à quel point dans le futur le raisonnement / modélisation doit avoir lieu.

Ces trois règles représentent en réalité des lois qui doivent être vérifiées par toute approche d'évitement d'obstacle ou de navigation dite *sûre*. Notons, que ces trois règles sont toutes liées au paramètre temps. Dans un environnement dynamique, ce paramètre est un élément primordial.

2.3 Modélisation du futur

Il a été montré dans le paragraphe précédent que la modélisation de l'environnement et particulièrement celle de son évolution dans le futur est nécessaire pour la sûreté du robot (critère 2). La question qui se pose est : quel modèle du futur doit être utilisé quand il s'agit d'objets dont le comportement futur est inconnu?

Différents types de représentation existent. Nous distinguons principalement trois types de modèles.

2.3.1 Les modèles déterministes

Dans ce type de représentation, il est supposé que chaque obstacle dans l'environnement du robot lui est assigné un mouvement futur nominal. Ce mouvement peut être soit *a priori connu* comme dans [172, 171] soit *estimé* en utilisant des technique de détection et de suivi des obstacles mobiles, où les modèles du futur sont basés sur l'extrapolation du comportement futur des obstacles mobiles à partir de leur état actuel. Dans la majorité des cas, l'extrapolation compte sur l'hypothèse d'un comportement constant (ex. [173, 174]) ou alors *appris* en utilisant des techniques de prédiction à long terme qui exploitent la nature de l'environnement en question et apprennent comment les obstacles mobiles se déplacent dans ce dernier (voir [175]).

2.3.2 Les modèles conservatifs

D'un point de vue sûreté de mouvement, les modèles déterministes sont utiles tant que leur prédiction de l'évolution du futur de l'environnement est fiable. Malheureusement cette fiabilité peut décroitre considérablement à long terme. Pour remédier à ce problème, les modèles conservatifs ont été proposés [5]. Dans ce type de représentation, le comportement du futur est estimé en utilisant les limites des contraintes dynamiques des obstacles mobiles (ex. vitesse maximale) ou des limites géométriques telles qu'une restriction de l'orientation à un intervalle de valeurs. Cet intervalle peut être lié à la nature de l'environnement ou la tâche à accomplir (par exemple quand le robot se déplace dans une route l'orientation des obstacles va correspondre aux directions de la route). Ces contraintes sont modélisées de manière à ce que le modèle du futur englobe tous les mouvements futurs possibles des obstacles. Par conséquent, chaque obstacle lui est attribué son ensemble atteignable, i.e. l'ensemble des états qu'il peut atteindre dans le futur, pour représenter son mouvement futur. C'est l'approche choisie par exemple dans [176, 5].

2.3.3 Les modèles probabilistes

Les modèles conservatifs sont satisfaisants d'un point de vue sûreté de mouvement du moment qu'ils peuvent garantir l'évitement de collision. Cependant, à cause de la croissance rapide des ensembles atteignables des obstacles, l'environnement finit par être entièrement interdit et le robot est alors bloqué. C'est pour remédier à ce problème que les modèles probabilistes ont été proposés. Dans ce type de modèles, le mouvement futur de chaque obstacle est caractérisé par une fonction de densité de probabilité. Les outils utilisés pour prédire le comportement futur des obstacles sont très variés ; ex. les chaines de Markov [177], les modèles de Markov cachés [178], et la simulation de Monté Carlo [179]. Les travaux [8, 9] ont traité le problème de la sûreté de mouvement en utilisant des modèles probabilistes du futur, qui sont les mieux adaptés pour prendre en charge les incertitudes concernant le comportement futur des obstacles mobiles. Cependant, une sûreté de mouvement stricte ne peut être garantie, à la place les risques de collision sont minimisés.

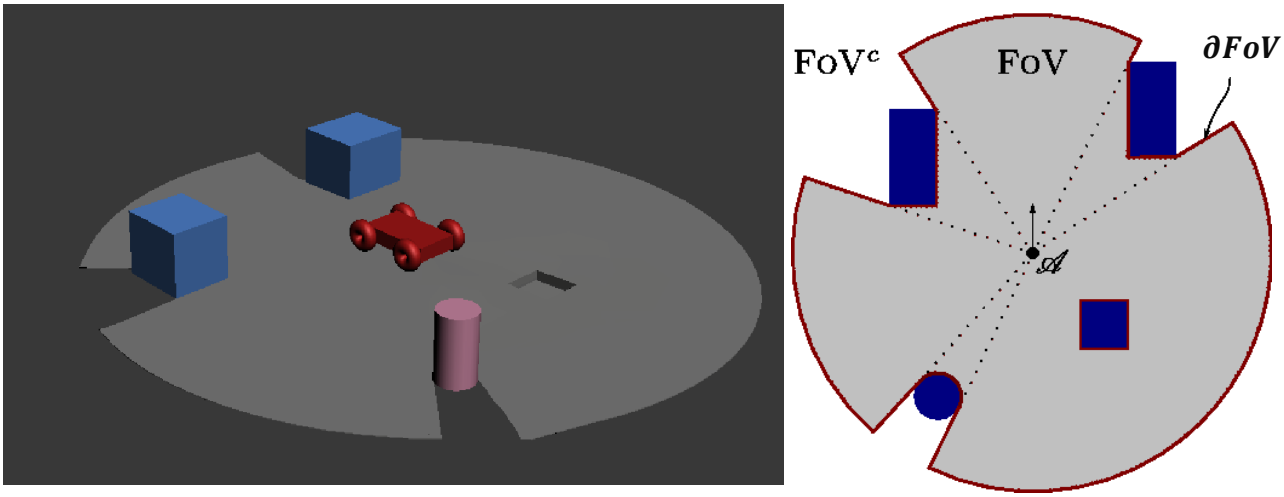
2.3.4 l'approche utilisée

Soit A désigne le robot mobile considéré dans cette thèse, qui opère dans un espace de travail 2D (WS). En supposant que A est équipé de capteurs de distance tels que des télémètres laser ou caméras, il peut uniquement percevoir un sous-ensemble de WS. Ce sous-ensemble est *le champ de vision de* A, il couvre 360°, sa forme est arbitraire, et il dépend de l'environnement actuel de A (voir figure 2.3.a). Il est noté FoV. WS est alors partitionné en trois sous-ensembles: (1) FoV, (2) FoV^c, la partie non perçue ($FoV^c = WS \backslash cl(FoV)$) et (3) ∂FoV, la limite entre les deux. Il parait raisonnable de supposer que A "explore autour de lui"; autrement dit que A se trouve toujours à l'intérieur de FoV. Pour tenir compte de l'existence de capteurs de distance 3D, e.g. Velodyne LIDAR ou caméra PrimeSensor, FoV peut contenir des "trous" représentant des objets entièrement perçus par le système sensoriel de A (voir figure 2.3.b). FoV représente la région de WS qui est libre d'objet à l'instant de détection. Ce modèle générique de champ de vision peut être enrichi si A peut faire la différence entre les objets fixes et mobiles. Dans ce cas, ∂FoV peut-être partitionné en trois

parties ∂FoV^f, ∂FoV^m et ∂FoV^u, correspondant respectivement aux objets fixes, mobiles et objets non perçus, i.e. les limites de détection et les lignes d'occultation (délimitant les régions cachées par des obstacles mobiles ou fixes), où:

$$\partial FoV = \partial FoV^f \cup \partial FoV^m \cup \partial FoV^u \qquad (2.7)$$

Quand les capteurs de A ne sont pas en mesure de faire la différence entre les objets fixes et mobiles, $\partial FoV = \partial FoV^u$.



(a) (b)

Figure 2.3 : Un robot avec un champ de vision limité dans un environnement inconnu (a) un scénario avec deux obstacles fixes, un obstacle mobile et un trou (la région en gris foncé est non perçue), (b) le champ de vision correspondant, où FoV est la région perçue (en gris), ∂FoV sa limite et FoV^c est la région non perçue.

L'environnement de A peut alors contenir aussi bien des objets perçus que des objets non perçus dont le comportement est imprévisible. C'est pourquoi parmi les trois modèles proposés dans les sous-paragraphes précédents, le plus adapté à gérer ce type de comportement afin de garantir la sûreté de mouvement est le second type de représentation ; à savoir *le modèle conservatif* : tous les mouvements futurs possibles pour un objet donné doivent être pris en considération.

Posons l'hypothèse d'un objet sous forme d'un point ayant une vitesse ne dépassant pas une valeur maximale et dont le comportement futur est inconnu. Etant donné sa position initiale, la région de l'espace de travail qui peut ne pas être sans

collision est modélisée par un disque dont le rayon croit en fonction du temps avec un rapport de croissance correspondant à la vitesse maximale de l'objet. Dans l'espace × temps, ces disques croissants sont représentés par un cône inversé (voir figure 2.4(3)). Ce cône est l'ensemble atteignable d'un objet représenté par un point dont la dynamique est caractérisée par une accélération infinie et une vitesse maximale. En général, les ensembles atteignables peuvent être utilisés pour représenter tous les mouvements futurs possibles d'un objet avec une dynamique arbitraire, ex. un objet avec une vitesse et une accélération maximales (voir figure 2.4(4)).

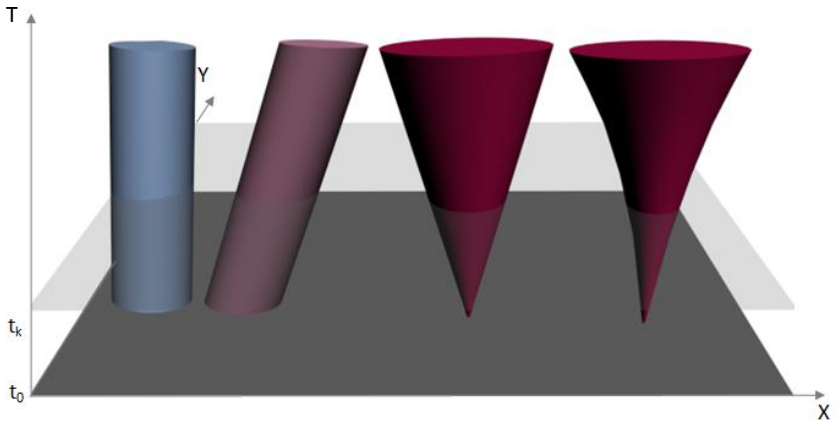


Figure 2.4 : Modèles du futur (de droite à gauche) : disque fixe (1), disque mobile avec vitesse constante (2), modèles conservatifs pour un point mobile avec un mouvement futur inconnu et une vitesse maximale (3) et une accélération et une vitesse maximale (4).

Dans une situation comme celle de la figure 2.3.b, comment les parties non perçues de l'espace de travail WS appartenant à ∂FoV^u et ∂FoV^c peuvent être prises en considération ? La réponse est, une fois encore, d'avoir un modèle conservatif et de traiter chaque point de ∂FoV^u et ∂FoV^c comme un objet mobile potentiel avec un comportement futur inconnu. En conclusion, le modèle espace × temps du futur peut être défini, pour différentes composantes du champ de vision du robot A (voir figure 2.5) comme suit:

- ∂FoV^u et ∂FoV^c (les objets non perçus) : chaque point de cet ensemble est modélisé comme un disque qui croit proportionnellement au temps (i.e. un cône dans l'espace × temps).
- ∂FoV^f (les objets fixes) : chaque point de cet ensemble reste constant à travers le temps (i.e. une ligne verticale dans l'espace × temps).
- ∂FoV^m (les objets mobiles) : si l'information sur leur comportement futur est disponible et fiable, chaque point de cet ensemble est modélisé selon cette information (i.e. une courbe dans l'espace × temps), sinon il est considéré comme un objet non perçu et modélisé comme un disque croissant.

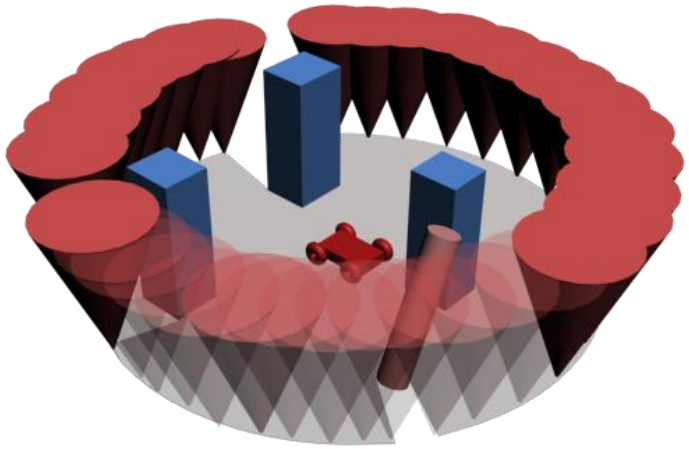


Figure 2.5 : Modèle conservatif du futur pour le scénario de la figure 2.3.a (représenté partiellement à des fins de visualisation).

Dans le cas particulier où le système de perception de A ne fait pas la différence entre les objets fixes et mobiles et qu'il n'a aucune information sur leur comportement futur, le minimum d'information à avoir sur l'environnement est une borne supérieure sur la vitesse des objets (sinon il est impossible de construire un modèle conservatif et utile du futur). Dans ce cas, chaque point de ∂FoV est modélisé comme un disque qui croit à travers le temps (un cône dans l'espace × temps). Notons que dans un tel modèle du futur, la région de WS qui est libre d'objets au moment de la détection, i.e. FoV, rétrécie graduellement et éventuellement disparaît (voir figure 2.6). $FoV(t)$ désigne la région de WS qui est libre d'objets au temps t

dans le modèle conservatif du futur et $\partial FoV(t)$ sa limite. Ce qui est important est de construire un modèle conservatif de manière à ce que la propriété de la sûreté de mouvement obtenue soit réellement garantie peu importe ce qui se passe dans l'environnement.

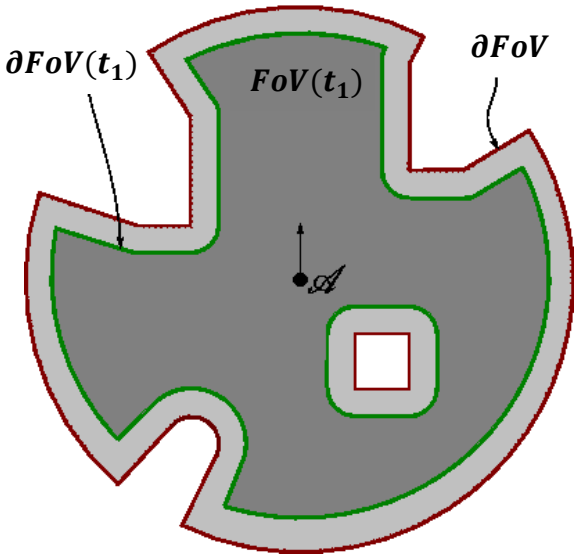


Figure 2.6 : Modèle conservatif du futur à l'instant t_1 (comment FoV rétrécit à travers le temps).

2.4 La sûreté de mouvement absolue vs passive

2.4.1 Sûreté absolue: est-elle possible?

Etant donné un modèle du futur, le problème de la sûreté de mouvement dans cette thèse est résolu en s'inspirant du concept *des états de collision inévitable* (ICS) (*inevitable collision state*) [11, 180]. Deux points sont mis en évidence :

(1) la sûreté de mouvement nécessite non seulement que la trajectoire du robot soit *sans collision* mais également doit être *sans ICS*, i.e. le robot doit être toujours dans un état où une trajectoire d'évitement est disponible.

(2) la sûreté de mouvement doit être toujours définie par rapport au modèle du futur utilisé.

ICS garantit une sûreté de mouvement absolue dans le sens où ; pour un état qui n'est pas ICS, il doit exister une trajectoire sans collision de durée infinie. Cependant, dans des cas semblables au notre où l'environnement est dynamique et inconnu, et la perception du robot se limite à son champ de vision, ce concept n'est plus valable. En effet, pour un objet mobile ayant un comportement futur inconnu dont le modèle est conservatif (comme décrit précédemment), à un certain point dans le futur, l'espace de travail sera entièrement couvert par les disques croissants. La résolution de ce problème constitue un défi du fait que le robot sera bloqué. Car, l'ensemble de l'espace des états du robot devient interdit et il sera alors impossible de trouver une trajectoire sans collision de durée infinie. C'est une situation où le concept ICS devient inefficace. La majorité des approches de navigation traitant le problème de sûreté de mouvement (comme décrit dans le paragraphe 2.1) supposent une connaissance a priori sur le comportement futur de l'environnement y compris le concept d'ICS. La réponse à ce challenge est de se contenter d'un niveau plus faible de sûreté de mouvement ; étant raisonnable, *mieux vaut garantir moins que ne rien garantir*. Même si c'est un niveau plus faible, le plus important est que la sûreté est garantie malgré les contraintes sévères imposées par le champ de vision limité du robot. Nous présentons ce niveau de sûreté dans ce qui suit.

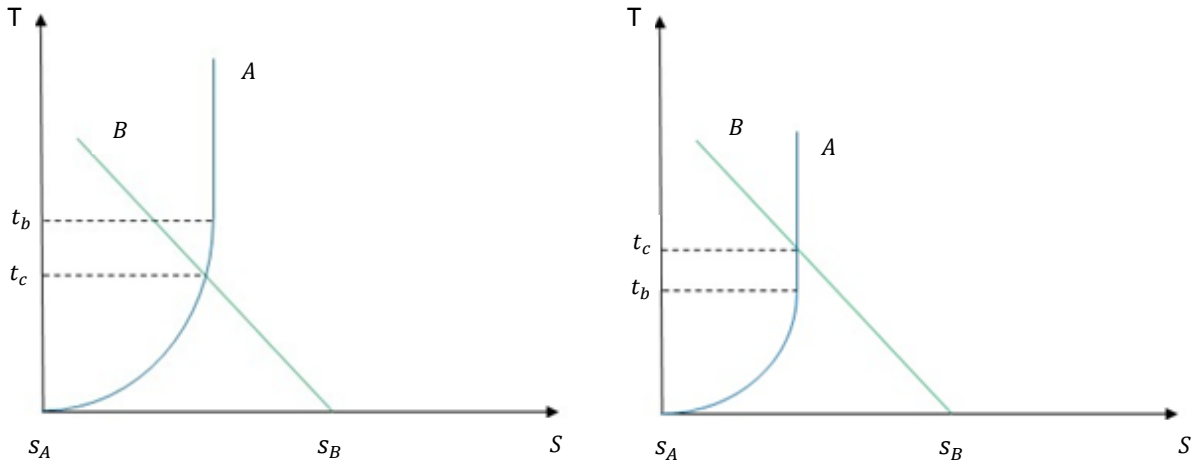
2.4.2 La sûreté de mouvement passive

Ce niveau de sûreté consiste à garantir que, si une collision a lieu, le robot sera à l'arrêt. Autrement dit, si une collision est inévitable, il peut être garanti que A aura toujours la possibilité de freiner et de s'arrêter avant que la collision ait lieu. Cela est une forme de sûreté passive dans un sens où A ne va jamais activement entrer en collision avec un objet. C'est pourquoi ce niveau de sûreté est appelé *sûreté de mouvement passive* (*passive motion safety*). Cette forme de sûreté a été proposée à l'origine par Macek et al. [181], où l'idée de base a été définie. Cependant, contrairement à [181], nous cherchons à garantir la sûreté de mouvement passive en

traitant les problèmes du champ de vision limité, des régions occultées et d'un comportement futur inconnu des obstacles. Par exemple, dans une situation réaliste, où le robot utilise uniquement ses capteurs embarqués, la sûreté de mouvement passive ne peut pas être garantie si un obstacle inattendu surgit d'une région occultée et qu'il n'était pas pris en considération. De plus, pour les approches que nous avons développées dans ce travail, la garantie de sûreté passive est formellement établie.

La sûreté de mouvement passive est alors assurée par une nouvelle version d'ICS qui est *ICS de freinage* (*braking ICS*) noté ICS^b (qui sera étudié dans le chapitre suivant) relativement à l'action de freinage pour mettre le robot à l'arrêt. Nous proposons ainsi la définition suivante :

Définition 1 (*sûreté passive*): *étant donné un modèle de l'évolution future de l'espace de travail, un état passivement sûr ou p-sûr pour A est un état s pour lequel il existe une trajectoire de freinage dont l'état initial est s et qui est sans collision jusqu'à ce que A s'arrête.*



(a) (b)

Figure2.7 : Deux scénarii correspondant à un objet mobile B approchant le robot A. (a) l'état s_A est non-p-sûr, (b) l'état s_A est p-sûr.

Un exemple illustratif est donné dans la figure 2.7, le comportement du robot A et d'un objet mobile B sont représentés dans le plan 1D. Dans la figure 2.7.a, malgré le freinage de A, une collision a lieu avant son arrêt ($t_c < t_b$, avec t_b le temps de freinage et t_c le temps de collision). Par conséquent, l'état de A; s_A est considéré comme *non-p-sûr*. Par contre, dans la figure 2.7.b, la trajectoire du robot est sans collision jusqu'à ce que A soit à l'arrêt. Bien qu'il y ait une collision après ($t_c > t_b$), elle n'est pas activement causée par A, donc s_A est considéré comme *p-sûr*.

Le chapitre 3 traite plus en détail ce concept de sûreté passive.

2.5 Conclusion

Dans ce chapitre, nous avons donné quelques réponses au problème de la sûreté de mouvement pour un robot mobile ayant un champ de vision limité et navigant dans des environnements inconnus traitant les obstacles fixes et mobiles avec un comportement futur inconnu. Nous avons proposé un niveau de sûreté plus faible quant à une garantie absolue (connaissance à l'infini) mais plus fort quant au contraintes imposées (champ de vision, futur inconnu), qui est la *sûreté de mouvement passive*. Elle est garantie par le concept *ICS de freinage* qui est étudié en détail dans le chapitre suivant.

CHAPITRE 3

ETATS DE COLLISION INEVITABLE DE FREINAGE : CONCEPT ET APPLICATION

L'objectif ultime de tout travail de recherche visant à concevoir un système robotique mené à imiter le comportement de l'être humain ou à le côtoyer, est de garantir la sûreté du mouvement du système à l'infini (sûreté absolu). Cependant, comme expliqué dans le chapitre précédent, cela ne peut pas être réalisé quand le système se déplace dans un environnement dynamique inconnu. C'est pour tenir compte des contraintes imposées par ce type d'environnement que nous avons développé un nouveau niveau de garantie ; la sûreté de mouvement passive. Le fondement de base du travail réalisé dans cette thèse est présenté dans ce chapitre. Il s'articule autour du concept d'*Etats de Collision Inévitable de freinage (ICS de freinage)*. La garantie de la sûreté du mouvement est basée sur ce concept, ce qui permet de concevoir un système de navigation passivement sûr. Ce concept s'inspire principalement du concept général des ICS, qui est présenté dans le deuxième paragraphe de ce chapitre où l'idée de base et les travaux en relation sont abordés. Par la suite, notre concept ICS de freinage est illustré de manière formelle, où les différentes définitions et propriétés sont fournies. Il est aussi montré que notre concept vérifie bien les trois critères généraux de sûreté présentés dans le chapitre précédent. Un exemple illustratif est donné afin de montrer l'aspect applicatif. Afin de vérifier si un état est ICS de freinage ou non, l'algorithme ICS^{b}-CHECK a été développé, il est illustré en détail dans ce chapitre. Il permet d'implémenter le concept ICS de freinage pour n'importe quel système de navigation.

3.1 Notation

La dynamique du robot mobile A est généralement décrite par l'équation différentielle de la forme :

$$\dot{s} = f(s, u) \qquad (3.1)$$

Où $s \in S$ est l'état de A, $\dot{s}$ sa dérivée par rapport au temps et $u \in U$ un contrôle. S et U désigne respectivement l'espace d'état et l'espace de contrôle de A. Soit $A(s)$ le sous-ensemble fermé de l'espace de travail WS occupé par A à l'état s.

Soit $\tilde{u}: [0, t_f] \rightarrow U$ une trajectoire de contrôle, i.e. une séquence temporelle de contrôles, t_f est la durée de $\tilde{u}$. L'ensemble de toutes les trajectoires de contrôle possibles est noté $\widetilde{U}$. En démarrant d'un état initial s_0, une trajectoire d'état $\tilde{s}$, i.e. une séquence temporelle d'états, est dérivée à partir d'une trajectoire de contrôle $\tilde{u}$ en intégrant l'équation (3.1); $\tilde{s}(s_0, \tilde{u}, t)$ désigne l'état atteint à l'instant t.

Une trajectoire de contrôle $\tilde{u}_b \in \widetilde{U}$, pour laquelle $\tilde{s}_b(s_0, \tilde{u}_b, t_b)$ est un état où A est à l'arrêt, est une trajectoire de freinage pour s_0 et t_b est son *temps de freinage.* L'ensemble de toutes les trajectoires de freinage possibles pour s_0 est noté $\widetilde{U}_b^{s_0}$.

Dans une situation semblable à celle de la figure 2.3, le sous ensemble FoV est la partie libre de l'espace de travail, alors que ∂FoV^f, ∂FoV^m, ∂FoV^u et FoV^c représentent les objets perçus et non perçus. Soit B_i le modèle espace × temps de l'évolution future des objets correspondants (selon les règles de modélisations définie dans le paragraphe 2.3.4 du chapitre 2). A l'instant 0 ; l'instant de détection, $B_i(0)$ correspond à un sous-ensemble de ∂FoV^f, ∂FoV^m, ∂FoV^u ou FoV^c. $B_i(t)$ désigne le sous ensemble de WS occupé par B_i à un instant t dans le modèle du futur. Il est supposé que $B_i(t)$ est un sous-ensemble fermé de WS et que le nombre total d'objets est n. De même, $B_i([t_1, t_2])$ désigne la région espace × temps occupée par l'objet pendant l'intervalle de temps $[t_1, t_2]$. Pour simplifier les notations, il est supposé que $B_i \equiv B_i([0, \infty))$.

3.2 Définition du concept "Etats de Collision Inévitable (ICS)"

Auparavant, la sûreté de mouvement se résumait à résoudre un problème d'évitement d'obstacles. Cependant, dans certains cas cela n'est pas suffisant. Par

exemple, dans le cas d'une voiture se déplaçant avec une très grande vitesse, confrontée à un mur, elle peut se retrouver dans une situation où même si elle n'est pas en collision dans le temps présent, il est impossible de contourner le mur ou de s'arrêter. La collision est alors inévitable quelle que soit la décision prise. C'est pourquoi il faut plus qu'un simple test de non collision pour garantir la sûreté du mouvement. Une solution adaptée à ce type de problème est le concept des *états de collision inévitable* (*Inevitable Collision States*) (ICS) développé dans [11], qui est un concept proche des deux concepts *ombre d'obstacles* [182] et *régions de collision inévitable* [183]. Il est aussi lié aux concepts : *noyaux de viabilité* [184], *ensembles atteignables arrières* [185], et *Barrier Certificates* [186]. Un ICS est un état pour lequel quelle que soit la trajectoire future du robot, une collision finit par se produire. L'ensemble des ICS définit une région de l'espace des états qui représente, d'un point de vue sûreté, une région qui doit être évitée par le robot.

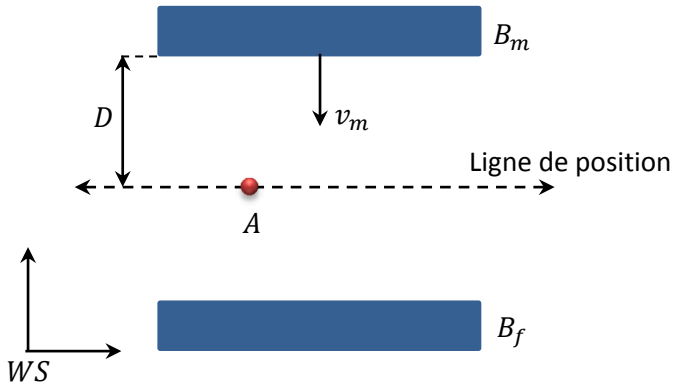
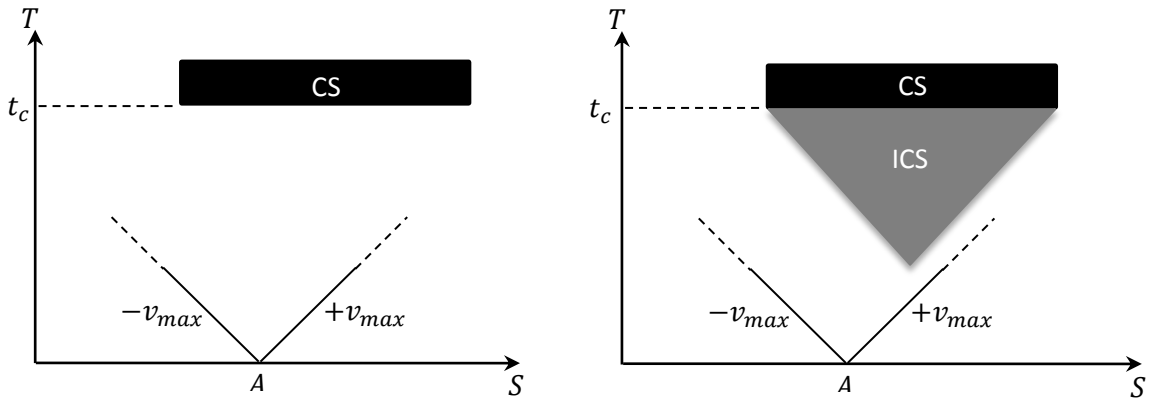


Figure 3.1: Scénario du compacteur.

Un simple exemple peut illustrer le principe des ICS, à savoir *le scénario du compacteur* (figure 3.1). Ce dernier est modélisé en 2D par deux plaques ; une qui est fixe (B_f) et l'autre qui est mobile (B_m) ayant une vitesse constante v_m et se déplaçant vers B_f et le robot A est placé entre les deux plaques. Pour simplifier le problème, nous supposons que A est traité comme un robot 1D qui peut se déplacer uniquement sur l'axe horizontal (appelé ligne de position). Dans ce cas l'unique moyen pour que A puisse sortir du compacteur et éviter d'être écrasé est de se

déplacer soit vers la gauche ou vers la droite jusqu'à ce qu'il sort du compacteur. En supposant que A est une masse ponctuelle contrôlée par sa vitesse (tel que $v \leq v_{max}$), sa dynamique est donnée par $\dot{s} = v$. L'espace d'état-temps de A est à deux dimensions ; position et temps. Durant son déplacement, B_m se croise avec la ligne de position à partir d'un temps de collision $t_c = D/v_m$ (avec D la distance entre A et B_m) pour une durée qui dépend de l'épaisseur de B_m et de sa vitesse v_m, résultant en un ensemble d'états-temps où le robot A entre en collision avec la plaque mobile B_m. Cet ensemble représente l'ensemble des états de collision CS (le rectangle étiqueté CS dans la figure 3.2.a) qui désigne la région interdite que le robot doit éviter. Cependant, cela n'est pas suffisant pour caractériser la région interdite pour le robot. La figure 3.2.b illustre, qu'en tenant compte d'une vitesse bornée du robot, il existe d'autres états (mis à part l'ensemble CS) pour lesquels; quel que soit le comportement du robot, une collision aura lieu dans le futur entre le robot A et la plaque mobile B_m. Ces états sont désignés par la région sous forme de triangle gris (en plus de l'ensemble CS) dans la figure 3.2.b. Ils représentent l'ensemble des ICS. Donc, dès que A occupe n'importe quel état de cet ensemble, il est condamné et il n'aura pas le temps de sortir du compacteur.



(a) (b)

Figure 3.2 : Représentation de la région interdite pour A dans le scénario de la figure 3.1: (a) états de collision (CS), (b) états de collision inévitable (ICS).

La définition de l'ensemble des ICS dépend à la fois de la dynamique du robot et de celle de l'environnement (comportement futur des obstacles). L'objectif de cet exemple est de montrer que la planification de mouvements sans collision n'est pas suffisante pour garantir la sûreté de mouvement. Il faut prendre en considération les régions ICS. Pour ce faire, garantir la sûreté de mouvement reviendrait à planifier des mouvements qui évitent les régions ICS.

3.3 ICS du point de vue de la sûreté passive

Bien que le concept ICS a été défini de manière à garantir une sûreté de mouvement absolue, il nécessite une connaissance a priori du comportement futur des obstacles mobiles. Dans des cas semblables au notre où l'environnement est inconnu, ce concept n'est plus valable.

Le problème de la sûreté de mouvement absolue a déjà été discuté dans le chapitre précédent. Il a été illustré que dans de tels types d'environnement ; dynamique et inconnu où la perception du robot se limite à son champ de vision, elle ne peut être garantie et qu'un niveau plus faible de sûreté doit être considéré.

C'est pourquoi dans cette thèse, nous avons opté pour une garantie de sûreté de mouvement passive, étant donné les contraintes imposées par l'environnement (présence d'objets non perçus, objets inattendus, etc.). Le principe de base de la sûreté de mouvement passive est de garantir que le robot soit à l'arrêt si une collision doit se produire ; d'où l'utilisation de trajectoire de freinage pour l'évaluation de la sûreté. C'est de là qu'est venue l'idée de développer un nouveau concept dérivé du concept ICS ; à savoir, le concept des *ICS de freinage* (*Braking ICS*) noté ICS^b [187, 188]. L'ensemble des ICS de freinage définit une région de l'espace d'états-temps qui représente, d'un point de vue sûreté, une région qui doit être évitée par le robot.

3.4 Définition des ICS de freinage

Un ICS de freinage (ICS^b) est informellement défini comme un état pour lequel quelle que soit la trajectoire de freinage future suivie par le robot, une collision se produit avant que A soit au repos ; d'où la définition formelle suivante :

Définition 2 (ICS de freinage) : s est un ICS^b si $\forall \tilde{u}_b \in \widetilde{U}_b^s, \exists t \in [0, t_b[,\ \tilde{s}(s, \tilde{u}_b, t)$ *est un état de collision à l'instant t.*

Il est à noter que lorsque A est dans un état où il est à l'arrêt, $\widetilde{U}_b^s$ se réduit à $\tilde{u}_b^0$ qui désigne la trajectoire de freinage où un contrôle nul est appliqué à A. Par conséquent A est toujours p-sûr (passivement sûr) (même si A est en collision, ce qui est dû à l'environnement).

Il est alors possible de définir l'ensemble des ICS^b donnant une collision avec un objet particulier B_i:

$$ICS^b(B_i) = \{s \in S | \forall \tilde{u}_b \in \widetilde{U}_b^s, \exists t \in [0, t_b[, A(\tilde{s}(s, \tilde{u}_b, t)) \cap B_i(t) \neq \emptyset\} \quad (3.2)$$

De même, l'ensemble des ICS^b donnant une collision avec l'ensemble des obstacles B:

$$ICS^b(B) = \{s \in S | \forall \tilde{u}_b \in \widetilde{U}_b^s, \exists t \in [0, t_b[, A(\tilde{s}(s, \tilde{u}_b, t)) \cap B(t) \neq \emptyset\} \quad (3.3)$$

L'ensemble ICS^b donnant une collision avec B_i pour une trajectoire donnée $\tilde{u}_b$ est définie comme suit :

$$ICS^b(B_i, \tilde{u}_b) = \{s \in S | \exists t \in [0, t_b[, A(\tilde{s}(s, \tilde{u}_b, t)) \cap B_i(t) \neq \emptyset\} \quad (3.4)$$

De même, l''ensemble ICS^b donnant une collision avec l'ensemble des obstacles B pour une trajectoire donnée $\tilde{u}_b$ est :

$$ICS^b(B, \tilde{u}_b) = \{s \in S | \exists t \in [0, t_b[, A(\tilde{s}(s, \tilde{u}_b, t)) \cap B(t) \neq \emptyset\} \quad (3.5)$$

Enfin, l'ensemble ICS^b donnant une collision avec B_i pour une trajectoire donnée $\tilde{u}_b$ à un instant t est :

$$ICS^b(B_i, \tilde{u}_b, t) = \{s \in S | A(\tilde{s}(s, \tilde{u}_b, t)) \cap B_i(t) \neq \emptyset\} \qquad (3.6)$$

3.5 Propriétés des ICS de freinage

La première propriété peut être établie à partir des deux équations (3.3) et (3.5). Elle montre que $ICS^b(B)$ peut être dérivé à partir de $ICS^b(B, \tilde{u}_b)$ pour chaque trajectoire de freinage possible $\tilde{u}_b$.

Propriété 1 (Intersection des entrées de contrôle)

$$ICS^b(B) = \bigcap_{\tilde{u}_b \in \tilde{U}_b} ICS^b(B, \tilde{u}_b)$$

Preuve:

$s \in ICS^b(B) \Leftrightarrow \forall \tilde{u}_b \in \tilde{U}_b^s, \exists t \in [0, t_b[, A(\tilde{s}(s, \tilde{u}_b, t)) \cap B(t) \neq \emptyset$

$\Leftrightarrow \forall \tilde{u}_b \in \tilde{U}_b^s, s \in ICS^b(B, \tilde{u}_b)$

$\Leftrightarrow s \in \bigcap_{\tilde{u}_b \in \tilde{U}_b} ICS^b(B, \tilde{u}_b)$

La propriété suivante quant à elle montre que $ICS^b(B, \tilde{u}_b)$ peut être dérivé à partir de $ICS^b(B_i, \tilde{u}_b)$ pour chaque objet B_i.

Propriété 2 (Union des obstacles) :

$$ICS^b\left(\bigcup_{i=1}^{n} B_i, \tilde{u}_b\right) = \bigcup_{i=1}^{n} ICS^b(B_i, \tilde{u}_b)$$

Preuve :

$$s \in ICS^b\left(\bigcup_{i=1}^{n} B_i, \tilde{u}_b\right) \Leftrightarrow \exists t \in [0, t_b[, A(\tilde{s}(s, \tilde{u}_b, t)) \cap \bigcup_{i=1}^{n} B_i(t) \neq \emptyset$$

$\Leftrightarrow \exists B_i, \exists t \in [0, t_b[, A(\tilde{s}(s, \tilde{u}_b, t)) \cap B_i(t) \neq \emptyset$

$\Leftrightarrow \exists B_i, s \in \ ICS^b(B_i, \tilde{u}_b)$

$\Leftrightarrow s \in \bigcup_{i=1}^{n} ICS^b(B_i, \tilde{u}_b)$

La combinaison des propriétés 1 et 2 permet d'établir la propriété suivante. Cette propriété permet de dériver $ICS^b(B)$ à partir de $ICS^b(B_i, \tilde{u}_b)$ pour chaque objet B_i et chaque trajectoire de freinage $\tilde{u}_b$.

Propriété 3 (caractérisation de ICS^b) :

$$ICS^b(B) = \bigcap_{\tilde{u}_b \in \tilde{U}_b} \bigcup_{i=1}^{n} ICS^b(B_i, \tilde{u}_b)$$

Preuve :

$$ICS^b(B) = ICS^b\left(\bigcup_{i=1}^{n} B_i\right)$$

En se basant sur la propriété 1, il peut être établi que :

$$ICS^b(B) = \bigcap_{\tilde{u}_b \in \tilde{U}_b} ICS^b\left(\bigcup_{i=1}^{n} B_i\ , \tilde{u}_b\right)$$

En se basant sur la propriété 2, il peut être établi que:

$$ICS^b(B) = \bigcap_{\tilde{u}_b \in \tilde{U}_b} (\bigcup_{i=1}^{n} ICS^b(B_i, \tilde{u}_b))$$

La propriété 1 a montré que $ICS^b(B)$ est calculée à partir de $ICS^b(B, \tilde{u}_b)$ pour chaque trajectoire de freinage $\tilde{u}_b \in \tilde{U}_b$. L'ensemble $\tilde{U}_b$ peut contenir une infinité de trajectoires de freinage possibles. D'un point de vue pratique, cela peut être très

coûteux surtout pour des applications en temps réel. Heureusement, il est possible d'établir une propriété qui permet de calculer une approximation conservative de $ICS^b(B)$ en utilisant uniquement un sous-ensemble de l'ensemble des entrées de contrôle possibles.

Propriété 4 (approximation de ICS^b) :

$$ICS^b(B) \subseteq ICS^b(B, \mathcal{E})$$

Avec $\mathcal{E} \subseteq \widetilde{U}_b$, un sous-ensemble de l'ensemble des trajectoires de freinage possibles.

Preuve :

Soit $\widetilde{U}_b = \mathcal{E} \cup \widetilde{U}_b \backslash \mathcal{E}$

$$ICS^b(B) = \bigcap_{\tilde{u}_b \in \mathcal{E}} ICS^b(B, \tilde{u}_b) \cap \bigcap_{\tilde{u}_b \in \widetilde{U}_b \backslash \mathcal{E}} ICS^b(B, \tilde{u}_b)$$

$$\subseteq \bigcap_{\tilde{u}_b \in \mathcal{E}} ICS^b(B, \tilde{u}_b)$$

Une des particularité du concept ICS est que des trajectoires de durée infinie sont vérifiées pour être sans collision ou non, i.e. l'horizon temporel est infini (c'est cet horizon temporel qui garantit la sûreté), tandis que, ICS^b considère des trajectoires $\tilde{u}_b$ de durée finie. La vérification de collision est limitée à l'intervalle de temps $[0, t_b[$, avec t_b le temps de freinage de $\tilde{u}_b$. t_b est le rapport entre la vitesse linéaire du robot et son accélération linéaire maximale. Pour un sous-ensemble arbitraire $\mathcal{E}$ de l'ensemble des trajectoires de freinage possibles, il existe un horizon temporel fini :

$$T_h = \max_{\tilde{u}_b \in \mathcal{E}} \{t_b\} \qquad (3.7)$$

T_h est un horizon temporel valide dans la mesute où ; afin de calculer $ICS^b(B, \mathcal{E})$, il suffit de considérer le modèle du futur jusqu'à un temps T_h. Cela est établi par la propriété suivante :

Propriété 5 (horizon temporel de ICS^b):

$$ICS^b(B, \mathfrak{E}) = ICS^b(B([0, T_h[), \mathfrak{E})$$

Preuve :

Cette propriété découle de la définition même de ICS^b, qui pour une trajectoire de freinage, il concerne uniquement les collisions qui ont lieu avant le temps $t_b < T_h$.

Rapellons que pour un robot qui a un champ de vision limité, B comprend une partie non perçue de WS: $\partial\mathrm{FoV}$ et $\mathrm{FoV}^{\mathrm{c}}$. Du point de vue de la sûreté de mouvement, la propriété suivante est très importante du moment qu'elle permet d'établir que FoV^c peut être ignoré dans le calcul de $ICS^b(B)$. Autrement dit, la considération de ∂FoV à lui seul est suffisante pour garantir la sûreté du mouvement.

Propriété 6 (limites du champ de vision) :

$$ICS^b(B) = ICS^b(\partial FoV \cup FoV^c) = ICS^b(\partial FoV)$$

Preuve :

La preuve de l'égalité entre $ICS^b(B)$ et $ICS^b(\partial FoV)$ est établie en deux étapes. Soit s, un état sans collision dont sa position correspondante est située à l'intérieur de FoV et tel que $s \in ICS^b(\partial FoV)$. Selon la propriété 2, il en découle que :

$\forall B_i, \forall B_j, ICS^b(B_i) \subseteq ICS^b(B_i \cup B_j)$

Par conséquent :

$s \in ICS^b(\partial FoV) \Rightarrow s \in ICS^b(\partial FoV \cup FoV^c)$

On suppose maintenant que $s \in ICS^b(FoV^c)$, ce qui veut dire que $\forall \tilde{u}_b \in \tilde{U}_b^s, \exists t \in [0, t_b[$ tel que $\tilde{s}(s, \tilde{u}_b, t)$ est en collision avec un point de $FoV^c(t)$. Du moment que s est situé à l'intérieur de FoV, il suffit d'un simple argument topologique pour réaliser que $\exists t' < t$ tel que $\tilde{s}(s, \tilde{u}_b, t')$ est en collision avec un point de $\partial FoV(t')$. Par conséquent, $s \in ICS^b(\partial FoV)$ et l'expression suivante est alors vérifiée :

$s \in ICS^b(\partial FoV \cup FoV^c) \Rightarrow s \in ICS^b(\partial FoV)$

En d'autre termes, il suffit de considérer $\partial \mathrm{FoV}$ afin de calculer $\mathrm{ICS}^{\mathrm{b}}(\mathrm{B})$.

3.6 ICS^b et les critères de sûreté

Dans le chapitre précédent (§2.2), nous avons présenté les trois critères de sûreté qui doivent être vérifiés par tout système de navigation. Si l'un de ces critères est violé le risque de collision est inévitable entre le robot et les objets de l'environnement. Dans notre cas, nous avons développé le concept ICS de freinage (ICS^b) afin de garantir la sûreté passive du mouvement, et qui constitue en fait le noyau de notre système de navigation. Grâce aux définitions et propriétés de ICS^b, il est possible de montrer que ce concept vérifie bien ces trois critères:

A partir de la définition même de ICS^b dans l'équation 3.2, il est clair que la région non p-sûre est définie à partir de $A(\tilde{s}(s, \tilde{u}_b, t))$ et $B_i(t)$ pour un intervalle de temps bien défini.

$A(\tilde{s}(s, \tilde{u}_b, t))$ désigne le sous-ensemble fermé de l'espace de travail WS occupé par le système A quand il est à l'état atteint à l'instant t en démarrant de s et en suivant la trajectoire $\tilde{u}_b$ et cela en intégrant l'équation différentielle de la dynamique de A (équation 3.1). Le premier critère de sûreté sur *la prise en compte de la dynamique* du robot est alors vérifié.

B_i désigne le modèle espace × temps de l'évolution future de l'objet en question (i.e modèle conservatif du futur), d'où $B_i(t)$ est le sous-ensemble de WS occupé par B_i à un instant t de ce modèle (où t appartient à un intervalle de temps bien défini). Le deuxième critère de sûreté concernant *le raisonnement sur le futur* est donc aussi vérifié.

Enfin, le dernier critère revient à utiliser un horizon temporel approprié. Le principe est déjà ancré dans le concept même de ICS^b, qui considère des trajectoires de durée finie. La vérification de collision est alors limitée à un intervalle de temps bien défini, à savoir $[0, T_h[$. La valeur de l'horizon temporel T_h est donné dans

l'équation 3.6. Le modèle du futur est alors considéré jusqu'à un temps T_h, ce qui est clairement illustré par la propriété 5.

3.7 Exemple d'application

Afin d'illustrer les définitions et propriétés introduites précédemment, un exemple simple est exploité. Il montre comment ICS^b est calculé pour différents types d'objets : statique ou mobile, pour différentes formes de l'objet : ponctuel, rectangle, cercle et enfin pour un seul objet ou un ensemble d'objets. Notons, cependant, que cet exemple traite principalement le côté représentation schématique de ICS^b sans se préoccuper de la précision du calcul ou de celle de la représentation, car l'objectif est de comprendre le principe de base. ICS^b est calculé rigoureusement dans la partie résultats (dans le chapitre 5).

Le système considéré est une masse ponctuelle A qui se déplace selon deux trajectoires de freinage possibles uniquement ($\tilde{u}_1$ et $\tilde{u}_2$). Le système A peut respectivement soit freiner avec un angle de braquage nul soit avec un angle de braquage constant positif jusqu'à ce qu'il s'arrête. Notons, que grâce à la propriété 4, il est possible de considérer seulement deux trajectoires au lieu d'une infinité de trajectoires de freinage possibles pour le calcul de ICS^b.

3.7.1 Cas d'un point statique

Soit B un point statique, ICS^b est calculé à partir de $ICS^b(B, \tilde{u}_{b_1})$ (représentée par la ligne en gris clair dans la figure 3.3) et $ICS^b(B, \tilde{u}_{b_2})$ (représentée par l'arc en gris foncé dans la figure 3.3) en se basant sur la propriété 1.

$$ICS^b(B) = ICS^b(B, \tilde{u}_{b_1}) \cap ICS^b(B, \tilde{u}_{b_2}) = B$$

L'ensemble des ICS^b est l'état correspondant à l'objet B. Par conséquent, A peut toujours éviter la collision à moins qu'il soit déjà en collision avec B.

Notons que, $ICS^b(B, \tilde{u}_{b_i}), i = 1,2$ est calculé en utilisant la définition de ICS^b (équation 3.4). Pour chaque pas de temps, le point de collision entre la trajectoire du système A et la trajectoire de l'objet B parcourues jusqu'à cet instant représente un état de collision. L'ensemble de ces états représente $ICS^b(B, \tilde{u}_{b_i})$.

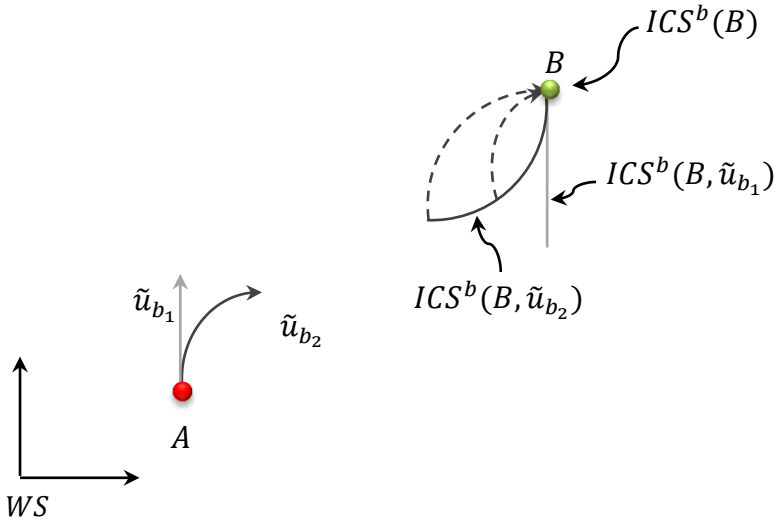


Figure 3.3 : Calcul de ICS^b pour un point statique.

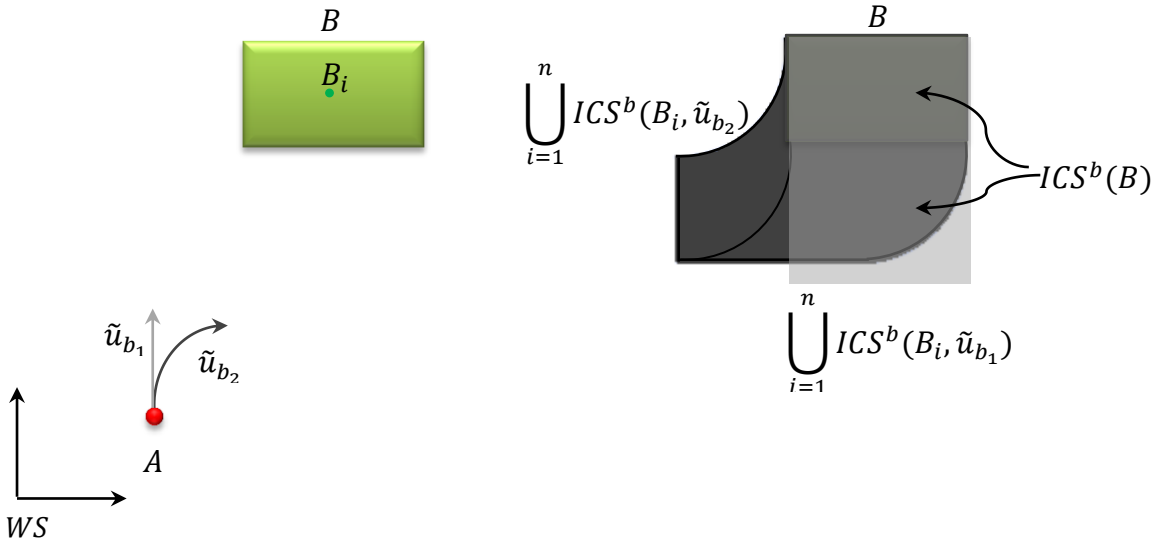
3.7.2 Cas d'un objet statique

Toujours dans le cas statique, mais ici l'obstacle n'est pas un point mais un objet de forme arbitraire (un rectangle par exemple (figure 3.4.a)) et qui représente l'union d'un ensemble de points $B = \cup_{i=1}^{n} B_i$. Dans ce cas, l'ensemble ICS^b est calculé à partir de ICS^b d'un point de cet objet en utilisant la propriété 2.

$$ICS^b(B) = ICS^b(\bigcup_{i=1}^{n} B_i) = ICS^b(\bigcup_{i=1}^{n} B_i, \tilde{u}_{b_1}) \cap ICS^b(\bigcup_{i=1}^{n} B_i, \tilde{u}_{b_2})$$

$$= \bigcup_{i=1}^{n} ICS^b(B_i, \tilde{u}_{b_1}) \cap \bigcup_{i=1}^{n} ICS^b(B_i, \tilde{u}_{b_2})$$

$$= ICS^b(B, \tilde{u}_{b_1}) \cap ICS^b(B, \tilde{u}_{b_2})$$



(a) Scenario avec un rectangle statique. (b) ICS^b

Figure 3.4 : Calcul de ICS^b pour un objet statique.

De cette manière, l'ensemble des états ICS^b est calculé pour chaque point B_i et pour une trajectoire de freinage donnée. L'union des ensembles correspondant à tous les points B_i représente ICS^b de l'obstacle B pour une trajectoire de freinage donnée. Par conséquent, deux ensembles en résultent $ICS^b(B, \tilde{u}_{b_1})$ et $ICS^b(B, \tilde{u}_{b_2})$ (voir figure 3.4.b). L'intersection de ces deux ensembles représente ICS^b de l'obstacle B (la région interdite pour le système A).

3.7.3 Cas d'un point mobile

Dans cet exemple, il est supposé que l'obstacle est un point mobile ayant une vitesse linéaire constante. ICS^b dépend alors de la vitesse relative entre A et B, ce qui est montré dans la figure 3.5. Soit $B = \bigcup_t B(t)$, la propriété 2 peut alors être exprimée de la même manière que pour un ensemble d'objets afin de calculer ICS^b. Cependant, selon la propriété 5 le temps ne peut être considéré à l'infini mais il est limité par un horizon temporel T_h. Dans ce cas, la trajectoire de B est considérée uniquement jusqu'à T_h. ICS^b de B est alors calculé comme suit :

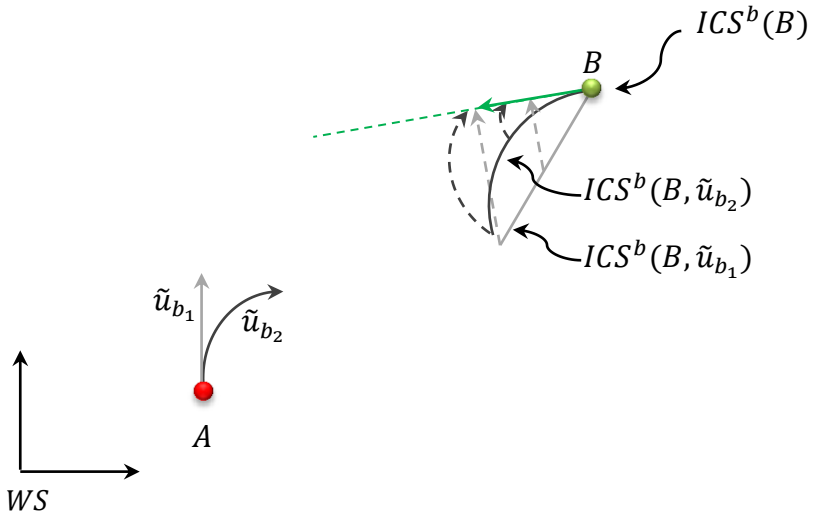


Figure 3.5 : Calcul de ICS^b pour un point mobile.

$$ICS^b(B) = ICS^b(\bigcup_{t=0}^{T_h} B(t)) = ICS^b(\bigcup_{t=0}^{T_h} B(t), \tilde{u}_{b_1}) \cap ICS^b(\bigcup_{t=0}^{T_h} B(t), \tilde{u}_{b_2})$$

$$= \bigcup_{t=0}^{T_h} ICS^b(B(t), \tilde{u}_{b_1}) \cap \bigcup_{t=0}^{T_h} ICS^b(B(t), \tilde{u}_{b_2}) = B(0)$$

La région interdite correspond à l'état du point B à $t = 0$. Donc, il suffit que A se situe n'importe où à part dans cet état pour éviter une collision certaine.

3.7.4 Cas d'un objet mobile

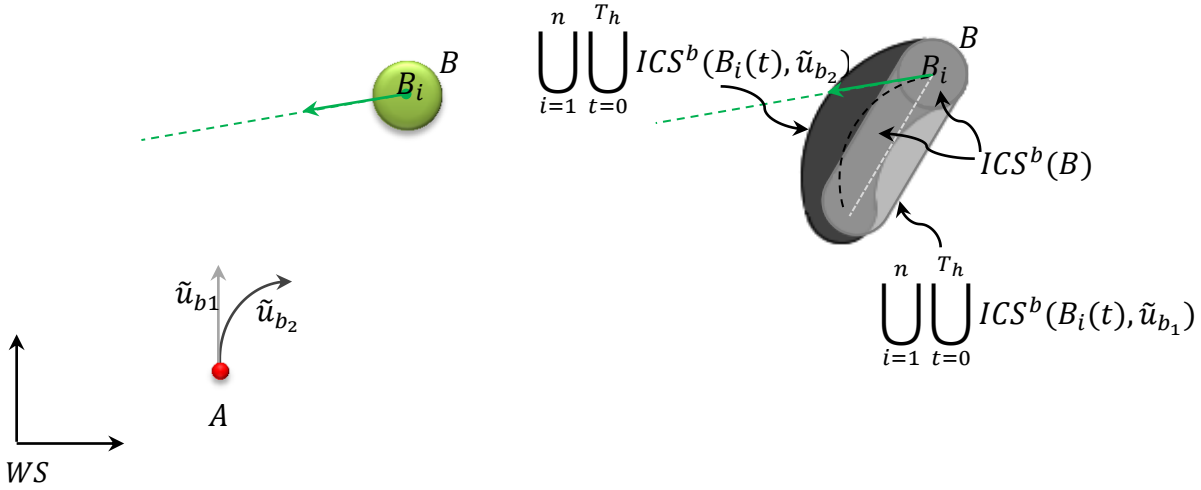
L'objet B est maintenant un disque mobile (figure 3.6.a). De même que dans le cas précédant, il a une vitesse linéaire supposée constante, considérée sur l'intervalle de temps $[0, T_h[$. L'objet B est l'union des points B_i. Par conséquent, B peut être exprimé par $B = \cup_{i=1}^{n} \cup_{t=0}^{T_h} B_i(t)$.

Selon les propriétés 1 et 2, ICS^b correspond à:

$$ICS^b(B) = ICS^b(\bigcup_{i=1}^{n}\bigcup_{t=0}^{T_h} B_i(t)) = ICS^b(\bigcup_{i=1}^{n}\bigcup_{t=0}^{T_h} B_i(t), \tilde{u}_{b_1}) \cap ICS^b(\bigcup_{i=1}^{n}\bigcup_{t=0}^{T_h} B_i(t), \tilde{u}_{b_2})$$

$$= \bigcup_{i=1}^{n} \bigcup_{t=0}^{T_h} ICS^b(B_i(t), \tilde{u}_{b_1}) \cap \bigcup_{i=1}^{n} \bigcup_{t=0}^{T_h} ICS^b(B_i(t), \tilde{u}_{b_2})$$

$$= ICS^b(B, \tilde{u}_{b_1}) \cap ICS^b(B, \tilde{u}_{b_2})$$



(a) Scenario avec un disque mobile. (b) ICS^b

Figure 3.6 Calcul de ICS^b pour un objet mobile.

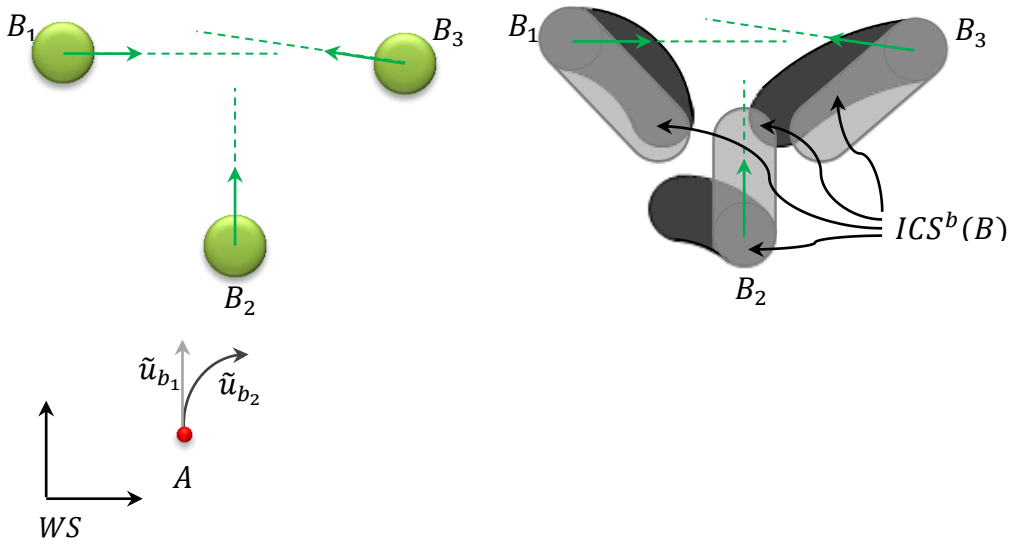
Le résultat est représenté dans la figure 3.6.b, où la région interdite résulte de l'intersection des deux bandes correspondant à $ICS^b(B, \tilde{u}_{b_1})$ et $ICS^b(B, \tilde{u}_{b_2})$ (représentées respectivement en gris clair et gris foncé). Cette région représente les états non sûrs que A doit éviter.

3.7.5 Cas de plusieurs objets

En général, l'environnement peut contenir plusieurs obstacles mobiles. Toujours à titre d'exemple, le scénario de la figure 3.7.a contient trois objets mobiles ($j = 1, \dots m$), avec $m = 3$, où les trois objets ont une vitesse linéaire constante. Supposons qu'ils contiennent le même nombre de points $i = 1, \dots n$. B, qui représente le modèle (espace × temps) de l'évolution future de l'environnement, peut être exprimé par $B = \bigcup_{j=1}^{m} \bigcup_{i=1}^{n} \bigcup_{t=0}^{T_h} B_{ji}(t)$. L'ensemble ICS^b de B devient alors:

$$ICS^b(B) = \bigcup_{j=1}^{m} \bigcup_{i=1}^{n} \bigcup_{t=0}^{T_h} ICS^b(B_i(t), \tilde{u}_{b_1}) \cap \bigcup_{j=1}^{m} \bigcup_{i=1}^{n} \bigcup_{t=0}^{T_h} ICS^b(B_i(t), \tilde{u}_{b_2})$$

Le résultat est illustré dans la figure 3.7.b, la zone d'intersection représente les états non-sûrs (ICS de freinage). Comme pour les cas précédents, c'est le résultat de l'intersection des deux ensembles de couleurs gris clair et gris foncé (correspondant respectivement à $ICS^b(B, \tilde{u}_{b_1})$ et $ICS^b(B, \tilde{u}_{b_2})$). Seulement dans ce cas, chaque ensemble ICS^b a été calculé pour plusieurs objets (au lieu d'un seul).



(a) Scenario avec trois disques mobiles. (b) ICS^b

Figure 3.7 : Calcul de ICS^b pour un environnement dynamique.

Dans cet exemple, des objets principalement circulaires ou rectangulaires ont été utilisés pour représenter les obstacles, mais l'approche peut être appliquée pour n'importe quel objet de forme arbitraire.

3.8 Détermination des états ICS^b

Dans les paragraphes précédents, le concept ICS^b a été formellement étudié. Cependant, notre objectif est que ce concept puisse être intégré dans des systèmes

de navigation afin de garantir la sûreté du robot. Pour ce faire, nous avons développé un *algorithme de vérification des états ICS^b* [187, 188]. Ce dernier vérifie si l'état du robot est p-sûr ou non.

3.8.1 L'algorithme général de vérification-ICS^b

Les étapes requises pour vérifier si un état s_c est un ICS^b ou non sont données dans l'algorithme 1. En plus de l'état à vérifier, l'algorithme a comme entrées B_i ; le modèle (espace × temps) de l'évolution future de l'environnement (modèle conservatif) (voir §2.3.4). L'algorithme général de vérification-ICS^b est fondé principalement sur les propriétés de ICS^b.

L'étape 1 de l'algorithme 1 vise à sélectionner un sous ensemble Ɛ de l'ensemble de trajectoires de freinage possibles, qui est utilisé pour calculer une approximation conservative de l'ensemble ICS^b. Les étapes 2, 3 et 4 sont la translation directe des propriétés 1 et 2. Enfin, dans les étapes 5 à 9 la valeur booléenne Vrai ou Faux est retournée pour désigner si l'état s_c est un ICS^b ou non.

Algorithme 1 : Algorithme général de vérification-ICS^b

Entrée: s_c, l'état à vérifier ; $B_i, i = 1 \dots n$.

Sortie : valeur booléenne.

1 Sélectionner $\mathcal{E} \subset \widetilde{U}_b^{s_c}$, un ensemble de trajectoires de freinage pour s_c ;

2 Calculer $ICS^b(B_i, \tilde{u}_b)$ pour chaque B_i et chaque $\tilde{u}_b \in \mathcal{E}$;

3 Calculer $ICS^b(B, \tilde{u}_b) = \cup_{i=1}^{n} ICS^b(B_i, \tilde{u}_b)$ pour chaque $\tilde{u}_b \in \mathcal{E}$;

4 Calculer $ICS^b(B, \mathcal{E}) = \cap_{\tilde{u}_b \in \mathcal{E}} ICS^b(B, \tilde{u}_b)$;

5 **if** $s_c \in ICS^b(B, \mathcal{E})$ **then**

6 | **return** VRAI **;**

7 **else**

8 | **return** FAUX ;

9 **end**

Malgré l'utilisation d'un sous ensemble ε de l'ensemble des trajectoires de freinage, l'algorithme général de vérification-ICS^b reste très coûteux en temps de calcul. Cela est dû d'une part, au coût de calcul de chaque ensemble $ICS^b(B_i, \tilde{u}_b)$ (étape 2) pour chaque trajectoire de l'ensemble ε et chaque objet B_i et puis de leur union et intersection (étapes 3 et 4). D'autre part, à la dimensionnalité de l'espace d'état S du robot A. Il est alors plus commode de proposer une autre version de l'algorithme de vérification-ICS^b (proposée dans ce qui suit) qui permet de palier aux problèmes posés.

3.8.2 ICS^b-CHECK: une version 2D de l'algorithme de vérification-ICS^b

Etant donné que l'espace de travail WS est de dimension 2, il est possible de concevoir une version 2D de l'algorithme de vérification-ICS^b que nous avons nommé *ICS^b-CHECK* [187, 188].

3.8.2.1 Raisonnement 2D

Quand A est planaire dont l'état s est un n-uplet de dimension arbitraire, il est alors possible de réécrire l'état s en $s = (x, y, \hat{z})$ avec (x, y) les coordonnées de l'espace de travail du point de référence de A, et $\hat{z}$ le reste des paramètres d'état [19, 171]. Le principe de base de ICS^b-CHECK est de calculer l'ensemble ICS^b correspondant à une tranche 2D de l'espace d'état S de A (au lieu d'effectuer le calcul pour la dimension complète de l'espace d'état) où chaque tranche correspond à un instant donné, et de vérifier ensuite si s appartient à cet ensemble. Supposons que l'état à vérifier est $s_c = (x, y, \hat{z}_c)$, la tranche 2D considérée est la tranche-$\hat{z}_c$. Si par exemple, $s_c = (x, y, \Theta, v, \xi)$ (avec Θ l'orientation d'un véhicule, v sa vitesse linéaire et ξ son angle de braquage), la tranche-$\hat{z}_c$ peut être représentée par un plan xy pour des paramètres (Θ, v, ξ) donnés (fixes), et où les contraintes de collision imposées par les objets de l'espace de travail peuvent être facilement calculées : comme la tranche-$\hat{z}_c$ est difféomorphe à l'espace de travail WS, cette propriété est exploitée pour calculer l'image des objets de WS dans la tranche-$\hat{z}_c$. Une fois cette image calculée par ICS^b-CHECK, il calcule l'ensemble ICS^b correspondant.

Dans la figure 3.8, un exemple illustratif est donné pour un robot A ayant la forme d'un disque. En démarrant d'un état arbitraire $s_c = (x, y, \hat{z}_c)$ appartenant à la tranche-$\hat{z}_c$, l'état atteint par A à un instant t en suivant la trajectoire $\tilde{u}_b$ est $s_t = \tilde{s}(s_c, \tilde{u}_b, t)$. s_t appartient alors à la tranche-$\hat{z}_t$, d'où la notation $s_t = (x, y, \hat{z}_t)$ (figure 3.8.a). Supposons maintenant que la collision entre A et B_i à lieu à cet instant t, dans ce cas l'ensemble des états de la tranche-$\hat{z}_t$ donnant une collision avec B_i (à savoir $A(s_t) \cap B_i(t)$) est $B_i(t) \ominus A(s_t)$, avec $\ominus$ la différence de Minkowski. C'est le principe de base du calcul de C-obstacle dans le cas 2D [19-chap3]. A est alors réduit à un point et la région de collision calculée est représentée avec une couleur grise dans la figure 3.8.b. Maintenant l'ensemble des états donnant une collision avec B_i doit être défini dans la tranche-$\hat{z}_c$. Etant donné que A est un corps rigide se déplaçant sur un plan, il existe une transformation géométrique unique traitant la translation et rotation de A, en décrivant son mouvement dans WS (à partir de l'état s_c jusqu'à l'état s_t). Cette transformation est notée $T_{\tilde{u}_b}(t)$, qui est en fonction de $\tilde{u}_b$ et t. Dans ce cas, $A(s_t) = T_{\tilde{u}_b}(t) A(s_c)$ et inversement $A(s_c) = T_{\tilde{u}_b}^{-1}(t)\, A(s_t)$. Par conséquent, l'image de $B_i(t) \ominus A(s_t)$ dans la tranche-$\hat{z}_c$ est obtenue en appliquant la transformation $T_{\tilde{u}_b}^{-1}(t)$ sur l'ensemble $B_i(t) \ominus A(s_t)$. Il est alors possible de définir l'ensemble ICS^b de la tranche-$\hat{z}_c$ qui mène à une collision avec B_i à un temps particulier $t \in [0, t_b[$ pour la trajectoire de freinage $\tilde{u}_b$ (voir figure 3.8.c):

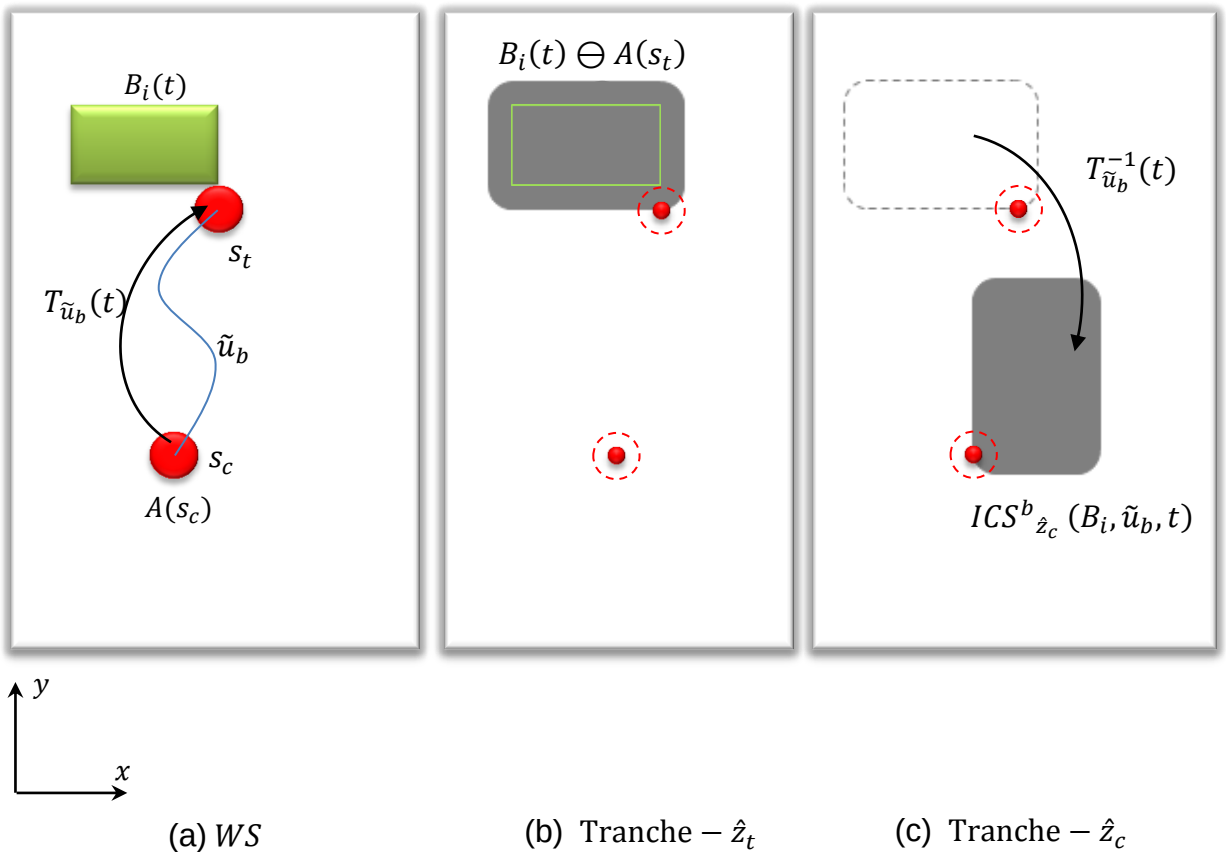
$$ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b, t) = \{s \in \text{tranche} - \hat{z}_c | A(\tilde{s}(s, \tilde{u}_b, t)) \cap B_i(t) \neq \emptyset\} \qquad (3.8)$$

En appliquant la transformation $T_{\tilde{u}_b}^{-1}(t)$, nous obtenons :

$$ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b, t) = \{s = (x, y, \hat{z}_c) | (x, y) \in T_{\tilde{u}_b}^{-1}(t)[B_i(t) \ominus A(s_t)]\} \qquad (3.9)$$

De même, l'ensemble ICS^b de la tranche-$\hat{z}_c$ qui mène à une collision avec B_i pour la trajectoire de freinage $\tilde{u}_b$ avec un temps de freinage t_b peut être défini (ligne 4 de l'algorithme 2):

$$ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b) = \bigcup_{t \in [0, t_b[} ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b, t) \qquad (3.10)$$



(a) WS (b) $\text{Tranche} - \hat{z}_t$ (c) $\text{Tranche} - \hat{z}_c$

Figure 3.8 : Calcul de $ICS^b{}_{\hat{z}_c}$ pour un objet B_i à un instant t.

$ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$ se calcule donc à partir de $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b, t)$ pour chaque instant t. Dans notre cas, chaque point de la limite du champ de vision de A est considéré comme un objet non perçu qui est représenté par un disque qui croit avec le temps (voir chapitre 2). Le comportement futur de cet objet est limité par l'horizon temporel T_h. Dans la figure 3.9, $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$ est calculé pour un tel objet. Tout d'abord, l'ensemble $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b, t)$ est calculé pour chaque disque (croissant) et pour un pas de temps donné. Par la suite, $ICS^b{}_{\hat{z}_c}$ de B_i est obtenu en unissant tous les ensembles calculés sur un intervalle de temps $t \in [0, T_h]$. Etant donné qu'une seule trajectoire de freinage $\tilde{u}_b$ est utilisée, $T_h = t_b$. L'équation (3.10) peut être réécrite par :

$$ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b) = \bigcup_{t \in [0, T_h[} ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b, t) \qquad (3.11)$$

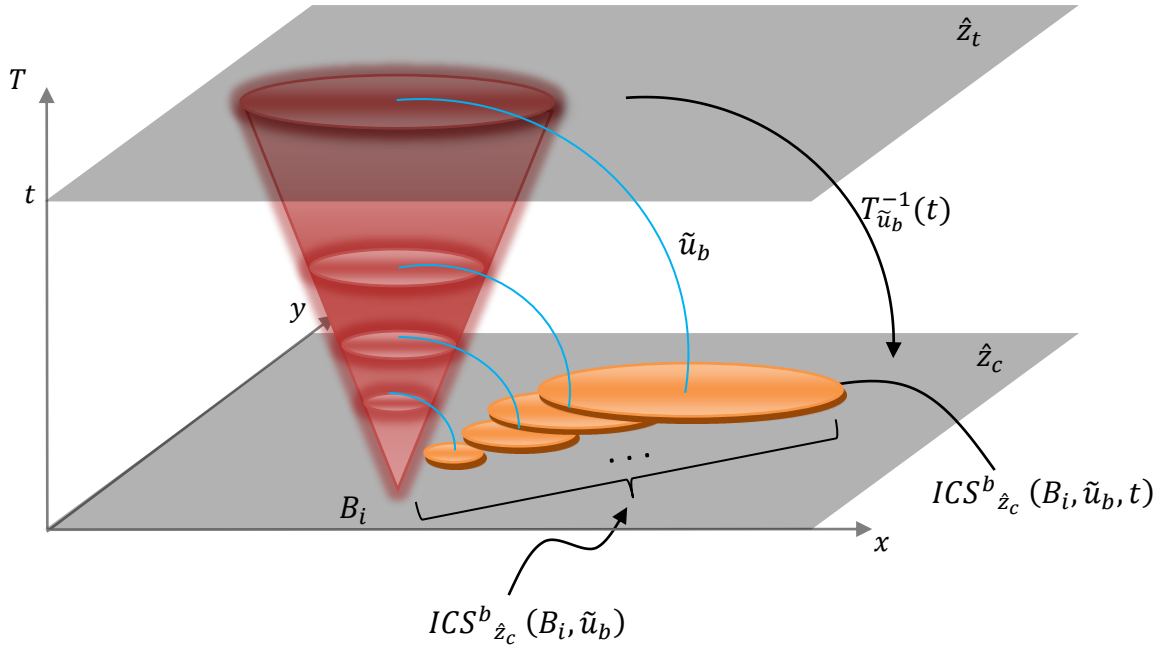


Figure 3.9 : Calcul de $ICS^b{}_{\hat{z}_c}$ pour un objet B_i.

3.8.2.2 l'algorithme ICS^b-CHECK

Comme illustré dans le paragraphe précédent, il est possible de calculer l'ensemble ICS^b de manière rigoureuse et efficace en raisonnant en 2D et donc de calculer l'ensemble ICS^b par rapport à la tranche-$\hat{z}_c$. L'algorithme ICS^b-CHECK (voir l'algorithme 2) opère alors de la même manière que l'algorithme général de vérification-ICS^b sauf que ICS^b est remplacé par $ICS^b{}_{\hat{z}_c}$.

Algorithme 2 : l'algorithme ICS^b-CHECK

Entrée: s_c, l'état à vérifier ; $B_i, i = 1 \dots n$.

Sortie : valeur booléenne.

1 Sélectionner $\mathcal{E} \subset \tilde{U}_b^{s_c}$, un ensemble de trajectoires de freinage pour s_c ;

2 **forall** $\tilde{u}_b \in \mathcal{E}$ **do**

3 **forall** B_i **do**

4 | Calculer $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$;

5 **end**

6 **C**alculer $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_b) = \bigcup_{i=1}^{n} ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$;

7 **end**

8 Calculer $ICS^b{}_{\hat{z}_c}(B, \mathfrak{E}) = \bigcap_{\tilde{u}_b \in \mathfrak{E}} ICS^b{}_{\hat{z}_c}(B, \tilde{u}_b)$;

9 **if** $\boldsymbol{s_c} \in ICS^b{}_{\hat{z}_c}(B, \mathfrak{E})$ **then**

10 | **return** VRAI **;**

11 **else**

12 | **return** FAUX ;

13 **end**

Les détails de l'implantation de l'algorithme ICS^b-CHECK et ses tests sont présentés dans le chapitre 5.

3.9 Conclusion

Au cours de ce chapitre, nous avons montré comment garantir la sûreté passive du mouvement pour un robot mobile ayant un champ de vision limité et aucune connaissance a priori de son environnement d'évolution. Cela a été réalisé grâce au concept *ICS de freinage*. Nous avons développé l'algorithme ICS^b-CHECK qui permet de définir si un état est ICS^b ou non. Le chapitre suivant est consacré à la mise en œuvre du concept ICS de freinage ; nous présentons deux schémas de navigation différents : un système d'évitement d'obstacles et un planificateur de mouvement. Tous les deux permettent de garantir la sûreté de mouvement passive.

CHAPITRE 4

NAVIGATION PASSIVEMENT SURE

Ce chapitre présente deux contributions importantes de ce travail. La première est un système d'évitement d'obstacles, que nous avons nommé *PASSAVOID*. Il permet de montrer l'efficacité du concept ICS de freinage (présenté dans le chapitre précédent) quant à la garantie d'une sûreté de mouvement passive. C'est un système réactif qui agit pendant un pas de temps donné pour générer un contrôle permettant de conduire le robot d'un état passivement sûr vers un autre. Le principe de l'approche ainsi que l'algorithme développé sont présentés dans la première partie du chapitre. Le reste du chapitre quant à lui est consacré à la deuxième contribution qui consiste à résoudre un problème de planification de mouvement. Dans une situation comme la nôtre où le robot mobile a un champ de vision limité et se déplace dans un environnement dynamique inconnu, une limite sur le temps de planification s'impose, d'où l'infaisabilité du calcul d'un mouvement complet vers le but. La solution proposée est un planificateur de mouvement partiel basé sur la garantie de la sûreté passive du mouvement appelé PASSPMP. Il permet de générer une trajectoire passivement sûre jusqu'au but où le concept ICS de freinage est utilisé. Pour cette partie, nous commençons par exposer l'idée générale et les objectifs à atteindre. Par la suite, nous expliquons le principe de la planification de mouvement partiel (PMP). Ce concept comble les lacunes des approches réactives (qui calculent le contrôle à appliquer durant l'étape suivante) et les approches délibératives (qui cherchent à calculer un mouvement complet vers le but). Une fois le concept PMP assimilé, PASSPMP peut être alors illustré. PASSPMP opère selon le principe de planification partielle, seulement il est régi par les règles de la sûreté passive du mouvement. Il est conçu de manière à calculer périodiquement des trajectoires partielles passivement sûres en utilisant une technique de diffusion adaptée et en sélectionnant la meilleure

trajectoire en se basant sur sa convergence vers le but. Le principe de l'approche et les algorithmes développés sont présentés en détail dans les paragraphes suivants.

4.1 Navigation réactive passivement sûre

Afin de démontrer la sûreté passive du mouvement et de valider le concept ICS de freinage, nous avons développé une méthode de navigation que nous nommons *PASSAVOID* [188, 189, 190] pour un robot mobile A ayant un champ de vision limité dans un environnement dynamique inconnu. La tâche primaire de PASSAVOID est de garder A dans des états p-sûrs ou de manière équivalente, de conduire A loin des états ICS^b. PASSAVOID garantit la sûreté passive du mouvement peu importe ce qui se passe dans l'environnement. Autrement dit, si une collision a lieu, il est garanti que A sera au repos quand ça se produira. PASSAVOID exploite ICS^b-CHECK pour accomplir sa mission.

4.1.1 Principe de PASSAVOID

PASSAVOID est une méthode de navigation réactive qui agit pendant un pas de temps δt. A chaque pas de temps, son but est de calculer un contrôle constant u qui sera appliqué au système robotique A durant le pas de temps suivant ; u doit être *admissible*, i.e. la trajectoire d'état correspondante doit être p-sûre (sans ICS^b).

PASSAVOID fonctionne comme la plupart des approches d'évitement d'obstacle (ex. [74, 77, 191, 192, 193]). De façon générale, leur principe de fonctionnement est de caractériser les régions interdites dans un espace de contrôle donné et puis de sélectionner un contrôle admissible, i.e. qui n'est pas interdit. Par conséquent, l'évitement d'obstacles dépend de l'habilité de la méthode d'évitement en question à trouver un contrôle admissible. En l'absence d'une caractérisation formelle des régions interdites, toutes les approches ont recours à une certaine forme d'échantillonnage de l'espace de contrôle avec le risque inhérent d'omettre les régions admissibles. PASSAVOID a aussi recours à l'échantillonnage pour trouver un contrôle admissible. Cependant, contrairement aux approches d'évitement d'obstacles standards, PASSAVOID est conçu de manière à ce qu'il garantisse toujours qu'un contrôle admissible existe et qu'il fait partie de l'ensemble d'échantillonnage.

Le principe de fonctionnement de PASSAVOID est illustré dans la figure 4.1. Soit s_0 l'état actuel de A et U_{ctrl} un ensemble échantillonné de contrôles : $U_{ctrl} = \{u_1 \dots u_m\}$. Un contrôle donné $u_j \in U_{ctrl}$ est appliqué à A pour une durée δt, qui conduit A de l'état s_0 à l'état $s_j = \tilde{s}(s_0, u_j, \delta t)$. Si s_j est p-sûr, i.e. ce n'est pas un ICS^b, alors u_j est admissible. Cela s'applique pour chaque contrôle de l'ensemble U_{ctrl} ; ce qui donne un ensemble de contrôles admissibles noté $U_{ctrl}{}^*$ à partir duquel PASSAVOID peut choisir un contrôle à appliquer durant le pas de temps suivant. Cette sélection peut être faite de façon arbitraire si nous nous intéressons uniquement à la survie de A ou sinon elle peut être faite de manière à assurer la convergence vers un but donné (en utilisant par exemple une fonction de navigation globale, un champ de potentiel, ou même un système de planification de mouvement partiel).

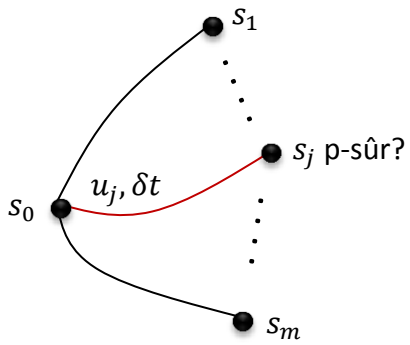


Figure 4.1 : Principe de fonctionnement de PASSAVOID (voir le texte).

4.1.2 Garantie de la sûreté passive du mouvement

Un système tel que PASSAVOID fonctionne bien tant qu'un contrôle admissible peut être trouvé dans U_{ctrl}. Cependant, si à la fin du processus, $U_{ctrl}{}^*$ est vide, cela signifie que chaque contrôle de U_{ctrl} conduit A vers un état qui est ICS^b. Donc, la sûreté passive du mouvement ne sera pas réalisée et une collision aura lieu quand A sera encore en déplacement. Pour remédier à ce problème, il est nécessaire de garantir que $U_{ctrl}{}^*$ n'est jamais vide et qu'il contient toujours au moins un contrôle admissible. Cela est possible grâce à un nombre de définitions et propriétés

nécessaires lors de la conception de PASSAVOID. Elles sont introduites dans ce qui suit, où δ-trajectoire de freinage et δ-sûreté passive sont définies en premier.

Définition 3 (δ-trajectoire de freinage) : une trajectoire de freinage $\tilde{u}^* \in \tilde{U}_b^{s_0}$ de durée t^* est une δ-trajectoire de freinage si elle est constante sur des intervalles de durée fixe δt.

δ-trajectoire de freinage est juste un type spécial de trajectoire de freinage (voir figure 4.2). Ce qui engendre un type de sûreté passive du mouvement qui lui correspond :

Définition 4 (δ-sûreté passive) : un état s_0 est δ-passivement sûr ou δ-p-sûr s'il existe une δ-trajectoire de freinage $\tilde{u}^*$ dont l'état initial est s_0 et qui est sans collision jusqu'à ce que A soit à l'arrêt.

Par conséquent, deux propriétés utiles sont établies :

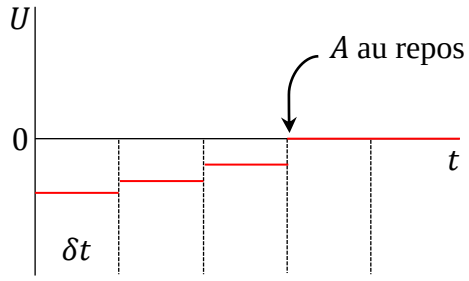


Figure 4.2 : Un exemple de δ-trajectoire de freinage (cas 1D).

Propriété 7 (Etats P-Sûrs)

Si l'état s_0 est p-sûr et que la trajectoire de freinage $\tilde{u}_b \in \tilde{U}_b^{s_0}$ dont l'état initial est s_0 est sans collision jusqu'à l'arrêt de A alors chaque état $\tilde{s}(s_0, \tilde{u}_b, t), 0 < t \leq t_b$ est aussi p-sûr.

Preuve:

Supposant que $\exists t_i \in]0, t_b]$ tel que $\tilde{s}(s_0, \tilde{u}_b, t_i)$ n'est pas p-sûr alors, par définition, $\forall \tilde{u}_j \in \tilde{U}_b^{s_i}$, $\tilde{u}_j$ produit une collision avant que A ne s'arrête. Cela s'applique également

à la trajectoire de freinage correspondant à la restriction de $\tilde{u}_b$ à l'intervalle de temps $[t_i, t_b]$ ce qui est contradictoire.

Notons que la propriété 7 s'applique également aux états δ-p-sûrs et à δ-sûreté passive.

Propriété 8 (garantie de la δ-Sûreté Passive)

Si l'état s_0 est δ-p-sûr alors il existe au moins un contrôle admissible u^* qui peut être utilisé pour conduire A à un état qui est également δ-p-sûr.

Preuve:

Comme s_0 est *δ-p-sûr, il existe alors au moins une trajectoire de freinage $\tilde{u}^*$ dont l'état initial est s_0 qui est sans collision jusqu'à ce que A s'arrête. Selon la propriété 7, l'état $\tilde{s}(s_0, \tilde{u}^*, \delta t)$ est δ-p-sûr. Soit u^* la valeur de $\tilde{u}^*$ sur l'intervalle de temps $[0, \delta t[$, u^* est un contrôle admissible.*

Pour PASSAVOID, la propriété 8 est fondamentale pour garantir la sûreté passive du mouvement. PASSAVOID doit conduire A d'un état δ-p-sûr à un autre. Supposons que s_0 est δ-p-sûr, la propriété 8 garantit l'existence d'au moins un contrôle admissible u^*, qui lorsqu'il est appliqué à A pour une durée δt, le conduit à un autre état δ-p-sûr. En général, un état δ-p-sûr s a plus qu'un seul contrôle admissible. Soit $K(s)$ l'ensemble des contrôles admissibles, il est nommé le *noyau de* s. Afin de garantir la sûreté passive de mouvement, PASSAVOID doit inclure $K(s_0)$ dans son ensemble d'échantillonnage de l'espace de contrôle. Ce qui se manifeste dans la ligne #2 de l'algorithme 3.

Algorithme 3 : PASSAVOID

Entrée: s_0, l'état δ-p-sûr actuel de A; $B_i, i = 1 \dots n$; δt, pas de temps.

Sortie : u

1 Echantillonner $U \rightsquigarrow U_{ctrl} = \{u_1 \dots u_m\}$; //sélectionner l'ensemble d'échantillonnage de l'espace de contrôle.

2 $U^*_{ctrl} = K(s_0)$; // Initialiser l'ensemble de contrôle admissible
3 **forall** $u_j \in U_{ctrl}$; // Calculer les contrôles admissibles
4 **do**
5 $s(\delta t) = \tilde{s}(s_0, u_j, \delta t)$;
6 **if** $s(\delta t)$ est δ-p-sûr **then**
7 | $U^*_{ctrl} = U^*_{ctrl} \cup \{u_j\}$; // u_j admissible
8 **end**
9 **end**
// Sélectionner et retourner un contrôle admissible
10 Sélectionner $u \in U^*_{ctrl}$;
11 **return** u **;**

PASSAVOID dispose de deux étapes importantes : calculer le noyau $K(s_0)$ (ligne#2 de l'algorithme 3) et vérifier si l'état $s(\delta t)$ est δ-p-sûr (ligne #6 de l'algorithme 3). Il s'avère que ces deux étapes sont liées et peuvent être calculées par une adaptation de ICS^b-CHECK. Etant donné que, par définition, un état qui n'est pas p-sûr est un état qui est ICS^b, ICS^b-CHECK est utilisé pour vérifier si un état est p-sûr ou pas. De manière similaire, ICS^b-CHECK peut être utilisé pour vérifier si un état est δ-p-sûr ou non en considérant plutôt des δ-trajectoires de freinage dans la ligne #1 de l'algorithme 1.

De plus, quand ICS^b-CHECK calcule $ICS^b(B, \tilde{u}_b)$ pour une trajectoire de freinage donnée $\tilde{u}_b$ (ligne #3 de l'algorithme 1), il est simple de déterminer si $\tilde{u}_b$ est sans collision en démarrant à partir de s_c (l'état à vérifier): i.e. quand $s_c \notin ICS^b(B, \tilde{u}_b)$. Dans ce cas, $\tilde{u}_b$ est un candidat pour $K(s_c)$; le noyau de s_c. L'algorithme 4 est une version de ICS^b-CHECK modifié de sorte à (1) vérifier si son entrée s_c est δ-p-sûr ou non (ligne#13 de l'algorithme 4), et (2) calculer le noyau de s_c (ligne #9 de l'algorithme 4). C'est cette version de ICS^b-CHECK qui est utilisée dans PASSAVOID.

Etant donné que l'état initial du système A est δ-p-sûr, la propriété 8 permet à PASSAVOID d'avoir à sa disposition un contrôle admissible qui peut être utilisé pour conduire A d'un état δ-p-sûr à un autre (pour toujours si besoin). Concernant

l'hypothèse que l'état initial est δ-p-sûr, elle est satisfaite quand A est à l'arrêt (voir paragraphe 3.4), et le contrôle nul est admissible. En d'autres termes, commençant par A à l'arrêt, PASSAVOID dispose d'un contrôle admissible facilement disponible qui peut être utilisé aussitôt si la situation l'exige (cela s'applique même si δt est très petit).

Il est prouvé que PASSAVOID est passivement sûr dans un sens où il est garanti que le robot A évitera toujours les états ICS^b peu importe ce qui se passe dans l'environnement.

Algorithme 4 : ICS^b-CHECK + Calcul du Noyau

Entrée: s_c, l'état à vérifier ; $B_i, i = 1 \ldots n$.

Sortie : valeur booléenne ; $K(s_c)$, le noyau de s_c ;

1 Sélectionner $\mathcal{E} \subset \widetilde{U}_b^{s_c}$, un ensemble de δ-trajectoires de freinage pour s_c ;

2 $K(s_c) = \emptyset$;

3 **forall** $\tilde{u}^* \in \mathcal{E}$ **do**

4 **forall** B_i **do**

5 | Calculer $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}^*)$;

6 **end**

7 **C**alculer $ICS^b{}_{\hat{z}_c}(B, \tilde{u}^*) = \cup_{i=1}^{n} ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}^*)$;

8 **if** $s_c \notin ICS^b{}_{\hat{z}_c}(B, \tilde{u}^*)$ **then**

9 | $K(s_c) = K(s_c) \cup u^*$; //s_c est δ-p-sûr pour $\tilde{u}^*$

10 **end**

11 **end**

12 Calculer $ICS^b{}_{\hat{z}_c}(B, \mathcal{E}) = \cap_{\tilde{u}_b \in \mathcal{E}} ICS^b{}_{\hat{z}_c}(B, \tilde{u}_b)$;

13 **if** $s_c \in ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$ **then**

14 | **return {**VRAI, $K(s_c)$**} ;** // dans ce cas $K(s_c) = \emptyset$

15 **else**

16 | **return {**FAUX, $K(s_c)$};

17 end

4.1.3 Navigation multi-robots passivement sûre

Comme expliqué précédemment, la sûreté passive permet de garantir qu'une collision ne va jamais se produire tant que le robot est en mouvement. Le robot ne va donc jamais causer activement une collision. Ne portant aucune hypothèse décisionnelle sur le comportement des objets mobiles, rien ne peut empêcher que ces derniers entrent en collision avec le robot malgré tout. Cependant, si chaque objet mobile dans l'environnement est passivement sûr (i.e. évite les états ICS^b) alors aucune collision ne devrait avoir lieu. Cette propriété peut être démontrée facilement.

Soit A_1 et A_2 deux robots fonctionnant avec un système de navigation passivement sûr tel que PASSAVOID. Selon les propriétés 7 et 8, les deux robots A_1 et A_2 sont dans un état δ-p-sûr à tout moment. Donc ce qui suit s'applique :

$$\forall t, s_1(t) \notin ICS_1^b \; et \; s_2(t) \notin ICS_2^b \quad (4.1)$$

Où, $s_i(t)$ et ICS_i^b désignent respectivement l'état à l'instant t et l'ensemble ICS^b correspondant, pour le robot $A_i, i = 1,2$.

Supposer qu'une collision peut avoir lieu entre A_1 et A_2 avec l'un deux ayant une vitesse non nulle, est une contradiction. Ce cas de figure ne peut avoir lieu.

4.2 Planification passivement sûre : concept général

Dans ce chapitre, l'objectif est de construire une trajectoire passivement sûre à partir d'un état initial jusqu'à un but donné, en vérifiant à la fois le critère de la sûreté et une meilleure convergence vers le but. La méthode proposée utilise un processus de re-planification (voir chapitre 1) qui chevauche la phase planification et la phase exécution de sorte que le robot puisse calculer des plans partiels. Cela conduit à concevoir un planificateur de mouvement partiel passivement sûr. Il est développé pour un robot mobile A ayant un champ de vision limité et placé dans un environnement dynamique inconnu.

Ce planificateur vise à conduire A de son état initial jusqu'à ce qu'il atteigne son but (en sélectionnant la meilleur trajectoire), tout en garantissant la sûreté

passive du mouvement peu importe ce qui se passe dans l'environnement (si une collision a lieu, il est garanti que A sera au repos).

Pour résumer, notre but est de concevoir un système de navigation passivement sûr. Ce concept est basé sur la planification de mouvement partiel (PMP) et garantit la sûreté passive du mouvement. Pour ce faire, un certain nombre de critères doivent être vérifiés et qui sont :

- Prise en compte des contraintes cinématiques et dynamiques du robot.
- Raisonnement sur le futur.
- Réagir pour un horizon temporel.
- Garantie de la sûreté passive (basée sur le concept ICS de freinage).
- Conduire le robot vers le but.

4.2.1 Principe de la planification de mouvement partiel (PMP)

La navigation d'un robot mobile nécessite le chevauchement de la perception, la prise de décision et l'exécution. Dans un environnement inconnu et dynamique, il est nécessaire de percevoir continuellement l'environnement et de mettre à jour régulièrement le modèle du monde sur lequel repose les décisions de navigation. De même, le module de prise de décision doit être appelé fréquemment et il a un temps fini pour décider quoi faire par la suite. Dans notre cas, ce module de prise de décision est basé sur un planificateur de mouvement partiel qui est la solution que nous proposons au problème posé pour un robot navigant dans ce type d'environnement. Il est spécialement conçu pour tenir compte de façon explicite de la contrainte temps réel caractérisant de tels environnements. En opposition aux méthodes réactives qui calculent à chaque fois la décision à prendre au pas de temps suivant et aux méthodes délibératives qui calculent une trajectoire complète du point de départ au point but, PMP calcule des trajectoires partielles. L'ensemble de ces trajectoires conduit le robot au but. Indépendamment de la façon dont le module de prise de décision fonctionne, le point primordial est qu'une trajectoire doit être toujours disponible à chaque cycle de prise de décision.

La chronologie du processus de prise de décision est illustrée dans la figure 4.3. A l'instant t_0, A est à l'état s_0, il est en cours d'exécuter une trajectoire Π_0 qui va

le conduire à l'état s_1 à l'instant t_1, le temps de cycle suivant. Soit $W(t_0)$ le modèle du monde disponible à l'instant t_0. Durant l'intervalle de temps $[t_0, t_1]$, le module de prise de décision doit calculer la trajectoire Π_1 qui doit être exécutée durant le prochain cycle de temps $[t_1, t_2]$. A l'instant t_1, le processus est répété à la base de $W(t_1)$, le modèle mis à jour de l'environnement, et ainsi de suite.

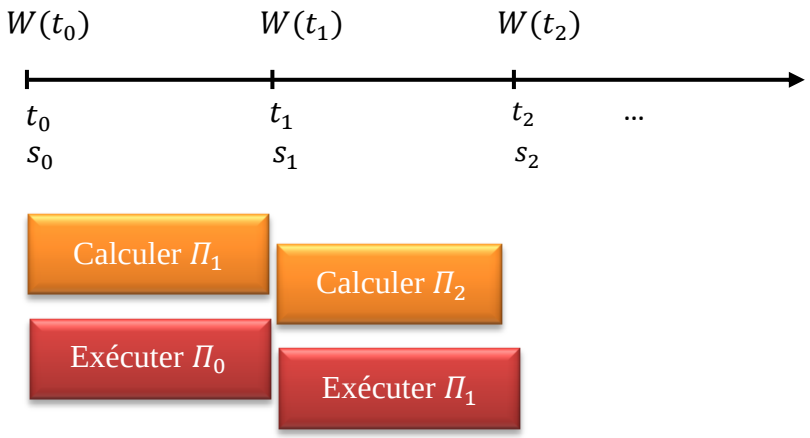


Figure 4.3 : La chronologie du processus de prise de décision et d'exécution dans un planificateur de mouvement partiel.

Du point de vue de la sûreté passive du mouvement, le concept ICS de freinage met en évidence deux points, à savoir :

- considérer que toutes les trajectoires Π_i soient sans collision est une condition insuffisante. Elles doivent être sans ICS de freinage, i.e. à chaque pas de temps, A doit être dans un état qui n'est pas ICS de freinage (si une collision a lieu, le robot est au repos).
- le concept ICS de freinage est toujours défini par rapport au modèle du monde utilisé (en particulier le modèle du futur).

L'introduction de cette notion de sûreté passive du mouvement dans le processus PMP nous a mené à développer la nouvelle version "PASSPMP" [194], décrite dans ce qui suit.

4.2.2 Planificateur de mouvement partiel passivement sûr (PASSPMP)

4.2.2.1 Principe de PASSPMP

PASSPMP est un planificateur partiel de mouvement qui opère pendant un cycle de temps donné δ_{cycle}. Durant ce temps, une trajectoire partielle p-sûre est calculée. Elle vérifie deux conditions : (1) les états correspondants doivent être p-sûrs (i.e. non ICS^b) et (2) la convergence vers le but désiré. Cette trajectoire à une durée de temps d'exécution δ_e. Cependant, seule la partie correspondant à la durée δ_{cycle} est exécutée, car à la fin de chaque cycle une nouvelle trajectoire partielle calculée est disponible (voir figure 4.4). Le cycle PASSPMP est choisi comme une période de temps fixe, afin d'avoir régulièrement une mise à jour de l'environnement. Il doit retourner une décision dans un délai borné qui dépend du modèle actuel de l'environnement (modèle du futur). En effet, le robot ne peut pas avoir tout son temps pour prendre une décision d'éviter la collision. Il doit respecter une contrainte de *temps de décision* δ_d. Ce temps dépend de l'état courant du robot et de l'environnement qui l'entoure. Généralement, sa valeur doit être choisie de manière à ce que le robot ne prenne pas plus de temps que le temps de collision (le temps pris pour que le robot rentre en collision avec un obstacle) pour décider ce qu'il doit faire et de manière à laisser assez de temps au robot pour éviter la collision. Mais, δ_d peut être formellement défini comme suit :

Etant donné la trajectoire courante que suit le robot, δ_d est la durée qui sépare l'instant courant est le premier état ICS^b sur cette trajectoire.

Ainsi, le cycle de temps doit être toujours inférieur à ce temps de décision:

$$\delta_{cycle} < \delta_d \qquad (4.2)$$

Durant le cycle PASSPMP, celui-ci calcule, sur la base du modèle du futur, une trajectoire partielle passivement sûre. Pour prouver qu'à chaque cycle de temps, une trajectoire partielle p-sûre doit être trouvée (prouver qu'une collision ne va jamais se produire quand A est en mouvement), certaines propriétés sont requises et donc introduites dans ce qui suit. Mais en premier, rappelons ce qu'est un état p-sûr (à partir de la définition 1).

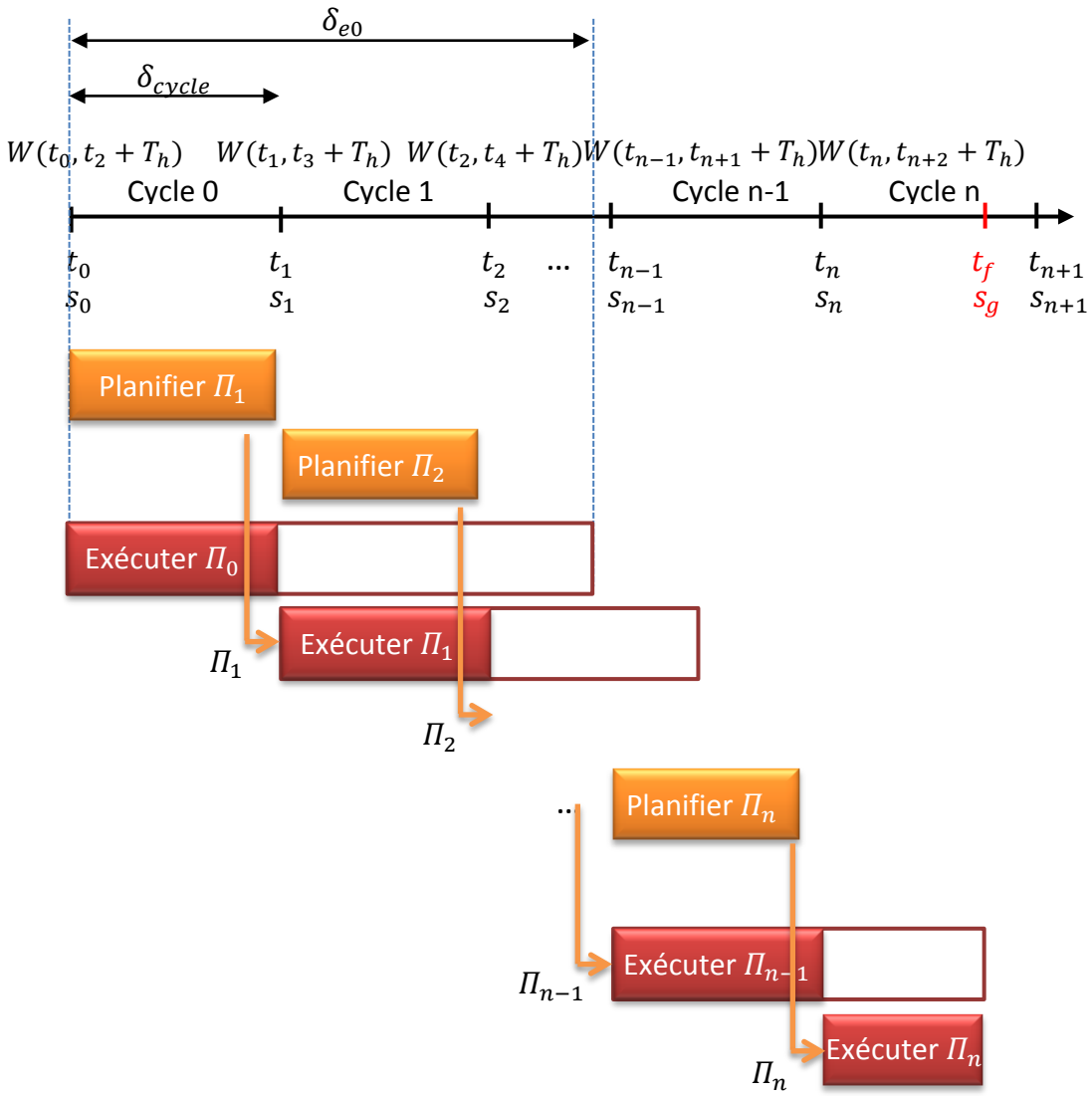


Figure 4.4 : Processus d'exécution de PASSPMP.

Un état s_0 est passivement sûr ou p-sûr (n'est pas un ICS de freinage) s'il existe au moins une trajectoire de freinage $\tilde{u}_b$ dont l'état initial est s_0 et qui est sans collision jusqu'à ce que A s'arrête.

Propriété 9 (Trajectoire P-Sûre)

Pour une trajectoire d'état donnée Π entre s_0 et s_j, si (1) la trajectoire d'état entre s_0 et s_j est sans collision (par rapport au modèle du futur) et (2) s_j est p-sûr, alors les états appartenant à Π sont p-sûrs, donc, Π est p-sûre.

Preuve :

Dans un premier temps, nous définissons un état s_i, avec $0 < i < j$ appartenant à une trajectoire d'état entre s_0 et s_j. Supposons que (1) s_2 est p-sûr, i.e. il existe au moins une trajectoire de freinage dont l'état initial est s_2 et qui est sans collision jusqu'à ce que A s'arrête. (2) s_1 n'est pas p-sûr, i.e. quelque soit la trajectoire de freinage existante, une collision a lieu (aucune trajectoire de freinage sans collision n'est disponible). Cependant, comme la trajectoire d'état entre s_0 et s_j est sans collision et qu'il existe une trajectoire de freinage pour l'état s_2, en démarrant de s_1, A peut freiner sans qu'une collision se produise. Par conséquent, s_1 est p-sûr : contradiction avec (2).

Propriété 10 (garantie de la sûreté passive)

Si l'état s_0 est p-sûr alors il existe au moins une trajectoire d'état qui est p-sûre et qui conduit A à travers des états qui sont également p-sûrs.

Preuve :

Selon la définition 1, si s_0 est p-sûr alors il existe au moins une trajectoire de freinage $\tilde{u}_b$ (entre s_0 et s_j) qui est sans collision jusqu'à ce que A s'arrête. A l'état s_j, le robot A est à l'arrêt alors s_j est p-sûr. En appliquant la propriété 9, comme la trajectoire de freinage $\tilde{u}_b$ est sans collision et que s_j est p-sûr, $\tilde{u}_b$ est désormais p-sûre et chaque état de cette trajectoire est également p-sûr. Soit $\tilde{u}_b$ est un cas particulier de trajectoires possibles, alors il est prouvé qu'il existe une trajectoire p-sûre.

La propriété 10 permet la conception de la version du planificateur PASSPMP qui est passivement sûre i.e. à chaque cycle de planification, il est garanti que PASSPMP peut planifier une trajectoire qui est p-sûre pour être exécutée au prochain cycle. De plus, même si PASSPMP échoue à calculer cette trajectoire, il y aura au moins une trajectoire de freinage disponible. C'est un processus itératif, répété jusqu'à ce que A atteigne son but.

Ayant un champ de vision limité, A peut uniquement percevoir un sous-ensemble de l'espace de travail WS. Ce qui représente la région de WS qui est libre d'objets (au moment de la perception) alors que le reste des régions de WS représente l'espace où il y a possibilité de présence d'objets fixes ou mobiles, perçus ou non perçus. La sûreté de mouvement nécessite un raisonnement sur le mouvement futur des objets dans l'environnement. Etant donné qu'aucune information n'est disponible sur les objets présents en dehors du champ de vision et de leur comportement futur, le modèle du futur utilisé dans ce cas est conservatif : connaissant la vitesse maximale des objets, chaque point de cette région est modélisé par un disque qui croit avec le temps, i.e. un cône dans l'espace×temps (pour plus de détail voir chapitre 2). Cependant, avec le temps la région de WS occupée par ces disques croissants augmente également. C'est pourquoi le modèle du futur doit être valide jusqu'à un temps fini, à savoir un horizon temporel T_h (tel défini à la propriété 5). Sa valeur est définie par l'équation (3.7). Pour garantir la sûreté du mouvement, il suffit de considérer le modèle du futur jusqu'à l'instant T_h.

Concernant le processus PASSPMP, celui-ci dispose d'un intervalle de temps $[t_0, t_1]$ pour calculer, sur la base du modèle du futur $W(t_0)$, la trajectoire Π_1 qui démarrera à l'instant t_1 (avec le robot à l'état s_1, début du cycle prochain $[t_1, t_2]$). Ce cycle se répète jusqu'à ce que le but soit atteint (voir figure 4.4). D'un point de vue sûreté de mouvement, il est garanti que s_1 est p-sûr, car s_1 est obtenu à partir de la trajectoire partielle p-sûre Π_0 (calculée pendant le cycle précédent). Grâce à la propriété 10, il est garanti qu'il existe une trajectoire Π_1 p-sûre (qui démarre à partir de l'état s_1 et conduit le robot à l'état s_2). Pour garantir qu'au cycle prochain ($[t_1, t_2]$) PASSPMP puisse calculer une trajectoire p-sûre (calcul de Π_2), s_2 doit être p-sûr par rapport à $W(t_0)$, mais s_2 correspond à l'instant t_2 alors que $W(t_0)$ commence à l'instant t_0. Si à l'instant t_2 (quand le modèle du monde $W(t_2)$ est disponible) s_2 s'avère être ICS de freinage par rapport à $W(t_2)$, la collision se produit. C'est pour cette raison qu'il faut considérer $W(t_0)$ jusqu'à $t_2 + T_h$. Le modèle du monde W doit donc rester valide sur l'intervalle de temps $[t_n, t_{n+2} + T_h]$ (pour chaque cycle n). Ainsi,

il est garanti que le processus PASSPMP fournit toujours une solution (à chaque cycle, une trajectoire p-sûre est générée).

4.2.2.2 L'algorithme PASSPMP

L'algorithme 5 décrit le comportement de PASSPMP durant un cycle k.

Algorithme 5 : PASSPMP

Entrée: Modèle du futur $W(t_k, t_{k+2} + T_h)$, but s_g, trajectoire partielle Π_k à exécuter.

Sortie : la trajectoire partielle sélectionnée Π_{k+1} à exécuter dans le cycle $k+1$.

1: TREE=EXPLORE_STATE_TIME($W(t_k, t_{k+2} + T_h), s_{k+1}$);

//Exploration de l'espace d'états-temps de A, $t \in [t_{k+1}, t_{k+2}]$

2: Π_{k+1}=SELECT_BEST_PTRAJ(TREE,s_g);

//Sélectionner la meilleure trajectoire partielle.

L'algorithme 5 a comme entrée le modèle du futur $W(t_k, t_{k+2} + T_h)$ disponible à l'instant t_k et valable jusqu'à l'instant $t_{k+2} + T_h$ (i.e. suffisant pour planifier la trajectoire partielle Π_{k+1} durant le cycle actuel k et l'exécuter durant le prochain cycle $k+1$). L'algorithme prend aussi comme entrée le but à atteindre, et la trajectoire partielle Π_k calculée durant le cycle $k-1$ à exécuter durant le cycle en cours. Cette trajectoire en plus de permettre au robot de se déplacer pendant la planification, elle fournit au planificateur l'état s_{k+1} à partir duquel commence la planification de Π_{k+1}.

L'algorithme PASSPMP fonctionne d'abord en explorant l'espace d'états-temps de A (avec $t \in [t_{k+1}, t_{k+2}]$) grâce à la fonction EXPLORE_STATE_TIME (ligne 1 de l'algorithme 5). Pour ce faire, une technique de diffusion est utilisée en étendant un arbre dont la racine est le nœud $\eta_{root=}s(t_{k+1})$ (avec $t_{k+1} = t_k + \delta_{cycle}$). Durant cette étape de l'algorithme, les trajectoires partielles sont calculées puis définies par rapport à la garantie de la sûreté passive (en se basant sur l'algorithme ICS^b-CHECK (voir algorithme 2)). L'approche développée est détaillée dans ce qui suit.

Dans l'étape #2 de l'algorithme 5, la fonction SELECT_BEST_PTRAJ sélectionne la meilleure trajectoire partielle en vérifiant le critère de sûreté passive et la convergence vers le but.

L'algorithme est répété de manière itérative jusqu'à atteindre le but (s_g).

4.2.2.3 Technique de diffusion : p-safe RRT

La phase d'exploration dans le processus de planification partielle est basée sur une recherche incrémentale dans l'espace d'états-temps du robot. Durant une étape de recherche incrémentale, il y a plusieurs transitions possibles à partir d'un état donné et qui sont limitées uniquement par les caractéristiques kinodynamiques du robot et le modèle du futur de l'environnement. Ces transitions sont appelées *primitives de trajectoire*. Différents niveaux d'expansion de l'arbre ou profondeurs (en anglais : *depth*) (i.e. étapes de recherche incrémentale) sont possibles (voir chapitre 1). La trajectoire finale résultante à partir d'une configuration initiale jusqu'à une configuration but est une concaténation de ces primitives (où chaque primitive appartient à une étape de recherche (profondeur) donnée de l'arbre).

Pour ce faire, afin d'explorer différentes trajectoires possibles, un arbre est étendu dans l'espace d'états-temps de A. En raison de simplicité et facilité d'adaptation au concept de mouvement partiel, la méthode adoptée est inspirée de la méthode RRT proposée dans [16] (pour plus de détail voir chapitre 1). Etant donné que la sûreté passive doit être un critère dans la sélection des trajectoires, nous nommons cette méthode *p-safe RRT* (i.e. *RRT p-sûr*) [194]. Les nœuds de l'arbre étendu représentent les états du robot qui sont reliés par des primitives de trajectoire (chaque primitive est une partie d'une trajectoire délimitée par deux nœuds). Ces primitives sont définies à partir du modèle dynamique du robot de l'équation (3.1). Les états (configurations du robot) de cette primitive sont obtenus en intégrant cette même équation. Pour ce faire, le vecteur de contrôle u requis pour le contrôle du niveau bas est directement dérivé à partir de la trajectoire planifiée. Cela permettra de générer des trajectoires faisables dans l'espace des solutions.

Les transitions explorées durant une seule étape incrémentale peuvent être étendues de plus en plus dans les étapes suivantes, résultant en une augmentation

exponentielle des états (nœuds) et primitives (transitions) possibles. C'est pourquoi, une attention spéciale doit être portée sur la façon dont cette exploration doit être effectuée. Il existe plusieurs méthodes d'extension de nœuds, citons les plus pertinentes [195] :

1. Exploration aléatoire : l'arbre est étendu vers une configuration aléatoire dans l'environnement.
2. Recherche du but : l'expansion de l'arbre se fait vers une configuration but bien définie.
3. Entrées de contrôles aléatoires : l'expansion se fait à partir d'un nœud de l'arbre (état) en utilisant des entrées de contrôles aléatoires.

Les deux premiers cas étendent l'arbre soit vers une configuration de l'espace de travail choisie de façon aléatoire soit vers une configuration but définie préalablement. Dans les deux cas, l'expansion se fait en cherchant toujours le nœud le plus proche de cette configuration et en choisissant un contrôle faisable le plus proche (tel le principe de l'approche RRT classique illustrée dans le chapitre 1). Quant au 3ème cas, il représente une recherche directe dans l'espace de contrôle des commandes actuelles kino-dynamiquement faisables. Un cas spécial est d'étendre l'arbre en utilisant une recherche exhaustive, i.e. utiliser un ensemble fixe de contrôles (par exemple aller tout droit, tourner à droite, tourner à gauche en ayant une accélération donnée et un angle de braquage donné dans le cas d'un robot de type voiture) au lieu d'un ensemble de contrôles aléatoires. Dans le cas des entrées de contrôles aléatoires, tout l'ensemble des contrôles disponible est exploré, alors que la recherche exhaustive est liée uniquement à un petit ensemble de contrôles afin de maintenir le calcul plus résoluble.

L'algorithme proposé p-safe RRT est illustré par l'algorithme 6. Il décrit la fonction EXPLORE_STATE_TIME de l'algorithme PASSPMP (ligne #1 de l'algorithme 5). Il vise à étendre l'arborescence (TREE), étant donné le modèle du futur (comportement futur des objets mobiles) représenté par le modèle conservatif présenté dans le chapitre 2 et un arbre initial (qui peut être un ensemble de nœuds et de transitions quelconques ou un nœud unique) dont la racine η_{root} est l'état initial du cycle à planifier (s_{k+1}). L'algorithme dispose d'un temps ($\delta_{cycle} - \delta t_{select}$) pour

étendre cette arborescence, avec δt_{select} le temps que va prendre la fonction SELECT_BEST_PTRAJ (ligne #2 de l'algorithme 5). Durant cette période de temps, l'arbre est étendu à travers différents niveaux d'expansion ou profondeurs, notés d_{Tree}. Chaque nœud lui est donc associé un d_{Tree}, désignant la profondeur à laquelle il appartient. Etant donné que l'arbre est enraciné à l'état η_{root}, un nœud η_i (un descendant de η_{root}) peut être alors également noté ${\eta_i}^{(\eta_{root}/d_{Tree})}$. N_1 est l'ensemble des nœuds appartenant à la profondeur actuelle. Un exemple simplifié du processus de diffusion est représenté dans la figure 4.5. Une approche d'expansion basée sur une recherche exhaustive est utilisée (la 3ème méthode d'expansion). L'expansion se fait pour chaque nœud de N_1 (qui au départ contient $\eta_{root} = s_{k+1}$ correspondant à la profondeur $d_{Tree} = 0$) selon un ensemble de contrôles, générant ainsi de nouveaux nœuds ${\eta_i}^{(s_{k+1}/1)}$ qui forment une nouvelle profondeur (correspondant à $d_{Tree} = 1$). Ces nœuds sont à leur tour étendus, et le processus se répète pendant la durée de la planification $(\delta_{cycle} - \delta t_{select})$.

Comme illustré dans l'algorithme 6, chaque nœud est étendu selon un ensemble d'échantillonnage de l'espace de contrôle sélectionné (ligne #6 de l'algorithme 6) en utilisant la fonction EXPANSION_WITH_SAFETY_ CHECK (ligne #8 de l'algorithme 6). Cette fonction étend un nœud η_i avec un contrôle u_j générant une nouvelle primitive de trajectoire qui est vérifiée par rapport à la sûreté passive. La primitive est générée par la fonction GENERATE_TRAJ_PRIM (ligne #22 de l'algorithme 6) ; les états de cette trajectoire sont obtenus en intégrant l'équation du modèle dynamique du robot (l'équation (3.1)). Quant à l'aspect sûreté passive, l'état est vérifié pour être p-sûr ou non en utilisant la fonction BRAKING_ICS_CHECK (ligne #24 de l'algorithme 6), qui est basée sur l'algorithme ICS^b-CHECK. Selon la propriété 10, il est garanti que si l'état s_{k+1} est p-sûr, il existe toujours une trajectoire partielle qui est p-sûre. Comme s_{k+1} appartient à la trajectoire partielle p-sûre précédente (à partir du cycle PASSPMP précédent), s_{k+1} est p-sûr donc il existe toujours une trajectoire partielle p-sûre jusqu'à ce que A atteigne son but. Selon la propriété 9, il est garanti qu'une primitive de trajectoire $\delta\Pi_{new}$ délimitée par les deux nœuds η_i (nœud à étendre) et η_{new} (le nouveau nœud) est p-sûre si $\delta\Pi_{new}$ est sans

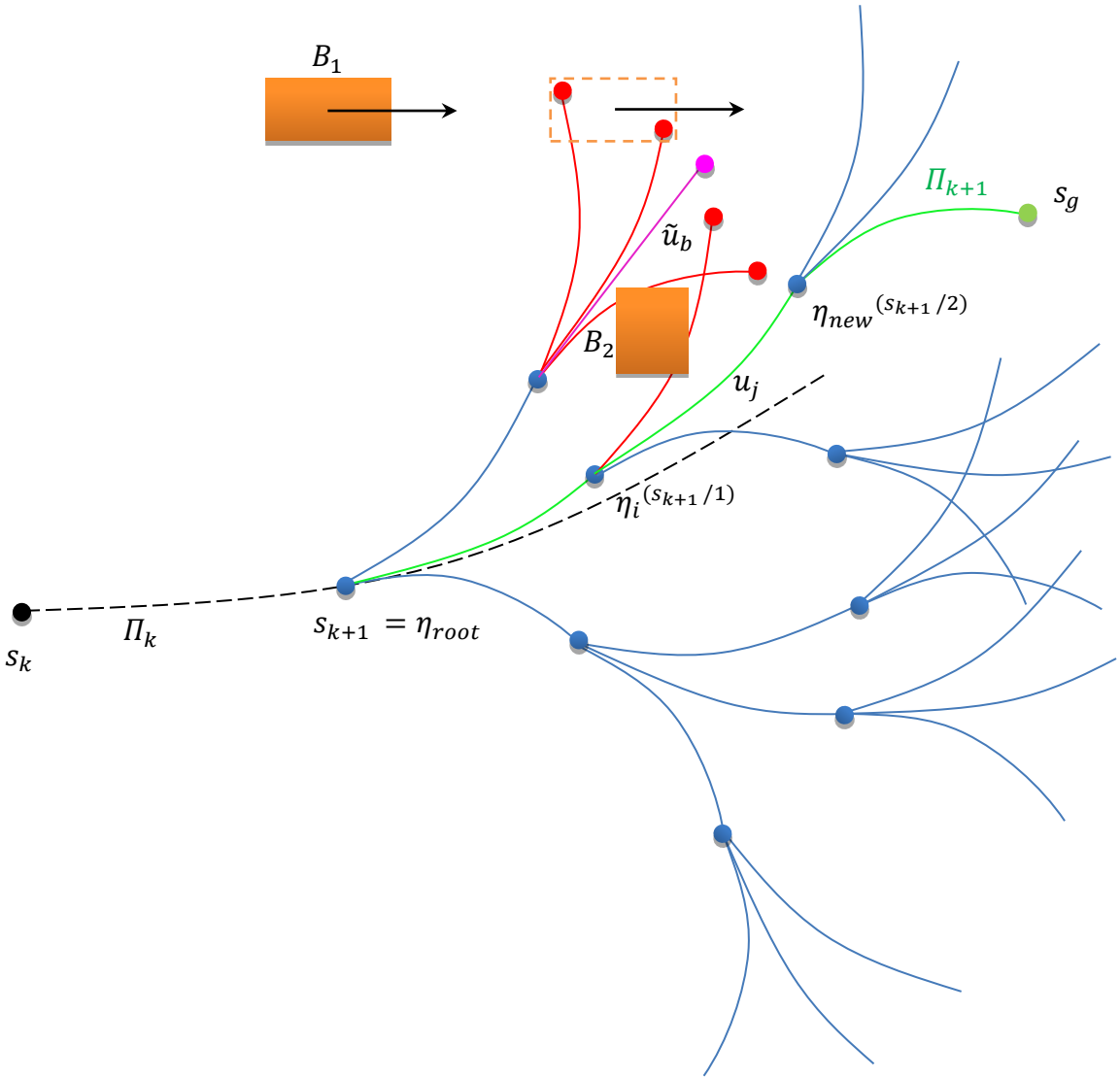


Figure 4.5 : Diffusion de la trajectoire du robot dans un environnement dynamique (avec un obstacle mobile B_1 et un obstacle fixe B_2) en utilisant p-safe RRT. A l'état s_k (début du cycle k), la trajectoire partielle Π_k planifiée au cours du cycle précédent $(k-1)$ est exécutée. En parallèle, la trajectoire Π_{k+1} est planifiée. L'arbre est enraciné à l'état s_{k+1}. Les primitives de trajectoire (respectivement les nœuds) représentées en bleu sont p-sûres. Celles en rouge ne sont pas p-sûres (ICS de freinage). Quant à la primitive en magenta, c'est une trajectoire de freinage sans collision. La trajectoire partielle planifiée Π_{k+1} (représentée en vert) est la concaténation de primitives p-sûres (en sélectionnant la combinaison optimale par rapport au but s_g).

collision et η_{new} est p-sûr. Si la propriété 9 est vérifiée, la primitive $\delta\Pi_{new}$ et le nœud étendu η_{new} correspondant sont alors ajoutés à l'arbre (lignes #27 et #28 de l'algorithme 6). Dans le cas où aucune primitive de trajectoire p-sûre n'est trouvé pour un nœud donné η_i, la fonction GENERATE_BRAKING_TRAJ (ligne #12 de l'algorithme 6) cherche une trajectoire de freinage qui est sans collision afin de garantir que le robot sera à l'arrêt si une collision se produirait (voir l'exemple de la figure 4.5). Etant donné que le nœud η_i est p-sûr, il est garanti qu'il existe au moins une trajectoire de freinage $\tilde{u}_b$ dont l'état initial est η_i et qu'elle est sans collision jusqu'à ce que A soit à l'arrêt (vérifiant la propriété de la sûreté passive du mouvement). A partir d'un ensemble de trajectoires de freinage donné, les trajectoires de freinage sans collision sont définies. Parmi ces dernières, une trajectoire est choisie selon le critère d'optimalité par rapport au but.

Algorithme 6 : l'algorithme p-safe RRT.

Entrée: Modèle du futur $W(t_k, t_{k+2} + T_h)$, δ_{cycle}, $\eta_{root} = s_{k+1}$ (début du cycle $k+1$, fourni par Π_k).

Sortie : TREE.

1: Initialisation: TREE $= \{\eta_{root}\}$, $N_1 = \{\eta_{root}\}$ (ensemble de nœuds appartenant à la profondeur actuelle), $d_{Tree} = 0$;
2: **While** $t < (\delta_{cycle} - \delta t_{select})$ **do**
3: $N_2 = \emptyset$;
4: **for** $i = 0$ to $|N_1|$ **do**
5: //sélectionner l'ensemble d'échantillonnage de l'espace de contrôle U_{ctrl}.
6: Echantillonner $U \rightsquigarrow U_{ctrl} = \{u_1 \dots u_n\}$
7: **Forall** $u_j \in U_{ctrl}$ **do**
8: $(\eta_{new}, \delta\Pi_{new})$ ← EXPANSION_WITH_SAFETY_ CHECK (η_i, u_j, W, TREE);
9: $N_2.push\_back(\eta_{new})$;
10: **end**

11: **if** $\eta_{new} = \emptyset$ **then**
12: GENERATE_BRAKING_TRAJ(η_i);
13: **end if**
14: **end for**
15: $N_1 \leftarrow N_2$;
16 : $d_{Tree} = d_{Tree} + 1$;
17: **end while**
18: **return** TREE;
19:
20: **Procedure** EXPANSION_WITH_SAFETY_ CHECK (η_i,u_j,W,TREE)
21: //verifier la p-sûreté
22: $(\eta_{new}, \delta\Pi_{new})$ ← GENERATE_TRAJ_PRIM(η_i,u_j)
23: //$\delta\Pi_{new}$: délimitée par les deux nœuds η_i et η_{new}.
24: **if** BRAKING_ICS_CHECK(η_{new}, W)=False **then**
25: //η_{new} est p-sûr
26: **if** $\delta\Pi_{new}$ est sans collision **then**
27: TREE = TREE ∪ η_{new};
28: TREE = TREE ∪ $\delta\Pi_{new}$;
29: **return** η_{new}, $\delta\Pi_{new}$;
30: **end if**
31: **end if**
32: **EndProcedure**
33:
34: **Procedure** GENERATE_BRAKING_TRAJ(η_i)
35: Sélectionner $U_{BM} \subset \widetilde{U}_b$, un ensemble de trajectoires de freinage pour η_i ;
36: $U_{BMCFree} = \emptyset$; // L'ensemble des trajectoires de freinage sans collision
37: **forall** $\tilde{u}_b \in U_{BM}$ **do**
38: **if** $\tilde{u}_b$ est sans collision **then**
39: $U_{BMCFree} = U_{BMCFree} \cup \tilde{u}_b$;
40: **end if**

41: **end for**
42: **forall** $\tilde{u}_b \in U_{BMCFree}$ **do**
43: COMPUTE_DISTANCE_TO_GOAL();
44: **end for**
45: $\delta\Pi_{new} \leftarrow$ MIN_DISTANCE_TO_GOAL($\tilde{u}_b \in U_{BMCFree}$);
46: TREE = TREE ∪ $\delta\Pi_{new}$;
47: TREE = TREE ∪ η_{new}; //η_{new} : état final de $\delta\Pi_{new}$
48: **EndProcedure**

4.2.2.4 La sélection de la trajectoire

Dans le paragraphe précédent, il a été montré comment p-safe RRT permet d'explorer l'espace d'états-temps du robot en construisant un arbre contenant différents nœuds et primitives de trajectoire qui sont testés pour être p-sûrs ou non (TREE), tout en garantissant que le robot sera toujours dans un état p-sûr. Plusieurs trajectoires partielles sont alors possibles, où chaque trajectoire est une concaténation de plusieurs primitives avec chaque primitive appartenant à une profondeur donnée de l'arbre.

La fonction SELECT_BEST_PTRAJ de l'algorithme PASSPMP (l'algorithme 5) permet de sélectionner la meilleure trajectoire partielle (Π_{k+1}) à partir de l'arborescence développée (utilisant l'algorithme 6). Cette sélection est basée sur deux conditions : (1) la sûreté passive de la trajectoire partielle. (2) la convergence vers le but : une fonction de coût pondérée est calculée basée sur une mesure de distance et le coût en temps.

Ainsi, la trajectoire partielle sélectionnée doit être en premier passivement sûre. Une trajectoire partielle p-sûre est une concaténation de primitives de trajectoire p-sûres. De cette façon, plusieurs possibilités de trajectoires partielles p-sûres sont disponibles.

Pour départager quelle trajectoire partielle p-sûre (entre les états s_{k+1} et son descendant s_{k+2}, avec s_{k+2}=$\eta_i{}^{(s_{k+1}/m)}$ où m est la profondeur totale de l'arbre) pourrait être sélectionnée, une fonction de coût pondérée est calculée. Elle est exprimée comme suit :

$$f_g(s_{k+2}) = w_d||s_{k+2} - s_g|| + w_t\Delta T_{\Pi} \qquad (4.3)$$

La fonction f_g détermine la meilleure trajectoire partielle à sélectionner Π_{k+1} i.e. l'optimale en distance et en temps par rapport au but. Elle dépend de la distance euclidienne entre l'état final de la trajectoire (s_{k+2}) et l'état du but (s_g) associée à un facteur de pondération de la distance w_d (minimisant la distance au but). De plus, cette fonction dépend du temps cumulé (ΔT_{Π}) (représentant la durée de la trajectoire) associée à un facteur de pondération du coût du temps w_t (minimisant le temps au but). Par exemple, dans le cas de la figure 4.5, la trajectoire optimale est représentée en vert.

La trajectoire partielle Π_{k+1} est le résultat de calcul d'un cycle de planification k. La trajectoire partielle planifiée peut ne pas atteindre le but, comme elle peut l'atteindre (tel le cas de l'exemple de la figure 4.5). D'autre part, elle peut être exécutée en entier ou en partie selon la durée du cycle PASSPMP. Le processus PASSPMP est répété de manière itérative jusqu'à ce que le robot atteigne son but, sachant qu'à chaque cycle une nouvelle mise à jour de l'environnement (modèle du futur) est considérée.

4.3 Conclusion

Durant ce chapitre, nous avons présenté deux approches de navigation *PASSAVOID* et PASSPMP. Ces approches sont capables de traiter à la fois des champs de vision limités, des régions occultées et un comportement futur inconnu des obstacles mobiles. *PASSAVOID* est un système d'évitement d'obstacles passivement sûr, conçu de manière à toujours permettre au robot d'éviter les états ICS^b. Il garantit la survie du robot mais sans se préoccuper d'aller vers un but bien précis. PASSPMP est basé sur le concept de planification de mouvement partiel qui vise à calculer des trajectoires partielles jusqu'à atteindre le but, et le concept ICS de freinage qui assure la sûreté passive de ces trajectoires et l'évitement des collisions. PASSPMP utilise p-safe RRT, une approche de diffusion afin d'explorer l'espace d'états-temps du robot de manière passivement sûre. Cette dernière est à son tour basée sur ICS^b-CHECK pour déterminer si un état est p-sûr ou non. Dans ce chapitre, il a été formellement prouvé que *PASSAVOID* et PASSPMP sont des approches qui

permettent au robot d'être toujours dans un état passivement sûr (éviter les ICS de freinage). Cependant, en comparaison avec PASSAVOID, PASSPMP a plus de perspicacité. Il vise à conduire le robot vers un but bien défini, au contraire de PASSAVOID dont la priorité est uniquement la survie du robot. Plus de réponses et preuves concernant tous les concepts développés dans ce chapitre et dans les chapitres précédents sont apportées lors des validations dans le prochain chapitre.

CHAPITRE 5

TESTS, RESULTATS ET VALIDATION

La problématique de cette thèse soulevait certains des problèmes les plus pointus dans le domaine du transport urbain et même de la robotique de service, à savoir, la garantie de la sûreté du mouvement du robot. Cette notion de sûreté lui permettrait de se déplacer en évitant les obstacles de l'environnement même ceux qui sont inattendus et de planifier sa trajectoire à partir d'un état initial jusqu'à un but. Ce qui doit être réalisé pour un robot mobile ayant un champ de vision limité et navigant dans un environnement dynamique inconnu. Ces problèmes ont été largement étudiés dans les chapitres précédents et des solutions ont été proposées. Principalement trois approches ont été développées ; ICS^{b}-CHECK, PASSAVOID et PASSPMP. Afin de démontrer et valider la sûreté passive du mouvement de ces approches et pour montrer leurs efficacités quant aux tâches à accomplir, ils ont été testés en simulation dans des scénarii traitant des objets fixes et mobiles dont le comportement futur est inconnu et en tenant compte des objets non perçus. Le modèle du robot utilisé correspond à la plateforme mobile *Robucar* disponible au CDTA[1].

5.1 Description du robot : *Robucar*

Le robot mobile *Robucar* est une plateforme expérimentale construite par la société *Robotsoft*[2] pouvant se déplacer dans un environnement extérieur (milieu urbain) (figure 5.1). C'est un prototype adapté pour tester des travaux visant la voiture

[1] Centre de développement des Technologies Avancées, équipe Navigation et Contrôle des Robots Mobiles autonomes (NCRM).

[2] www.robotsoft.fr

électrique intelligente. En fait, notre objectif est d'implémenter nos algorithmes sur des voitures commercialisées afin de les munir d'assez d'autonomie pour se déplacer sans conducteurs.

Le *Robucar* est un robot mobile de type voiture, il contient quatre roues motrices directives réparties en deux trains ; avant et arrière, de manière à être utilisé en mode simple braquage (motricité par le train avant, train arrière fixe) ou double braquage (motricité par le train avant et arrière). Pour être le plus fidèle possible au modèle de la voiture, le mode simple braquage est alors utilisé.

Le *Robucar* est équipé d'un PC embarqué, d'un système odométrique et de plusieurs capteurs extéroceptifs (caméra, ultrasons, lasers). Pour le travail réalisé dans cette thèse, le modèle de l'environnement sera uniquement basé sur les données collectées à partir des capteurs laser.

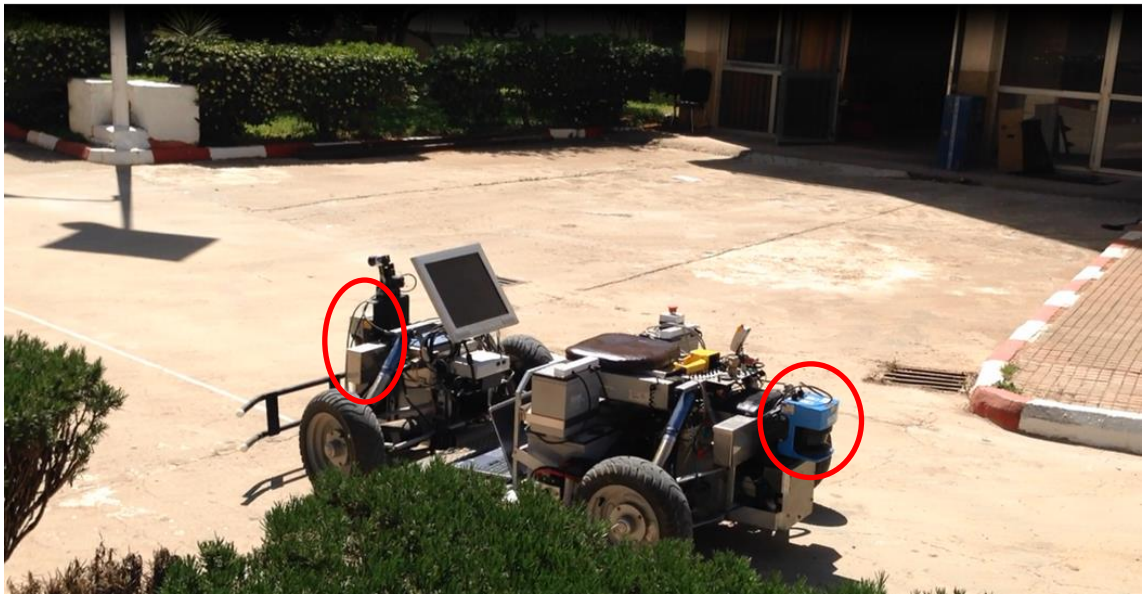


Figure 5.1 : Le robot mobile *Robucar* (les deux cercles rouges désignent les deux capteurs laser).

5.2 Modèle du robot

Le modèle du *Robucar* A est celui d'un système de type voiture avec deux roues arrières fixes et deux roues avants orientables (figure 5.2). Un état de A est

défini par un 5-uplet $s = (x, y, \Theta, v, \xi)$ avec (x, y) les coordonnées du centre de l'axe des roues arrières de A, Θ l'orientation de A, v la vitesse linéaire de A, et ξ l'orientation des roues avants (angle de braquage). Un contrôle de A est défini par le couple $u = (u_\alpha, u_\xi)$ avec u_α l'accélération linéaire de ses roues arrières et u_ξ la vitesse de braquage. Soit L la distance entre l'axe des roues avants et l'axe des roues arrières du système A. Le mouvement de A est régi par les équations différentielles suivantes :

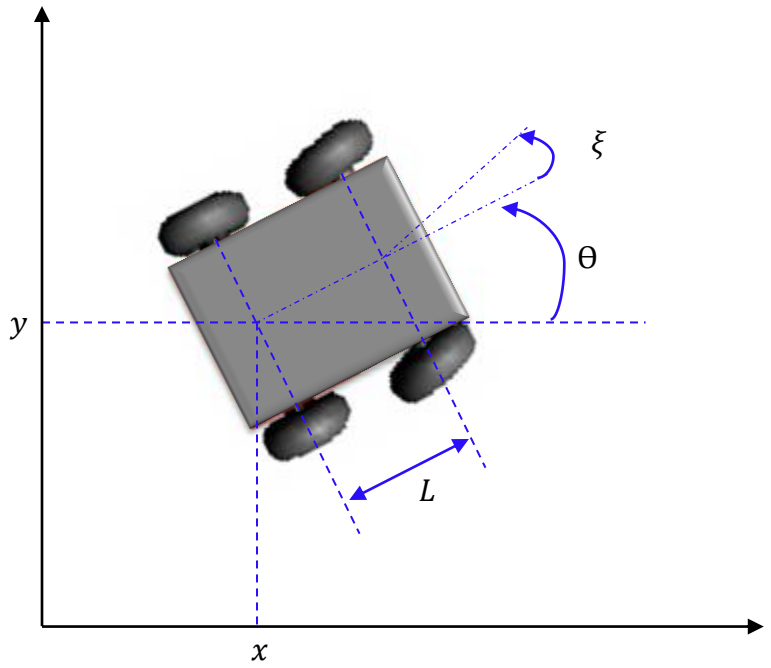


Figure 5.2 : Le véhicule de type voiture A.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\Theta} \\ \dot{v} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} v\cos\Theta \\ v\sin\Theta \\ v\tan\xi/L \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_\alpha + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_\xi \qquad (5.1)$$

Les contraintes additionnelles suivantes sont considérées:

$$|v| \leq v_{max}, |\xi| \leq \xi_{max}, |u_\alpha| \leq u_{\alpha_{max}}, |u_\xi| \leq u_{\xi_{max}} \qquad (5.2)$$

Avec :

$$v_{max} = 20m/s, \xi_{max} = 0.314rad, u_{\alpha_{max}} = 7m/s^2, u_{\xi_{max}} = 0.314rad/s \qquad (5.3)$$

5.3 Modèle de l'espace de travail

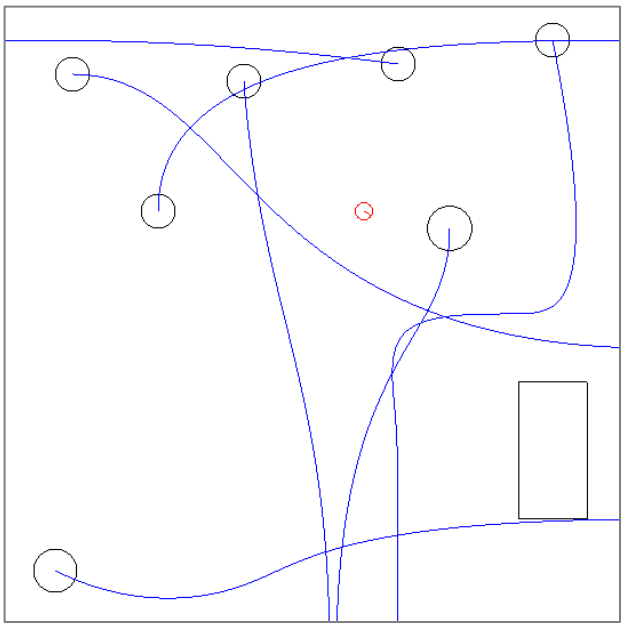


Figure 5.3: Un exemple de l'espace de travail: qui contient un rectangle fixe et sept disques mobiles (avec leurs trajectoires futures en bleu). Le robot A est le disque rouge.

L'environnement du robot mobile A est représenté par un espace de travail 2D ; WS. Sa taille est de 180×180 mètres. WS est peuplé d'objets sous forme de polygone ou disque, qui peuvent être statiques ou mobiles avec une vitesse maximale $v_{B_{max}} = 20m/s$. Le nombre d'obstacles dans l'environnement peut diminuer en fonction des obstacles sortant ou restant dans l'espace de travail (selon leur comportement futur). A est modélisé par un disque (en rouge), où la ligne liée au centre montre son orientation durant le déplacement. La Figure 5.3 illustre le type de scénario considéré ainsi que les trajectoires des obstacles mobiles.

5.4 ICS^b-CHECK

Afin de valider le concept ICS^b et démontrer son utilité, l'algorithme ICS^b-CHECK a été implémenté en simulation. Pour des besoins de calcul effectué par l'algorithme, celui-ci doit avoir recours à des techniques standards de rendu graphique afin d'augmenter son efficacité (voir paragraphe 5.4.1). L'algorithme ICS^b-CHECK a été testé dans des environnements dynamiques inconnus où le robot utilise uniquement l'information de son champ de vision limité (voir paragraphe 5.4.2).

5.4.1 Préliminaires

D'après l'algorithme 2, ICS^b-CHECK nécessite d'effectuer des opérations d'union et d'intersection (ligne #4, 6 et 8). La question qui se pose est que d'un point de vue pratique quel est le meilleur moyen pour y parvenir ? Une solution idéale par rapport à la simplicité et à la facilité d'implémentation, est de faire ces calculs graphiquement, et de bénéficier des performances du processeur graphique (GPU). Cela est réalisé grâce à la bibliothèque OpenGL[3] et en tirant profit du code couleur RGB (Red-Green-Blue). Pour des régions de dimensions arbitraires, ces opérations d'union et d'intersection peuvent être couteuses à implémenter. Cependant, dans notre cas, comme les régions sont planes, le problème ne se pose pas.

Le calcul d'unions et d'intersections de formes 2D arbitraires peut être alors réalisé en dessinant ces régions dans le buffer OpenGL, où chaque pixel correspond à un état donné. Le buffer OpenGL correspond dans notre cas à la tranche-$\hat{z}_c$. Le principe est d'assigner une couleur RGB à chaque trajectoire de freinage $\tilde{u}_b \in \mathcal{E}$. L'ensemble $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$ aura par conséquent la couleur correspondant à la trajectoire $\tilde{u}_b$ en question. Les différents ensembles $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$ (pour toutes les trajectoires de $\mathcal{E}$) sont alors dessinés en effectuant une opération logique "et" ($\vee$) au niveau du pixel entre la couleur de la trajectoire de freinage et la couleur du buffer OpenGL actuel (initialisée avec la couleur blanche, i.e. #FFFFFF, mais par la suite elle contient toutes les couleurs correspondant aux différents ensembles $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$).

[3] http://www/opengl.org.

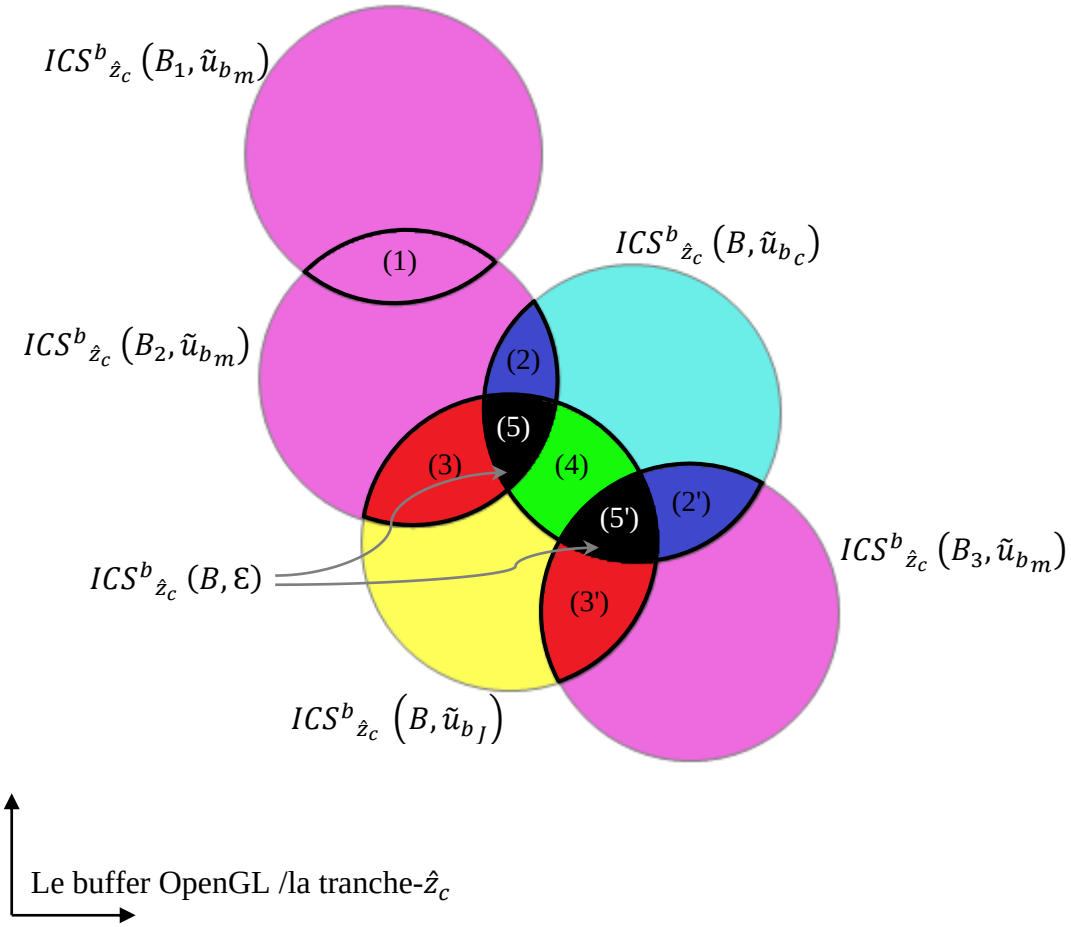


Figure 5.4 : Un exemple illustratif du calcul graphique de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$.

L'union est obtenue en dessinant une région sur une autre de la même couleur ou sur du blanc. Tandis que l'intersection est obtenue en dessinant une région sur une autre de couleur différente. L'intersection de toutes les couleurs (avec chaque couleur correspondant à une trajectoire de freinage de l'ensemble $\mathcal{E}$) donne la couleur noir (#000000). Donc, si un état donné correspond à un pixel noir, cela signifie que c'est un ICS^b (toutes les trajectoires de freinage conduisent à une collision). De ce fait, il faut établir la condition suivante et qui doit être vérifiée lors de l'affectation des couleurs. Soit clr_j désigne la couleur assignée à $\tilde{u}_b$ et $\delta\mathcal{E}$ un sous ensemble arbitraire de $\mathcal{E}$, il est établi que :

$$\bigvee_{\tilde{u}_b \in \mathcal{E}} clr_j = \#000000 \; et \bigvee_{\tilde{u}_b \in \delta\mathcal{E} \subset \mathcal{E}} clr_j \neq \#000000 \qquad (5.4)$$

Un exemple illustratif est donné dans la figure 5.4 montrant le buffer OpenGL (équivalent à la tranche-$\hat{z}_c$). $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$ est calculé pour trois objets B_1, B_2 et B_3 (avec $B = \bigcup_i B_i$) et pour un ensemble de trois trajectoires de freinage $\mathcal{E} = \{\tilde{u}_{b_m}, \tilde{u}_{b_c}, \tilde{u}_{b_J}\}$, ayant respectivement les couleurs magenta (#FF00FF), cyan (#00FFFF) et jaune (#FFFF00). Notons que les couleurs sont associées aléatoirement aux différentes trajectoires et que l'affectation des couleurs obéit à la condition (5.4). Supposons que tout d'abord, les trois ensembles $ICS^b{}_{\hat{z}_c}(B_1, \tilde{u}_{b_m})$, $ICS^b{}_{\hat{z}_c}(B_2, \tilde{u}_{b_m})$ et $ICS^b{}_{\hat{z}_c}(B_3, \tilde{u}_{b_m})$ sont dessinés. Ces trois ensembles leur sont affectés la même couleur, celle de la trajectoire de freinage $\tilde{u}_{b_m}$, à savoir le magenta. Le chevauchement des deux ensembles $ICS^b{}_{\hat{z}_c}(B_1, \tilde{u}_{b_m})$ et $ICS^b{}_{\hat{z}_c}(B_2, \tilde{u}_{b_m})$ (représentés par deux cercles) en utilisant l'opérateur logique "et" résulte en une région dont la couleur est également le magenta (la zone (1)). Cela est dû au fait que les trois ensembles correspondent à la même trajectoire de freinage. C'est ainsi que l'union est effectuée. L'ensemble $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_c})$ lui est affecté la couleur cyan (#00FFFF), i.e. la couleur attribuée à la trajectoire de freinage $\tilde{u}_{b_c}$. Quand cette région est dessinée, il apparait un chevauchement avec $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_m})$ qui est désigné par les deux zones (2) et (2'). Ces régions sont de couleur bleu (#0000FF) ; une couleur différente que celles des deux ensembles. Ces régions correspondent aux états pour lesquels les deux trajectoires de freinage ne sont pas sans collision. C'est ainsi que l'intersection de $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_m})$ et $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_c})$ est effectuée. La même chose arrive quand l'ensemble $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_J})$ est dessiné ; cinq nouvelles régions de chevauchement apparaissent : les deux régions (3) et (3') en rouge (#FF0000) résultant de l'intersection de $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_J})$ et $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_m})$, la région (4) en vert (#00FF00) résultant de l'intersection de $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_J})$ et $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_c})$, et enfin les deux régions (5) et (5') en noir qui est le résultat de l'intersection de

$ICS^b{}_{\hat{z}_c}\left(B, \tilde{u}_{b_J}\right)$, $ICS^b{}_{\hat{z}_c}\left(B, \tilde{u}_{b_c}\right)$ et $ICS^b{}_{\hat{z}_c}\left(B, \tilde{u}_{b_m}\right)$. Ces deux régions désignent $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$, i.e. les états pour lesquels, il n'existe aucune trajectoire de freinage sans collision.

5.4.2 ICS^b-CHECK en action

L'algorithme ICS^b-CHECK permet de mettre en œuvre le concept ICS de freinage et de tester si un état est ICS^b ou non (resp. non p-sûr ou p-sûr) en utilisant le modèle du futur de l'environnement. Afin de montrer comment ICS^b-CHECK fonctionne, il a été testé pour différents environnements contenant des obstacles fixes et mobiles. Pour simplifier l'illustration du calcul des ICS^b, dans un premier cas d'étude, uniquement les obstacles perçus sont traités en considérant un champ de vision illimité. Le deuxième cas d'étude est plus complexe, et répond plus à notre problématique en considérant un champ de vision limité. Le fonctionnement de l'algorithme ICS^b-CHECK y est montré en tenant compte de la présence d'obstacles perçus et non perçus (i.e. les limites du champ de vision, les régions occultées et les obstacles présents à l'extérieur du champ de vision). Une fois le principe de ICS^b-CHECK est assimilé, nous nous intéressons à analyser l'influence de certains paramètres sur l'ensemble ICS^b résultant (respectivement la région libre, i.e. ensemble d'états p-sûrs). Tous d'abord, l'impact de l'ensemble des trajectoires de freinage choisi est analysé : dans le troisième cas d'étude, c'est l'influence de l'accélération des trajectoires de freinage qui est analysée et dans le cas d'étude 4, c'est l'influence du nombre de ces trajectoires qui est analysée. Enfin, dans le dernier cas d'étude, l'influence de la vitesse maximale des obstacles mobiles est analysée.

5.4.2.1 Cas d'étude 1 : champ de vision illimité (obstacles perçus)

Dans un premier temps, afin d'illustrer comment le calcul des ICS^b est effectué, le problème est simplifié en supposant que tous les objets de l'environnement sont perçus (cette supposition ne sera plus valable pour le reste du document). De cette manière, il est montré comment calculer ICS^b pour un objet perçu statique ou mobile ayant un comportement futur donné, et par la suite comment calculer ICS^b pour

l'ensemble des objets de l'environnement et enfin déduire si l'état du robot est un ICS de freinage ou non.

Dans ce cas, nous considérons un espace de travail contenant 7 obstacles mobiles sous forme de disques et un obstacle statique polygonal (celui de la figure 5.3). L'état à vérifier pour être ICS^{b} ou non est $s_c = (30{,}20, -1{,}20{,}0)$ (représentant respectivement les coordonnées cartésiennes du robot (en mètres), son orientation (en radians), sa vitesse linéaire (en mètres/secondes) et enfin son angle de braquage (en radians)). ICS^{b}-CHECK commence par calculer l'ensemble ICS^{b} pour la tranche-$\hat{z}_c$ correspondante; $\hat{z}_c = (-1{,}20{,}0)$ et par la suite de trouver la couleur du pixel $(30{,}20)$ dans la tranche-$\hat{z}_c$. Si la couleur trouvée est noire alors l'état s_c est un ICS^{b}, sinon c'est un état p-sûr. La tranche-$\hat{z}_c$ est représentée dans la figure 5.5 ; les obstacles de l'environnement ont été isotropiquement agrandis (les courbes en bleu sont leurs trajectoires respectives). Le robot est réduit à un point, sa position correspond à l'état s_c (pour des raisons de clarté, il est représenté par un cercle rouge).

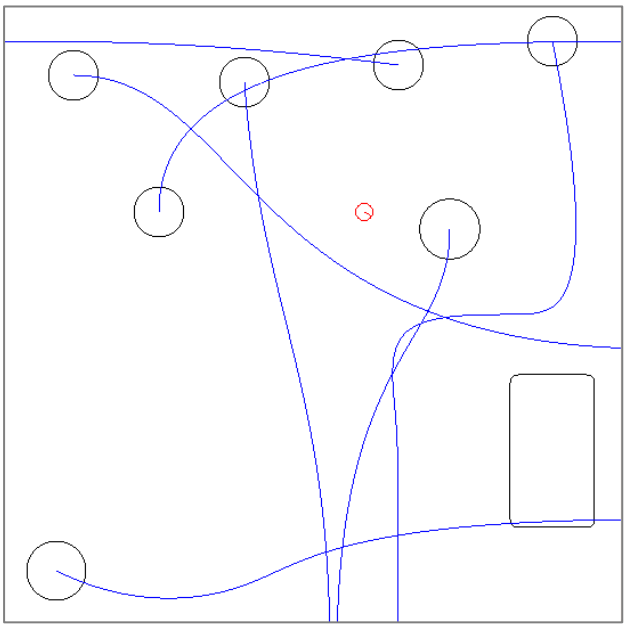


Figure 5.5 : Un environnement dynamique contenant des obstacles (isotropiquement agrandis) perçus par le robot (la tranche-$\hat{z}_c$).

L'ensemble des trajectoires de freinage utilisé par ICS^{b}-CHECK pour le calcul d'une approximation conservative de l'ensemble ICS^{b} (ligne #1, algorithme 2) est représenté dans la figure 5.6 (treize trajectoires sont considérées).

Le calcul de ICS^{b} de la tranche-$\hat{z}_c$ i.e. $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$, passe par plusieurs étapes tel expliqué dans l'algorithme 2 (voir chapitre 3) en exploitant les techniques du rendu graphique (voir paragraphe 5.4.1).

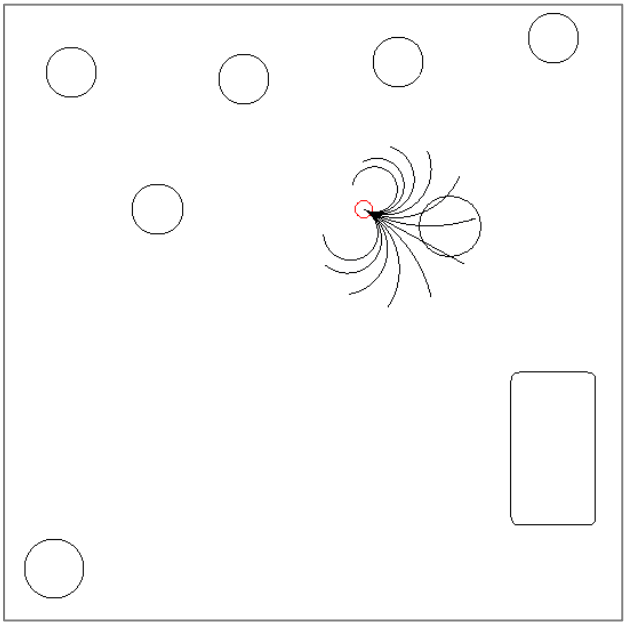
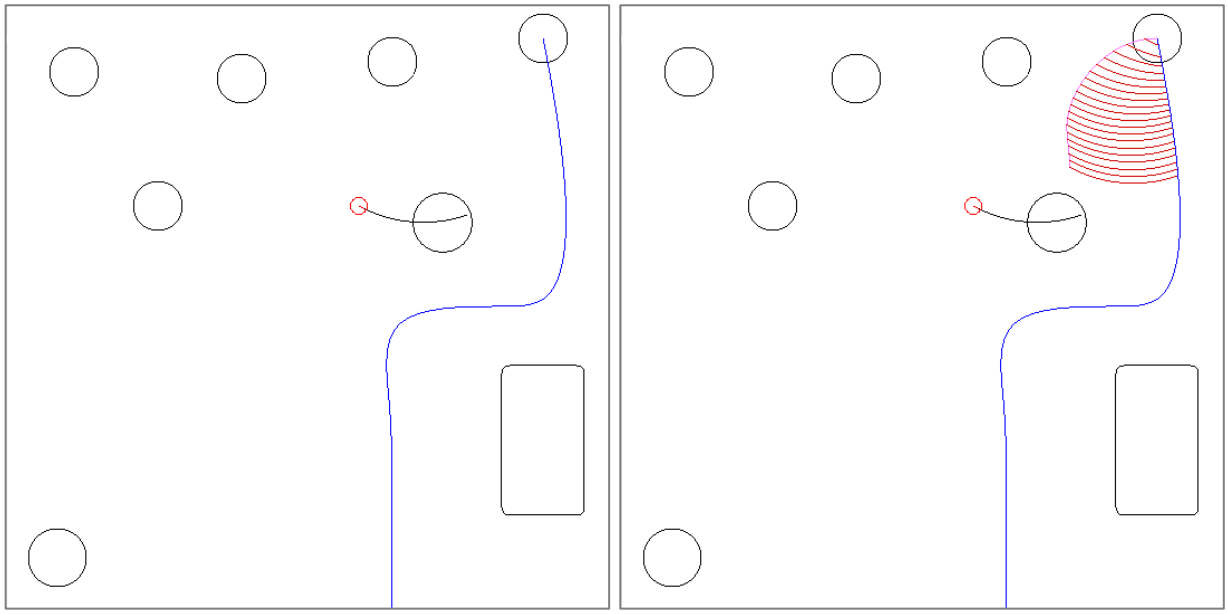


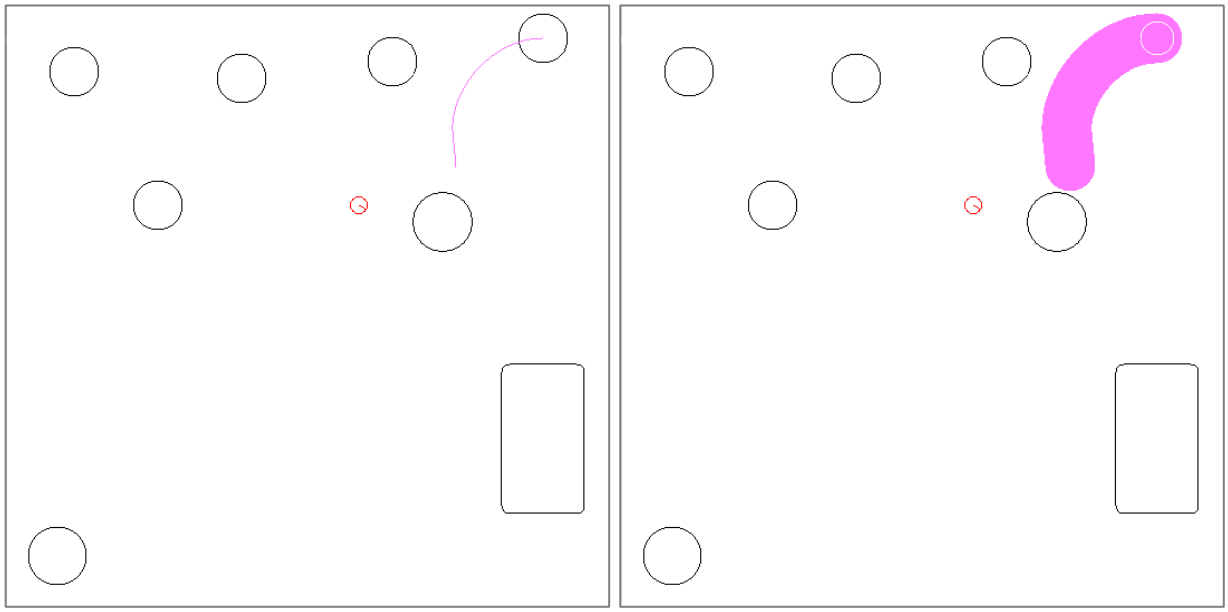
Figure 5.6 : L'ensemble des trajectoires de freinage considérées pour le calcul de ICS^{b}.

- Calcul de $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$:

L'algorithme ICS^{b}-CHECK commence par calculer $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$ (ligne #4, algorithme 2), i.e. l'ensemble ICS^{b} pour chaque trajectoire de freinage $\tilde{u}_b$ de l'ensemble $\mathcal{E}$ et chaque objet B_i de l'ensemble des objets de l'environnement. La figure 5.7 illustre comment ce calcul est réalisé. La trajectoire de freinage $\tilde{u}_b$ et l'objet B_i sont illustrés dans la figure 5.7.a. $\tilde{u}_b$ est représentée en noir, et B_i lui est associée la trajectoire future correspondante en bleu. Dans la figure 5.7.b, $ICS^b{}_{\hat{z}_c}(b_i, \tilde{u}_b, t)$ est calculé; où



(a) $\tilde{u}_b$, B_i (b) $ICS^b{}_{\hat{z}_c}(b_i, \tilde{u}_b, t)$



(a) $ICS^b{}_{\hat{z}_c}(b_i, \tilde{u}_b)$ (b) $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$

Figure 5.7 : Calcul de $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$.

b_i est le centre de l'objet B_i. En appliquant la transformation $T_{\tilde{u}_b}^{-1}(t)$ (voir paragraphe 3.7.2.1), il est alors possible de définir ICS^b de la tranche-$\hat{z}_c$ (trajectoire en magenta)

qui mène à une collision avec b_i à un temps particulier. Notons que l'objet se déplace selon une trajectoire infinie, $ICS^b{}_{\hat{z}_c}(b_i, \tilde{u}_b, t)$ doit être alors calculé jusqu'à l'infini. Cependant, selon le critère de la sûreté passive du mouvement, ce temps doit être limité par un horizon temporel (T_h). Le résultat du calcul $ICS^b{}_{\hat{z}_c}(b_i, \tilde{u}_b)$ pour tous les temps $t \in [0, T_h[$ est montré dans la figure 5.7.c. Enfin, la figure 5.7.d illustre le résultat du calcul de l'ensemble $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$.

- Calcul de $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_b)$:

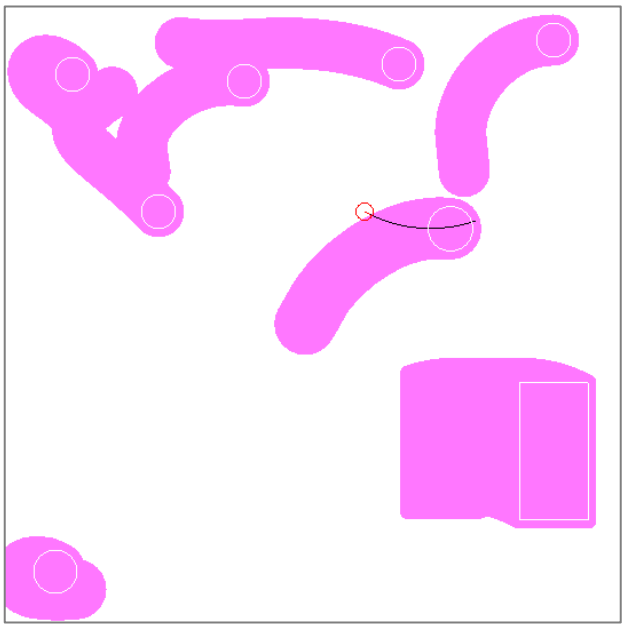
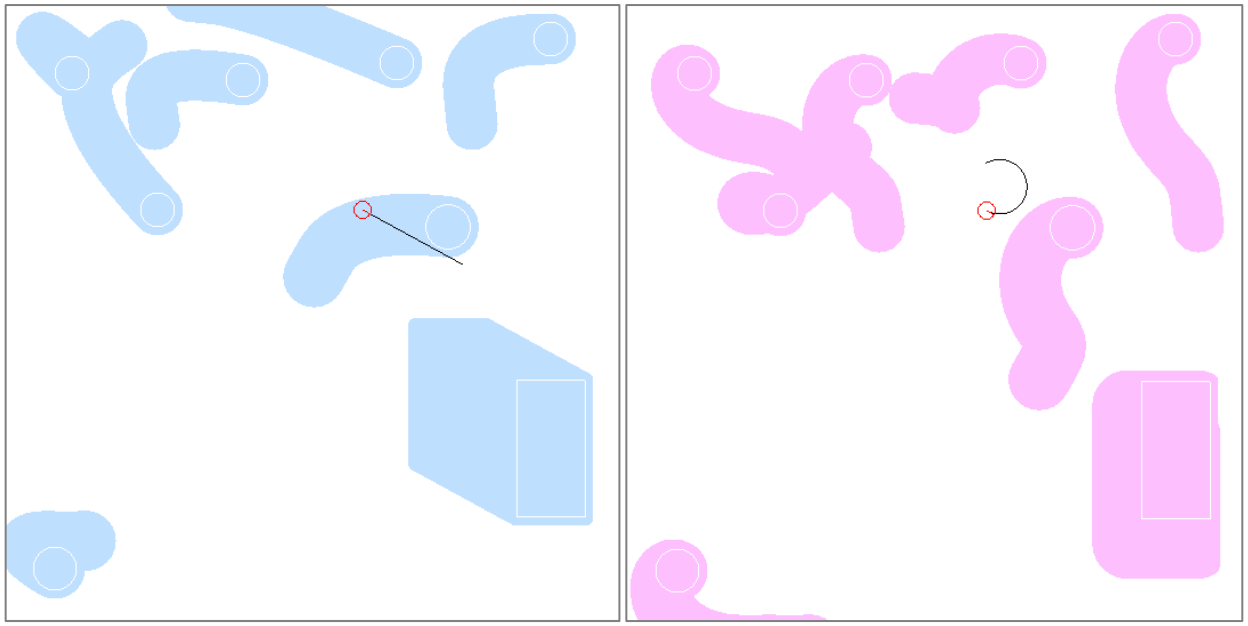
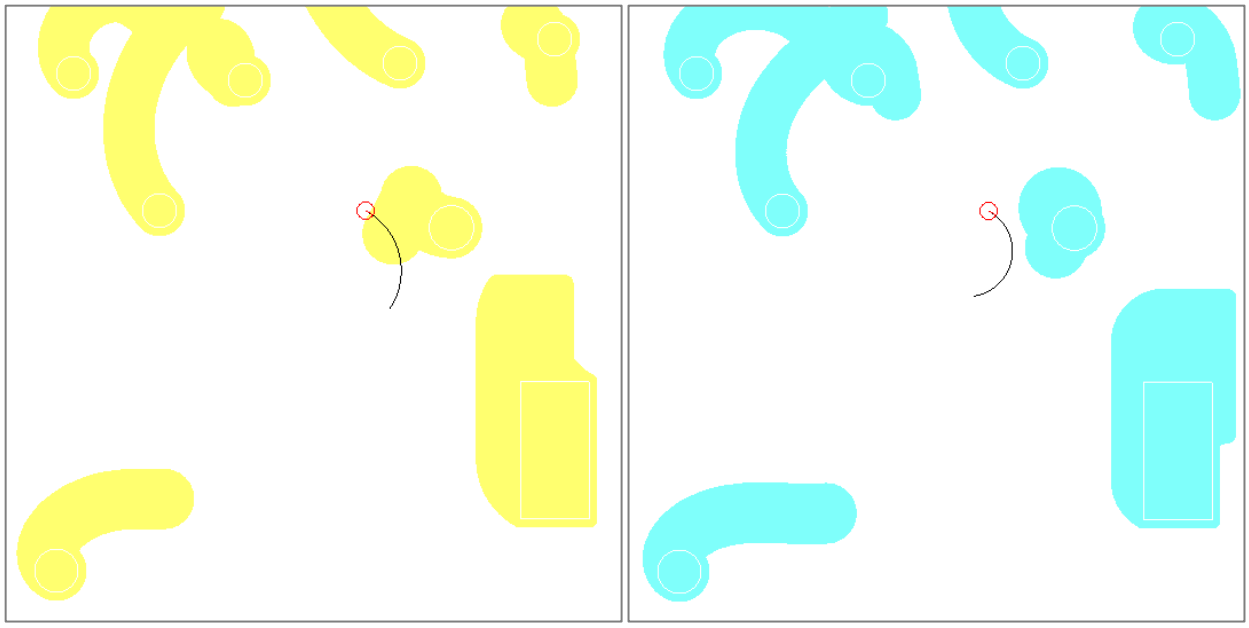


Figure 5.8 : Calcul de $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_b)$.

Après avoir calculé l'ensemble $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$, pour une trajectoire de freinage donnée et un objet donné. Le même calcul est effectué pour tous les objets de la tranche-$\hat{z}_c$ et pour la même trajectoire de freinage $\tilde{u}_b$. L'union des ensembles calculés résulte en l'ensemble $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_b)$ (ligne #6, algorithme 2) (voir figure 5.8). De la même manière, le calcul de l'ensemble $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_i})$ est répété pour chaque trajectoire de freinage $\tilde{u}_{b_i} \in \mathcal{E}$. La figure 5.9 montre les ensembles $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_i})$ calculés pour différentes trajectoires de freinage de l'ensemble $\mathcal{E}$ représenté dans la figure 5.6.



(a) $ICS^{b}{}_{\hat{z}_c}\left(B, \tilde{u}_{b_0}\right)$ (b) $ICS^{b}{}_{\hat{z}_c}\left(B, \tilde{u}_{b_5}\right)$

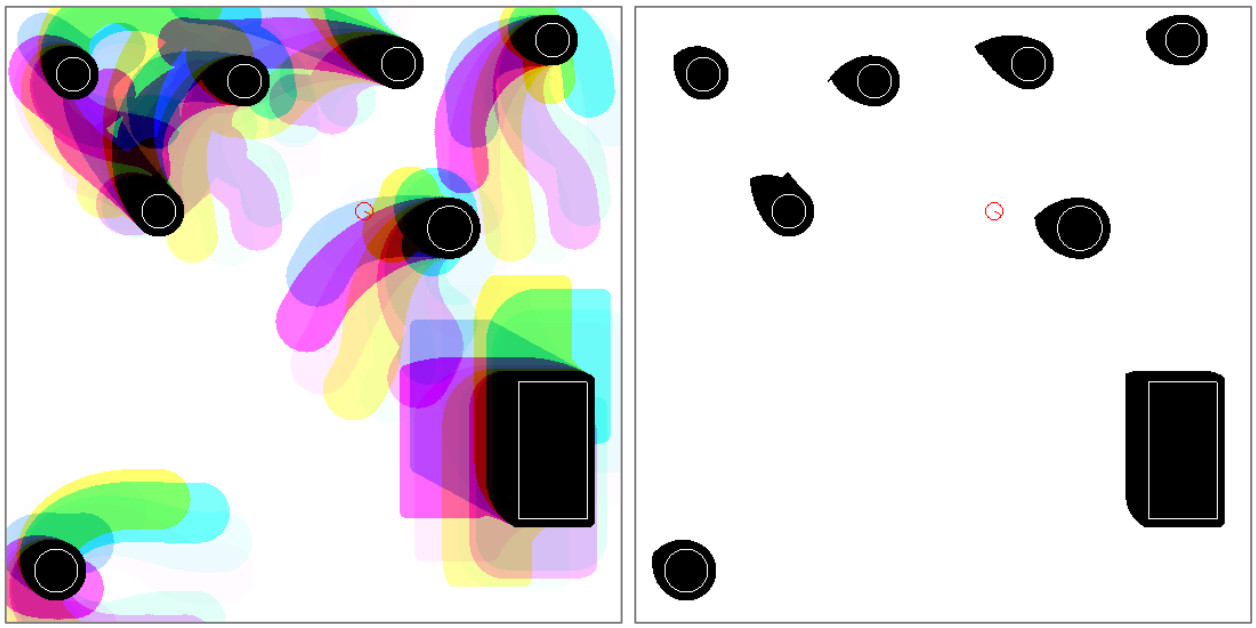


(c) $ICS^{b}{}_{\hat{z}_c}\left(B, \tilde{u}_{b_8}\right)$ (d) $ICS^{b}{}_{\hat{z}_c}\left(B, \tilde{u}_{b_9}\right)$

Figure 5.9 : Calcul de $ICS^{b}{}_{\hat{z}_c}\left(B, \tilde{u}_{b_i}\right)$ pour différentes trajectoires de freinage $\tilde{u}_{b_i} \in \mathcal{E}$.

- Calcul de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$:

Enfin, le fait de représenter les différents ensembles $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_b)$, $\tilde{u}_b \in \mathcal{E}$ dans le même buffer OpenGL, résulte en l'ensemble final $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$ pour tous les objets de l'environnement (B) et toutes les trajectoires de freinage ($\mathcal{E}$) (ligne #8 de l'algorithme 2), tel représenté dans la figure 5.10.



(a) $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$.

(b) Version noir et blanc de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$.

Figure 5.10 : Calcul de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$.

La dernière étape de ICS^{b}-CHECK consiste tout simplement à vérifier la couleur du pixel correspondant à l'état du robot s_c. Si le pixel est noir alors s_c est ICS^b, sinon il ne l'est pas (il est p-sûr) (ligne #9 à #13, algorithme 2). Dans le cas présent, s_c n'est pas ICS^b (i.e. p-sûr) car la couleur du pixel correspondant est différente du noir dans la figure 5.10.a. La figure 5.10.b est une version noir/blanc de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$ qui simplifie la représentation ; la région en blanc représente les états p-sûrs alors que les régions en noir représentent les états ICS^b (non p-sûrs). s_c appartient à la région blanche donc c'est bien un état p-sûr.

5.4.2.2 Cas d'étude 2 : champ de vision limité (obstacles perçus et non perçus)

Tel expliqué dans la partie problématique de cette thèse, l'une des grandes contraintes à considérer quant à la prise en compte du comportement futur de l'environnement pour la garantie de la sûreté du mouvement du robot, est son champ de vision limité. D'où la connaissance partielle et incomplète de l'environnement. Ce deuxième cas d'étude vise à montrer comment l'algorithme ICS$^{b\text{-}}$CHECK relève le défi, et illustrer comment il fonctionne réellement en tenant compte de ces contraintes pour une garantie de la sûreté passive du mouvement. L'algorithme a été testé dans deux situations différentes qui sont présentées dans ce qui suit. En opposition avec l'hypothèse du cas le plus défavorable considéré dans le paragraphe 2.3.4, il est supposé que le système de perception peut faire la différence entre les obstacles statiques et les obstacles mobiles et que le modèle du futur des obstacles mobiles perçus est disponible.

Situation #1 :

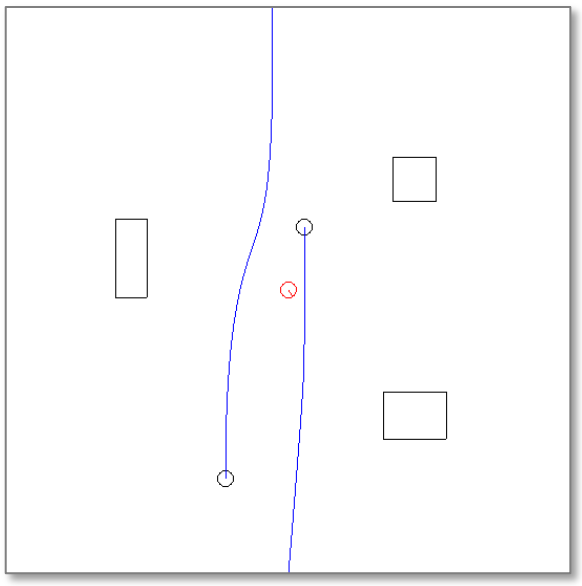


Figure 5.11 : Un environnement dynamique contenant deux obstacles mobiles et trois obstacles statiques.

Dans ce cas de figure, le robot se déplace dans un environnement contenant trois obstacles statiques (de forme polygonale) et deux obstacles mobiles (disques) (voir la figure 5.11). Les obstacles mobiles ont des vitesses aléatoires v_B limitées par la vitesse maximale $v_{B_{max}}$ ($v_B \leq v_{B_{max}}$).

- Description de la limite du champ de vision

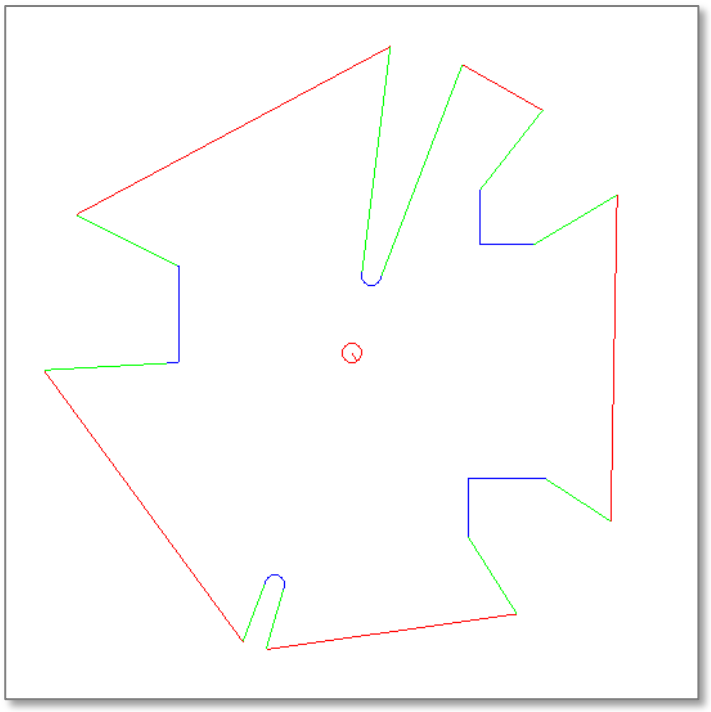


Figure 5.12 : Champ de vision FoV et sa limite ∂FoV (pour le cas de la figure 5.11) : les obstacles mobiles et fixes (perçus) sont en bleu, les limites de la perception sont en rouge et les occlusions son en vert.

En supposant que A est équipé d'un télémètre laser omnidirectionnel monté au centre de A, le champ de vision de A est alors un cercle dont le rayon correspond à la portée maximale du capteur laser. La figure 5.12 représente le champ de vision de A pour l'environnement de la figure 5.11. Il a été calculé en utilisant la technique du "graphe de visibilité" [29, 196, 197, 198]. Les arcs circulaires correspondant à la portée maximale du laser ont été remplacés par des segments de droites ; cette simplification conservative pourrait facilement être levée. Dans cette figure, il est

illustré comment la limite du champ de vision ∂FoV est partitionnée en trois parties ; ∂FoV^f, ∂FoV^m et ∂FoV^u correspondant respectivement aux objets fixes, mobiles et non perçus. Les objets fixes et mobiles (perçus) sont représentés en bleu. Les objets non perçus désignent les limites de perception (représentés en rouge) et les occlusions (représentés en vert). Le modèle du futur utilisé pour ∂FoV^f, ∂FoV^m est construit selon les principes énoncés dans le paragraphe 2.3.4.

- ICS^{b}-CHECK à l'œuvre

ICS^{b}-CHECK est appelé pour déterminer si l'état du robot est ICS^{b} ou non. L'état à vérifier est $s_c = (0{,}0, -1{,}20{,}0)$. De la même façon que pour le cas d'étude précédent, ICS^{b}-CHECK calcule l'ensemble ICS^{b} correspondant à la tranche-$\hat{z}_c$ avec $\hat{z}_c = (-1{,}20{,}0)$.

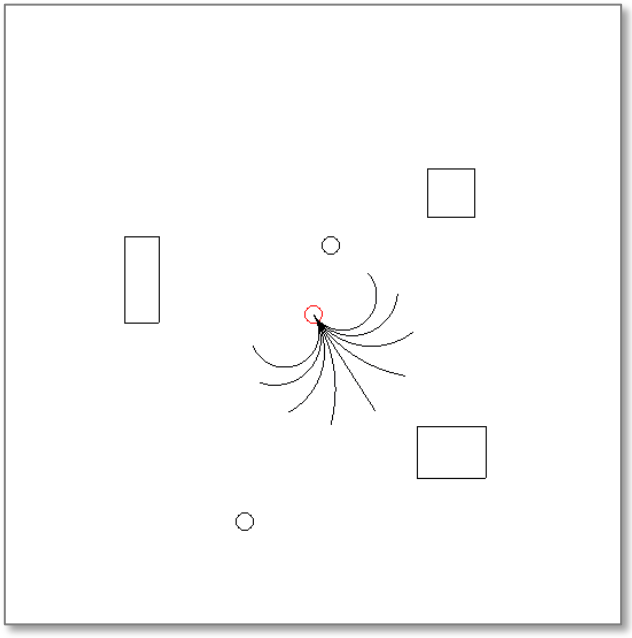


Figure 5.13 : Ɛ, l'ensemble des trajectoires de freinage considéré par ICS^{b}-CHECK.

Un ensemble de trajectoires de freinage Ɛ doit être sélectionné. Ces trajectoires peuvent être choisies arbitrairement grâce à l'approximation conservative de l'ensemble ICS^{b} (selon la propriété 4). L'ensemble d'échantillonnage de l'espace de contrôle U est obtenu grâce à une discrétisation régulière de l'ensemble de contrôle 2D $[-u_{\alpha_{max}}, u_{\alpha_{max}}] \times [-u_{\xi_{max}}, u_{\xi_{max}}]$, à partir de là, l'ensemble des

trajectoires de freinage ℇ est sélectionné. Dans ce cas, ℇ comprend neuf trajectoires de freinage définies par une décélération linéaire constante $u_\alpha = -u_{\alpha_{max}}$ et une vitesse de braquage constante $|u_\xi| \leq u_{\xi_{max}}$. Ces trajectoires de freinage sont représentées dans la figure 5.13.

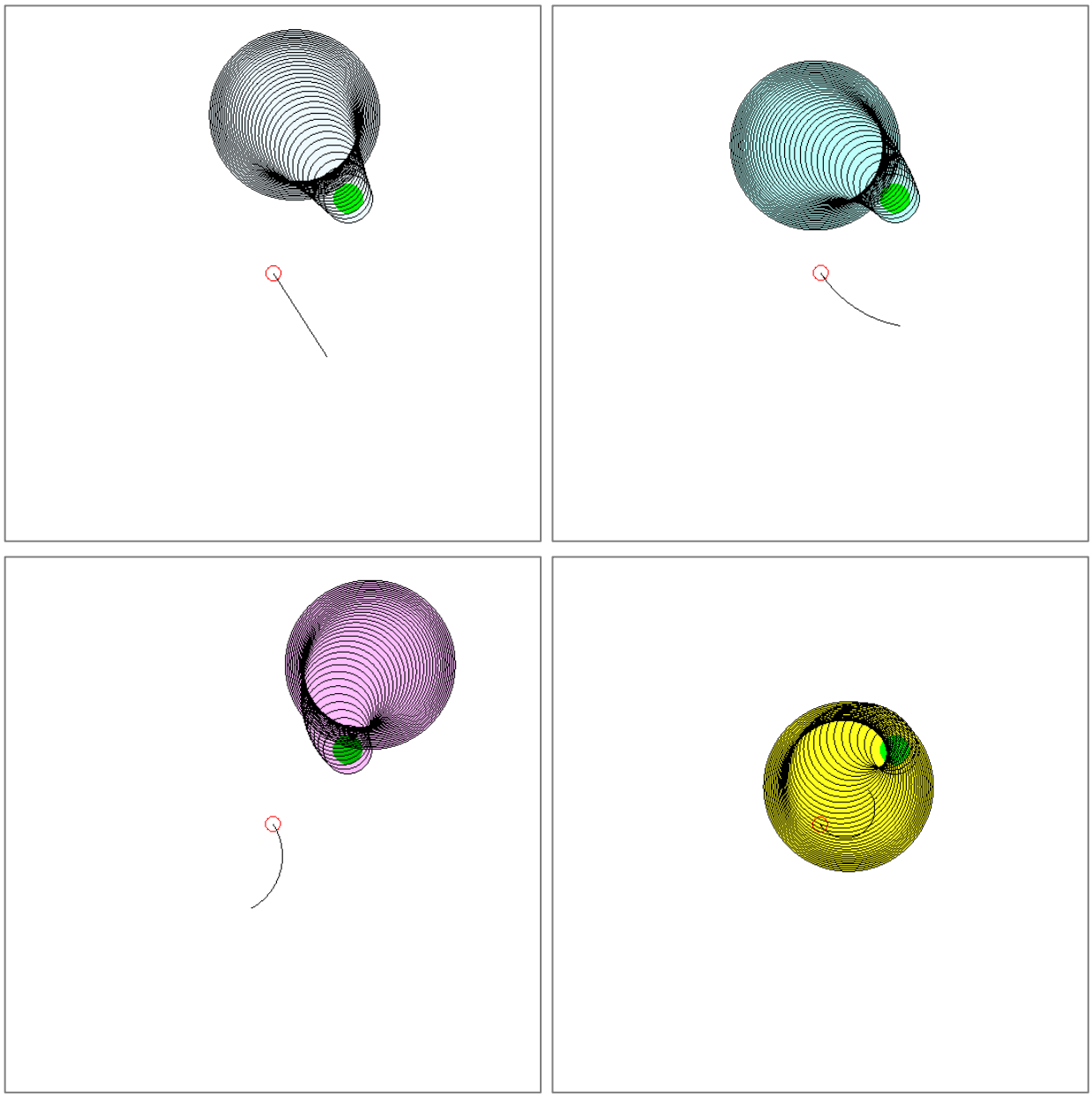


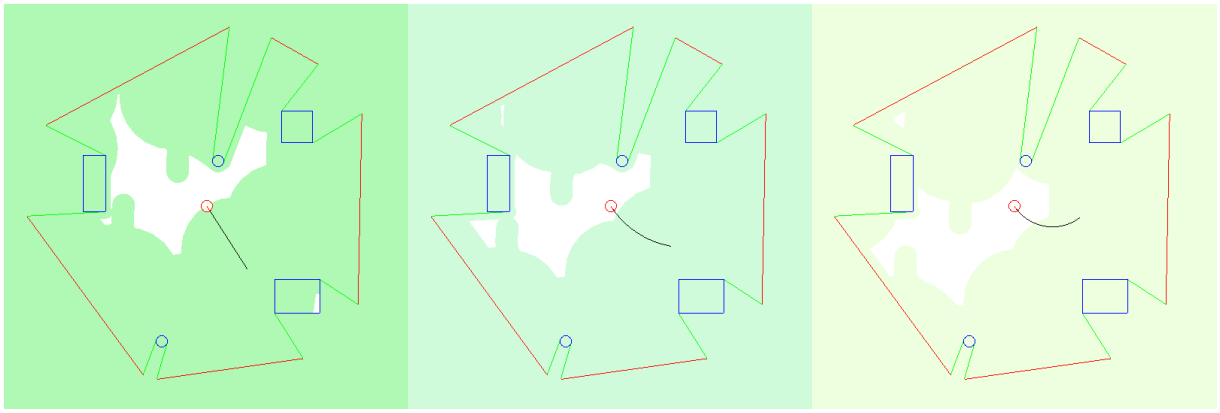
Figure 5.14 : Calcul de $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$ pour un objet inattendu (le disque en vert) et pour différentes trajectoires de freinage.

Pour chaque trajectoire de freinage $\tilde{u}_b \in \mathcal{E}$, l'ensemble $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_b)$ est calculé. Comme illustré dans le cas d'étude précédent, $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_b)$ est obtenu en calculant l'ensemble $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$ pour chaque objet de l'environnement. Seulement, dans ce cas, les objets peuvent être de différentes natures et avec des modèles du futur différents. Ils peuvent être perçus ou inattendus (non perçus). Les objets perçus (fixes ou mobiles) suivent une trajectoire nominale, le calcul de l'ensemble $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$ se fait comme pour le premier cas d'étude. Cependant, pour les objets inattendus, i.e. limite du champ de vision et les occlusions, le comportement futur de chaque point de ces ensembles est modélisé par un cône (des disques croissants).

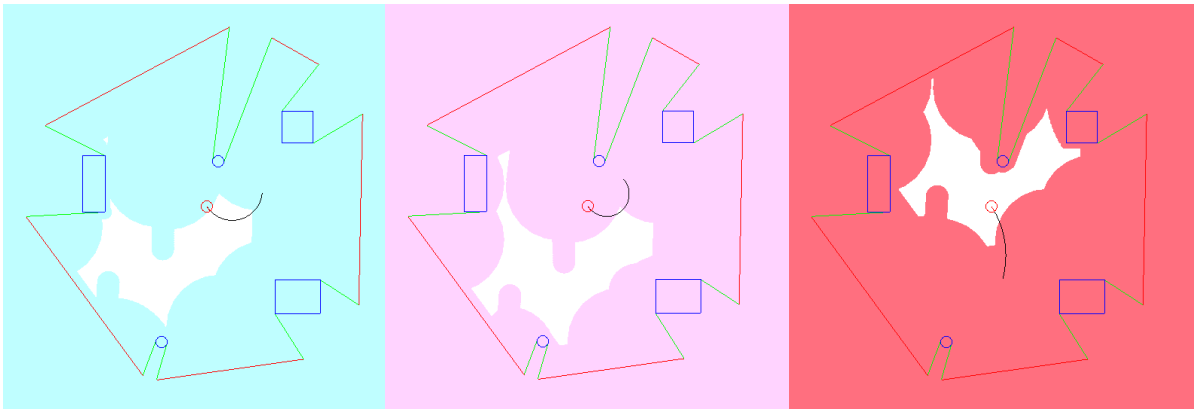
Un exemple de calcul de $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$ pour un objet inattendu quelconque est donné dans la figure 5.14. Notons qu'utilisant un modèle conservatif, le disque peut croitre jusqu'à l'infinie mais comme expliqué dans le paragraphe 2.3.4, cette croissance est limité par l'horizon temporel (dans ce cas $T_h = 2.8s$). Par ce fait, la croissance des régions ICS^b est aussi limitée par ce même élément (voir propriété 5), ce qui est clair dans la figure.

Pour l'environnement de la figure 5.11, $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}_b)$ est calculé pour chaque point de $\partial FoV = B$ (selon le modèle du futur correspondant). En exploitant les techniques du rendu graphique, $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_b)$ donne une région d'une couleur donnée dans le buffer OpenGL représentant la tranche-$\tilde{z}_c$. La Figure 5.15 illustre les régions correspondant à chacune des neuf trajectoires de freinage. Pour une trajectoire de freinage donnée $\tilde{u}_b$, un état correspondant à un pixel dans la région colorée est un ICS^b. Toutes les étapes de ICS^b-CHECK nécessitant le calcul d'unions et d'intersections de formes arbitraires sont accomplies de manière très efficace dans ce buffer OpenGL, en tirant profit du code couleur Rouge-Vert-Bleu et de l'application de l'opérateur logique "et" (voir paragraphe 5.4.1). Le résultat final est illustré dans la figure 5.16, ce qui correspond à $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$: résultat de l'intersection de tous les $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_b)$ calculés pour chaque trajectoire $\tilde{u}_b$ (figure 5.15.a-i). La version noir et blanc (N/B) de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$ est donnée dans la figure 5.17. Il apparait que $s_c =$

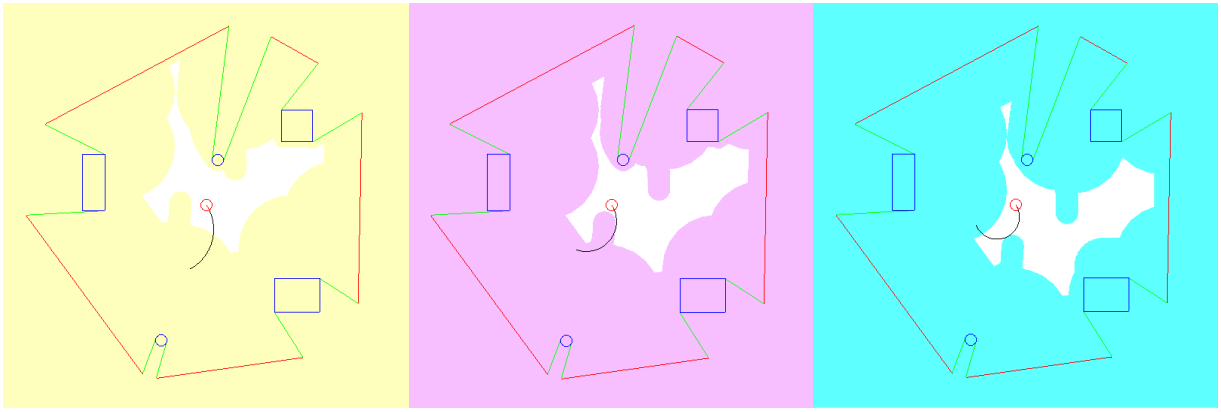
$(O, 0, -1{,}20, 0)$ n'est pas un ICS^b : la couleur du pixel $(0,0)$ dans la tranche-$\hat{z}_c$ n'est pas noire. Par ce fait, s_c est un état p-sûr.



(a) $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_0})$ (b) $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_1})$ (c) $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_2})$



(d) $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_3})$ (e) $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_4})$ (f) $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_5})$



(g) $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_6})$ (h) $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_7})$ (i) $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_{b_8})$

Figure 5.15 : $ICS^b{}_{\hat{z}_c}(B, \tilde{u}_b)$ pour différentes trajectoires de freinage.

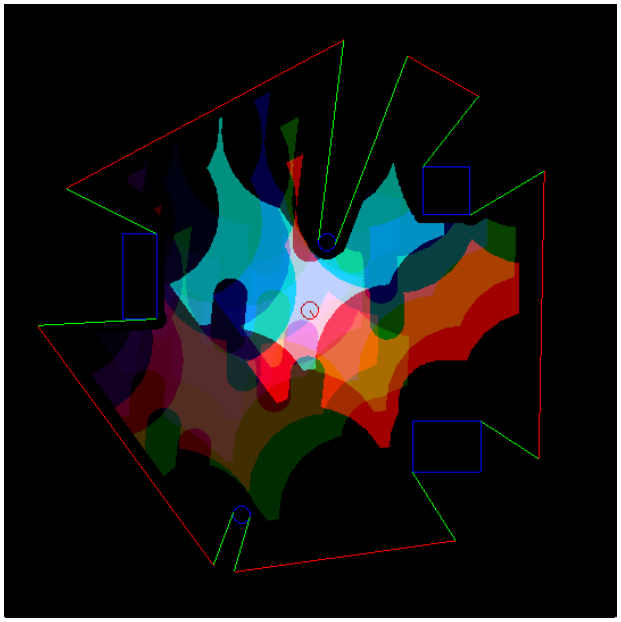


Figure 5.16 : La tranche-$\hat{z}_c$ 2D de l'espace d'état 5D de A : une région d'une couleur donnée indique que c'est un ICS^b pour la trajectoire de freinage correspondante. Les régions noires sont ICS^b.

Figure 5.17 : La version noir et blanc de la figure 5.16 : les régions blanches correspondent aux états p-sûrs.

Situation#2 :

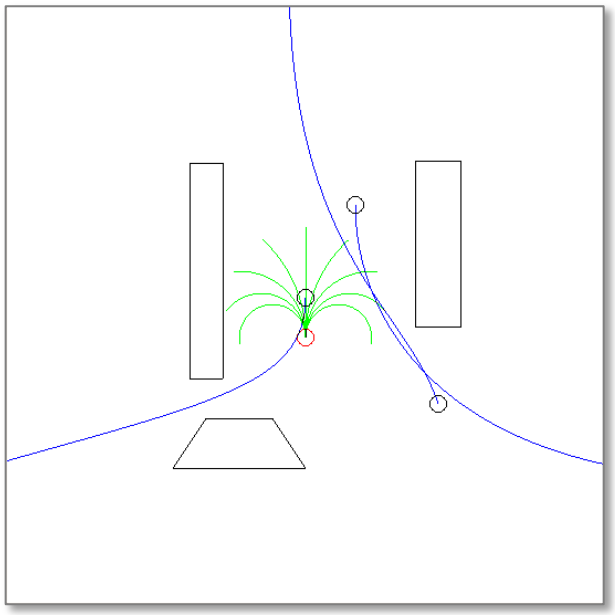


Figure 5.18 : Un environnement dynamique contenant trois obstacles fixes et trois obstacles mobiles. Les trajectoires de freinage utilisées par ICS^{b}-CHECK (l'ensemble $\mathcal{E}$) sont représentées en vert.

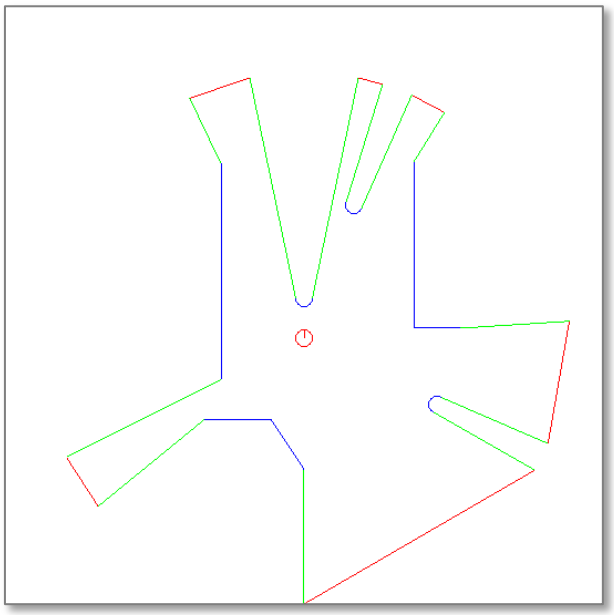


Figure 5.19 : Champ de vision FoV et sa limite ∂FoV (pour l'environnement de la figure 5.18) : les obstacles mobiles et fixes (perçus) sont en bleu, les limites de la perception sont en rouge et les occlusions son en vert.

Cet espace de travail contient trois obstacles fixes polygonaux et trois obstacles mobiles sous forme de disques (voir figure 3.18). Comme pour le cas précédent, à cause du champ de vision limité du robot et pour garantir la sûreté du mouvement, les obstacles inattendus sont également considérés, i.e. ∂FoV^{u} : les limites de perception et les occlusions. La limite du champ de vision de A est représentée dans la figure 5.19. Cependant, dans ce cas de figure, le robot se trouve dans une situation de difficulté ; pour sortir d'un couloir créé par deux obstacles fixes, il est bloqué par un obstacle mobile qui arrive droit sur lui. L'état à vérifier pour être ICS^{b} ou non est $s_c = (0{,}0{,}1.57{,}20{,}0)$. ICS^{b}-CHECK calcule l'ensemble ICS^{b} correspondant à la tranche-$\hat{z}_c$, avec $\hat{z}_c = (1.57{,}20{,}0)$. L'ensemble des trajectoires de freinage $\mathcal{E}$ sélectionné est représenté dans la figure 3.18. La sortie finale du processus est illustrée dans la figure 5.20 et 5.21.

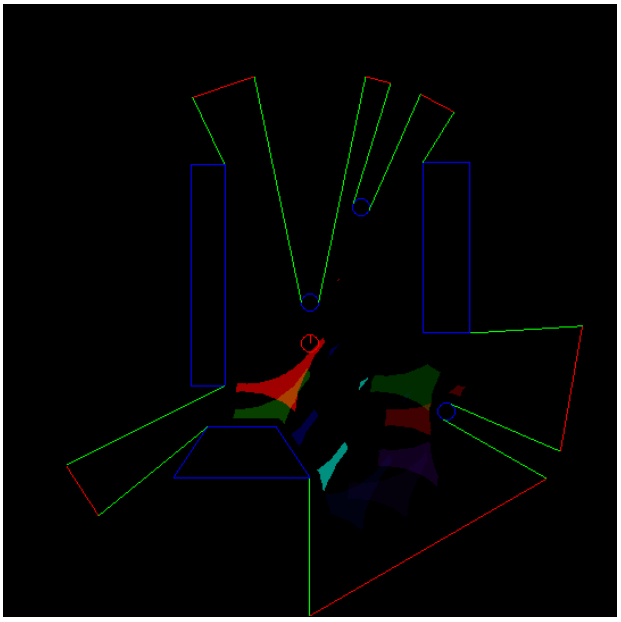


Figure 5.20 : $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$

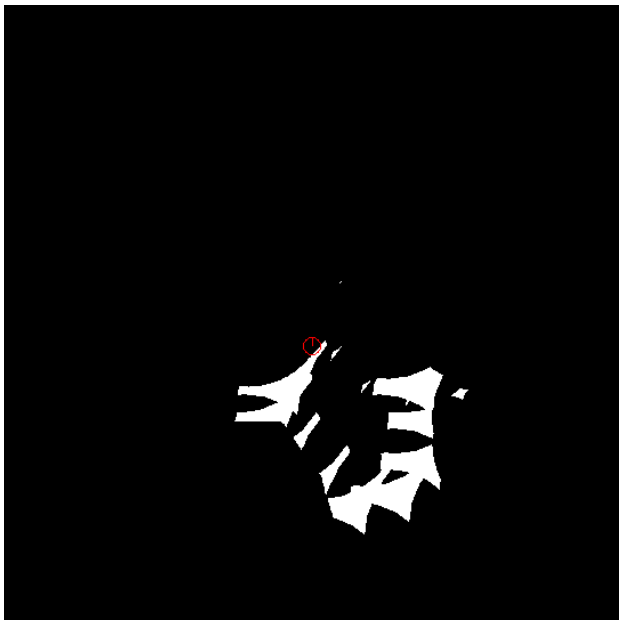
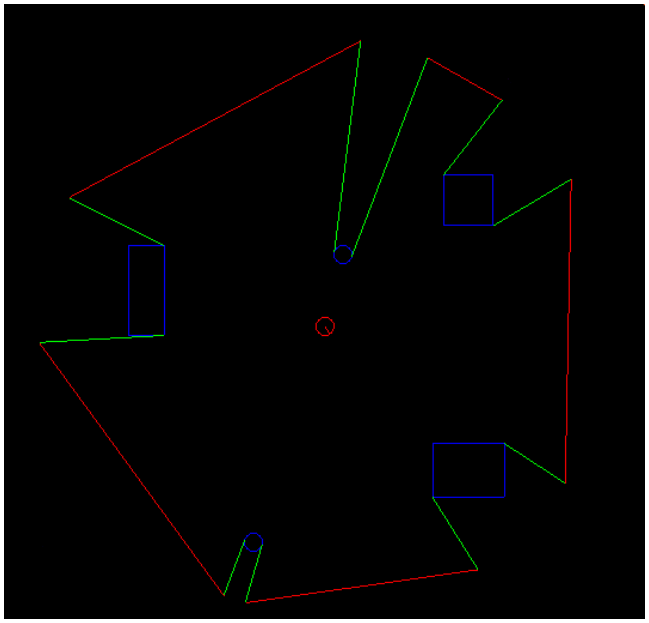


Figure 5.21 : Version N/B de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$.

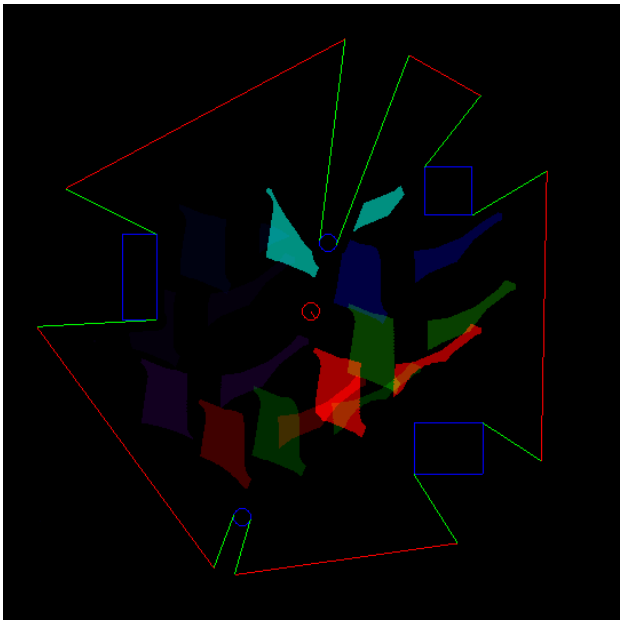
L'état s_c est un ICS^b, car il fait partie de la région interdite (le pixel correspondant à l'état est de couleur noir). Donc, c'est un état qui est non p-sûr.

5.4.2.3 Cas d'étude 3 : (Calcul de ICS^b pour différentes valeurs de l'accélération)

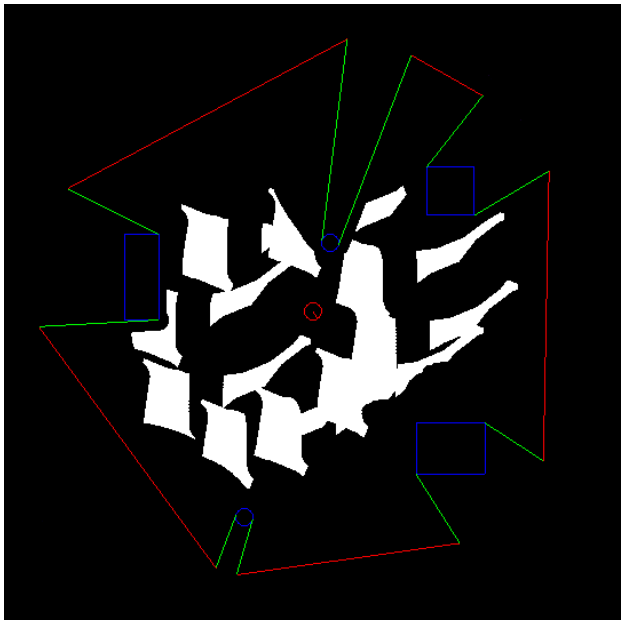
En utilisant un modèle conservatif de l'environnement, le calcul des ensembles ICS^b nécessite que le raisonnement sur le futur soit limité par un horizon temporel fini (propriété 5), dont la valeur est déterminée par l'équation (3.6). Cet horizon temporel dépend des trajectoires de freinage utilisées ; selon l'équation (3.6), il dépend du temps de freinage qui à son tour est en fonction de l'accélération (le temps de freinage est le rapport entre la vitesse linéaire du robot et son accélération linéaire). Afin de montrer plus concrètement l'influence de l'horizon temporel sur la sûreté du mouvement, l'ensemble ICS^b est calculé pour différentes valeurs de l'accélération. L'environnement de test représenté dans la figure 5.11 (dans le cas d'étude 1) est repris dans ce cas d'étude. L'ensemble des trajectoires de freinage utilisé par ICS^b-CHECK est fixé à neuf trajectoires (voir figure 5.13), qui sont définies par une décélération linéaire constante $u_\alpha = -u_{\alpha_{max}}$ et une vitesse de braquage constante $|u_\xi| \leq u_{\xi_{max}}$ qui varie entre $-0.12\ rad/s$ et $0.12\ rad/s$ avec un pas de $0.03\ rad/s$. La figure 5.22 montre le résultat du calcul de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$ pour cinq valeurs différentes de l'accélération (avec $|u_\alpha| \leq u_{\alpha_{max}}$), à savoir $u_\alpha = 3m/s^2, 4m/s^2, 5m/s^2, 6m/s^2$ et $7m/s^2$.



(a) Calcul de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$ pour $u_\alpha = 3m/s^2$.

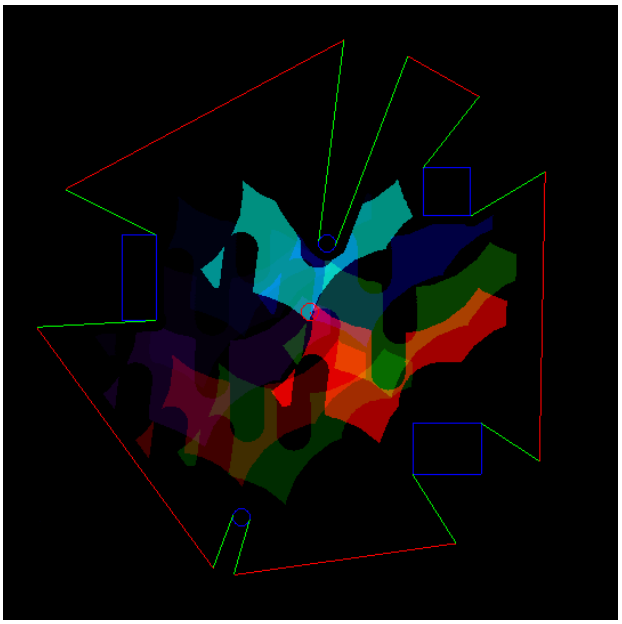


$ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$

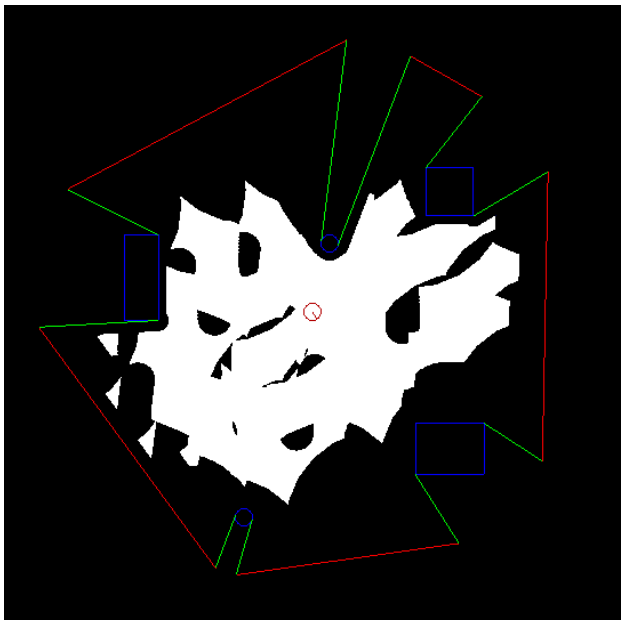


Version N/B de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$

(b) calcul de l'ensemble ICS^b pour $u_\alpha = 4m/s^2$.

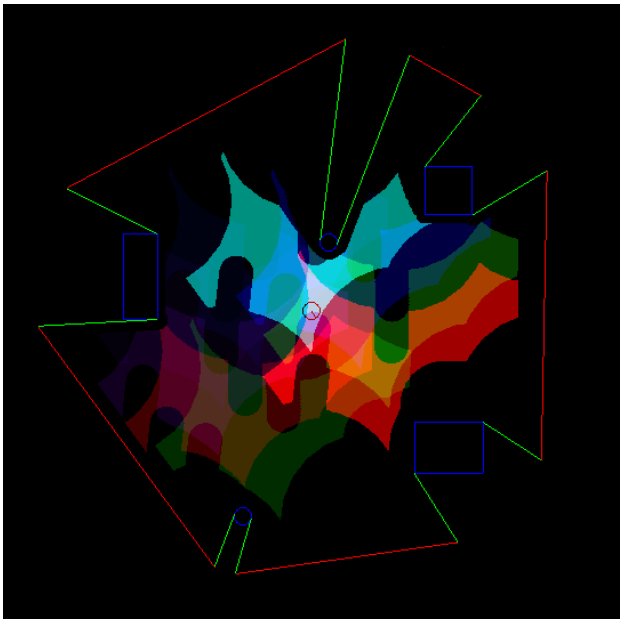


$ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$

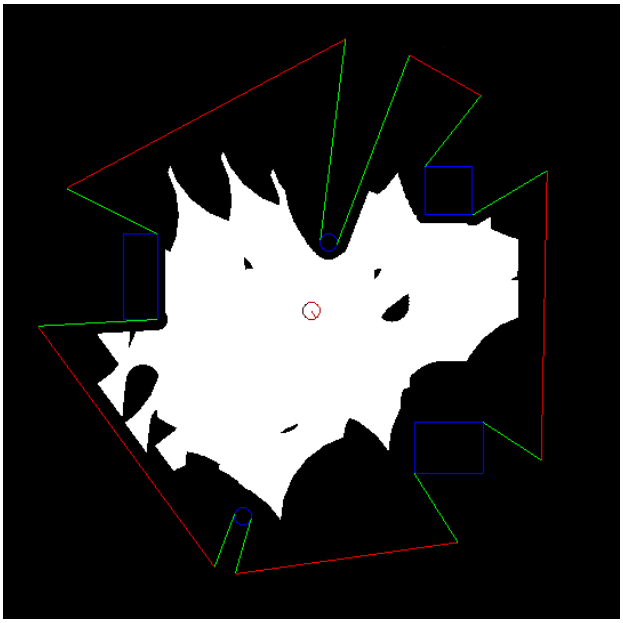


Version N/B de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$

(c) calcul de l'ensemble ICS^b pour $u_\alpha = 5m/s^2$.

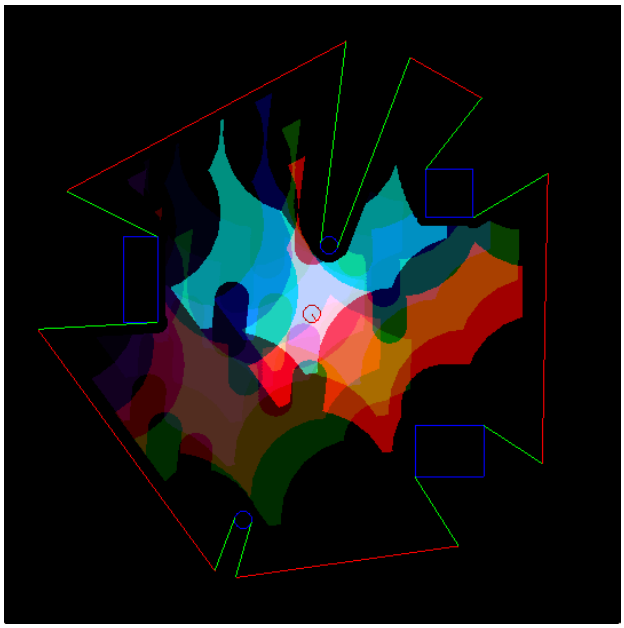


$ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$

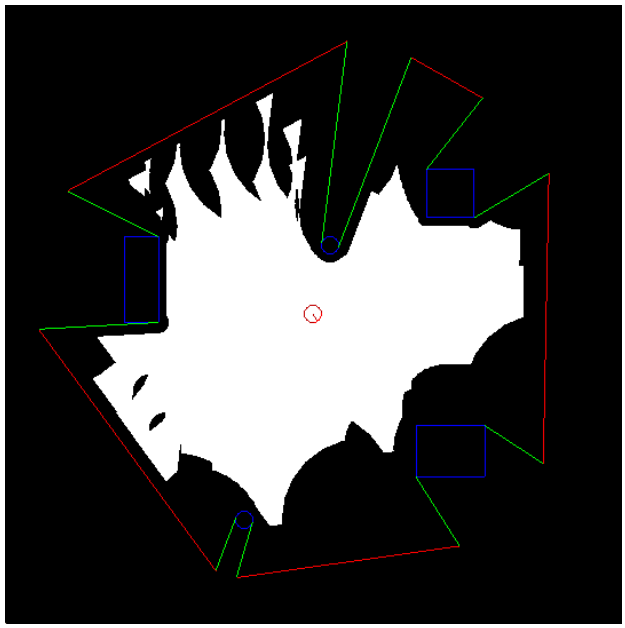


Version N/B de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$

(d) calcul de l'ensemble ICS^b pour $u_\alpha = 6m/s^2$



$ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$



Version N/B de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$

(e) calcul de l'ensemble ICS^b pour $u_\alpha = 7m/s^2$.

Figure 5.22 : Calcul de l'ensemble ICS^b pour différentes valeurs de l'accélération.

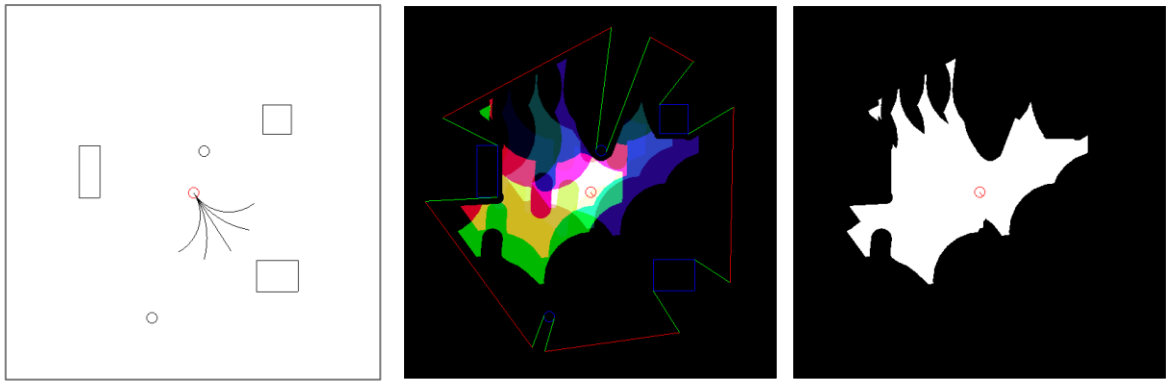
Il est clair dans cette expérience que la région libre (les états p-sûrs) croit avec la croissance de la valeur de l'accélération. Une analyse quantitative vient soutenir ces résultats ; le pourcentage d'états p-sûrs (pixels de couleur blanche) est calculé en fonction des différentes valeurs de l'accélération (voir le tableau 5.1). La première colonne du tableau indique qu'aucun état p-sûr n'est présent (cas de la figure 5.22.a). Cela peut s'expliquer par rapport à l'horizon temporel résultant. Etant inversement proportionnel à l'accélération, l'horizon temporel décroit quand l'accélération croit. Quand T_h est grand, tout l'environnement devient interdit (tous les états sont ICS^b). Ce qui est dû au fait que, utilisant un modèle conservatif, la taille des objets inattendus croit avec le temps jusqu'à ce qu'ils occupent tout l'environnement (à un certain temps dans le futur). Au contraire, quand T_h est plus petit (l'accélération est plus grande), l'environnement devient plus libre (voir les figure 5.22.b, 5.22.c, 5.22.d par exemple). Nous constatons également à partir du tableau que le pourcentage le plus élevé correspond à l'accélération maximale. Ce cas contient alors le plus d'états p-sûrs (voir la figure 5.22.e). C'est pour cette raison que dans notre cas, nous utilisons des trajectoires de freinage avec une accélération maximale.

Tableau 5.1 : Le pourcentage d'états p-sûrs dans l'environnement en fonction de l'accélération.

$u_\alpha(m/s^2)$	3	4	5	6	7
Pourcentage d'états p-sûrs (pixels blancs) (%)	0	8.6	17.5	20.3	22.4

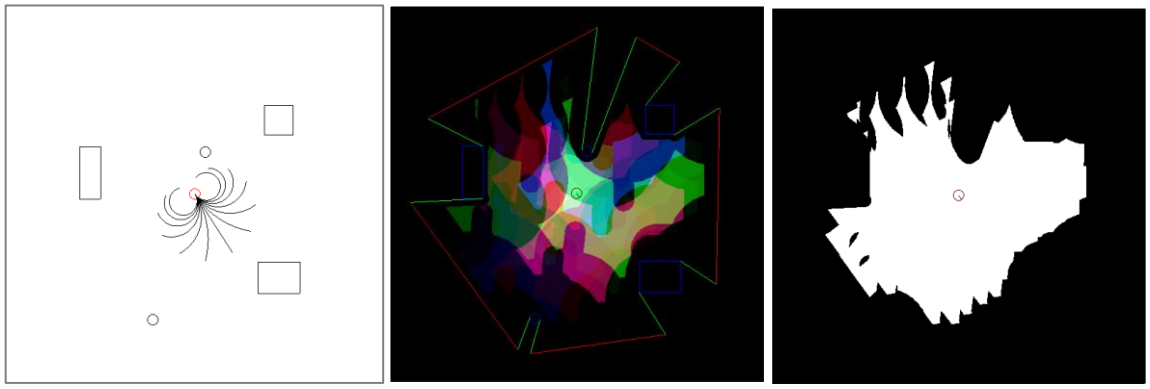
5.4.2.4 Cas d'étude 4 : (Calcul de ICS^b pour différents nombres de trajectoires de freinage)

Dans ce cas, il est question de comparer le résultat du calcul de l'ensemble ICS^b pour un même environnement de test mais pour différents nombres de trajectoires de freinage. L'environnement de test utilisé est celui de la figure 5.11.



(a) ε^- (b) $ICS^b{}_{\hat{z}_c}(B, \varepsilon^-)$ (c) version N/B de $ICS^b{}_{\hat{z}_c}(B, \varepsilon^-)$

Figure 5.23 : Calcul de l'ensemble ICS^b utilisant un ensemble de trajectoire de freinage ε^- (pour l'environnement de la figure 5.11).



(a) ε^+ (b) $ICS^b{}_{\hat{z}_c}(B, \varepsilon^+)$ (c) version N/B de $ICS^b{}_{\hat{z}_c}(B, \varepsilon^+)$

Figure 5.24 : Calcul de l'ensemble ICS^b utilisant un ensemble de trajectoire de freinage ε^+ (pour l'environnement de la figure 5.11).

Dans un premier temps, et relativement aux résultats obtenus dans le cas d'étude précédent, l'accélération est fixée à la valeur maximale. Nous considérons trois ensembles dont les trajectoires de freinage sont définies par une décélération linéaire constante $u_\alpha = -u_{\alpha_{max}}$ en plus d'une vitesse de braquage constante $|u_\xi| \leq u_{\xi_{max}}$. Le premier ensemble de trajectoires de freinage est ε représenté dans la figure 5.13, il contient neufs trajectoires possibles avec les couples de contrôle (u_α, u_ξ), où $u_\alpha = -u_{\alpha_{max}}$ et u_ξ variant entre $-0.12\, rad/s$ et $0.12\, rad/s$ avec un pas

de $0.03\,rad/s$. Le deuxième ensemble est ε^- contenant cinq trajectoires (voir la figure 5.23.a), dont les couples de contrôle correspondant sont définis par $u_\alpha = -u_{\alpha_{max}}$ et u_ξ qui varie entre $-0.06\,rad/s$ et $0.06\,rad/s$ avec un pas de $0.03\,rad/s$. Enfin, ε^+ un ensemble de treize trajectoires de freinage (voir la figure 5.24.a), avec les couples de contrôle (u_α, u_ξ), où $u_\alpha = -u_{\alpha_{max}}$ et u_ξ variant entre $-0.18\,rad/s$ et $0.18\,rad/s$ avec un pas de $0.03\,rad/s$.

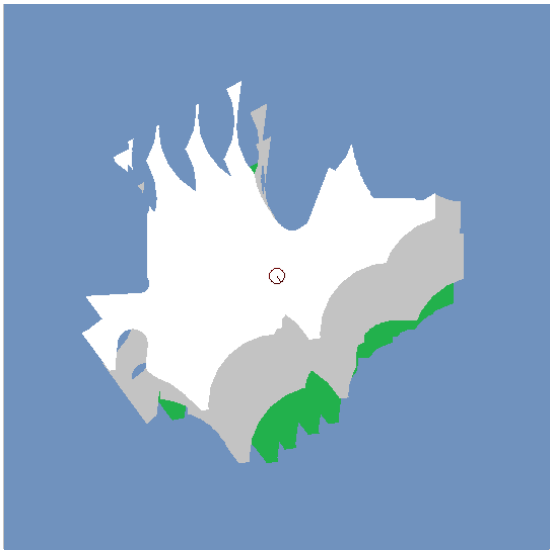


Figure 5.25 : Superposition des $ICS^b{}_{\hat{z}_c}(B)$ représentés dans les figures 5.23.c, 5.17, et 5.24.c (utilisant respectivement les ensembles de trajectoires de freinage ε^-, ε, ε^+) correspondant respectivement aux couleurs gris, vert et bleu.

Le principe est de recalculer $ICS^b{}_{\hat{z}_c}(B)$ pour chacun de ces trois ensembles. Les figures 5.16, 5.23.b et 5.24.b montrent respectivement le résultat du calcul de $ICS^b{}_{\hat{z}_c}(B, \varepsilon)$, $ICS^b{}_{\hat{z}_c}(B, \varepsilon^-)$ et $ICS^b{}_{\hat{z}_c}(B, \varepsilon^+)$. Il est clair à partir de ces résultats que la région libre (en blanc) dans la figure 5.23.c (avec l'ensemble ε^-) a rétréci par rapport à celle de la figure 5.17 (avec l'ensemble ε). Et au contraire, dans le cas de la figure 5.24.c (avec l'ensemble ε^+), la région libre s'est dilatée par rapport aux deux premiers cas (figure 5.17 et 5.23.c). Pour plus d'illustration, les trois versions N/B de $ICS^b{}_{\hat{z}_c}(B)$ correspondant aux trois ensembles ε, ε^- et ε^+ ont été superposées en

attribuant à chacune une couleur différente (uniquement afin de différencier les trois cas). D'après la figure 5.25, il peut être établi que :

$ICS^b{}_{\hat{z}_c}(B, \varepsilon^+) \subset ICS^b{}_{\hat{z}_c}(B, \varepsilon) \subset ICS^b{}_{\hat{z}_c}(B, \varepsilon^-)$, où $\varepsilon^- \subset \varepsilon \subset \varepsilon^+$

C'est pour cette raison que les trajectoires de freinage peuvent être choisies arbitrairement, vue la propriété d'approximation conservative de ICS^b.

Toutefois, pour mieux illustrer l'impact du nombre de trajectoires de freinage choisi sur l'ensemble ICS^b (ou sur la région libre), une analyse quantitative a été effectuée. Le pourcentage d'états p-sûrs (pixels de couleur blanche) est calculé pour différents nombres de trajectoires de freinage (voir le tableau 5.2).

Tableau 5.2: Le pourcentage d'états p-sûrs dans l'environnement en fonction du nombre de trajectoires de freinage.

Nombre de trajectoires	2	3	5	7	9	11	13	15
Pourcentage d'états p-sûrs (pixels blancs) (%)	7.7	9.8	14.3	19	22.4	24.1	24.4	24.5

Il est évident à partir de ces résultats, que la région libre (états p-sûrs) se dilate quand le nombre de trajectoires de freinage augmente (i.e. le pourcentage de pixels blancs augmente avec le nombre de trajectoires). Cependant, bien que le nombre d'états p-sûrs augmente avec le nombre de trajectoires, l'inconvénient est que le temps de calcul augmente également. C'est pourquoi, il faut choisir l'ensemble des trajectoires de freinage en respectant le coût de calcul (d'où le sens d'avoir utilisé la propriété d'approximation conservative (propriété 4)). Une analyse quantitative concernant ce point sera effectuée par la suite dans le paragraphe 5.7.

Rappelons que dans ce cas, les différents tests ont été réalisés pour une valeur maximale de l'accélération. La question qui se pose est que peut-il se produire si nous faisons varier au même temps le nombre de trajectoires de freinage et la valeur de l'accélération?

Pour répondre à cette question, nous considérons tout d'abord trois ensembles de trajectoires de freinage $\mathcal{E}_1^5$, $\mathcal{E}_2^9$ et $\mathcal{E}_3^{13}$ contenant respectivement cinq, neuf et treize trajectoires, avec $\mathcal{E}_1^5 \subset \mathcal{E}_2^9 \subset \mathcal{E}_3^{13}$ et où l'accélération est attribué de manière croissante. Les trois ensembles sont définis par les couples de contrôle (u_α, u_ξ). Nous considérons une vitesse de braquage u_ξ constante variant avec un pas de $0.03\, rad/s$, tel que $|u_\xi| \leq 0.06\, rad/s$ pour l'ensemble $\mathcal{E}_1^5$, $|u_\xi| \leq 0.12\, rad/s$ pour l'ensemble $\mathcal{E}_2^9$ et $|u_\xi| \leq 0.18\ rad/s$ pour l'ensemble $\mathcal{E}_3^{13}$. Concernant la décélération ; les trajectoires de l'ensemble $\mathcal{E}_1^5$ sont générées avec une décélération $u_\alpha = -4\, m/s^2$, pour l'ensemble $\mathcal{E}_2^9$, les trajectoires rajoutées ($\mathcal{E}_2^9 \backslash \mathcal{E}_1^5$) sont définies par une décélération $u_\alpha = -5\, m/s^2$ et enfin les trajectoires rajoutées dans le cas de l'ensemble $\mathcal{E}_3^{13}$ ($\mathcal{E}_3^{13} \backslash \mathcal{E}_2^9$) sont définies par une décélération $u_\alpha = -7\, m/s^2$. L'ensemble $ICS^b{}_{\hat{z}_c}(B)$ est alors calculé pour chacun de ces ensembles. Les résultats obtenus sont représentés dans le tableau 5.3, où le pourcentage d'états p-sûrs (région libre) est calculé pour chaque ensemble de trajectoires de freinage.

Tableau 5.3: Le pourcentage d'états p-sûrs dans l'environnement en fonction des ensembles de trajectoires de freinage $\mathcal{E}_1^5$, $\mathcal{E}_2^9$ et $\mathcal{E}_3^{13}$.

Ensembles de trajectoires de freinage	$\mathcal{E}_1^5$	$\mathcal{E}_2^9$	$\mathcal{E}_3^{13}$
Pourcentage d'états p-sûrs (pixels blancs) (%)	7.5	15.2	19.5

Par la suite, nous considérons trois autres ensembles de trajectoires de freinage $\mathcal{E}_4^5$, $\mathcal{E}_5^9$ et $\mathcal{E}_6^{13}$ contenant respectivement cinq, neuf et treize trajectoires (avec $\mathcal{E}_4^5 \subset \mathcal{E}_5^9 \subset \mathcal{E}_6^{13}$), mais dans ce cas, l'accélération est attribuée de manière décroissante. L'ensemble $ICS^b{}_{\hat{z}_c}(B)$ est alors calculé pour chacun de ces ensembles qui sont définis par les couples de contrôle (u_α, u_ξ), tel que : $|u_\xi| \leq 0.06\, rad/s$ pour l'ensemble $\mathcal{E}_4^5$, $|u_\xi| \leq 0.12\, rad/s$ pour l'ensemble $\mathcal{E}_5^9$ et $|u_\xi| \leq 0.18\ rad/s$ pour l'ensemble $\mathcal{E}_6^{13}$, avec un pas de $0.03\, rad/s$. $u_\alpha = -7\, m/s^2$ pour l'ensemble $\mathcal{E}_4^5$,

$u_\alpha = -5\, m/s^2$ pour les trajectoires rajoutées dans l'ensemble $\mathcal{E}_5^9$ ($\mathcal{E}_5^9 \backslash \mathcal{E}_4^5$) et $u_\alpha = -4\, m/s^2$ pour les trajectoires rajoutées dans l'ensemble $\mathcal{E}_6^{13}$ ($\mathcal{E}_6^{13} \backslash \mathcal{E}_5^9$). Le tableau 5.4 montre les résultats obtenus en représentant le pourcentage d'états p-sûrs (région libre) en fonction de l'ensemble des trajectoires de freinage.

Tableau 5.4: Le pourcentage d'états p-sûrs dans l'environnement en fonction des ensembles de trajectoires de freinage $\mathcal{E}_4^5$, $\mathcal{E}_5^9$ et $\mathcal{E}_6^{13}$.

Ensembles de trajectoires de freinage	$\mathcal{E}_4^5$	$\mathcal{E}_5^9$	$\mathcal{E}_6^{13}$
Pourcentage d'états p-sûrs (pixels blancs) (%)	14.3	20.3	20.8

Nous remarquons principalement à partir des deux tableaux 5.3 et 5.4 que la différence entre ces deux résultats est que, quand l'accélération des trajectoires ajoutées décroit (tableau 5.4), les écarts entre les différents pourcentages diminuent par rapport à ceux du tableau 5.3 (où l'accélération au contraire croit). C'est logique car même si le nombre de trajectoires a augmenté, la valeur de l'accélération a diminué. D'autre part, si nous comparons les résultats de ces deux tableaux à ceux du tableau 5.2 (pour les mêmes nombres de trajectoires de freinage), nous constatons que les pourcentages les plus élevés correspondent aux valeurs du tableau 5.2 qui correspondent à des trajectoires avec une accélération maximale.

Enfin, la conclusion logique à l'analyse effectuée dans les cas d'étude 3 et 4 est la nécessité de choisir des trajectoires de freinage "à fond", i.e. avec une décélération $u_\alpha = -u_{\alpha_{max}}$.

5.4.2.5 Cas d'étude 5 : (l'influence de la vitesse maximale des obstacles mobiles sur l'ensemble ICS^b)

Dans les cas d'études précédents nous avons montré comment le calcul des ICS^b peut être effectué pour un environnement dynamique inconnu en considérant aussi bien les obstacles perçus que les obstacles non perçus. Ce qui est possible

grâce au modèle conservatif de l'environnement qui permet de prendre en compte tous les mouvements futurs possibles d'un objet donné avec une dynamique arbitraire en considérant uniquement sa vitesse maximale. La vitesse maximale des objets mobiles est donc un paramètre important pour le modèle du futur. C'est pourquoi, dans ce cas d'étude, l'influence de la vitesse maximale des obstacles sur l'ensemble ICS^b (respectivement l'ensemble des états p-sûrs) est analysée. Pour ce faire, l'ensemble ICS^b est calculé pour différentes valeurs de la vitesse maximale des obstacles mobiles.

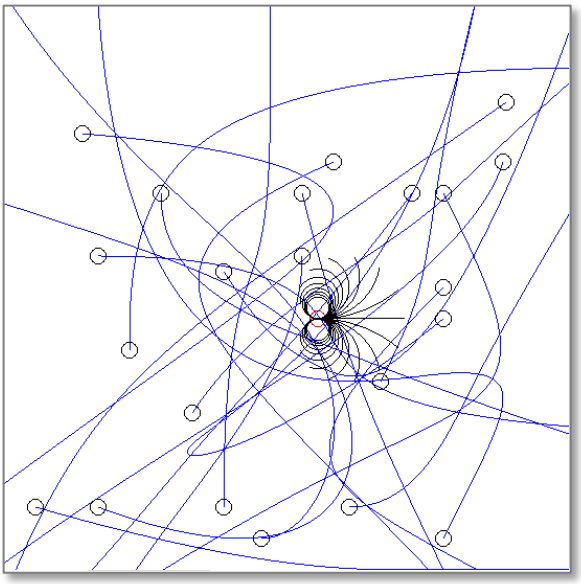


Figure 5.26 : Un environnement dynamique contenant 22 obstacles mobiles, avec des trajectoires arbitraires. Les trajectoires de freinage utilisées par ICS^b-CHECK (l'ensemble Ɛ) sont représentées en noir.

L'expérience est réalisée dans un environnement dynamique contenant 22 obstacles mobiles se déplaçant de manière arbitraire (cas de la figure 5.26) et pour un robot mobile avec les mêmes conditions expérimentales (contraintes kinodyamiques, champ de vision) que celles définies dans les paragraphes 5.2 et 5.3. L'ensemble des trajectoires de freinage utilisé par ICS^b-CHECK comprend 23 trajectoires de freinage définies par une décélération linéaire constante $u_\alpha = -u_{\alpha_{max}}$ et une vitesse de braquage constante $|u_\xi| \leq u_{\xi_{max}}$ qui varie entre $-0.33\,rad/s$ et $0.33\,rad/s$ avec un pas de $0.03\,rad/s$. Le but de cette expérience est de calculer

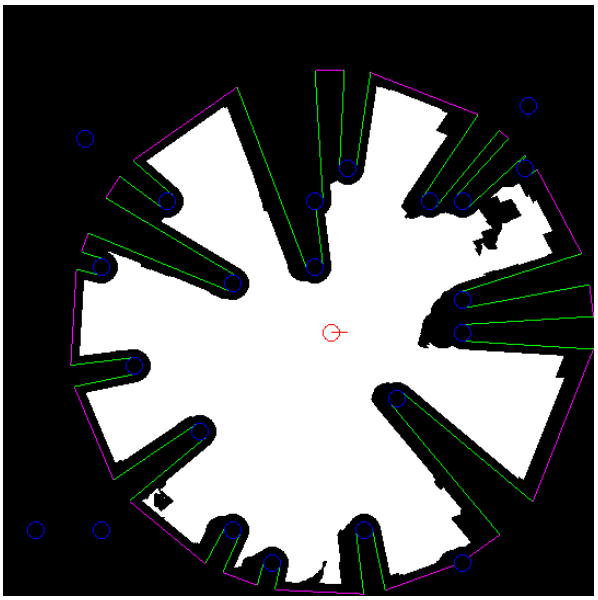
l'ensemble ICS^b pour différentes valeurs de la vitesse maximale des obstacles mobiles. Les résultats obtenus sont représentés dans le tableau 5.5, où le pourcentage d'états p-sûrs (pixels de couleur blanche) est donné en fonction des différentes valeurs de la vitesse maximale des mobiles $v_{B_{max}}$.

Tableau 5.5 : Le pourcentage d'états p-sûrs dans l'environnement en fonction de la vitesse maximale des obstacles mobiles.

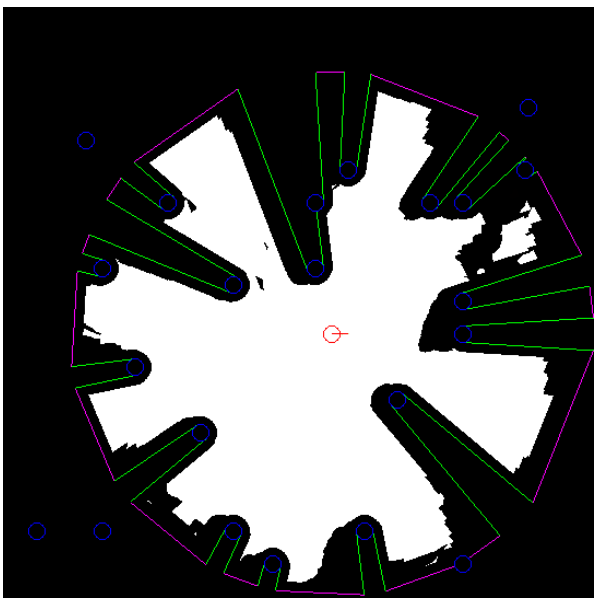
Vitesse maximale des obstacles $v_{B_{max}}\ (m/s)$	5	10	20	30	40	50
Pourcentage d'états p-sûrs (pixels blancs) (%)	35.7	34.2	31.8	21	16	11.8

Il est clair à partir du tableau 5.5 que plus la vitesse maximale des mobiles est grande plus la région libre rétrécit (i.e. nombre d'états p-sûrs diminue). Comme nous considérons chaque point des limites du champ de vision et des régions occultées comme un objet potentiel avec un comportement futur inconnu, le comportement futur de chaque point de ces ensembles est modélisé par un disque qui croit proportionnellement au temps avec un rapport de croissance correspondant à la vitesse maximale de l'objet (un cône dans l'espace $\times$ temps). C'est pourquoi, quand la vitesse maximale des mobiles varie, le rapport de croissance des disques varie. Ainsi, quand la vitesse croit, l'espace occupé par les disques croit également (l'espace libre diminue). Nous pouvons conclure que l'ensemble des états p-sûrs diffère selon la nature des objets présents dans l'environnement. Par exemple, si les obstacles mobiles sont des piétons avec une vitesse maximale de 5m/s (un piéton qui court), la région d'états p-sûrs (région en blanc) est relativement grande (voir la figure 5.27.a). Alors que, si c'est des véhicules avec une vitesse maximale de 20m/s, la région p-sûre rétrécit un peu (voir la figure 5.27.b), et avec une vitesse maximale

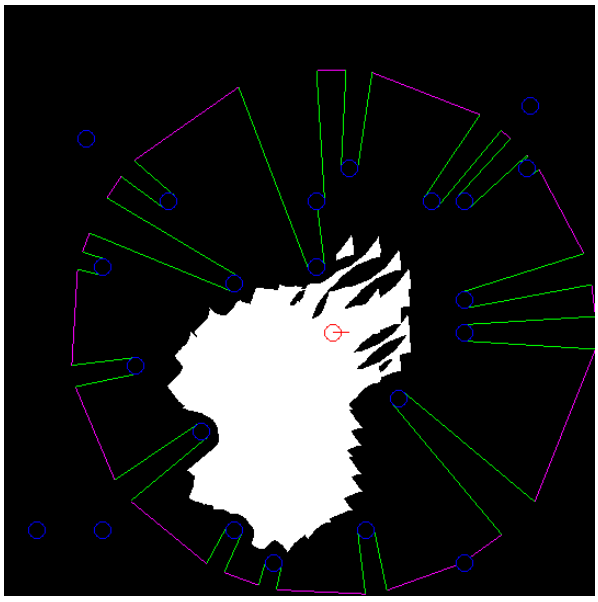
encore plus grande par exemple 50m/s la région p-sûre rétrécit encore plus (voir la figure 5.27.c).



(a) Version N/B de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$ pour $v_{B_{max}} = 5m/s$.



(b) Version N/B de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$ pour $v_{B_{max}} = 20m/s$.



(c) Version N/B de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$ pour $v_{B_{max}} = 50m/s$.

Figure 5.27 : Calcul de l'ensemble ICS^b pour différentes valeurs de la vitesse maximale des obstacles mobiles.

Plus de validations et preuves sur la sûreté passive du mouvement du robot sont présentées dans le paragraphe suivant.

5.5 PASSAVOID en action

PASSAVOID est une approche d'évitement d'obstacles conçue pour conduire le robot d'un état p-sûr vers un autre. Afin d'illustrer le fonctionnement de PASSAVOID, trois scénarii ont été sélectionnés. Le premier est appelé *scénario du Compacteur 1D*, il est simple mais il aide à comprendre le type de comportement que PASSAVOID va avoir quand A est confronté à une situation dangereuse clairement identifiée. Le second scénario est appelé *scénario d'un environnement urbain simple*, avec un environnement qui présente un plus grand nombre d'obstacles. Enfin, le troisième scénario appelé *scenario de la foule aveugle (Blind Crowd scenario)*, dont le premier objectif est d'illustrer les performances de PASSAVOID dans des situations complexes. Ces trois scénarios sont expliqués dans le paragraphe suivant.

Dans les trois cas, PASSAVOID n'a aucune information quant au comportement futur de l'environnement. PASSAVOID ne tente pas de conduire A vers un but donné mais de garder A dans des états p-sûrs. Son second objectif est de maintenir A en mouvement. En d'autres termes, la sélection d'un contrôle admissible (ligne #10 de l'algorithme 3) est biaisée en faveur des contrôles donnant une vitesse linéaire non nulle. Ce choix a été fait de manière à éviter la réponse au problème de la sûreté passive du mouvement, qui est simplement de freiner et de s'arrêter pour toujours (de cette façon, A atteint et reste toujours dans un état p-sûr).

Notons que pour les trois scénarii, les contraintes du robot (contraintes kinodynamiques (vitesse maximale, accélération maximale, etc.), champ de vision, etc.) et de l'environnement (vitesse maximale des objets mobiles) restent les mêmes que celles définies dans les paragraphes 5.2 et 5.3.

5.5.1 Scénario du Compacteur 1D

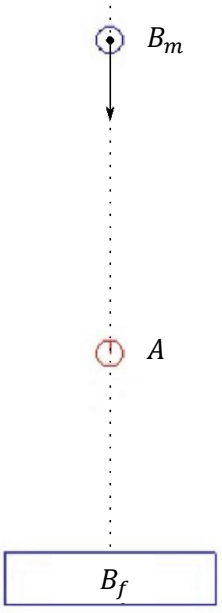


Figure 5.28 : Scénario du compacteur 1D.

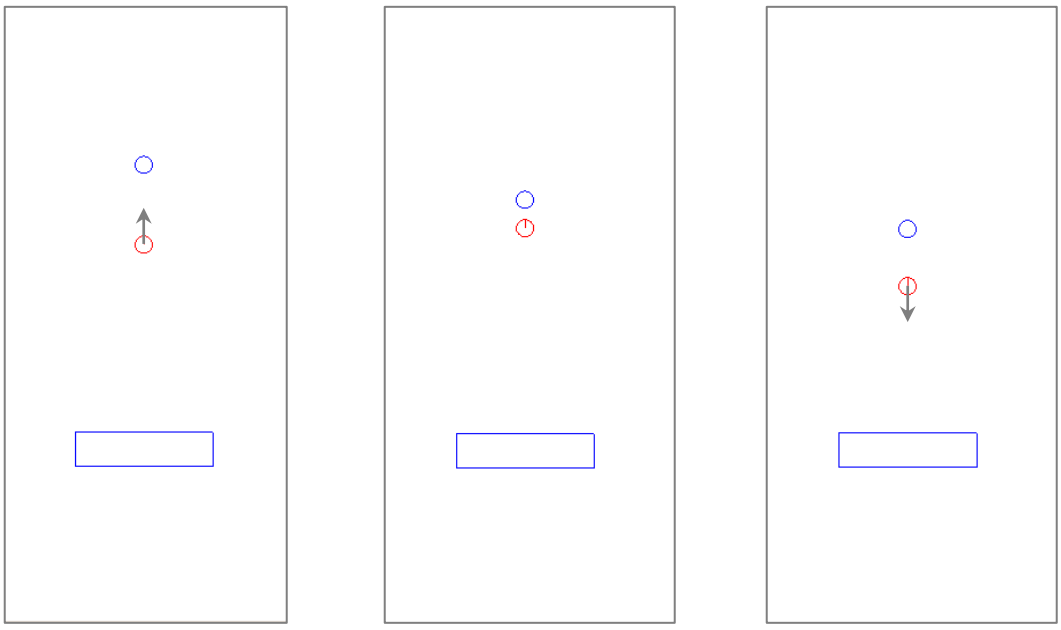
Le scénario du compacteur 1D contient un objet fixe B_f et un objet mobile B_m. L'objet mobile se déplace en direction de l'objet fixe (voir figure 5.28). Les deux objets B_f et B_m sont comme les deux mâchoires d'un compacteur (d'où le nom du scénario). A est placé entre B_f et B_m, à la position $s_0 = (0{,}7{,}1.57{,}20{,}0)$. Il est supposé que A peut uniquement se déplacer le long de la ligne verticale liant B_f et B_m. Au départ, A se déplace vers le haut avec une vitesse linéaire positive. Dans une telle situation, il est clair que l'état initial s_0 est un ICS (peu importe ce que A fait, il finira par être écrasé par B_m). Cependant, il est possible de sélectionner la position initiale de A et sa vitesse linéaire de manière à ce que s_0 soit p-sûr.

Dans ce scénario, lorsque PASSAVOID est appliqué, A présente le comportement suivant pour rester toujours dans des états p-sûrs (la séquence correspondant à ce comportement est représentée dans la figure 5.29):

1. L'approche croissante de B_m oblige A à graduellement décroitre sa vitesse (figure 5.29.a) jusqu'à ce qu'il s'arrête (voir figure 5.29.b).

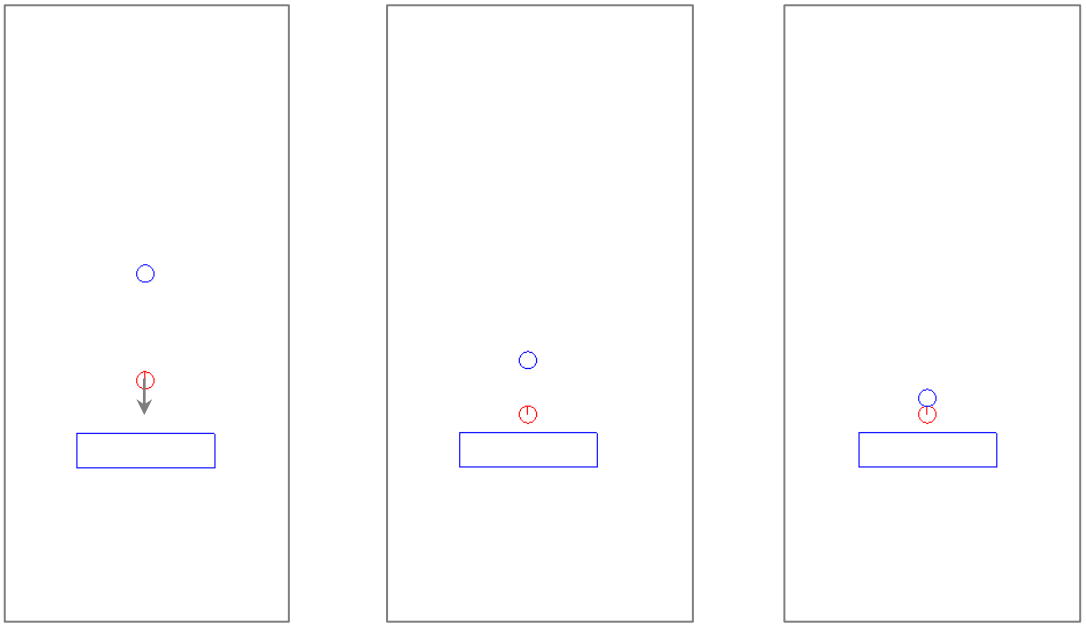
2. A fait marche arrière afin d'éviter la collision avec B_m (rappelons que PASSAVOID est biaisé en faveur de garder A en mouvement) (voir figure 5.29.c).
3. Pendant sa marche en arrière, A se rapproche de B_f. A un certain point, B_f oblige A à réduire sa vitesse (voir figure 5.29.d).
4. A est maintenant au repos à côté de B_f, il sera bientôt heurté par B_m (voir figure 5.29.e).
5. A est en collision avec B_m ($t = 7s$) (voir figure 5.29.f).
6. Lorsque la collision avec B_m est terminée[4], A reprend son mouvement vers l'avant (voir figure 5.29.g).

L'évolution de la vitesse linéaire de A est représentée dans la figure 5.30. Aussi simple que cela puisse paraître, ce scénario montre comment PASSAVOID cherche à éviter une collision avec B_m de manière naturelle (en freinant et en se déplaçant en sens inverse). Cependant, lorsque A est piégé, PASSAVOID garantit que le robot sera au repos quand la collision se produit.

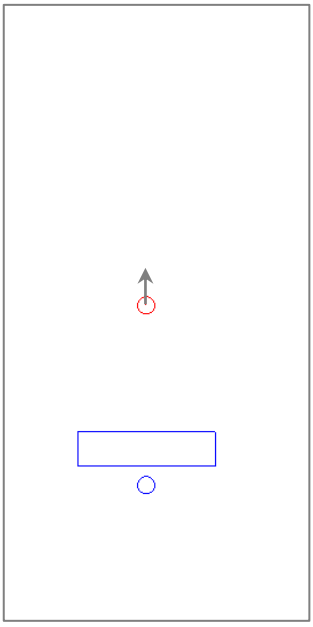


(a) (b) (c)

[4] En supposant que B_m peut passer à travers A.



(d) (e) (f)



(g)

Figure 5.29 : PASSAVOID en action pour le scénario du compacteur 1D. (a)-(g) : Séquence du mouvement de A.

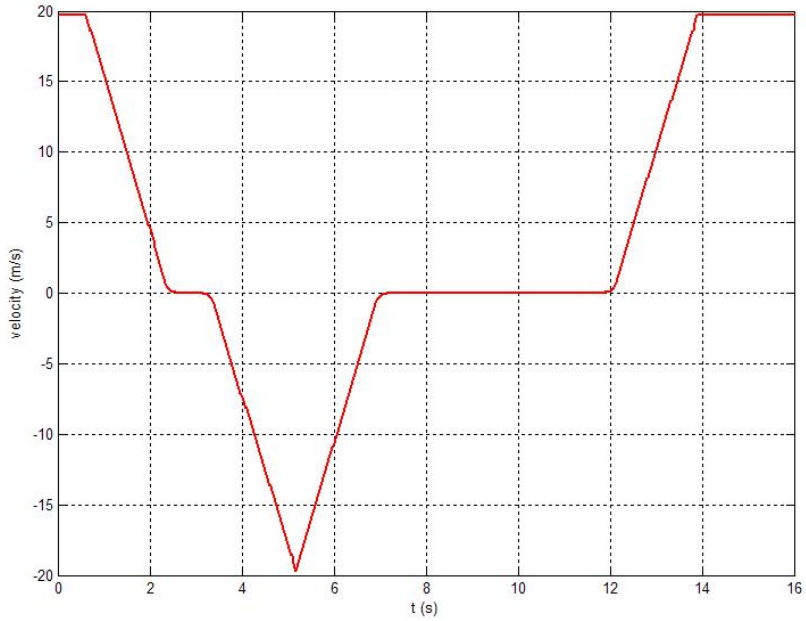


Figure 5.30 : Profile de vitesse de A pour le scénario du compacteur 1D.

Pour la majorité des approches d'évitement d'obstacle même celles traitant le problème de la sûreté du mouvement (voir chapitre 1), ce scénario reste un défi : où la sûreté du mouvement ne peut être vérifiée (la collision est inévitable). Pour notre cas, le test ci-dessus montre que PASSAVOID est capable de toujours garder le robot dans un état p-sûr. La sûreté passive du mouvement est donc bien vérifiée.

5.5.2 Scénario d'un environnement urbain simple

Dans ce cas, nous reprenons le scénario de la figure 5.11. Tout environnement urbain peut contenir des objets fixes (des constructions par exemple) ou mobiles (les voitures, les piétons, etc.), ce scénario est un exemple simple qui contient trois obstacles fixes et deux obstacles mobiles. A chaque pas de temps, une nouvelle mise à jour de l'environnement est disponible (i.e. information sur les obstacles et la limite du champ de vision), d'où la mise à jour du modèle du futur. Un nouvel ensemble ICS^b est alors calculé. PASSAVOID détermine l'ensemble de contrôles admissibles (le noyau K) en vérifiant si le contrôle en question conduit le robot vers un état p-sûr (grâce à l'algorithme ICS-CHECK). Par la suite, un contrôle est choisi

pour être appliqué. Ce contrôle conduit le robot vers un état p-sûr. Notons que, l'ensemble de trajectoires de freinage utilisé par ICS-CHECK comprend neuf trajectoires définies par une décélération linéaire constante $u_\alpha = -u_{\alpha_{max}}$ et une vitesse de braquage constante $|u_\xi| \leq u_{\xi_{max}}$.

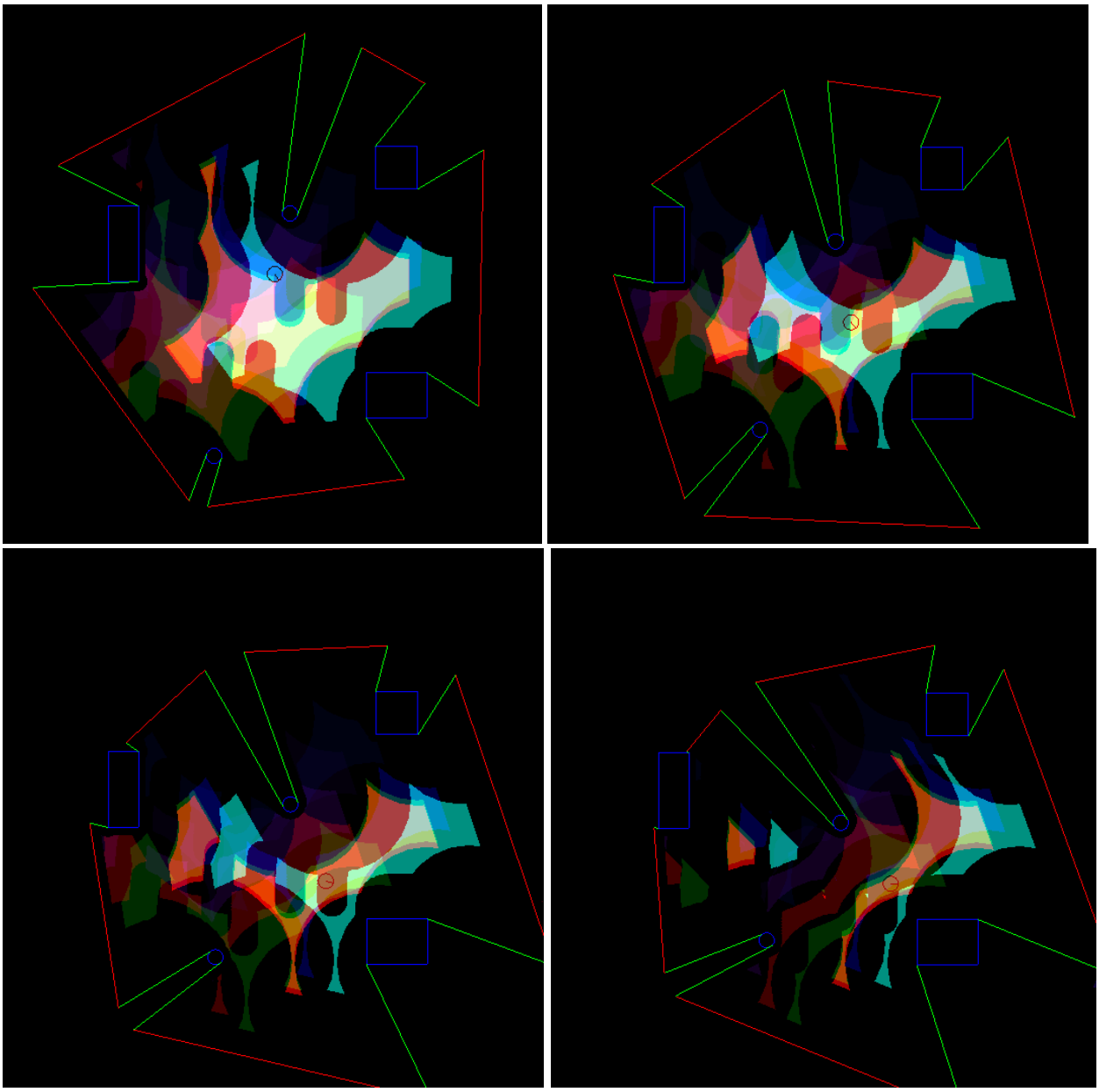


Figure 5.31 : PASSAVOID en action dans le scénario d'un environnement urbain simple: séquence des mouvements de A.

La figure 5.31 illustre le comportement de PASSAVOID dans ce scénario. Il est clair que A évite non seulement les obstacles de l'environnement mais également les régions interdites (les états de collision inévitable de freinage (ICS^b)) représentées en noir.

Cette expérience démontre la capacité de PASSAVOID à faire respecter la sûreté passive du mouvement. D'autres évaluations sont apportées dans le paragraphe suivant, avec un scénario impliquant un plus grand nombre d'objets se déplaçant à des vitesses aléatoires.

5.5.3 Scenario de la Foule Aveugle

Le scénario de la foule aveugle est plus contraignant, il dispose de 22 objets mobiles se déplaçant arbitrairement dans un espace de travail 2D, dont la position initiale du robot est $(0,0,0)$ (voir figure 5.32). Les objets sont aveugles dans le sens où leur mouvement n'est pas affecté par les autres objets.

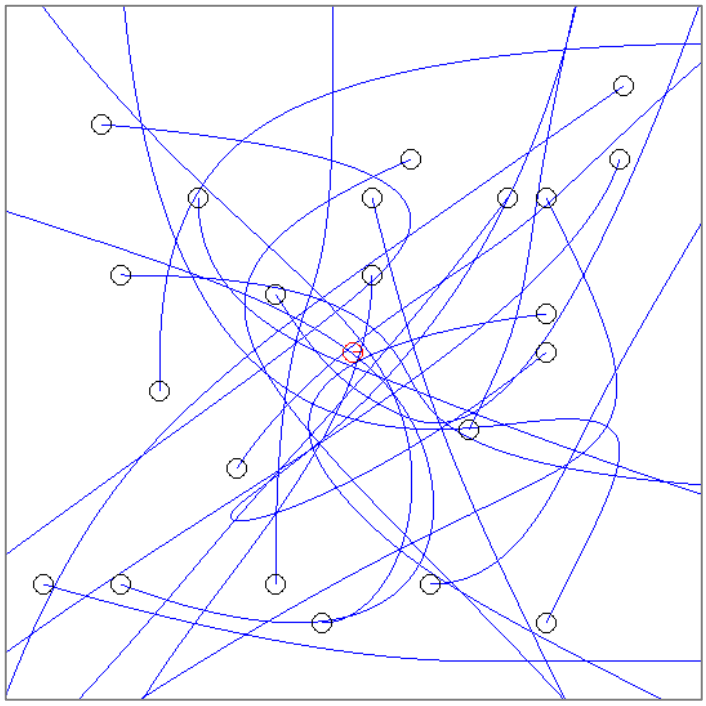


Figure 5.32 : Scénario de la foule aveugle.

L'ensemble d'échantillonnage de l'espace de contrôle U est obtenu grâce à une discrétisation régulière de l'ensemble de contrôle 2D $[-u_{\alpha_{max}}, u_{\alpha_{max}}] \times [-u_{\xi_{max}}, u_{\xi_{max}}]$, et l'ensemble de trajectoires de freinage ε utilisé par ICS^b-CHECK comprenant 23 trajectoires de freinage (voir figure 33) définies par une décélération linéaire minimale constante $u_\alpha = -u_{\alpha_{max}}$ et une vitesse de braquage $|u_\xi| \leq u_{\xi_{max}}$.

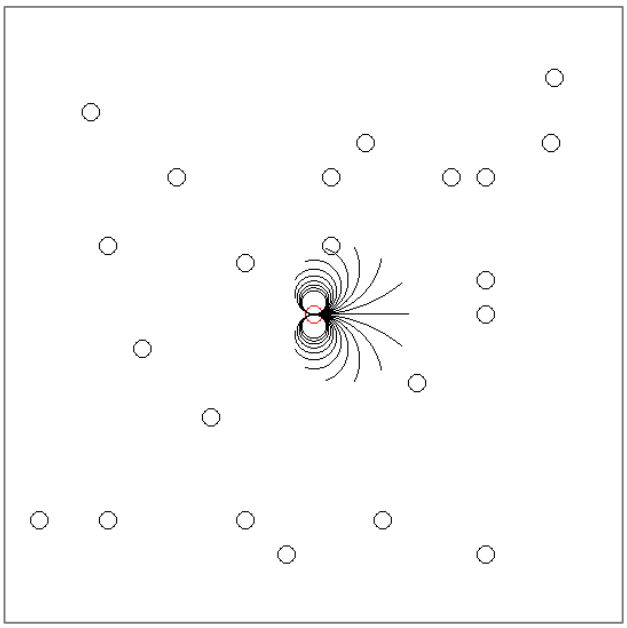
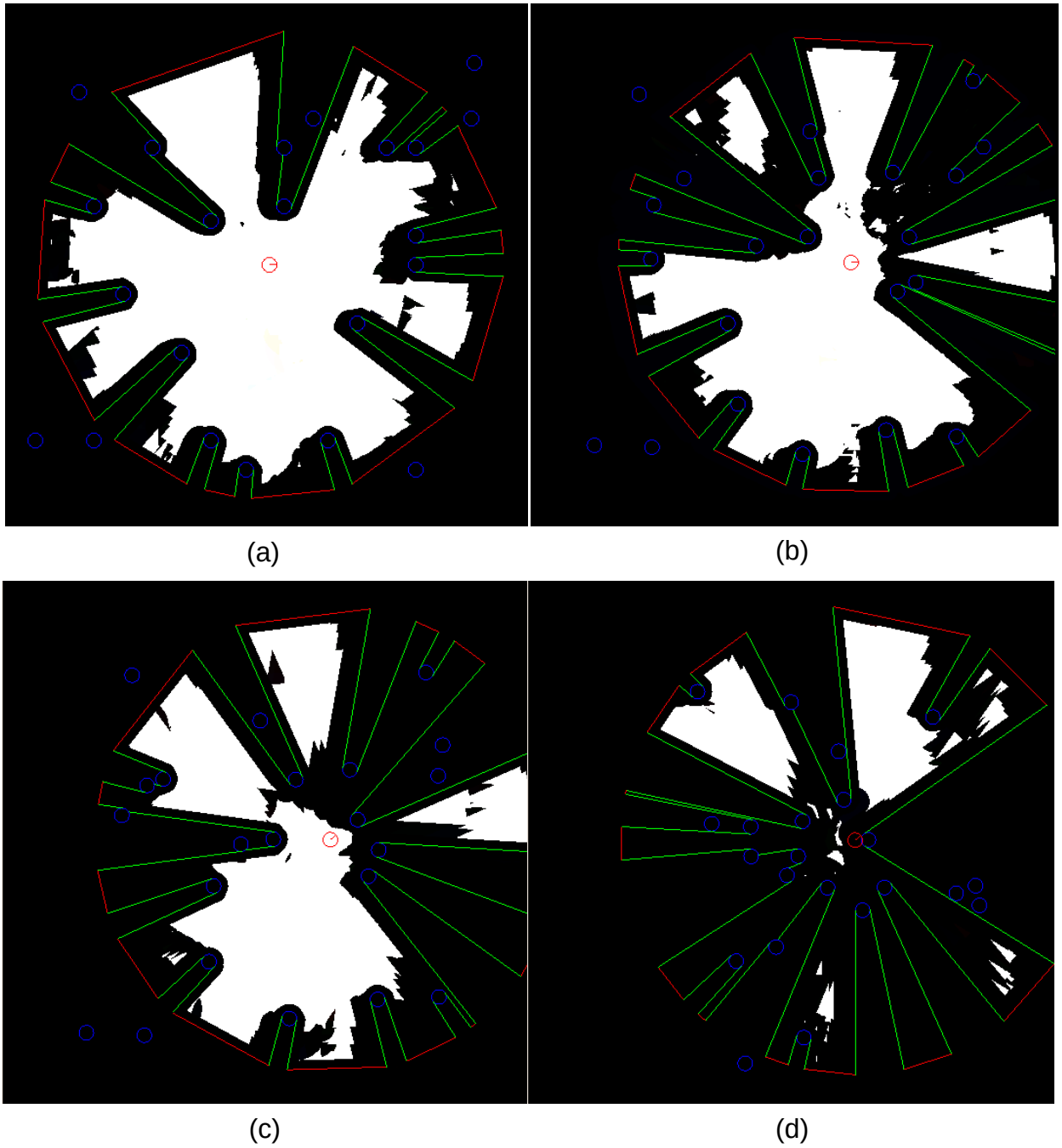


Figure 5.33 : L'ensemble des trajectoires de freinage ε considéré par ICS^b-CHECK dans le scénario de la foule aveugle.

La figure 5.34 illustre quatre situations prises à différents pas de temps de l'exécution de PASSAVOID dans ce scénario. Chaque situation présente le robot A, les objets mobiles et le champ de vision correspondant. L'ensemble des ICS^b est également représenté dans la figure (la région noire). Dans la situation de la figure 5.34.a, le robot se déplace librement dans la région p-sûre, mais en se rapprochant de la région ICS^b (voir la figure 5.34.b), A va tenter d'éviter cette région. Cependant, dans la situation de la figure 5.34.c, A est au repos car une collision est imminente. Par exemple, dans le cas de la figure 5.34.d, une collision a lieu mais le robot est au repos quand cela se produit.



La figure 5.34 : Quatre situations ((a)-(d)) de PASSAVOID en action dans le scénario de la foule aveugle, illustrant les ICS^b (région noire).

Après plusieurs exécutions, ces expériences ont montré la capacité de PASSAVOID à permettre au robot de se déplacer dans un environnement encombré d'obstacles mobiles de manière passivement sûre: chaque fois qu'une collision eu lieu, A était au repos, i.e. il est garanti que le robot est toujours dans un état p-sûr.

5.6 PASSPMP en action

PASSPMP est un planificateur de mouvement basé sur le principe de la planification de mouvement partiel et sur la garantie de la sûreté de mouvement passive. A chaque cycle PASSPMP, une trajectoire partielle est calculée. L'ensemble des trajectoires partielles calculées conduit le robot vers son but. Le mouvement partiel peut être décrit comme une concaténation de plusieurs primitives géométriques où chacune est vérifiée pour être p-sûre ou non. La trajectoire partielle sélectionnée est choisie par rapport au critère de la sûreté du mouvement et à la convergence vers le but à atteindre, où la distance et le temps sont tous les deux les facteurs à prendre en compte.

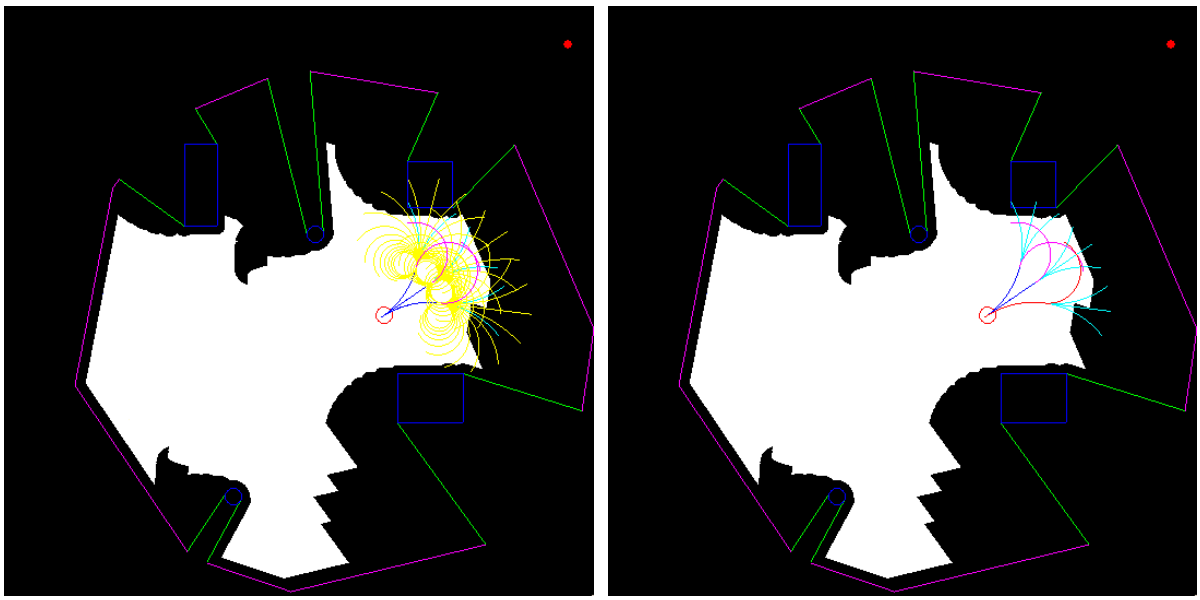
Afin de démontrer et valider la sûreté passive du mouvement de PASSPMP, il a été implémenté et testé en simulation dans deux scénarii différents ; le *scénario d'un environnement urbain simple* (voir la figure 5.11) qui contient des obstacles fixes et mobiles et le *scenario de la foule aveugle* (voir figure 5.32), qui est beaucoup plus complexe avec un grand nombre d'obstacles mobiles qui se déplace selon des trajectoires arbitraires. Dans les deux cas, les conditions expérimentales concernant le robot (contraintes kinodyamiques, champ de vision) et les objets mobiles (valeur de la vitesse maximale, comportement futur inconnu) restent les même que celles définies dans les paragraphes 5.2 et 5.3.

5.6.1 Scénario d'un environnement urbain simple

Ce scénario contient trois obstacles fixes et deux obstacles mobiles (voir la figure 5.11). PASSPMP doit conduire le robot mobile à partir de son état initial jusqu'à l'état but en considérant les obstacles présents dans l'environnement et le champ de vision ∂FoV (obstacles non perçus).

La figure 5.35.a montre un exemple de construction de l'arbre passivement sûr (p-safe RRT) pour un cycle donné de PASSPMP. L'arbre est étendu selon l'ensemble de contrôles définis par une accélération linéaire maximale constante $u_\alpha = u_{\alpha_{max}}$ et une vitesse de braquage constante $|u_\xi| \leq u_{\xi_{max}}$ (à savoir les couples de contrôle : $(u_\alpha, -u_\xi), (u_\alpha, 0), (u_\alpha, u_\xi)$). Il est vérifié par rapport à la sûreté passive du mouvement

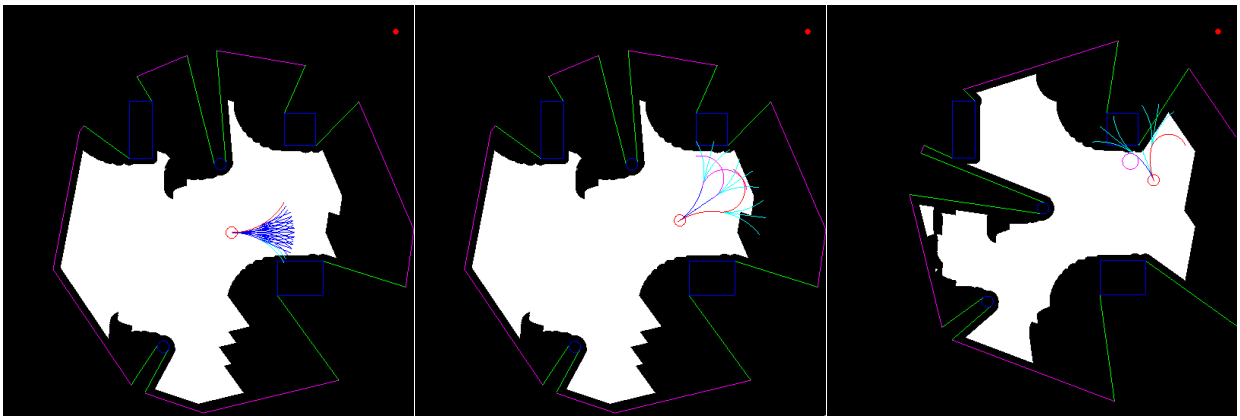
en utilisant l'algorithme ICS^{b}-CHECK (ligne #8 de l'algorithme 6) ; où l'ensemble des trajectoires de freinage utilisé par ICS-CHECK comprend cinq trajectoires de freinage. Comme illustrer dans la figure 5.35.a, les primitives de trajectoire représentées en bleu sont p-sûres alors que celles représentées en cyan ne le sont pas (ICS^{b}). L'arbre est étendu de manière à ce que chaque nœud est étendu avec au moins une primitive de trajectoire p-sûre. Dans le cas où aucune primitive de trajectoire p-sûre n'est trouvée, le nœud est alors étendu avec une trajectoire de freinage sans collision (lignes #11 et 12 de l'algorithme 6) pour garantir que le robot s'arrête avant qu'une collision ait lieu (la garantie de la sûreté passive du mouvement). Par conséquent, parmi un ensemble de trajectoires de freinage possible, 21 dans cet exemple (représentées en jaune dans la figure 5.35.a), la trajectoire de freinage sélectionnée est celle qui est sans collision et la plus proche au but (elle est représentée en magenta).



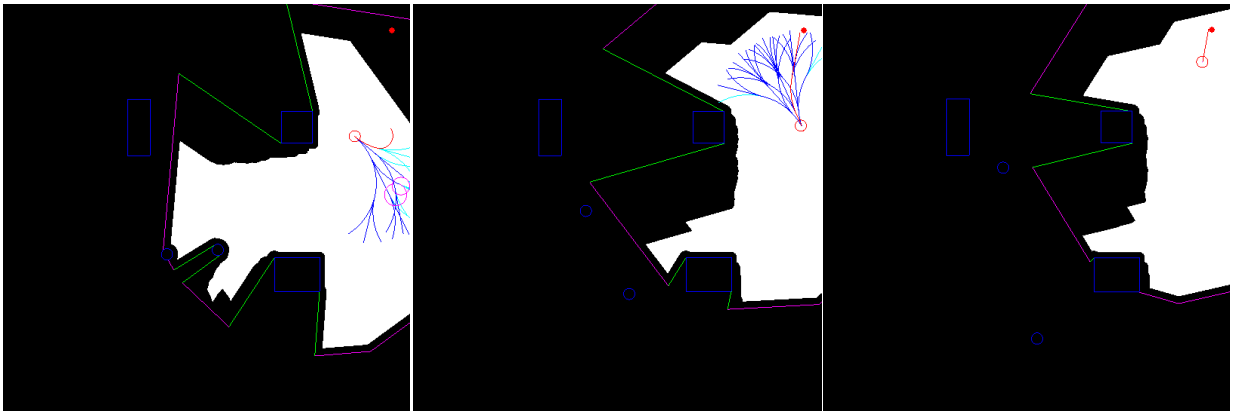
(a) (b)

Figure 5.35 : p-safe RRT en action: (a) la construction de p-safe RRT durant un cycle de PASSPMP (b) la trajectoire partielle sélectionnée correspondante (en rouge). Le robot A est le disque rouge au centre. Les obstacles sont représentés en bleu, les disques sont les obstacles mobiles et les polygones sont les obstacles fixes. La marque rouge est le but à atteindre par A (voir le texte).

Durant un cycle de planification, la meilleure trajectoire partielle est sélectionnée étant donné deux conditions : (1) la sûreté passive du mouvement de la trajectoire partielle et (2) l'optimisation du temps et de la distance par rapport au but (ligne #2 de l'algorithme 5) en utilisant l'équation (4.1). Les deux facteurs de pondération de la distance et du temps sont fixés expérimentalement, où $w_d = 0.2$ et $w_t = 0.8$ respectivement (une étude concernant ce point est effectuée par la suite). Dans le cas de l'arbre étendu de la figure 5.35.a, la trajectoire partielle sélectionnée est représentée en rouge (voir la figure 5.35.b). Elle correspond à la concaténation d'une primitive p-sûre et une trajectoire de freinage sans collision.



(a) Plan Π_0, t_0 (b) Plan Π_1, cycle 0 (c) Plan Π_2, cycle 1



(d) Plan Π_3, cycle 2 (e) Plan Π_4, cycle 3 (f) Plan Π_5, cycle 4

Figure 5.36 : p-safe RRTs correspondant à différents cycles de PASSPMP. La trajectoire partielle p-sûre sélectionnée à chaque cycle est représentée en rouge.

Pour atteindre le but d'une manière passivement sûre, PASSPMP conduit le robot suivant un ensemble de trajectoires partielles concaténées (calculées à différents cycles). Selon le processus PASSPMP illustré dans le paragraphe 4.2.2.1, à chaque cycle k, une trajectoire partielle p-sûre Π_{k+1} est calculée. La figure 5.36 montre les arbres étendus (*p-safe RRTs*) à travers les différents cycles PASSPMP, ainsi que les trajectoires sélectionnées jusqu'à ce que A atteigne son but, où la durée du cycle PASSPMP est $\delta_c = 2.4s$. Pendant le cycle correspondant à la figure 5.36.f, une trajectoire de freinage sans collision est calculée pour permettre au robot de s'arrêter à l'état but s_g.

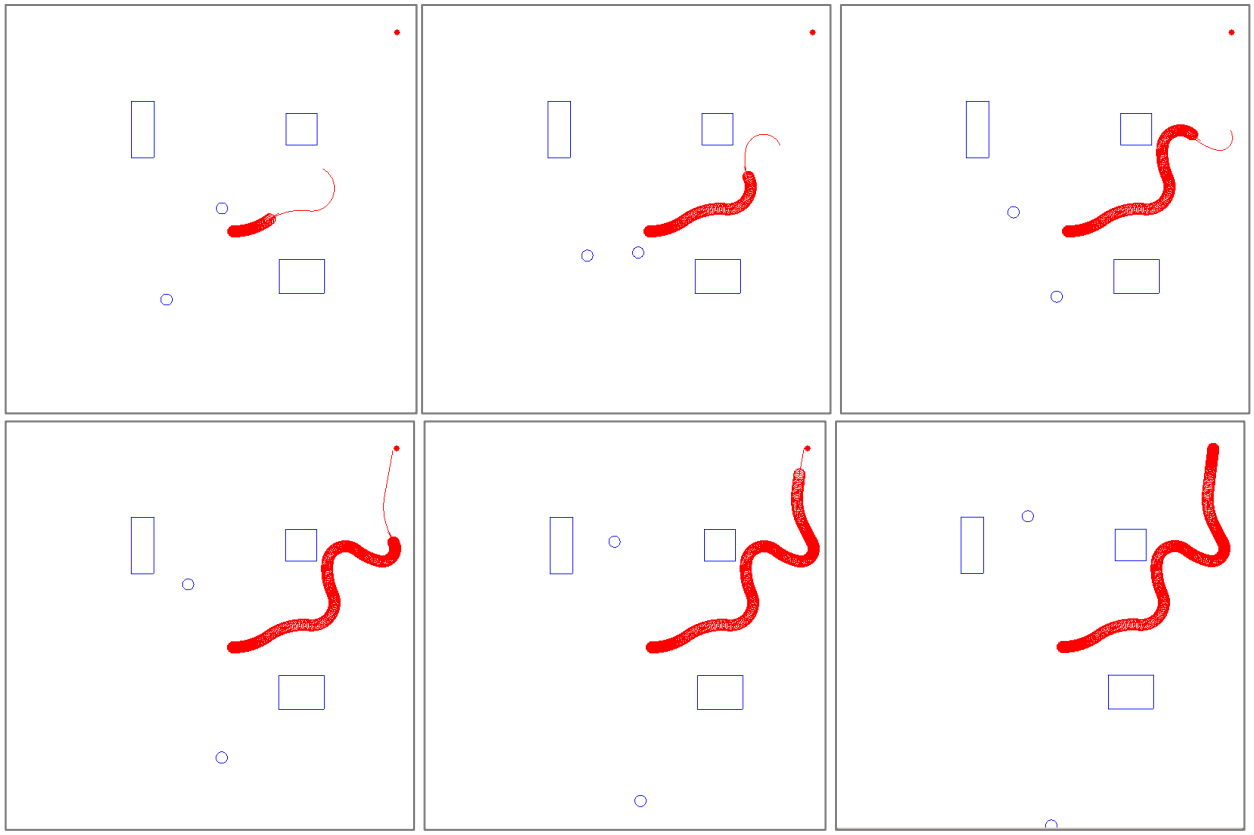


Figure 5.37 : La trajectoire exécutée par le robot dans le scénario d'un environnement urbain simple à travers les différents cycles PASSPMP. La trace rouge laissée derrière le robot représente la trajectoire exécutée et la trajectoire rouge devant lui est la trajectoire planifiée.

Notons qu'à l'instant t_0, la trajectoire Π_0 doit être disponible (une entrée de l'algorithme 5). C'est pour cette raison que ce plan est calculé à l'état initial (s_0), étant à l'arrêt (cas de la figures 5.36.a).

La trajectoire partielle planifiée peut être exécutée en entier ou en partie durant le cycle PASSPMP suivant. La séquence de trajectoires exécutées par le robot à partir de s_0 jusqu'à s_g est illustrée dans la figure 5.37. Le robot montre un comportement passivement sûr ; il évite les obstacles perçus et non perçus (risque imminent à partir des limites du champ de vision et des régions occultées) de l'environnement, et freine quand l'évitement n'est pas possible.

L'évolution de la vitesse de A est illustrée dans la figure 5.38, où A présente le comportement suivant :

1) Durant le premier cycle, A augmente sa vitesse jusqu'à ce qu'il atteigne la valeur $v = 15m/s$. PASSPMP maintient A dans des états p-sûrs durant son déplacement (aucune collision ne se produit).
2) A continut d'augmenter sa vitesse jusqu'à ce qu'il atteigne la valeur maximale. Vers la fin du deuxième cycle, même si A peut se déplacer sans risque de collision, il réduit graduellement sa vitesse (jusqu'à $v = 10m/s$). Cela est dû à la présence d'un obstacle statique (perçu) et au champ de vision limité de A : comme les objets inattendus sont considérés, la possibilité de présence d'un objet oblige A à freiner jusqu'à ce qu'il s'arrête pour rester dans un état p-sûr (si la collision se produit, A sera au repos).
3) Au cycle suivant, un nouveau plan est considéré utilisant une nouvelle mise à jour de l'environnement. A arrête de réduire sa vitesse et l'augmente de nouveau jusqu'à la valeur $v = 17m/s$. Une fois encore, il freine (garantie de la sûreté passive du mouvement) jusqu'à $v = 7.2m/s$ (toujours à cause de la considération du champ de vision limité). Ainsi, ce processus se répète à travers les différents cycles.
4) Durant le dernier cycle, A atteint le but. Il sélectionne une trajectoire de freinage sans collision qui le conduit à l'état le plus proche du but avec une vitesse nulle (A au repos) (voir figure 5.36).

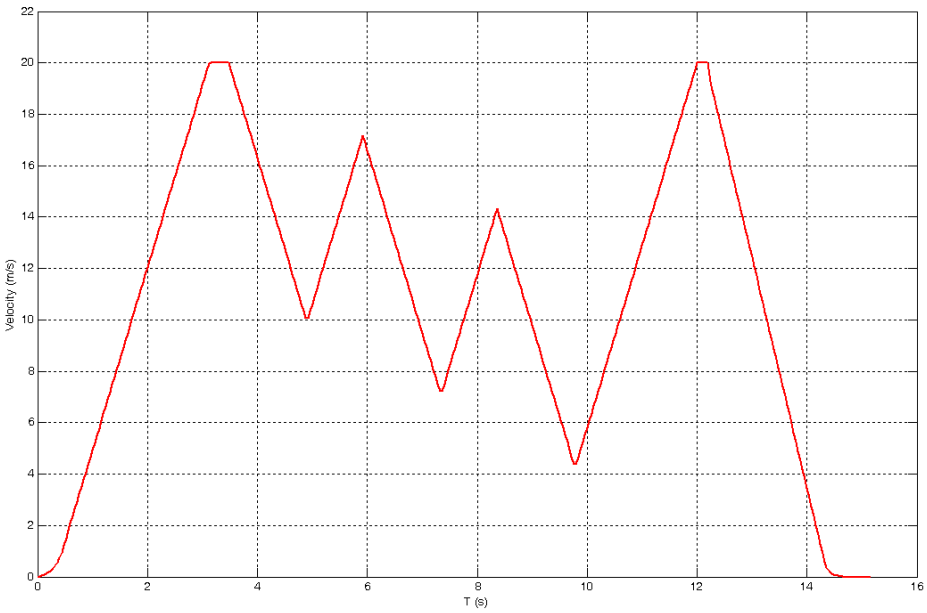


Figure 5.38 : Profile de vitesse de A.

5.6.2 Scenario de la Foule Aveugle

Le scénario de simulation dispose de 22 objets mobiles se déplaçant de manière arbitraire dans un espace de travail 2D (voir figure 5.32). Leur mouvement n'est pas affecté par les autres objets.

Les tests effectués dans un tel scénario visent à évaluer les performances de PASSPMP dans un environnement complexe encombré d'obstacles mobiles. La figure 5.39 montre une séquence de trajectoires générées durant les différents cycles de PASSPMP (la trajectoire en rouge devant le robot), pendant qu'il exécute la trajectoire planifiée durant le cycle précédent (la trace rouge laissée derrière le robot). La durée du cycle PASSPMP est $\delta_c = 1.4s$.

Cette expérience a démontré la capacité de PASSPMP à conduire le robot à un but prédéfini tout en garantissant la sûreté passive du mouvement, en prenant en compte la dynamique du système, celle de l'environnement et le champ de vision limité du robot.

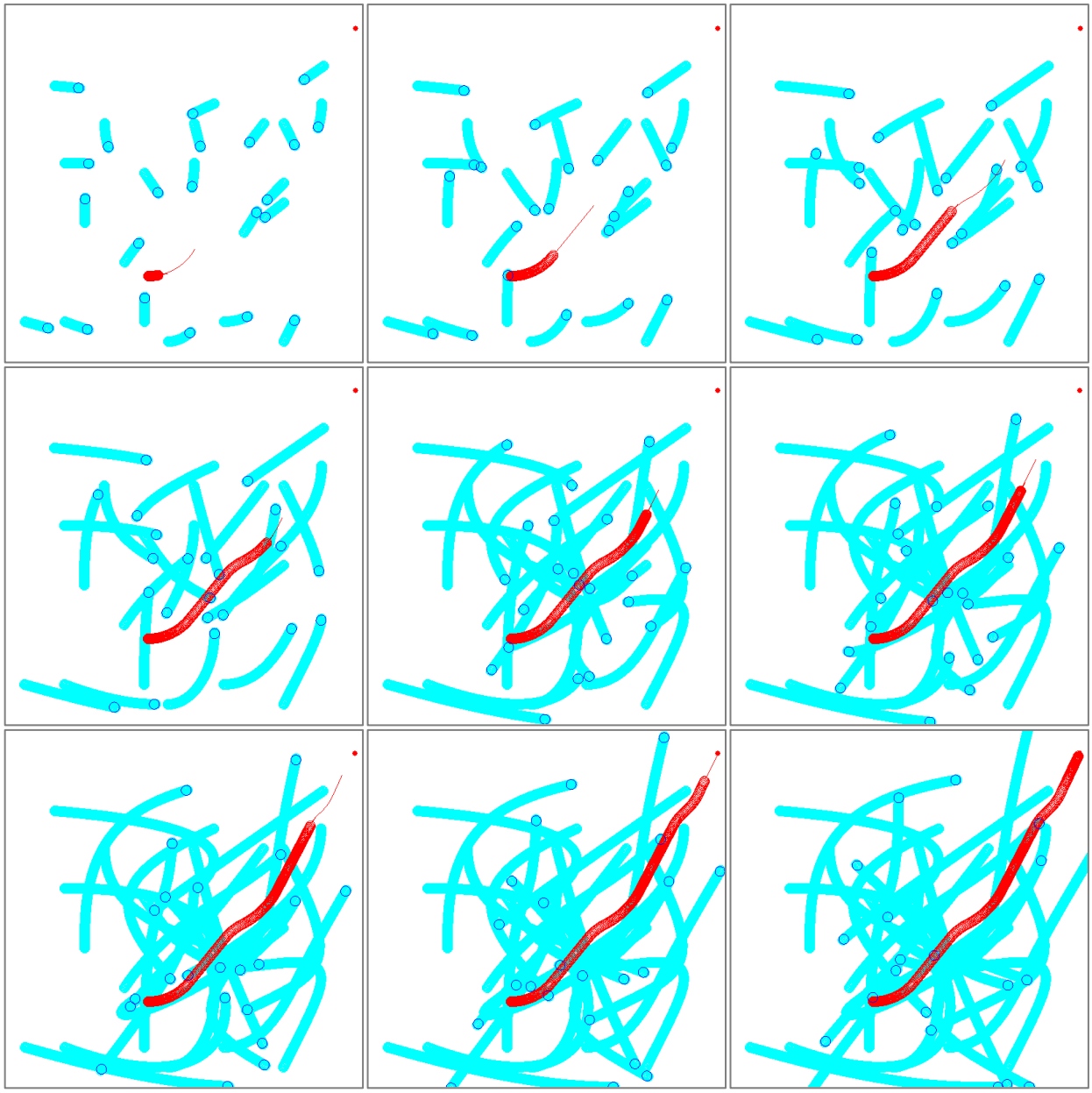


Figure 5.39 : PASSPMP en action dans le scénario de la foule aveugle : séquence de mouvements de A prise à différents instants du processus. La trace rouge laissé derrière le robot est la trajectoire exécutée. La trajectoire rouge devant le robot est la trajectoire planifiée (qui va être exécutée). Les traces en cyan derrière les obstacles représentent leur déplacement.

5.6.3 L'influence du choix des facteurs de pondération w_d et w_t

Pour montrer l'influence du choix des deux facteurs de pondération de la distance et du temps (respectivement w_d et w_t) sur la convergence de PASSPMP vers le but, parmi une centaine de tests effectués, quelques résultats illustratifs sont représentés dans les tableaux tableau 5.6 et tableau 5.7. D'après l'algorithme d'expansion de l'arbre p-safe RRT (l'algorithme 6), plusieurs trajectoires partielles p-sûres peuvent être générées. Ces trajectoires peuvent être de durées différentes (temps cumulé (ΔT_{Π})) et de distances euclidiennes différentes (distance qui sépare l'état final de la trajectoire partielle et l'état but, et que nous notons Δd_g). Certaines trajectoires peuvent être optimales par rapport au temps, i.e. le temps cumulé ΔT_{Π} correspondant est le minimum par rapport à celui des autres trajectoires, mais en contrepartie, la distance Δd_g est très grande. C'est le cas où la fonction f_g est minimisée uniquement par rapport au temps (i.e. $w_d = 0$, donc $f_g = \Delta T_{\Pi}$) (cas de la première colonne du tableau 5.6). D'autres trajectoires sont optimales par rapport à la distance (Δd_g minimale), alors que la durée de la trajectoire est grande (relativement aux durées des autres trajectoires). Dans ce cas la fonction f_g est minimisée uniquement par rapport à la distance (i.e. $w_t = 0$, donc $f_g = \Delta d_g$) (cas de la dernière colonne du tableau 5.6). Contrairement à ces deux cas, notre objectif est de minimiser la fonction f_g par rapport au temps et la distance, c'est pourquoi il faut trouver un compromis entre ces deux paramètres pour la sélection de la trajectoire partielle. Notre choix est de perdre un peu en distance pour gagner en temps. Ce qui revient au choix des facteurs de pondération w_d et w_t. A partir du tableau 5.6, il est clair que pour des valeurs de w_t inférieures à 0.8 (respectivement $w_d > 0.2$), la trajectoire est sélectionnée en minimisant la distance (i.e. Δd_g est minimale). Cependant, quand $w_t \geq 0.8$ (respectivement $w_t \leq 0.2$), les deux paramètres temps et distances sont minimisés. Δd_g correspondant à la trajectoire sélectionnée est plus grand que Δd_g minimale de $4.5m$, mais la durée de la trajectoire gagne $2.4s$ en moins. De plus, généralement, la distance traversée pendant ce temps ($2.4s$) est plus grande que la distance perdue ($4.5m$). Si par exemple le robot se déplace avec uniquement une vitesse de $5m/s$, la distance traversée pendant ce temps ($2.4s$) est $12m$, ce qui

est bien supérieur à $4.5m$. Notons que dans certaines situations, il se peut qu'il existe une trajectoire avec au même temps une durée ΔT_{Π} minimale et une distance au but Δd_g minimale. Dans ce cas, quel que soit la valeur des facteurs de pondération, la trajectoire sélectionnée est la même.

Tableau 5.6 : L'influence du choix des deux facteurs de pondération de la distance et du temps (respectivement w_d et w_t) sur la convergence de PASSPMP vers le but ; tests effectués dans un environnement contenant 22 obstacles et pour un état donné du but.

w_d	0	0.1	0.2	0.3	0.5	1
w_t	1	0.9	0.8	0.7	0.5	0
f_g	4.8	9.7	9.7	12.5	12.5	24.9
$\Delta d_g\ (m)$	29.4, 39.9	29.4	29.4	24.9	24.9	24.9
$\Delta T_{\Pi}\ (s)$	4.8	4.8	4.8	7.2	7.2	7.2

Nous avons effectué les mêmes tests que ceux représentés dans le tableau 5.6 pour différents états du but. Les résultats correspondant sont résumés dans le tableau 5.7 ; pour chaque état but, uniquement les résultats correspondant au couple de facteurs (w_d,w_t) à partir duquel les deux paramètres temps et distance sont minimisés, sont représentés dans le tableau (i.e. pour chaque état but, uniquement la colonne en gris dans le tableau 5.6 est représentée). Pour chaque couple de facteurs (w_d,w_t), nous déterminons comme dans le tableau 5.6 la fonction f_g, la distance au but Δd_g et la durée de la trajectoire ΔT_{Π}, et en plus, nous calculons la différence de distance au but en plus entre la trajectoire sélectionnée et la trajectoire avec la distance Δd_g minimale (que nous notons $Diff_{\Delta d}$), ainsi que la différence de temps en moins entre les deux trajectoire (que nous notons $Diff_{\Delta T}$). La principale remarque déduite à partir du tableau 5.7 est que plus le facteur w_d est grand (resp. plus w_t est petit), plus la différence de distance $Diff_{\Delta d}$ est petite. Cela est logique, car plus w_d est grand, plus le paramètre distance est favorisé. D'un

autre côté, en comparant les résultats obtenus pour des valeurs faibles et élevées de w_t, il peut paraître qu'il vaut mieux utiliser des faibles valeurs de w_t (relativement aux valeurs de $Diff_{\Delta d}$ et $Diff_{\Delta T}$). Cependant, si par exemple nous utilisons un facteur de temps $w_t = 0.2$ (resp. $w_d = 0.8$), dans le cas du test du tableau 5.6, uniquement la distance sera minimisée, et non pas les deux paramètres temps et distance. C'est pourquoi c'est plus commode d'utiliser des valeurs élevées de w_t. Quand $w_t = 0.9$, $Diff_{\Delta d}$ est relativement très grande, mais dans le cas où $w_t = 0.8$, la valeur de $Diff_{\Delta d}$ est acceptable relativement au temps gagné ($Diff_{\Delta T}$). Toutefois une note importante est que dans le cas où $w_t = 0.9$, la trajectoire partielle sélectionnée ne finit jamais par une trajectoire de freinage (sauf bien sûr dans le cas où toutes les trajectoires finissent par une trajectoire de freinage). De ce fait, si dans un cas particulier il n'est pas souhaitable de favoriser les trajectoires de freinage, il suffit de fixer le facteur w_t à cette valeur.

Tableau 5.7 : L'influence du choix des deux facteurs de pondération w_d et w_t sur la convergence de PASSPMP vers le but pour différents états du but.

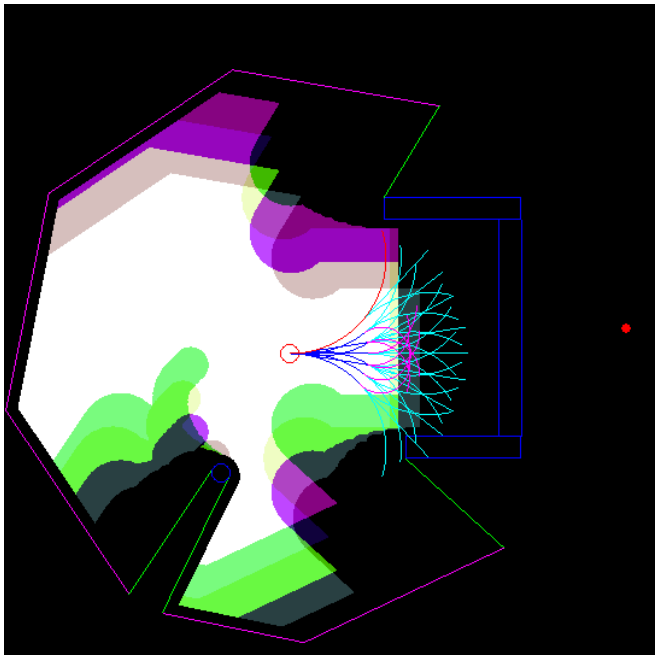
w_d	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
w_t	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2
f_g	17.9	9.7	7.4	7.9	18.1	9.2	24.2	10
$\Delta d_g\ (m)$	131.1	29.4	12.5	11.9	30.9	12.2	32.4	11.2
$\Delta T_{\Pi}\ (s)$	5.3	4.8	5.3	5.3	5.3	4.8	5.3	5.3
$Diff_{\Delta d}\ (m)$ (en plus)	17	4.5	3.4	2.2	1.6	1.1	0.7	0.4
$Diff_{\Delta T}\ (s)$ (en moins)	1.9	2.4	1.9	1.9	1.9	2.4	1.9	1.9

5.6.4 PASSPMP vs. PASSAVOID

D'après les tests présentés dans les paragraphes 5.5, 5.6.1 et 5.6.2, les deux méthodes PASSAVOID et PASSPMP sont toutes les deux passivement sûres, mais leur comportement est différent. PASSAVOID manque de prévoyance, c'est purement réactif, son seul but est de calculer un nouveau contrôle à appliquer pour le pas de temps suivant. En plus, il ne se préoccupe pas de conduire A vers un but donné. Par contre, PASSPMP raisonne sur plusieurs pas de temps en générant une trajectoire partielle vers le but.

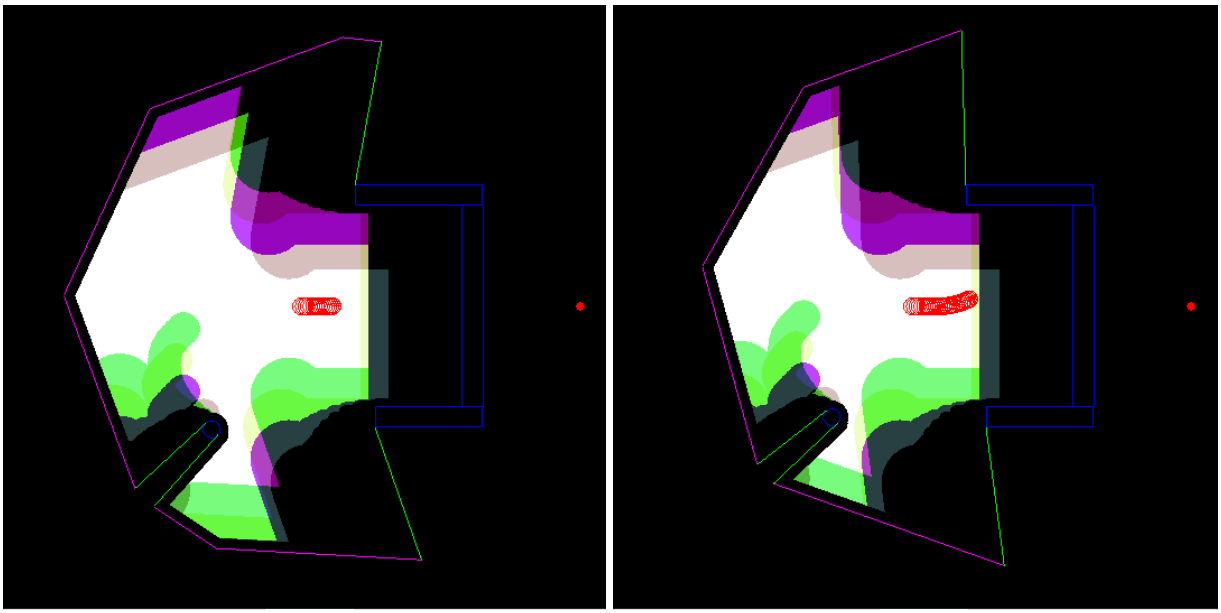
Un problème commun des méthodes réactives est le problème des minima locaux. Dans un cas pareil, le robot peut se retrouver bloqué à l'intérieur d'un obstacle de forme concave, et par conséquent, il ne pourra jamais atteindre son but. Pour tester la réaction des algorithmes PASSAVOID et PASSPMP dans une telle situation, les deux algorithmes ont été exécutés dans un environnement contenant un obstacle de forme concave. Les conditions expérimentales concernant le robot (contraintes kinodyamiques, champ de vision) et les objets mobiles (vitesse maximale) restent les même que celles définies dans les paragraphes 5.2 et 5.3. L'ensemble des trajectoires de freinage $\mathcal{E}$ utilisé par ICS^b-CHECK comprend 4 trajectoires.

La figure 5.40 montre l'efficacité de PASSPMP à générer une trajectoire (celle représentée en rouge) vers le but qui évite un obstacle statique de forme U. Cette trajectoire évite également la région ICS^b pour toujours garder le robot dans un état p-sûr. Cependant, notons que dans ce cas nous avons choisi les facteurs de pondération w_t et w_d (tel que $w_t = 0.9$ et $w_d = 0.1$) de manière à utiliser des primitives de trajectoire qui favorisent l'expansion de l'arbre pour éviter les cas où le robot doit freiner et s'arrêter pour demeurer passivement sûr (i.e. le cas des trajectoires qui finissent par des trajectoires de freinage (représentées en Magenta dans la figure 5.40)).

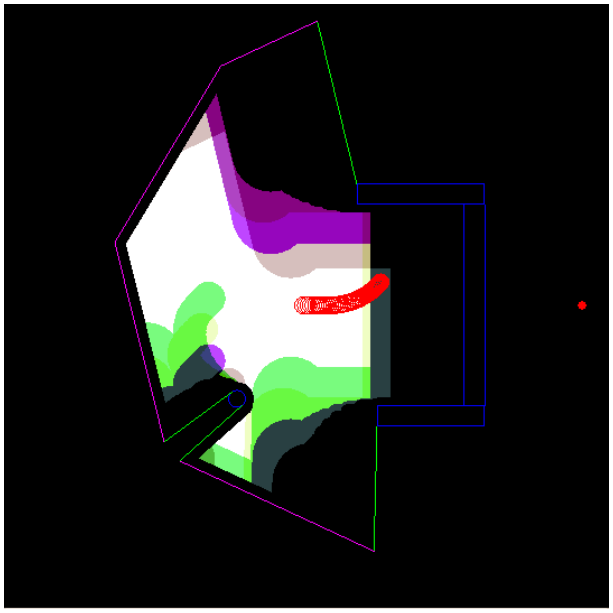


La figure 5.40 : PASSPMP permet au robot d'éviter un obstacle statique de forme U (la trajectoire sélectionnée par PASSPMP est représentée en rouge).

Le même test précédent a été effectué pour PASSAVOID, le robot doit se déplacer vers un but fixé devant lui, mais sa trajectoire est obstruée par un obstacle statique de forme U. De plus, il est supposé que dans ce cas de figure, PASSAVOID ne permet pas la marche à arrière. La figure 5.41 montre le comportement de PASSAVOID dans une telle situation ; tous d'abord le robot se déplace tout droit vers le but (voir la figure 5.41.a), puis à un instant donné, il détecte une région ICS^b (correspondant à l'obstacle statique), alors il modifie sa trajectoire pour l'éviter (voir la figure 5.41.b). Cependant, il se retrouve bloqué par une autre région ICS^b, et dans ce cas il est obligé de de s'arrêter pour rester passivement sûr (voir la figure 5.41.c). Par conséquent, le robot reste bloqué à l'intérieur de l'obstacle statique de forme U et ne peut plus atteindre son but. Toutefois, notons que dans d'autres situations où le robot peut faire marche arrière et où l'ensemble de contrôles U_{ctrl} (ligne #1 de l'algorithme 3) utilisé est moins restreint, le robot peut trouver une issue et sortir de ce cul de sac.



(a) (b)



(c)

Figure 5.41 : Comportement de PASSAVOID en présence d'un obstacle statique de forme U (situations (a)-(c)).

Cette expérience a montré l'avantage de PASSPMP par rapport à PASSAVOID quant à la résolution du problème des minima locaux.

5.7 Complexité et performance

L'implémentation des algorithmes développés au cours de ce travail de thèse a été faite en C++ utilisant openGL comme moteur de rendu et testée sur un PC portable équipé d'un Core i7 (1,6GHz x 2CPU, 4 GB RAM, ATI Mobility Radeon HD 4500 Series GPU, SE : Linux (ubunto)). Pour plus de démonstrations concernant les performances des algorithmes ICS^b-CHECK, PASSAVOID et PASSPMP, la complexité en temps de ces algorithmes est évaluée par rapport à différents paramètres.

Pour l'algorithme ICS^b-CHECK, le cœur ou noyau de cet algorithme est le calcul de $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}^*)$ (ligne #5 de l'algorithme 4). En supposant un modèle du futur temporellement discret (avec un pas de temps fixe Δt), cette étape peut être effectuée pour $n_t = T_h/\Delta t$, le nombre de pas de temps utilisé pour représenter le modèle du futur. Comme illustré dans le paragraphe 5.4.2, $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$ est obtenu à partir du calcul de $ICS^b{}_{\hat{z}_c}(B_i, \tilde{u}^*)$ pour chaque trajectoire de freinage et pour chaque objet en effectuant des unions et des intersections (selon les lignes #7 et #12 de l'algorithme 4), avec n_b le nombre de trajectoires de freinage (l'ensemble $\mathcal{E}$) et n_o le nombre d'objets de l'environnement.

Le temps d'exécution de ICS^b-CHECK a été évalué par rapport à plusieurs paramètres, tous liés au calcul de $ICS^b{}_{\hat{z}_c}(B, \mathcal{E})$; à savoir, n_o, n_b, n_t et R_{FoV}, (le rayon du champ de vision de A). La figure 5.42 illustre la variation du temps d'exécution par rapport au nombre d'objets n_o. La figure 5.43 représente la variation du temps d'exécution par rapport à différents ensembles de trajectoires de freinage n_b. La figure 5.44 quant à elle montre le changement du temps d'exécution par rapport à l'horizon temporel, à travers le nombre de pas du modèle du futur n_t . Enfin, il est évalué pour différents rayons du champ de vision de A, R_{FoV} (tel représenté dans La figure 5.45). Notons que chaque figure représente le temps d'exécution moyen pour un nombre d'expériences réalisées égale à 10.

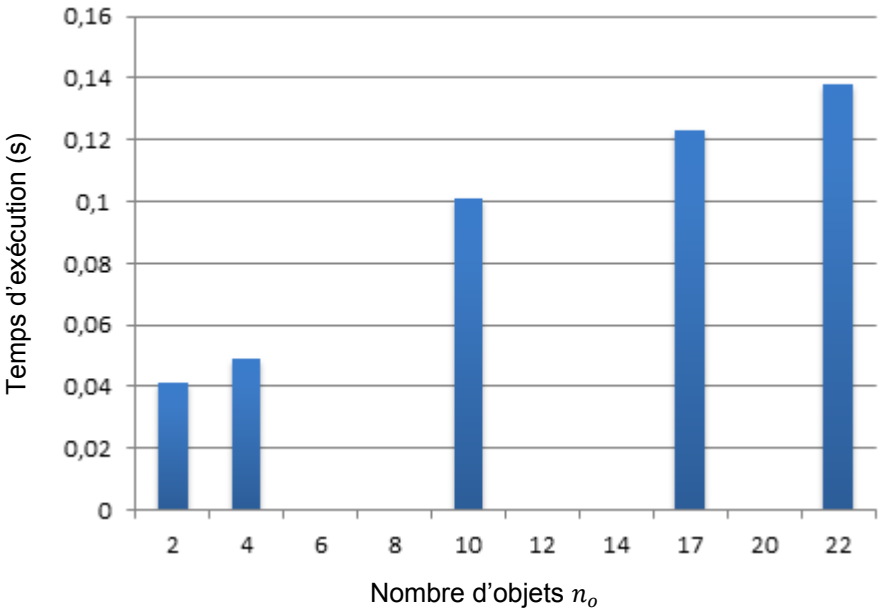


Figure 5.42 : Le temps d'exécution moyen de ICS^b-CHECK par rapport à n_o, le nombre d'objets ($n_b = 9,\ n_t = 71$ et $R_{FoV} = 80m$).

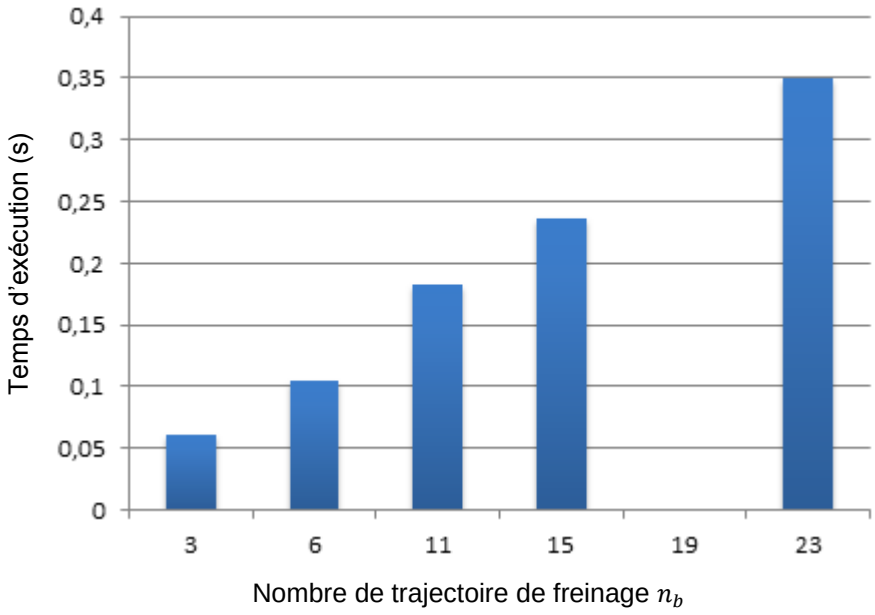


Figure 5.43 : Le temps d'exécution moyen de ICS^b-CHECK par rapport à n_b, le nombre de trajectoires de freinage ($n_0 = 22,\ n_t = 71$ et $R_{FoV} = 80m$).

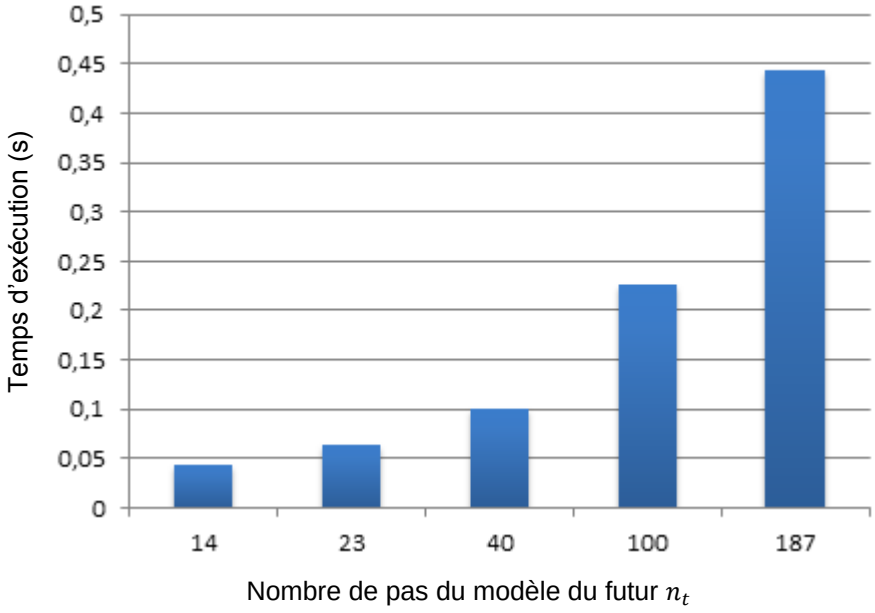


Figure 5.44 : Le temps d'exécution moyen de ICS^{b}-CHECK par rapport à n_t, le nombre de pas du modèle du futur ($n_0 = 22$, $n_b = 9$ et $R_{FoV} = 80m$).

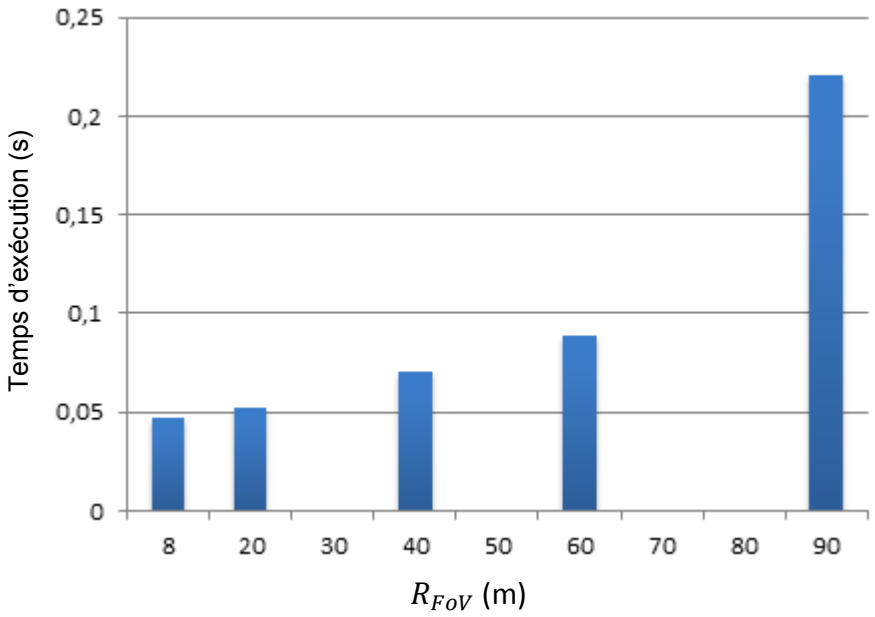


Figure 5.45 : Le temps d'exécution moyen de ICS^{b}-CHECK par rapport à R_{FoV}, le rayon du champ de vision de A ($n_0 = 22$, $n_b = 9$ et $n_t = 71$).

Nous déduisons à partir des figures 5.42, 5.43, 5.44 et 5.45 que le temps d'exécution de ICS^b-CHECK, croît avec n_o, le nombre d'objets (les boucles **forall** de l'algorithme 4), n_b, la taille de l'ensemble des trajectoires de freinage, n_t, le nombre de pas du modèle du futur et enfin le rayon du champ de vision de A. Pour ce dernier paramètre, il affecte directement ∂FoV, à savoir les limites de la perception et les occlusions (chaque point est un objet potentiel, selon le modèle du futur adopté). Cela peut être alors exprimé par n_{np}, le nombre d'objets non perçus (considérés dans ∂FoV).

Concernant PASSAVOID, sa complexité en temps augmente linéairement avec n_s, la taille de l'ensemble d'échantillonnage de l'espace de contrôle U_{ctrl} (la boucle **forall** de l'algorithme 3), et avec la complexité de ICS^b-CHECK (dans l'algorithme 4). Comme la complexité totale de ICS^b-CHECK croît linéairement avec n_o, n_b, n_t et n_{np}, la complexité en temps de PASSAVOID est donc une fonction de n_s, n_b, n_o, n_{np} et n_t. Toutefois, notons que le temps d'exécution de PASSAVOID dépend principalement du temps d'exécution de ICS^b-CHECK, et même s'il croit avec la taille de l'ensemble d'échantillonnage de l'espace de contrôle U_{ctrl} (i.e. n_s), ça reste minime (même pour un grand nombre de contrôles, par exemple $n_s = 23$, la procédure de vérification de la p-sûreté prend uniquement $9ms$). Les résultats obtenus sont satisfaisants quant à l'implémentation en temps réel de PASSAVOID. Même dans le cas où le nombre d'objets et le nombre de trajectoires de freinage sont tous les deux grands, le temps d'exécution reste approuvable. Ces résultats pourraient être encore améliorés grâce à un code d'optimisation[5] ou l'utilisation d'une machine plus puissante.

Contrairement à PASSAVOID qui calcule un contrôle à appliquer à chaque pas de temps, PASSPMP génère une trajectoire partielle p-sûre vers le but à chaque cycle de temps. D'une part, nous estimons dans ce qui suit la durée δ_e de cette trajectoire pour différentes valeurs du cycle PASSPMP ainsi que le nombre de nœuds générés durant le cycle.

[5] Une implémentation CUDA est en perspective.

Le tableau 5.8 montre que la durée de la trajectoire croit par rapport à la durée du cycle. Plus la durée du cycle est grande plus le planificateur dispose de temps pour explorer l'espace de recherche. Par conséquent, la trajectoire générée est plus grande et se rapproche plus du but. Cependant, notons que lorsque la durée du cycle est très petite, elle peut être insuffisante pour calculer une trajectoire p-sûre (voir le cas de la première valeur du tableau), car le test de la sûreté (calcul des ICS^b) prend plus de temps que δ_c.

Le tableau 5.8 montre également que le nombre de nœuds croit avec la durée du cycle, ce qui est évident du moment que l'expansion de l'arbre s'étend avec le temps et qu'elle est limitée par la durée du cycle (voir l'algorithme 6). Le nombre de nœuds croit exponentiellement, mais pour les deux dernières valeurs du tableau, ce n'est plus le cas, où le nombre de nœuds générés aurait dû être plus grand. Cela est dû aux régions non p-sûres (les états ICS^b) ; plus l'arbre s'étend, plus les primitives de trajectoire générées se rapprochent des états ICS^b (correspondant principalement à ∂FoV, la limite du champ de vision). Par conséquent, beaucoup de nœuds ne peuvent pas être étendus (i.e. il n'est pas possible de générer un ensemble de primitives p-sûres à partir du nœud en question). Dans ce cas une trajectoire de freinage sans collision est générée pour garantir la sûreté passive (voir ligne #12 de l'algorithme 6). Par exemple, pour la dernière valeur du tableau, la trajectoire partielle est le résultat de la concaténation de plusieurs primitives p-sûres mais qui finit par une trajectoire de freinage pour garantir que le robot soit à l'arrêt avant une éventuelle collision. Notons que, cette trajectoire de freinage est sélectionnée à partir d'un ensemble de trajectoires (dans ce test 21 trajectoires de freinage). Ainsi, pour chaque nœud non expansible, l'ensemble des trajectoires de freinage est testé par rapport à la non collision, puis la trajectoire qui conduit le robot à un état plus proche au but est sélectionnée. Par conséquent la complexité en temps de PASSPMP croît avec le nombre de trajectoires de cet ensemble et le nombre de nœuds non expansibles.

Tableau 5.8 : Evaluation de la durée de la trajectoire générée et le nombre de nœuds étendus pour différentes valeurs de la durée du cycle PASSPMP (avec $n_0 = 22$, $n_b = 23,\ R_{FoV} = 80m$ et $n_t = 71$).

La durée du cycle PASSPMP $\delta_c\ (s)$	0.3	1.5	2.5	3.5	4.5	5.5
Nombre de nœuds	1	13	40	135	176	237
Durée de la trajectoire $\delta_e\ (s)$	0	1.76	2.64	3.52	7.12	8

D'autre part, rappelons que PASSPMP étend un arbre de manière passivement sûre en utilisant la fonction EXPANSION_WITH_SAFETY_ CHECK (ligne #8 de l'algorithme 6). Cette fonction est basée sur ICS^b-CHECK. Comme la complexité totale de ICS^b-CHECK croît linéairement avec n_o, n_b, n_t et n_{np}, pour un cycle de temps donné, le temps disponible pour la construction de l'arbre diminue quand la valeur de l'un de ces paramètres augmente, et ainsi la durée (δ_e) de la trajectoire diminue également.

5.8 Conclusion

Ce chapitre a été consacré à la mise en œuvre des algorithmes développés dans les chapitres précédents et aux tests des différentes approches proposées pour des scénarii traitant des objets fixes ou mobiles avec un comportement futur inconnu, pour un robot mobile de type voiture ayant un champ de vision limité. Grace à ICS^b-CHECK, il a été montré comment le calcul des états de collision inévitable de freinage peut être effectué pour un modèle conservatif de l'environnement (contenant des objets perçus et des objets non perçus), et puis de vérifier si l'état du robot est passivement sûr ou non. Ces informations sont utilisées dans le système d'évitement d'obstacles (PASSAVOID) et dans le système de planification de mouvement partiel (PASSPMP). Les résultats des expériences effectuées ont montré la capacité de PASSAVOID et PASSPMP à garantir la sûreté passive du mouvement dans un

environnement dynamique inconnu (de manière à ce qu'une collision n'ait jamais lieu tant que le robot est en mouvement) et à agir en temps réel. Cependant, PASSAVOID cherche uniquement à éviter les ICS^b sans se préoccuper où aller alors que PASSPMP planifie la trajectoire du robot vers un but bien défini de manière passivement sûre en tenant compte des facteurs temps et distance.

CONCLUSION GENERALE ET PERSPECTIVES

Ce travail a traité le problème de la navigation de manière sûre d'un robot mobile avec un champ de vision limité placé dans un environnement dynamique inconnu. Etant donné que, la sûreté absolue est impossible à garantir dans une telle situation, nous avons introduit dans cette thèse un niveau plus faible de sûreté de mouvement appelé *sûreté de mouvement passive* : elle garantit que, si une collision se produit, le robot sera à l'arrêt. C'est un choix raisonnable compte tenu des contraintes sévères imposées par un champ de vision limité et un manque d'information sur l'environnement et son évolution future. Aussi limité que cela puisse paraitre, la sûreté de mouvement passive est intéressante pour deux raisons : (1) elle permet de fournir au moins une forme de garantie de la sûreté de mouvement dans des scénarii rigoureux, et (2) si chaque objet mobile dans l'environnement l'applique, alors aucune collision n'aura jamais lieu.

Parmi les contributions principales du travail présenté dans ce document se trouve le concept *ICS de freinage*, i.e., une version de ICS correspondant à la sûreté de mouvement passive. La sûreté de mouvement passive peut être obtenue en évitant les ICS de freinage à tout moment. Il a été montré que le concept ICS de freinage vérifie des propriétés qui ont permis la conception d'un algorithme efficace de vérification des ICS de freinage (ICS^{b}-CHECK). ICS^{b}-CHECK a été par la suite intégré dans un système de navigation réactive appelé PASSAVOID, dont la sûreté de mouvement passive a été formellement établie. PASSAVOID peut faire naviguer un robot planaire avec un champ de vision limité dans un environnement dynamique inconnu et il peut garantir que le robot évite toujours les ICS de freinage peu importe ce qui se produit dans l'environnement.

Afin de permettre à la fois, au robot d'atteindre un but bien défini et garantir la sûreté du mouvement, un planificateur de mouvement passivement sûr appelé PASSPMP a été développé : il est basé sur le principe de la planification de mouvement partielle et sur le concept ICS de freinage pour la garantie de la sûreté

du mouvement. A l'inverse des approches réactives qui calculent uniquement le contrôle à appliquer au pas de temps suivant et ainsi manquent de convergence vers le but d'une part, et des approches délibératives qui calculent un chemin complet vers le but mais qui nécessitent le plus souvent une connaissance a priori de l'environnement et sont peu adaptées pour les environnements dynamiques d'autre part, PASSPMP calcule périodiquement une trajectoire partielle passivement sûre, générée pour conduire le robot vers l'état but. Tout comme PASSAvoid, ce système opère dans un environnement dynamique inconnu et prend en considération le comportement futur des obstacles mobiles, en traitant aussi bien les obstacles perçus dans le champ de vision du système de perception que les obstacles non perçus. Mais contrairement à PASSAvoid, il permet de calculer la trajectoire optimale (en temps et en distance) à partir d'un état initial jusqu'à un état but donné. Cette trajectoire représente l'ensemble des trajectoires partielles passivement sûres sélectionnées (en se basant sur les critères sûreté et convergence vers le but) à chaque cycle de PASSPMP. Pour explorer l'espace d'états-temps, PASSPMP utilise une technique de diffusion basée sur l'expansion d'un arbre de recherche, l'approche développée est *p-safe RRT*, qui est une extension de la technique RRT (Rapidly exploring Random Trees). A la différence des approches de diffusion standards, celle-ci intègre le critère de la sûreté de mouvement ; elle utilise ICS^b-CHECK pour déterminer si un état est passivement sûr ou non. Ainsi, elle peut déterminer les nœuds et les branches de l'arbre qui sont passivement sûrs.

Les résultats des expériences effectuées montrent la capacité des algorithmes développés à agir dans des scénarii traitant des environnements inconnus encombrés d'objets fixes et mobiles avec un comportement futur inconnu. Le système robotique utilisé est de type voiture, équipé d'un capteur laser avec un champ de vision limité. Le modèle du futur utilisé est conservatif qui tient compte des régions occultées et des limites du champ de vision. Ainsi, le système de navigation élaboré permet au robot mobile de se déplacer dans de telles conditions, tout en garantissant au robot de ne jamais rentrer en collision avec les objets de l'environnement et que dans le cas où la collision est inévitable (causée par un objet inattendu par exemple), le robot sera à l'arrêt (i.e. la garantie de la sûreté passive).

Ce travail pourrait être étendu dans les directions suivantes :

- Pour augmenter l'efficacité de l'algorithme ICS^{b}-CHECK, le calcul des états de collision inévitable de freinage s'est fait de manière graphique. Etant donné la nature graphique des opérations à effectuer, cet algorithme pourra être implémenté directement sur le processeur graphique en utilisant la plateforme Cuda. De ce fait, nous bénéficierons de plus de rendement et parallélisme, et ainsi l'algorithme peut être optimisé.
- Implémenter la solution proposée dans ce travail sur une plateforme expérimentale (le Robucar) sous l'environnement ROS[1], où nous exploiterons le concept pair à pair (peer-to-peer). Ce concept facilitera énormément l'échange d'informations et rendra le flux du transfert de données fortement parallélisé pour donner de meilleurs résultats. Nous pourrons ainsi différencier entre les entités exécutables réactives (ex. les drivers), les entités cognitives (réflexion et prise de décision) et les entités de visualisation, d'affichage et de rendu. Cela va rendre notre solution flexible et modulaire, ainsi d'autres modules nécessaires pour améliorer la solution proposée peuvent être facilement intégrés, comme par exemple un module de suivi des objets mobiles.
- Dans certaines applications, la sûreté de mouvement passive peut être limitée, vu qu'elle s'occupe uniquement de la nocivité du robot envers l'environnement, i.e. quand une situation de collision survient, le robot prend toutes ses responsabilités sans les partager avec les autres obstacles de l'environnement. Il serait plus intéressant d'explorer des niveaux de sûreté plus élaborés comme *la sûreté de mouvement amicalement passive* : elle garantit que si une collision a lieu, le robot sera à l'arrêt et que l'objet en question aurait eu le temps de s'arrêter ou éviter la collision (s'il le désirait). Un tel niveau de sûreté de mouvement suppose que les objets mobiles ont des capacités cognitives, qu'ils peuvent aussi réagir et éviter les collisions et pour lesquels une certaine connaissance de leurs propriétés dynamiques est nécessaire (ce qui peut s'appliquer dans

[1] ROS (Robot operating System) est un outil (middle-wear) de développement spécifique pour les problématiques liées à la robotique.

plusieurs situations). En général, il peut être intéressant d'explorer d'autres formes de sûreté de mouvement en fonction de la particularité du problème de navigation considéré.

- A long terme, notre objectif est d'utiliser la technologie pour réduire le pourcentage des pertes humaines dans les accidents de la route, en dotant le robot mobile de la capacité de naviguer dans des environnement dynamiques inconnus de manière autonome et en se basant sur son système de perception, toute en garantissant la sûreté de son mouvement et celle de son entourage (les autres voitures, les piétons, etc.). Les voitures sans conducteur sont un avenir très prometteur dans le domaine du transport.

REFERENCES

1. Fletcher L., Teller S., Olson E., Moore D., Kuwata Y., How J., Leonard J., Miller I., Campbell M., Huttenlocher D., Nathan A. and Kline F.R., “The MIT—Cornell collision and why it happened”, International Journal of Field Robotics, V. 25, Issue 10, (2008), pages 775-807.
2. Thrun S., “What we’re driving”, at Google official blog, (9 October 2010).
3. Fraichard Th., “A short paper about motion safety”, IEEE Int. Conf. on Robotics and Automation, (2007), pages 1140-1145.
4. Pallottino L., Scordio V., Bicchi A., Frazzoli E., “Decentralized cooperative policy for conflict resolution in multivehicle systems”, IEEE Trans Robotics 23(6), V. 23, Issue 6, (2007), pages 1170-1183.
5. Van den Berg J. and Overmars M., “Planning time-minimal safe paths amidst unpredictably moving obstacles”, Int Journal Robotics Research V. 27, (2008), pages 1173-1174.
6. Lalish E., Morgansen K., “Decentralized reactive collision avoidance for multivehicle systems”, In: IEEE Conf. Decision and Control, Cancun (MX), (2008), pages 1218-1224.
7. Bekris K., Tsianos K. and Kavrak L., “Safe and distributed kinodynamic replanning for vehicular networks”, Mobile Networks and Applications V. 14, Issue 3, (2009), pages 292-308.
8. Bautin A., Martinez-Gomez L. and Fraichard T., “Inevitable collision states: a probabilistic perspective”, Proceedings of the IEEE International Conference on Robotics and Automation, (May 2010), pages 4022-4027.

9. Althoff D., Althoff M., Wollherr D. and Buss M., “Probabilistic collision state checker for crowded environments”, IEEE International Conference on Robotics and Automation, (2010), pages 1492-1498.
10. Fraichard T. and Howard T., “Iterative Motion Planning and Safety Issue”, Handbook of Intelligent Vehicles, Springer, (2012), pages 1433-1458.
11. Fraichard Th. and Asama H., “Inevitable collision states. a step towards safer robots?”, Advanced Robotics, V. 18, Issue 10, (2004), pages 1001-1024.
12. Fraichard T, “Will the driver seat ever be empty?” INRIA” Research Report, 2014.
13. Lumelsky V. and Tiwari S., “Velocity bounds for motion planning in the presence of moving planar obstacles”, In IEEE-RSJ int. conf. intelligent robots and systems, München, Germany, (1994).
14. Kohout R., Hendle J. and Musliner D., “Guaranteeing safety in spatially situated agents”, Proc. of National Conf. on Artificial Intelligence, Portland, V.2, (1996), pages 909-914.
15. S.Petti and T. Fraichard, “Safe motion planning in dynamic environments”, Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, (2005), pages 2210-2215.
16. LaValle S. M. and Kuffner J. J., “Randomized kinodynamic planning”, International Journal of Robotics Research, V. 20, Issue 5, (2001), pages 378–400.
17. Lozano-Pérez T., “Spatial planning. A configuration space approach”, IEEE Transactions on Computers, V. C-32, Issue 2, (1983), pages 108-120.
18. Lozano-Pérez T. and Wesley M., “An algorithm for planning collision-free paths among polyhedral obstacles”, Communications of the ACM, V. 22, Issue10, (1979), pages 560-570.
19. Latombe J.-C., “Robot Motion Planning”, Livre, Kluwer, Boston, MA, (1991).
20. Choset H., Lynch K. M., Hutchinson S., Kantor G. A., Burgard W., Kavraki L. E. and Thrun S., “Principles of Robot Motion: Theory, Algorithms, and Implementation”, Livre, MIT Press, (2005).

21. Avnaim F. and Boissonnat J.-D., "Polygon placement under translation and rotation", In Proceedings of Annual Symposium on Theoretical Aspects of Computer Science, V. 294, (1988), pages 322–333.
22. LaValle S., "Planning Algorithms", Livre. Cambridge University Press, (2006).
23. Fraichard T. and Laugier C., "Kinodynamic planning in a structured and time-varying 2D workspace", IEEE International Conference on Robotics and Automation, V. 2, (1992), pages 1500-1505.
24. Fraichard T., " Trajectory Planning in a Dynamic Workspace: a `State-Time Space' Approach", Advanced Robotics, V13, No. 1, (1999), pages 75-94.
25. Laumond J.P., "Robot Motion Planning and Control", Livre, ISBN 978-3-540-76219-5, (1998).
26. Siegwart R., Nourbakhsh I. and Scaramuzza D., "Introduction to Autonomous Mobile Robots", Livre, The MIT Press, (2004).
27. Ge S. S. and Lewis F. L., "Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications", Livre, SBN 9780849337482, (2006).
28. Casal A., "Reconfiguration Planning for Modular Self-Reconfigurable Robots", PhD thesis, Stanford University, Stanford, CA, (2002).
29. Lavalle S. M., Lin D., Guibas L. J., Latombe J. C. and Motwani R., "Finding an unpredictable target in a workspace with obstacles", In IEEE International Conference on Robotics and Automation, V. 1, (1997), pages 737-742.
30. Hart P.E., Nilsson N.J. and Raphael B., "A formal basis for the heuristic determination of minimum cost paths", IEEE Trans. on Systems Science and Cybernetics, V. 4, Issue 2, (1968), pages 100–107.
31. Dijkstra E. W., "A note on two problems in connexion with graphs", Numerische Mathematik, V. 1, Issue 1, (1959), pages 269-271.
32. Sekhavat S. and Laumond J.-P., "Topological property of trajectories computed from sinusoidal inputs for nonholonomic chained form systems", In IEEE International Conference on Robotics and Automation, V. 4, (1996), pages 3383–3388.

33. Chazelle B., “Approximation and decomposition of shapes”, In J. T. Schwartz and C. K. Yap, editors, Algorithmic and Geometric Aspects of Robotics, Lawrence Erlbaum Associates, Hillsdale, NJ, (1987), pages 145–185.
34. Lingas A., “The power of non-rectilinear holes”, In In Proceedings 9th International Colloquium on Automata, V. 140, (1982), pages 369–383.
35. Schwartz J. T. and Sharir M., “On the Piano Movers’ Problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers”, Communications on Pure and Applied Mathematics, V. 36, (1983), pages 345–398.
36. Schwartz J. T. and Sharir M., “On the “Piano Movers” Problem. II. General techniques for computing topological properties of algebraic manifolds”, Advances in Applied Mathematics, V. 12, (1983), pages 298–351.
37. Brock O. and Kavraki L.E., “Decomposition-based motion planning: a framework for real-time motion planning in high-dimensional configuration spaces”, In Proceedings IEEE International Conference on Robotics and Automation, V. 2, (2001), pages 1469–1474.
38. Vandapel N., Kuffner J., and Amidi O., “Planning 3-d path networks in unstructured environments”, In Proceedings IEEE International Conference on Robotics and Automation, (2005), pages 4624–4629.
39. Alami R., Simon T., and Laumond J. P., “A geometrical approach to planning manipulation tasks”, In International Symposium on Robotics Research, (1989), pages 113–119.
40. Lozano-Perez T., “Automatic Planning of Manipulator Transfer Movements”, IEEE Transactions on Systems, Man and Cybernetics, V. 11, Issue 10, (1981), pages 681-698.
41. Brooks R.A. and Lozano-Perez T., “A subdivision algorithm in configuration space for findpath with rotation”, IEEE Transactions on Systems, Man and Cybernetics, V. 15, Issue 2, (1985), pages 224 – 233.
42. Zhu D.J. and Latombe J.C., “New heuristic algorithms for efficient hierarchical path planning”, IEEE Transactions on Robotics and Automation, V. 7, Issue 1, (1991), pages 9-20.

43. Kavraki L. E., Svestka P., Latombe J. C. and Overmars M. H., “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”, IEEE Transactions on Robotics and Automation, V. 12, Issue 4, (June 1996), pages 566–580.
44. Rohnert H., “Shortest path in the plane with convex polygonal obstacles”, Information Processing Letters, V. 23, Issue 2, (1986), pages 71–76.
45. Van den Berg J. and Overmars M., “Roadmap-based motion planning in dynamic environments”, IEEE Transaction on Robotics, V. 21, Issue 5, (2005), pages 885-897.
46. LaValle S. M., “Rapidly-exploring random trees: A new tool for path planning”, Technical Report (Computer Science Department, Iowa State University) (TR 98-11), (October 1998).
47. Hsu D., Latombe J.C. and Motwani R., “Path planning in expansive configuration spaces”, In Proc. IEEE Int. Conf. on Robotics and Automation, V.3, (1997) pages 2719–2726.
48. Bekris K. E., Chen B., Ladd A., Plaku E. and Kavraki L., “Multiple query motion planning using single query primitives”, In IEEE/RSJ International Conference on Intelligent Robots and Systems, V. 1, (2003), pages 656–661.
49. Akinc M., Bekris K.E., Chen B.Y., Ladd A.M., Plaku E., Kavraki L.E., “Probabilistic Roadmaps of Trees for Parallel Computation of Multiple Query Roadmaps”, The International Symposium on Robotics Research, V.15, (2005), pages 80-89.
50. Ferguson D. and Stentz A., “Anytime RRTs”, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, (2006), pages 5369-5375.
51. Bruce J. and Veloso M., “Real-time randomized path planning for robot navigation”, In RoboCup 2002: Robot Soccer World Cup VI, Lecture Notes in Computer Science, V.2752, (2003), pages 288–295.
52. Zucker M, Kuffner J. and Branicky M., “Multipartite RRTs for Rapid Replanning in Dynamic Environments”, IEEE International Conference on Robotics and Automation, (2007), pages 1603 – 1609.

53. Macek K., Becked M. and Siegwart R., “Motion Planning for Car-Like Vehicles In Dynamic Urban Scenarios”, IEEE/RSJ International Conference on Intelligent Robots and Systems, (2006), pages 4375 – 4380.
54. Heon-Cheol L., Touahmi Y. and Beom-Hee L., “Grafting: A Path Replanning Technique for Rapidly-Exploring Random Trees in Dynamic Environments”, Advanced Robotics, V.26, Issue 18, (2012), pages 2145-2168.
55. Bessiere P., Ahuactzin J., El-Ghazali T. and Mazer E., “The ariane’s clew algorithm : Global planning with local methods”, In IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, V. 2, (1993), pages 1373–1380.
56. Mazer E., Ahuactzin J. and Bessiere P., “The ariane’s clew algorithm”, Journal of Artificial Intelligence Research, V. 9, (1998), pages 295-316.
57. Ahuactzin J.M. and Portilla A., “A basic algorithm and data structures for sensor-based path planning in unknown environments”, IEEE/RSJ International Conference on Intelligent Robots and Systems, V.2, (2000), pages 903 – 908.
58. Hsu D., Latombe J.-C. and Motwani R., “Path planning in expansive configuration spaces”, Int. Journal Computational Geometry & Applications, V. 9, (1999), pages 495–512.
59. Phillips J.M., Bedrossian N., Kavraki L.E., “Guided Expansive Spaces Trees: a search strategy for motion- and cost-constrained state spaces”, 2004 IEEE International Conference on Robotics and Automation, (ICRA '04), V. 4, (2004), pages: 3968 – 3973.
60. Milnor J., “Morse Theory”, Princeton University Press, Princeton, NJ, (1963).
61. Kavraki L. E. and Latombe J. C., “Randomized preprocessing of configuration space for path planning”, In IEEE International Conference on Robotics and Automation, V. 3, (1994), pages 2138–2145.
62. Kavraki L. E. and Latombe J. C., “Probabilistic roadmaps for robot path planning”, Practical Motion Planning in Robotics: Current Approaches and Future Directions, John Wiley, West Sussex, England, (1998), pages 33–53.

63. Haro, F. and Torres, M., “A Comparison of Path Planning Algorithms for Omni-Directional Robots in Dynamic Environments”, IEEE 3rd Latin American Robotics Symposium, LARS '06, (2006), pages 18-25.
64. Zhu Y., Zhang T., Song J., and Li X., “A new bug-type navigation algorithm for mobile robots in unknown environments containing moving obstacles,” The Industrial Robot, V. 39, (2012), pages 27-39.
65. Khatib O., “Real-time obstacle avoidance for manipulators and mobile robots”, International Journal of Robotics Research, V. 5, Issue 1, (1986), pages 90-98.
66. Koren Y. and Borenstein J., “Potential field methods and their inherent limitations for mobile robot navigation”, In IEEE International Conference on Robotics and Automation, V. 2, (April 1991), pages 1398-1404.
67. Khatib M. and Chatila R., “An Extended Potential Field Approach for Mobile Robot Sensor-Based Motions”, In Proceedings of the Intelligent Autonomous Systems IAS-4, IOS Press, Karlsruhe, Germany, (March 1995), pages 490–496.
68. Montano L. and Asensio J.R., “Real-Time Robot Navigation in Unstructured Environments Using a 3D Laser Range Finder”, In Proceedings of the IEEE/RSJ International Conference on Intelligent Robot and Systems, IROS 97, IEEE Press, V. 2, Grenoble, France, (1997), pages. 526–532.
69. Feder H.J.S. and Slotin J-J.E., “Real-Time Path Planning Using Harmonic Potentials in Dynamic Environments”, In Proceedings of the IEEE International Conference on Robotics and Automation, V. 1, (1997), pages 874-88.
70. Kamon I., Rivlin E., and Rimon E., “A new range-sensor based globally convergent navigation for mobile robots”, In IEEE Int’l. Conf. on Robotics and Automation, V. 1, (1996), pages 429-435.
71. GE S.S. and CUI Y.J., “Dynamic Motion Planning for Mobile Robots Using Potential Field Method”, Autonomous Robots, V. 13, Issue 3, (2002), pages 207–222.
72. Huang L., “Velocity planning for a mobile robot to track a moving target - a potential field approach”, Robotics and Autonomous Systems, V. 57, Issue 1, (2009), pages 55–63.

73. Yaghmaie F. A., Mobarhani A. and Taghirad H. D., “A New Method for Mobile Robot Navigation in Dynamic Environment: Escaping Algorithm”, Proceeding of the 2013 RSI/ISMInternationaI Conference on Robotics and Mechatronics, (2013), pages 212-217.
74. Fox D., Burgardand W. and Thrun S., “The dynamic window approach to collision avoidance”, IEEE Robotics and Automation Magazine V. 4, Issue 1, (1997), pages 23-33.
75. Brock O. and Khatib O., “High-speed navigation using the global dynamic window approach”, Proceedings of International Conference on Robotics and Automation, V. 1, (1999), pages 341-346.
76. Seder M., Petrovic I., “Dynamic window based approach to mobile robot motion control in the presence of moving obstacles”, In: IEEE Int. Conf. Robotics and Automation, Roma (IT), (2007), pages 1986-1991.
77. Fiorini P. and Shiller Z., “Motion planning in dynamic environments using velocity obstacles”, Int. Journal Robotics Research V. 17, Issue 7, (1998), pages 760-772.
78. Large F., Laugier C. and Shiller Z., “Navigation among moving obstacles using the NLVO: Principles and applications to intelligent vehicles”, Autonomous Robots, V. 19, (2005), pages 159-171.
79. Wilkie D., van den Berg J. and Manocha D., “Generalized velocity obstacles”, In Proceedings IEEE International Conference on Intelligent Robots and Systems, (2009), pages 5573–5578.
80. van den Berg J., Lin M. and Manocha D., “Reciprocal velocity obstacles for real-time multi-agent navigation”, Proceedings of the IEEE International Conference on Robotics and Automation, (2008), pages 1928-1935.
81. Snape J., van den Berg J., Guy S. J. and Manocha D., “Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles”, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, (2009), pages 5917-5922.

82. Parthasarathi R. and Fraichard T., “An inevitable collision state checker for a car-like vehicle”, in IEEE International Conference on Robotics and Automation, (2007), pages 3068-3073.
83. Lawitzky A., Nicklas A., Wollherr D. and Buss M., “Determining states of inevitable collision using reachability analysis”, IEEE/RSJ International Conference on Intelligent Robots and Systems, (2014), pages 4142-4147.
84. Martinez-Gomez L. and Fraichard T., “Collision avoidance in dynamic environments: an ics-based solution and its comparative evaluation”, IEEE International Conference on Robotics and Automation, (2009), pages 100-105.
85. Bekris K. E., “Avoiding Inevitable Collision States: Safety and Computational Efficiency in Replanning with Sampling-based Algorithms”, in the Workshop on Guaranteeing Safe Navigation in Dynamic Environments, held in conjunction with the Intl. Conf. on Robotics and Automation (ICRA), (2010).
86. Borenstein J. and Korem Y., “The vector field histogram _ fast obstacle avoidance for mobile robots”, IEEE Transactions on Robotics and Automation, V. 7, Issue 3, (1991), pages 278-288.
87. Ulrich I. and Borenstein J., “Vfh+ : Reliable obstacle avoidance for fast mobile robots”, V. 2, (1998), pages 1572–1577.
88. Ulrich I. and Borenstein J., “Vfh* : Local obstacle avoidance with look-ahead verification”, IEEE International Conference on Robotics and Automation, San Francisco, USA, V. 3, (2000), pages 2505-2511.
89. Jiea D., Xueming M. and Kaixiang P., “IVFH∗: Real-time dynamic obstacle avoidance for mobile robots”, 11th International Conference on Control Automation Robotics & Vision (ICARCV), (2010), pages 844 – 847.
90. Jianming Guo, Zhang Shouping, Xu Jia and Zhou Shenghui, “Kalman prediction based VFH of dynamic obstacle avoidance for intelligent vehicles”, International Conference on Computer Application and System Modeling (ICCASM), V.3, (2010), pages 6-10.

91. Simmons R., “The curvature velocity method for local obstacle avoidance”, Proceedings of the International Conference on Robotics and Automation, V. 4, (1996), pages 3375-3382.
92. Ko N.Y. and Simmons R. R., “The lane-curvature method for local obstacle avoidance”, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, V. 3, (October 1998), 1615-1621.
93. Molinos E., Llamazares A., Ocana M. and Herranz F., ”Dynamic obstacle avoidance based on curvature arcs”, IEEE/SICE International Symposium on System Integration (SII), (2014), pages 186 – 191.
94. Minguez J. and Montano L., “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, V. 3, (2000), pages 2094-2100.
95. Minguez J., Montano L. and Santos-Victor J., “Reactive navigation for non-holonomic robots using the ego kinematic space”, Proceedings IEEE International Conference on Robotics and Automation, (2002), pages 3074-3080.
96. Minguez J., Montano L., Simeon T. and Alami R., “Global nearness diagram navigation (gnd)”, Proceedings of the IEEE International Conference on Robotics and Automation, V. 1, (2001), pages 33-39.
97. Minguez J. and Montano L., “Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios", IEEE Transactions on Robotics and Automation, V. 20, (2004), pages 45-59.
98. Durham J.W. and Bullo F., “Smooth Nearness-Diagram Navigation”, IEEE/RSJ International Conference on Intelligent Robots and Systems, (2008), pages 690-695.
99. Chakravarthy A. and Ghose D., “Obstacle Avoidance in a Dynamic Environment: A Collision Cone Approach”, IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, V. 28, NO. 5, (1998), pages 562-574.

100. Ferrara A. and Rubagotti M., “Sliding Mode Control of a Mobile Robot for Dynamic Obstacle Avoidance Based on a Time-Varying Harmonic Potential Field”, In ICRA2007 Workshop Perception, Planning and Navigation for Intelligent Vehicles, (2007).
101. Gopalakrishnan B., Singh A.K. and Krishna K.M., “Time scaled collision cone based trajectory optimization approach for reactive planning in dynamic environments”, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), (2014), pages 4169-4176.
102. Owen E. and Montano L., “Motion planning in dynamic environments using the velocity space”, in proceedings IEEE International Conference on Intelligent Robots and Systems, (2005), pages 2833–2838.
103. Blanco J. L., González J. and Fernández-Madrigal J. A., “Extending obstacle avoidance methods through multiple parameter-space transformations”, in Autonomous Robots, V. 24, Issue 1, (2008) pages, 29–48.
104. Tychonievich L.A., Burton R.P. and Tychonievich, L.P., “Versatile reactive navigation”, in IEEE/RSJ International Conference on Intelligent Robots and Systems, (2009), pages 2966 - 2972.
105. Shuai C., Junhao X. and Huimin L, " Real-time obstacle avoidance using subtargets and Cubic B-spline for mobile robots”, IEEE International Conference on Information and Automation (ICIA), (2014), pages 634-639.
106. Yanyan Lu, Zhonghua Xi and Jyh-Ming Lien, "Collision prediction among polygons with arbitrary shape and unknown motion”, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), (2014), pages 4148-4153.
107. Petti S. and Fraichard T., “Partial motion planning framework for reactive planning within dynamic environments”, Proc. of the IFAC/AAAI Int. Conf. on Informatics in Control, Automation and Robotics, (2005), pages 199-204.
108. Petti S. and Fraichard T., “Safe Navigation of a Car-Like Robot in a Dynamic Environment”, in Proc. of the European Conf. on Mobile Robots, (2005).

109. Benenson R., Petti S., Fraichard T. and Parent M., “Integrating Perception and Planning for Autonomous Navigation of Urban Vehicles”, In IEEE-RSJ Int. Conf. on Intelligent Robots and Systems (IROS), (2006), pages 98-104.
110. Kelly A. and Stentz T., “Rough terrain autonomous mobility - part 2: an active vision and predictive control approach”, Autonomous Robots, V.5, Issue 2, (1998), pages 163–198.
111. Bonnafous D. Lacroix S. and Simeon T., "Motion generation for a rover on rough terrains", in IEEE/RSJ International Conference on Intelligent Robots and Systems, V. 2, (2001), pages 784 – 789.
112. Kelly A., Stentz T., Amidi O., Bode M., Bradley D., Mandelbaum R., Pilarski T., Rander P., Thayer S., Vallidis N. and Warner R., “Toward reliable off-road autonomous vehicles operating in challenging environments”, the International Journal of Robotics Research, V. 25, No. 5-6, (2006), pages 449–483.
113. Wettergreen D., Tompkins P., Urmson C., Wagner M. and Whittaker W., “Sun-synchronous robotics exploration: technical description and field experimentation”, the International Journal of Robotics Research, V. 24, No. 1, (2005), pages 3–30.
114. Biesiadecki J.J. and Maimone M. W., “The mars exploration rover surface mobility flight software: Driving ambition”, In Proceedings of the 2006 IEEE aerospace conference, (2006).
115. Knepper R., Srinivasa S. and Mason M., “An equivalent relation for local path sets”, In: Proceedings of the ninth international workshop on the algorithmic foundations of robotics, V. 68, (2010), pages 19-35.
116. Rogers-Marcovitz F. and Kelly A., “On-line mobile robot model identification using integrated perturbative dynamics”, In Proceedings of the 12th international symposium on experimental robotics, (2010), pages 417-431.
117. Howard T., Green C., Kelly A. and Ferguson D., “State space sampling of feasible motions for highperformance mobile robot navigation in complex environments”, Journal of Field Robotics, V. 25, No 6–7, (2008), pages 325–345.

118. Howard T. and Kelly A., “Optimal rough terrain trajectory generation for wheeled mobile robots”, International Journal of Robotics Research, V. 26, No 2, (2007), pages 141–166.
119. Stentz A., “Optimal and efficient path planning for unknown and dynamic environments”, International Journal of Robotics and Automation, V. 10, (1993), pages 89–100.
120. Stentz A., "The focussed d* algorithm for real-time replanning”, in Proceedings of the International Joint Conference on Artificial Intelligence, (1995), pages 1652–1659.
121. Koenig S. and Likhachev M., “Improved fast replanning for robot navigation in unknown terrain”, In IEEE Int. Conf. on Robotics and Automation, V. 1, (2002), pages 968-975.
122. Stentz A., “Constrained d*”, In Proceedings of the Eighteenth National Conference on Artificial Intelligence, (2002), pages 605-611.
123. Horwood J., Aragon N. and Poore A., “Gaussian sum filters for space surveillance: theory and simulation”, Journal of Guidance, Control, and Dynamics, V.34, No. 6, (2011), pages 1839–1851.
124. Bestaoui Sebbane Y., “Planning and Decision Making for Aerial Robots”, Intelligent Systems, Control and Automation: Science and Engineering, V. 71, (2014).
125. Zilberstein S. and Russell S., ”Approximate reasoning using anytime algorithms”, In Imprecise and Approximate Computation, Kluwer Academic Publishers, (1995).
126. Dean T. and Boddy M., “An analysis of time dependent planning”, In Proceedings of the National Conference on Artificial Intelligence (AAAI), (1988).
127. Zhou R. and Hansen E., “Multiple sequence alignment using A*”, In Proceedings of the National Conference on Artificial Intelligence (AAAI), (2002).

128. Likhachev M., Gordon G. and Thru S., “ARA*: Anytime A* with provable bounds on sub-optimality”, In Advances in Neural Information Processing Systems. MIT Press, (2003).
129. Likhachev M., Ferguson D., Gordon G., Stentz A. and Thrun S., “Anytime dynamic a*: An anytime replanning algorithm”, Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), (2005), pages 262-271.
130. van den Berg J., Ferguson D. and Kuffner J., “Anytime Path Planning and Replanning in Dynamic Environments”, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), (2006), pages 2366-2371.
131. Quinlan S. and Khatib O., “Elastic bands: Connecting path planning and control”, In Proceedings IEEE International Conference on Robotics and Automation, V. 2, (1993), pages 802–807.
132. Khatib O., “Real-time obstacle avoidance for manipulators and mobile robots”, PhD thesis, LAAS-CNRS, (1996).
133. Brock O. and Khatib O., “Real time replanning in high-dimensional configuration spaces using sets of homotopic paths”, Proc. IEEE Intl. Conf. on Robotics and Automation, V. 1, (2000), pages 550-555.
134. Khatib M., Jaouni H., Chatila R. and Laumond J.-P., “Dynamic path modification for car-like nonholonomic mobile robots”, In Proceedings IEEE International Conference on Robotics and Automation, V. 4, (1997), pages 2920–2925.
135. Lamiraux F., Bonnafous D. and Lefebvre O., "Reactive path deformation for nonholonomic mobile robots”, IEEE Transactions on Robotics and Automation, V. 20, Issue 6, (2004), pages 967-977.
136. Laforet S., "Evaluation de la sûreté de techniques de navigation autonomes", thèse de master, conservatoire national des arts et métiers, (2006).
137. Kurniawati H. and Fraichard T., “From path to trajectory deformation”, In Proceedings IEEE International Conference on Intelligent Robots and Systems, (2007), pages 159-164.

138. Delsart V. and Fraichard T., “Navigating dynamic environments using trajectory deformation”, In Proceedings IEEE International Conference on Intelligent Robots and Systems, (2008), pages 226-233.
139. Chen H. and Allgower F., “A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability,” Automatica, V.34, No.10, (1998), pages 1205–1217.
140. Mayne D., Rawlings J., Rao C. and Scokaert P., “Constrained model predictive control: Stability and optimality,” Automatica, V.36, No.6, (2000), pages 789-814.
141. Goodwin G. C., Seron M. M. and De Doná J. A., “Constrained Control and Estimation: an Optimization Approach”, livre, Springer, (2005).
142. Leung C., Huang S., Kwok N. and Dissanayake G., "Planning under uncertainty using model predictive control for information gathering," Robotics and Autonomous Systems, V.54, (2006), pages 898-910.
143. Richalet J., Rault A., Testud J.L. and Papon J., “Model Predictive Heuristic Control : Applications to Industrial processes”, Automatica, V.14, (1978), pages 413-428.
144. Morari M. and Lee H., "Model predictive control: past, present and future", Computers & Chemical Engineering, V.23, Issue 4-5, (1999), pages 667-682.
145. Qin S. J. and Badgwell T.A., “An Overview of Nonlinear Model Predictive Control Applications,” Progress in Systems and Control Theory, V.26, (2000), pages 369-392.
146. Qin S. J. and Badgwell T.A., “A Survey of Industrial Model Predictive Control Technology”, Control Engineering Practice, V.11, Issue 7, (2003), pages 733-764.
147. Piovesan J. L. and Tanner H. G., “Randomized model predictive control for robot navigation”, IEEE International Conference on Robotics and Automation, (2009), pages 94-99.
148. Brooks A., Kaupp T. and Makarenko A., “Randomised mpc-based motion-planning for mobile robot obstacle avoidance”, IEEE International Conference on Robotics and Automation, (2009), pages 3962-3967.

149. Farrokhsiar M. and Najjaran H., "Unscented predictive motion planning of a nonholonomic system", IEEE International Conference on Robotics and Automation, (2011), pages 4480 – 4485.
150. Maniatopoulos S., Panagou D. and Kyriakopoulos K.J., " Model Predictive Control for the navigation of a nonholonomic vehicle with field-of-view constraints", American Control Conference (ACC), (2013), pages 3967 – 3972.
151. Schouwenaars T., "Safe trajectory planning of autonomous vehicles", Ph.D. dissertation, Massachusetts Institute of Technology, (2006).
152. Yoon Y., Shin J., Kim H. J., Park Y. and Sastry S., "Model-predictive active steering and obstacle avoidance for autonomous ground vehicles", Control Engineering Practice, V.17, (2009) , pages 741-750.
153. Tahirovic A. and Magnani G., "Passivity-based model predictive control for mobile robot navigation planning in rough terrains", IEEE/RSJ International Conference on Intelligent Robots and Systems, (2010), pages 307–312.
154. Dunlap D.D., Caldwell C.V. and Collins E.G J.r., " Nonlinear Model Predictive Control using sampling and goal-directed optimization", IEEE International Conference on Control Applications (CCA), (2010), pages 1349–1356.
155. Goerzen C., Kong Z. and Mettler B., "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance", Journal of Intelligent and Robotic Systems, V.57, (2010), pages 65-100.
156. Kim H.J. and Shim D.H, "A flight control system for aerial robots: Algorithms and experiments", Control Engineering Practice, V.11, No.12, (2003), pages 1389–1400.
157. Mohandes A., Farrokhsiar M. and Najjaran H., "A Motion Planning Scheme for Automated Wildfire Suppression", 2014 IEEE 80th Vehicular Technology Conference, (2014), pages 1-5.
158. Ardiyanto I. and Miura J., "Cameraman robot: Dynamic trajectory tracking with final time constraint using state-time space stochastic approach", 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, (2014), pages 3108–3115.

159. Siegwart R. and Nourbakhsh I. R. and Scaramuzza D., “Introduction to Autonomous Mobile Robots”, Livre, MIT Press, (2011).
160. Pan J., Zhang L. and Manocha D., “Collision-free and smooth trajectory computation in cluttered environments”, Journal of Robotics Research, V. 31, Issue 10, (2012), pages 1155–1175.
161. Sarid S., Xu B. and Kress-Gazit H., “Guaranteeing High-Level Behaviors while Exploring Partially Known Maps”, Robotics: Science and Systems, MIT Press, (2012).
162. Täubig H., Frese U., Hertzberg C., Lüth C., Mohr S., Vorobev E. and Walter D., “Guaranteeing functional safety: design for provability and computer-aided verification”, Autonomous Robots, V. 32, (2012), pages 303–331.
163. Hsu D., Kindel R., Latombe J.C., Rock S., “Randomized kinodynamic motion planning with moving obstacles”, The International Journal of Robotics Research V. 21, no. 3, (2002), pages 233-255.
164. Bekris K., Kavraki L., “Greedy but safe replanning under kinodynamic constraints”, In: IEEE Int. Conf. Robotics and Automation, (2007), pages 704-710.
165. Althoff D., Kuffner J., Wollherr D. and Buss, M., “Safety assessment of robot trajectories for navigation in uncertain and dynamic environments”, Autonomous Robots, V. 32, (2012), pages 285–302.
166. Seward D., Pace C. and Agate R., “Safe and effective navigation of autonomous robots in hazardous environments”, Autonomous Robots, V. 22, (2007), pages 223–242.
167. van den Berg J., Abbeel P. and Goldberg K., “LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information”, In Robotics: Science and Systems, (2010).
168. Sadou M., Polotski V. and Cohen P., “Occlusion in obstacle detection for safe navigation”, In IEEE Intelligent Vehicles Symposium, (2004), pages 716-721.
169. Chung W., Kim S., Choi M., Choi J., Kim H., Moon C. and Song J., “Safe navigation of a mobile robot considering visibility of environment”, IEEE

Transactions on Industrial Electronics V. 56, Issue 10, (2009), pages 3941-3950.

170. Bouraine S., Fraichard T., "passive safe navigation for autonomous mobile robot in dynamic environments", Rapport de recherche, Inria, Grenoble, France, (2010).
171. Martinez-Gomez L. and Fraichard T., “An effcient and generic 2d inevitable collision state-checker”, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, (2008), pages 234-241.
172. Bekris K.E., Tsianos K. I and Kavraki L. E., “A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online”, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, (2007), pages 3784-3790.
173. Fiorini P. and Shiller Z., “Motion planning in dynamic environments using velocity obstacles”, International Journal of Robotics Research, V. 17, Issue 7, (1998), pages 760–772.
174. Elnagar A. and Gupta K., “Motion prediction of moving objects based on autoregressive model”, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, V. 28, Issue 6, (1998), pages 803-810.
175. Vasquez D., “Incremental learning for motion prediction of pedestrians and vehicles”, PhD thesis, Institut Nationale Polytechnique de Grenoble, (2007).
176. Schmidt C., Oechsle F. and Branz W., “Research on trajectory planning in emergency situations with multiple objects”, In IEEE intelligent transportation systems conference, (2006), pages 988-992.
177. Rohrmuller F., Althoff M., Wollherr D. and Buss M., “Probabilistic mapping of dynamic obstacles using markov chains for replanning in dynamic environments”, In IEEE/RSJ international conference on intelligent robots and systems, (2008), pages 2504-2510.
178. Vasquez D., Fraichard T. and Laugier C., “Growing hidden Markov models: a tool for incremental learning and prediction of motion”, International Journal of Robotics Research, V. 28, (2009), pages 1486–1506.

179. Broadhurst A., Baker S., Kanade T., “Monte Carlo road safety reasoning”, In IEEE intelligent vehicles symposium, (2005), pages 319-324.
180. Fraichard T., Asama H., “Inevitable collision states. A step towards safer robots?”, IEEE/RSJ International Conference on Intelligent Robots and Systems, V. 1, (2003), pages 388 - 393.
181. Macek K., Vasquez-Govea D.A., Fraichard T. and Siegwart R., “Safe Vehicle Navigation in Dynamic Urban Scenarios”, IEEE Conference on Intelligent Transportation Systems, (2008), pages 482 - 489.
182. Reif J., Sharir M., “Motion planning in the presence of moving obstacles”, In IEEE symposium on the foundations of computer science, Cambridge, (1985), pages 144-154.
183. LaValle S., Kuffner J., "Randomized kinodynamic planning”, In IEEE international conference on robotics and automation, V. 1, (1999), pages 473-479.
184. Aubin JP., “Viability theory”, Birkhuser, Boston, (1991).
185. Mitchell I. and Tomlin C., “Overapproximating reachable sets by hamilton-jacobi projections”, Journal of Scientific Computing, V. 19, Issue 1–3, (2003), pages 323-346.
186. Prajna S., Jadbabaie A., Pappas G., "A framework for worst-case and stochastic safety verification using barrier certificates”, IEEE Transactions on Automatic Control, V. 52, Issue 8, (2007), pages 1415-1428.
187. Bouraine S., Fraichard T. and Salhi H., “Relaxing the Inevitable Collision State concept to address provably safe mobile robot navigation with limited field-of-views in unknown dynamic environments”, IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, California. September 25-30, (2011), pages 2985-2991.
188. Bouraine S., Fraichard T. and Salhi H., “Provably safe navigation for mobile robots with limited fieldof- views in dynamic environments”, Autonomous Robots, Volume 32, Issue 3, (April 2012), pages 267-283.
189. Fraichard T. and Bouraine S., “Provably Safe Navigation for Mobile Robots with Limited Field-of-Views in Dynamic Environments”, A workshop of the 2011

Robotics: Science and Systems Conference, Los Angeles, CA (US), (June 27, 2011).

190. Bouraine S., Fraichard T. and Salhi H., “Provably Safe Navigation for Mobile Robots with Limited Field-of-Views in Unknown Dynamic Environments”, IEEE International Conference on Robotics and Automation (ICRA 2012), Saint Paul, Minnesota, USA. May 14-18, (2012), pages 174-179.
191. Fulgenzi C., Spalanzani A. and Laugier C., “Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid”, IEEE International Conference on Robotics and Automation, (2007), pages 1610-1616.
192. Althoff M., Stursberg O. and Buss M., “Model-based probabilistic collision detection in autonomous driving”, IEEE Transactions on Intelligent Transportation Systems, V. 10, (2009), pages 299 – 310.
193. Snape J., van den Berg J., Guy S. and Manocha D., “The Hybrid Reciprocal Velocity Obstacle”, IEEE Transactions on Robotics, V. 27, Issue 4, (2011), pages 696-706.
194. Bouraine S. and Fraichard T. and Azouaoui O. and Salhi H., “Passively Safe Partial Motion Planning for Mobile Robots with Limited Field-of-Views in Unknown Dynamic Environment”, 2014 IEEE International Conference on Robotics and Automation (ICRA 2014), (2014), pages 3576-3582.
195. Macek K., Vasquez-Govea D.A., Fraichard T. and Siegwart R., “Towards safe vehicle navigation in dynamic urban scenarios”, Automatika, V. 50, no. 3-4, (2009), pages 184-194.
196. Mason M. T., “Mechanics of Robotic Manipulation”, MIT Press, (2001).
197. Rosenbloom P. S., Laird J. E. and Newell A., “The Soar Papers: Research on Integrated Intelligence”, MIT Press, Cambridge, MA, (1993).
198. Dunlaing C. and Yap C., “Generic Transformation of Data Structures”, Proc. 23rd IEEE Foundations of Computer Science, (1982), pages 186-195.