Optimisation de la navigation robotique

Sawssen Jalel

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Signal, Image, Acoustique et Optimisation

Présentée et soutenue par :

Mme SAWSSEN JALEL

le vendredi 16 décembre 2016

Titre :

Optimisation de la navigation robotique

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Institut de Recherche en Informatique de Toulouse (I.R.I.T.)

Directeur(s) de Thèse :

M. PHILIPPE MARTHON

M. KHALED BSAIES

Rapporteurs :

M. MOHAMED JALLOULI, ECOLE NALE D'INGENIEUR DE SFAX TUNISIE

M. THIERRY FRAICHARD, INRIA GRENOBLE - RHONE ALPES

Membre(s) du jury :

M. PIERRE SPITERI, INP TOULOUSE, Président

M. FAOUZI MOUSSA, UNIVERSITE DE TUNIS EL-MANAR, Membre

M. KHALED BSAIES, UNIVERSITE DE TUNIS EL-MANAR, Membre

M. LEON RATSIFANDRIHANA, SEGULA TECHNOLOGIES, Membre

Mme ANA MARIA ROCHA, UNIVERSITE DE MINHO BRAGA, Membre

M. PHILIPPE MARTHON, INP TOULOUSE, Membre

À la mémoire de ma mère

Remerciements

Cette thèse constitue une riche expérience qui ne peut s'achever sans remercier les personnes qui m'ont encadré, soutenu et aidé à la réalisation de ce travail.

Mes premiers remerciements sont adressées à mes directeurs de thèse M. Philippe Marthon et M. Khaled Bsaïes auxquels j'exprime ma profonde gratitude et ma reconnaissance d'avoir bien voulu diriger mes travaux de thèse. Je remercie M. Philippe Marthon pour son soutien amical, sa disponibilité, son expérience et sa patience ainsi que pour sa bonne humeur. Ses conseils, tout au long de ce travail, toujours pertinents et sa rigueur scientifique m'ont été très utiles pour mener à bien ce travail. Je le remercie également de m'avoir accordé sa confiance lors des enseignements que j'ai réalisés. Je ne le remercierai jamais assez et je lui serai toujours reconnaissante. Je tiens aussi à remercier M. Khaled Bsaïes pour la confiance qu'il m'a témoignée tout au long de ces années, pour tous ses conseils et directives généreusement donnés et pour ses encouragements qui ont poussé en avant mes travaux de recherche.

Je souhaite également exprimer mes sincères remerciements et reconnaissance à mon co-encadrant M. Atef Hamouda, Maître Assistant à la Faculté des Sciences de Tunis, qui m'encadre depuis 2009, et avec qui j'ai effectué mon premier stage de recherche en Master. Je le remercie pour son souci constant de l'avancement de ma thèse ainsi que son suivi continu de mon travail. Ses conseils, ses encouragements ainsi que la confiance qu'il m'a toujours témoigné m'ont été d'un grand apport tout au long de mes travaux. Monsieur, je n'oublierai jamais que c'était grâce à vous que je fais de la recherche aujourd'hui. Merci de m'avoir soutenue, merci infiniment pour votre précieuse présence !

Mes remerciements vont ensuite aux membres du jury, qui m'ont fait l'honneur d'évaluer mon travail de thèse. Je vous en suis très reconnaissante. Tout d'abord, mes remerciements vont à M. Pierre Spiteri, professeur émérite à L'INP-ENSEEIHT pour avoir accepté de présider mon jury de thèse. Je remercie également M. Thierry Fraichard, chargé de recherche au centre INRIA Grenoble Rhone-Alpes et M. Mohamed Jallouli, professeur à l'Université de Gabès pour leur long travail de rapporteur. Merci infiniment pour le temps que vous avez consacré à la lecture de mon manuscrit et à la rédaction de vos rapports éclairés.

Mes vifs remerciements s'adressent également à M. Faouzi Moussa, Professeur à la faculté des sciences de Tunis, Mme. Ana Rocha, Professeur à l'Université de Minho,

et M. Léon Ratsifandrihana, Docteur à SEGULA Aéronautique et Défense, pour avoir bien voulu faire partie de mon jury.

Merci à mes amis de la faculté des sciences de Tunis : Zouhour, Hmida, Mohamed, Aymen, Ines, Bilel, Malek, Imen, Balkis, Souad, Amina, Nidhal, Rami, Khaoula, Nejib, Fatma... Je les remercie pour tous les bons moments passés ensemble, pour les échanges instructifs que nous avons eus et aussi pour le soutien moral qu'apporte le groupe.

Un grand merci à Philippe Papaix, Ghania, Saliha, Arselan, Hafidha, Sara, Amel, Ibtissem et Henda.

Un merci tout particulier à ma chère amie Rania. Je te remercie Rannou pour ta bonne humeur, ta gentillesse et tes encouragements. Je ne peux que te souhaiter beaucoup de bien, de joie et de bonne chance.

Pour finir, j'envoie une pensée émue à tous les membres de ma famille. Leur soutien sans faille et leurs conseils ont constitué un réconfort permanent. Je remercie en particulier mon cher frère Mokded, sans qui cette thèse n'aurait pas été. Mention spéciale à mon mari chéri. Merci Raouf d'être toujours là pour moi, Merci de ta patience, de ton support et de pouvoir subir mon stress...

Merci à tous,

Sawssen

Résumé

La robotique mobile autonome est un axe de recherche qui vise à donner à une machine la capacité de se mouvoir dans un environnement sans assistance ni intervention humaine. Cette thèse s'intéresse à la partie décisionnelle de la navigation robotique à savoir la planification de mouvement pour un robot mobile non-holonome, pour lequel, la prise en compte des contraintes cinématiques et non-holonomes est primordiale. Aussi, la nécessité de considérer la géométrie propre du robot et la bonne maîtrise de l'environnement dans lequel il évolue constituent des contraintes à assurer.

En effet la planification de mouvement consiste à calculer un mouvement réalisable que doit accomplir le robot entre une position initiale et une position finale données. Selon la nature de l'environnement, notamment les obstacles qui s'y présentent, deux instances du problème se distinguent : la planification de chemin et la planification de trajectoire.

L'objectif de cette thèse est de proposer de nouveaux algorithmes pour contribuer aux deux instances du problème de planification de mouvement. La méthodologie suivie repose sur des solutions génériques qui s'appliquent à une classe de systèmes robotiques plutôt qu'à une architecture particulière. Les approches proposées intègrent les B-splines Rationnelles non uniformes (NURBS) dans le processus de modélisation des solutions générées tout en s'appuyant sur la propriété de contrôle local, et utilisent les algorithmes génétiques pour une meilleure exploration de l'espace de recherche.

Mots clés : Navigation autonome, planification de trajectoires, courbes NURBS, évitement d'obstacles, environnements statiques et dynamiques, algorithmes génétiques.

Abstract

The mobile robotics is an area of research that aims to give a machine the ability to move in an environment without assistance or human intervention. This thesis focuses on the decisional part of robotic navigation, namely motion planning for a non-holonomic mobile robot, for which, the consideration of kinematic and non-holonomic constraints is paramount. Also, the need to consider the specific geometry of the robot and the good control of the environment in which it operates are constraints to insure.

Indeed, motion planning is to calculate a feasible movement to be performed by the robot between an initial and a final given position. Depending on the nature of the environment, two instances of the problem stand out : the path planning and the trajectory planning.

The objective of this thesis is to propose new algorithms to contribute to the two instances of motion planning problem. The followed methodology is based on generic solutions that are applicable to a class of robotic systems rather than a particular architecture. The proposed approaches include the Non-Uniform Rational B-Spline (NURBS) in the modeling process of the generated solutions while relying on the local control property. Also, they use genetic algorithms for better exploration of the search space.

Key words : Autonomous navigation, planning optimal trajectories, NURBS curves, obstacle avoidance, static and dynamic environments, genetic algorithms.

Table des matières

Liste des figures

Liste des tableaux

Introduction Générale

DE nos jours, l'émergence de la robotique "intelligente" conduit à rendre les robots de plus en plus autonomes et de plus en plus faciles à piloter. Un système robotique est défini comme un système artificiel, doté de capteurs et d'actionneurs, conçu pour agir sur le monde extérieur. Au début, ces automates ont été installés dans les industries pour accomplir des tâches pénibles, répétitives ou impossibles pour les humains. De nos jours, nous pouvons trouver les robots dans des différents domaines : industriel, domestique, médical, militaire, etc.

Contrairement aux robots manipulateurs, nous appelons robots mobiles l'ensemble des robots à base mobile et le plus souvent "les robots à roues" sinon il faut préciser le type de locomotion (marcheurs ou humanoïdes, sous marins, aérien). Le robot "Shakey", qui représente le sujet de recherche du Stanford Research Institut (SRI) de 1966 à 1972, était le premier robot mobile avec des capacités de perception et de raisonnement sur son environnement.

D'une manière générale, un robot mobile est un système équipé de modules de perception, de décision et d'action lui permettant d'agir sur son environnement en vue d'atteindre un objectif qui lui a été attribué, et ce d'une manière autonome ou supervisé. La gestion de ces différents éléments est définie par une architecture de contrôle, qui peut potentiellement faire appel à un modèle interne de l'environnement lui permettant la planification de ses actions à long terme. Dans ce manuscrit, nous nous intéressons à une sous famille de robots appelée "Les robots mobiles autonomes". Le critère d'autonomie se réfère à la capacité de ces machines de choisir ses actions pour achever certains objectifs et de réaliser correctement ses tâches même dans des situations inattendues et sans intervention humaine.

Pour opérer dans le monde réel, les robots doivent être capables de détecter, de raisonner, et d'agir. La capacité de générer un mouvement valide est cruciale pour le succès de toute l'opération. Sans cette aptitude, un robot ne sera pas en mesure d'accomplir sa tâche et par conséquent, les résultats de détection et de raisonnement

deviendront nuls. Cette thèse est une contribution dans le domaine de la génération des trajectoires pour les robots mobiles. Ceci inclut la planification des chemins et l'évitement d'obstacles, qui représentaient des sujets de recherches depuis le début de la robotique, et en particulier la robotique mobile (depuis la fin des années soixante au début des années soixante-dix).

D'après Levitt et Lawton (1990), le concept de la navigation autonome est défini comme étant le processus permettant de répondre aux trois questions suivantes [Levitt and Lawton, 1990] :

1. "Où suis-je ?"
2. "Où sont les autres lieux par rapport à moi ?"
3. "Comment puis-je atteindre ces autres lieux depuis l'endroit où je me trouve ?"

Pour répondre à ces questions, un robot mobile doit avoir la capacité de déterminer ses coordonnées par rapport à un système de référence (étape de localisation). Ensuite, détecter et localiser les éléments de l'environnement qui l'entoure (étape de perception) et enfin planifier le mouvement à effectuer pour atteindre son objectif en respectant un certain nombre de contraintes liées à l'environnement (présence des obstacles) et aux capacités du robot (étape de planification). Ainsi, le processus complet qui permet à un robot de naviguer dans son environnement se résume en trois modules fortement interdépendantes : la perception, la localisation et la planification.

La perception : C'est l'étape de capture des informations et données requises à partir de l'environnement extérieur et en utilisant les capteurs de différents types que disposent le robot (caméras, capteurs ultrasons, capteurs laser, capteurs optiques,...). Cette étape est primordiale pour la navigation d'un robot mobile autonome puisqu'elle permette la modélisation de l'environnement (élaboration d'une représentation mathématique) et au final de créer un modèle, plus ou moins simplifié, des interactions du robot avec son entourage.

La localisation : Suite à l'étape de perception, le robot doit déterminer sa posture (position et orientation) courante en traitant des données de natures différentes (mesures de divers capteurs, diverses cartes,...). La mission de localisation peut se diviser en deux phases. La première consiste à conditionner les mesures capteur afin

de construire le modèle sensoriel. Tandis que la deuxième est consacrée à la mise en correspondance de ce modèle avec la carte de l'environnement ce qui revient à apparier les observations avec des primitives cartographiques. Là, il s'agit de l'étape dominante pour le calcul de la configuration du robot.

La planification : Cette étape consiste à calculer un mouvement réalisable que doit accomplir le robot entre une position initiale et une position finale données. Selon la nature de l'environnement, notamment les obstacles qui s'y présentent, nous distinguons :

- *La planification de chemin :* Représente le cas de planification de mouvement le plus simple dans lequel le robot se déplace dans un environnement peuplé par des obstacles statiques. Ainsi, nous nous limitons à l'aspect géométrique du problème ce qui revient à calculer une suite continue de configurations sans collision entre deux configurations données.
- *La planification de trajectoire :* Dans certains problèmes, l'espace de travail peut être dynamique. L'emplacement des obstacles peut changer au fil du temps, ce qui rend la dimension temporelle un paramètre important dans l'évaluation de la validité du mouvement du robot. Ainsi, la planification de trajectoire devient incontournable dès que le problème nécessite la prise en compte des contraintes dynamiques du système robotique que ce soit celles concernant l'environnement (présence d'obstacles mobiles) ou celles qui décrivent la dynamique du robot.

Parmi ces différents modules, cette thèse s'intéresse à la partie décisionnelle de la navigation robotique à savoir la planification de mouvement pour un robot mobile non-holonome (exemple une voiture) pour lequel la prise en compte des contraintes cinématiques est primordiale ; couplée avec la nécessité de considérer sa propre géométrie et sans oublier la bonne maîtrise de l'environnement dans lequel il évolue.

Étant donné un système robotique R et un environnement Ω, la notion de planification de chemin/trajectoire sans collision revient à déterminer une trajectoire C permettant à ce robot de rejoindre une position finale P_f en partant d'une position initiale P_i. Une trajectoire est dite lisse si elle est continue et dérivable, ne contenant pas, donc, des points anguleux qui, s'ils existaient, obligent le robot à s'arrêter brusquement pour changer de direction. Ce dernier est soumis, par conséquent, à

des variations brusques de vitesses pouvant provoquer le glissement des roues. Aussi, l'arrêt et le redémarrage fréquents engendrent une perte d'énergie inévitable ! Deux questions s'imposent : La première c'est **comment modéliser notre solution** et la deuxième concerne la **spécification des critères** à prendre en compte lors de la génération de cette solution. Ainsi, plusieurs techniques d'interpolation et d'approximation ont été introduites pour modéliser une trajectoire. Les plus répandus ces jours sont les splines [Shikin and Plis, 1995] [Egerstedt and Martin, 2010], les courbes de Bézier [Prautzsch et al., 2002], les B-splines et les B-splines rationnelles non uniformes (NURBS) [Piegl and Tiller, 1997]. Chacune de ces splines a ses propres mérites et lacunes, qui seront présentés en détails ultérieurement.

Étant une généralisation des B-Splines, les B-Splines rationnelles non uniformes ou NURBS (pour Non-Uniform Rational Basis Splines) sont les plus avantageuses pour la modélisation et l'ajustement des courbes en donnant plus de souplesse, de précision et de contrôle local. En fait, l'incorporation des poids pour les points de contrôle améliore la fléxibité et permet aux NURBS de synthétiser les courbes analytiques aussi bien que celles de formes libres en modifiant les points de contrôl, le vecteur nodal ou les poids. Ces derniers représentent ainsi les degrés de liberté i.e en changeant au moins un de ces paramètres, différentes allures de courbes peuvent être obtenues. Par conséquent, une question judicieuse se pose : **Comment attribuer des valeurs à ces variables afin d'avoir la forme souhaitée de la solution ?** Plusieurs recherches ont été menées dans le domaine de la paramétrisation des NURBS. En effet, trouver la bonne paramétrisation lors du calcul de la courbe est d'une importance indéniable pour les techniques d'ajustement des courbes et surfaces.

La performance d'un algorithme de planification dépend de la façon cruciale des contraintes et des critères pris en comptes. Une contrainte se caractérise par le fait qu'elle doit être satisfaite. Un critère par le fait qu'il doit être optimisé (minimisé ou maximisé). Parmi les contraintes imposées à la trajectoire, nous trouvons par exemple une limite inférieure sur le rayon de courbure du robot, ou le fait que la trajectoire soit libre de collision. Parmi les critères (à minimiser), nous citons la distance que parcourt le robot entre les positions initiale et finale (plus la trajectoire est courte, mieux cela vaut), l'énergie, la consommation du fuel, le temps mis pour suivre cette trajectoire, etc.

De ce fait, **résoudre un problème de planification de trajectoire revient à résoudre un problème d'optimisation avec contraintes**. Quelles sont alors les méthodes permettant de résoudre de tels problèmes d'optimisation ? Ces problèmes sont non convexes et non différentiables de sorte que pour les résoudre, il faut généralement utiliser des méta-heuristiques. Ces méthodes sont souvent inspirées des systèmes naturels, qu'ils soient pris en biologie de l'évolution (algorithmes génétiques), en physique (recuit simulé) ou en éthologie (algorithmes de colonies de fourmis ou essaims particulaires).

Nos contributions à la planification de mouvement concernent les deux instances du problème à savoir la planification du chemin et la planification de trajectoire. La méthodologie suivie repose sur l'intention de concevoir des solutions génériques qui s'appliquent à une classe de systèmes robotiques plutôt qu'à une architecture particulière. Les travaux présentés dans ce manuscrit se classifient selon la nature de l'application considérée. Pour un système mono-robot, nos contributions se résument en :

1. Développement d'une nouvelle approche de planification de chemin avec contrainte de courbure en utilisant les courbes NURBS dans un environnement statique [contexte déterministe] ([Jalel et al., 2013], [Jalel et al., 2015b]) .
2. Optimisation de la méthode précédente par l'intégration des algorithmes génétiques lors de la paramétrisation de la courbe NURBS modélisant la solution [contexte déterministe] ([Jalel et al., 2015a], [Jalel et al., 2015c]).
3. Intégration de la méthode de Hooke et Jeeves dans le processus de modélisation du chemin en termes d'une courbe NURBS [contexte déterministe] ([Jalel et al., 2016]).
4. Proposition d'un nouvel algorithme de déformation de trajectoire pour un robot mobile non holonome évoluant dans un environnement dynamique [contexte stochastique].

La suite du présent manuscrit est organisée comme suit : nous proposons dans le chapitre 1 un état de l'art non exhaustif sur la planification de mouvement pour les robots mobiles. Dans le chapitre 2, nous présentons les techniques de lissage les plus utilisées pour la modélisation des chemins pour les robots mobiles non-holonomes. Nous détaillerons ensuite, dans le chapitre 3, l'emploie des courbes NURBS et des algorithmes génétiques, en tant que technique d'optimisation, pour la résolution du problème de planification de chemin en tenant compte des contraintes du système étudié. Dans le

même contexte, nous proposons une deuxième solution au problème en utilisant la méthode de Hooke et Jeeves. Le chapitre 4 s'intéresse à la navigation en environnement dynamique stochastique et décrit ainsi l'algorithme de déformation de trajectoire proposé. Finalement, ce mémoire se clôture par des conclusions et perspectives où nous concluons cette thèse, discutons les limites de nos algorithmes et suggérons les orientations futures ainsi que les perspectives de notre travail.

CHAPITRE 1

Synthèse des principaux travaux en planification de mouvement

1.1 Introduction

Dans sa formulation générale, la planification de mouvement pour un robot R évoluant dans un environnement W, revient à déterminer une séquence d'actions lui permettant de se déplacer entre deux configurations données tout en respectant un certain nombre de contraintes et de critères. Cette séquence d'actions se définit par une succession de points de passage appartenant à la trajectoire du mobile considéré qui doit respecter un certain nombre de contraintes et de critères qui dépendent généralement de la nature de la tâche à réaliser, de la nature de l'environnement et des caractéristiques du robot lui même (sa géométrie, cinématique et dynamique).

Ce sujet a suscité de nombreuses études. Toutefois, la difficulté d'un tel problème dépend en grande partie de la nature de l'environnement qui peut être soit ***statique*** c'est à dire qui ne change pas au cours du temps (comme les murs, les meubles, les équipements,...) , soit ***dynamique*** c'est à dire qu'il existe des entités (des obstacles ou d'autres robots), outre que le robot, qui s'y déplacent. Il faut noter, bien évidemment, que la plupart des environnements réels sont dynamiques.

Dans ce chapitre, nous commençons par un rappel sur la formulation standard du problème de planification de mouvement. Nous enchainons avec une vue d'ensemble des grandes approches proposées dans la littérature en adoptant une classification selon le type de l'environnement (statique ou dynamique). Cette présentation sera suivie d'un bilan de synthèse résumant les points forts et les limites de chaque méthode citée.

1.2 Formulation du problème

Un problème de planification de mouvement est défini dans un espace de travail W (scène 2D ou 3D) qui est modélisé par l'espace euclidien $\mathbb{R}^N$, avec $N = 2$ ou 3. Cet espace comporte des entités géométriques de différents types : le robot et les obstacles qui peuvent être fixes ou mobiles.

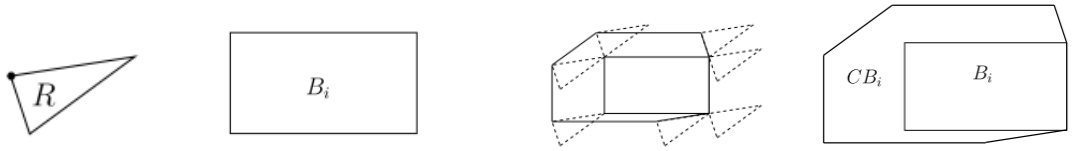
Un robot R représente un système mécanique auquel on associe une représentation cinématique et géométrique. Il est constitué d'un ou de plusieurs solides reliés entre eux par des liaisons cinématiques qui contraignent leur mouvement relatif. L'ensemble de ces liaisons définit la chaîne cinématique du robot ainsi que l'ensemble de ses degrés de libertés, c'est à dire, l'ensemble minimal de paramètres nécessaires pour définir parfaitement son état géométrique. Un robot doit être muni d'un ensemble de capteurs extéroceptifs pour être capable d'avoir un retour sensoriel de l'environnement et d'actionneurs pour se déplacer dans W.

La majorité des travaux de planification de mouvement utilisent la notion de l'espace des configurations introduite au début des années 80 par Lozano-Pérez [Lozano-Pérez, 1983]. En fait, une configuration q détermine l'ensemble des variables permettant de caractériser le robot dans son environnement W. L'ensemble des configurations possibles du robot est appelé espace des configurations C et est décomposé en 2 sous ensembles $C = C_{free} \cup C_{obst}$. Le premier noté C_{free} désigne le sous-espace des configurations libre des obstacles. Le deuxième noté C_{obst} correspond au sous-espace du robot en collision. Ce concept permet de ramener le problème à la planification de mouvement d'un point, représentant la configuration du robot, dans ce nouvel espace.

Une difficulté principale de cette formulation réside dans la représentation du sous-espace C_{obst}. Une formalisation détaillée est présentée dans l'ouvrage de Latombe [Latombe, 1991]. En fait une configuration q du robot R est définie par un vecteur multidimensionnel tel que chaque composante correspond à la valeur d'un de ses degrés de liberté. notons $R(q)$ le sous-ensemble de W occupé par R dans la configuration q. Un obstacle B_i de W est représenté dans l'espace C par un C-obstacle CB_i défini par :

$$CB_i = \{q \in C | R(q) \cap B_i \neq \varnothing\} \tag{1.1}$$

L'union de tous les CB_i de C définit le sous-espace du robot en collision $C_{obst} =$



(a) Robot de forme triangulaire. (b) Obstacle de forme rectangulaire. (c) Robot au contact de l'obstacle. (d) Obstacle grossi CBi.

Figure 1.1 — Calcul du C-obstacle [LaValle, 2006].

$\bigcup_{i=1}^{m} CB_i$ avec m le nombre d'obstacles dans W. Ainsi, le sous-espace libre des obstacles C_{free} se définit par :

$$C_{free} = C \setminus C_{obst} = \{q \in C | R(q) \cap C_{obst} = \varnothing\} \tag{1.2}$$

En s'appuyant sur cette notion de l'espace des configurations C dans laquelle la posture du robot dans son espace de travail devient un point q dans C, le problème de planification de mouvement revient à calculer selon la nature de l'environnement :

1. Un *chemin* représenté comme une fonction continue de $[0, 1]$, qui à toute valeur s associe une configuration $q(s)$ et tel que $s(0) = q_{initiale}$ et $s(1) = q_{finale}$ avec $q_{initiale}$ et q_{finale} sont, respectivement, les configurations initiale et finale du robot :
$$q : [0, 1] \longrightarrow C_{free}$$
$$s \longrightarrow q(s)$$
2. Une *trajectoire* représentée par une fonction continue de $[t_{initial}, t_{final}]$, qui à toute valeur du temps $t \in [t_{initial}, t_{final}]$ associe une configuration $q(t)$:
$$q : [t_{initial}, t_{final}] \longrightarrow C_{free}$$
$$t \longrightarrow q(t)$$

La solution est faisable si elle est libre des collisions et qu'elle respecte les contraintes cinématiques et dynamiques liées au système robotique considéré.

1.3 Approches de planification en environnement statique

La planification de chemin est la formulation la plus simple du problème de planification de mouvement puisque l'environnement dans lequel évolue le robot est statique.

Ainsi, le problème se réduit à l'aspect géométrique de la solution et par conséquent toute courbe qui assure les contraintes de continuité et de la non-collision est acceptable comme solution et le défi revient, en général, à calculer le plus court chemin entre la configuration initiale et la configuration finale.

Plusieurs méthodes ont été développées et en fonction de la connaissance à priori que le robot dispose de son environnement, deux familles d'approches se présentent :

- **Les approches globales (délibératives ou déterministes) :** Le principe est de déterminer une solution complète du problème si elle existe, avant que le robot débute son déplacement, en se basant sur une connaissance aussi complète que possible de l'environnement de travail.
- **Les approches locales (réactives) :** Dans plusieurs applications, l'environnement n'est pas connu à priori. Dans ces conditions, il n'est pas possible de déterminer un mouvement complet jusqu'au but avant que le robot ne commence à se déplacer. La planification correspond alors à gérer dynamiquement les informations relatives à l'environnement. En effet, le principe est de calculer uniquement le mouvement à appliquer au prochain pas temporel et c'est à partir des données capteurs recueillies par le système robotique à chaque instant. Par conséquent, la représentation de l'environnement est construite au fur et à mesure du déplacement du robot.

Ci-dessous, nous présenterons un aperçu de ces différentes approches.

1.3.1 Approches de planification globales

1.3.1.1 Les roadmaps

Cette classe de solutions au problème de planification vise à capturer la topologie de l'espace des configurations libres des obstacles C_{free} afin de simplifier le problème à une recherche d'un plus court chemin dans un graphe connectant la configuration initiale à la configuration finale. Ces méthodes se déroulent, donc, en deux phases : En premier lieu la construction du graphe dans l'espace de recherche. Et ensuite, le parcours de ce graphe dans l'objectif de calculer une solution. Cette deuxième étape s'appuie sur la théorie des graphes en faisant appel aux algorithmes de recherche de plus court chemin comme *Bellman*, *Dijkstra*, A^*. Quant à l'étape de construction du graphe, nous présentons ci-dessous les méthodes les plus répandues.

Les graphes de visibilité Les graphes de visibilité sont proposés par Nilsson en 1969, suite à l'idée que le chemin le plus court pour joindre la configuration finale à partir de l'initiale, passe par les sommets des obstacles polygonaux placés dans l'environnement [Nilsson, 1969]. Ce graphe est construit en reliant avec des droites tous les sommets des obstacles entre eux, sans que ces droites ne coupent d'autres obstacles (voir figure 1.2). En effet, la configuration initiale est reliée par des droites aux sommets visibles des obstacles. Ce processus est réitéré entre les sommets des obstacles et les sommets visibles des obstacles suivants jusqu'à atteindre la configuration finale. Vu que cette méthode autorise le contact entre le robot et les obstacles, elle devient relativement peu employée pour la résolution du problème de planification.

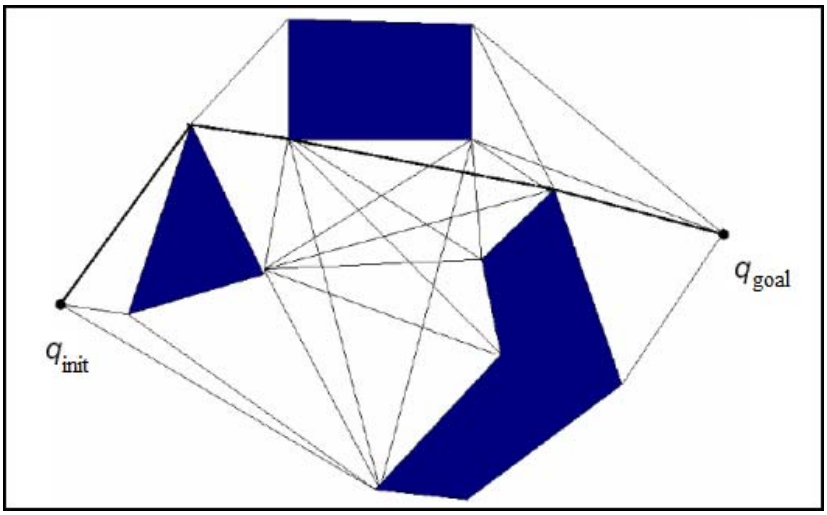


Figure 1.2 — Chemin calculé entre deux configurations q_{init} et q_{goal} à partir d'un graphe de visibilité

Les diagrammes de voronoï Une deuxième méthode pour construire une roadmap est de se servir des diagrammes de Voronoï [Takahashi and Schilling, 1989] qui, contrairement aux graphes de visibilité, évitent le contact avec les obstacles et valident, par conséquence, la distance de sécurité. La construction d'un diagramme de Voronoï se fait en traçant des lignes d'égale distance aux obstacles et au bord de l'environnement exploré. Ensuite le graphe est obtenu en raccordant la configuration initiale au point le plus proche de ce diagramme et de même pour la configuration finale. Cette procédure est illustrée par la figure 1.3.

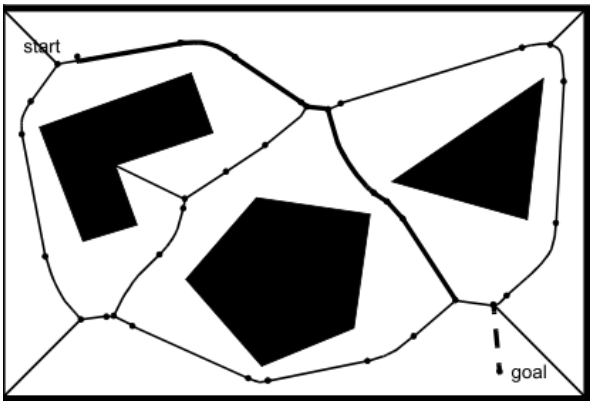
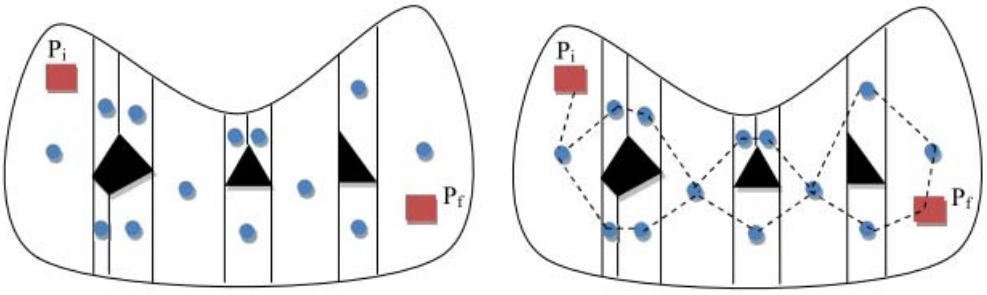


Figure 1.3 — Planification de chemin à partir d'un diagramme de voronoï [Siegwart and Nourbakhsh, 2004]

La décomposition cellulaire Les approches par décomposition cellulaire consistent à partitionner l'espace de configurations libres en un ensemble de cellules adjacentes (régions connexes). Il existe différentes techniques de décomposition de l'espace qui se classifient en deux catégories : les méthodes exactes et les méthodes approchées. Les méthodes exactes effectuent une décomposition qui se base sur un recouvrement exact de l'espace des configurations libres. Quant aux méthodes approchées, elles tentent de rapprocher la structure de l'espace libre avec des cellules qui ont une forme simple comme la forme rectangulaire. Ainsi, si la décomposition est exacte, l'union de ces cellules permet de retrouver l'espace des configurations libres. Un graphe de connexion peut alors être construit à partir des barycentres de ces cellules et les configurations initiale et finale du robot (voir figure 1.4).



(a) Décomposition en cellules. (b) Génération du graphe.

Figure 1.4 — Construction d'une roadmap par la technique de décomposition cellulaire [GUECHI, 2010].

1.3.1.2 Les Rapidly-exploring Random Trees (RRT)

Initialement proposée par Lavalle [Lavalle, 1998], la RRT représente l'une des approches les plus populaires au cours de ces dernières années. Elle consiste à construire un arbre, en partant de la position initiale du robot qui est le noeud racine de l'arbre. L'algorithme RRT explore progressivement l'espace des configurations pour trouver un chemin atteignant la position but. En fait, le principe est le suivant : à partir d'une configuration initiale q_{init}, l'environnement du robot est exploré en choisissant arbitrairement à chaque itération une nouvelle configuration q_{rand} non obstruée. La deuxième étape consiste à déterminer le noeud q_{near} le plus proche de q_{rand} dans l'arbre existant. L'idée ensuite serait d'essayer d'étendre l'arbre à partir de q_{near} dans la direction de q_{rand} d'une longueur ε. Enfin si cette extension réussit, la nouvelle configuration créée q_{new} sera ajoutée dans l'arbre. Ce processus sera réitéré jusqu'à ce que la configuration finale q_{goal} soit atteinte. Ce principe est illustré dans la figure 1.5.

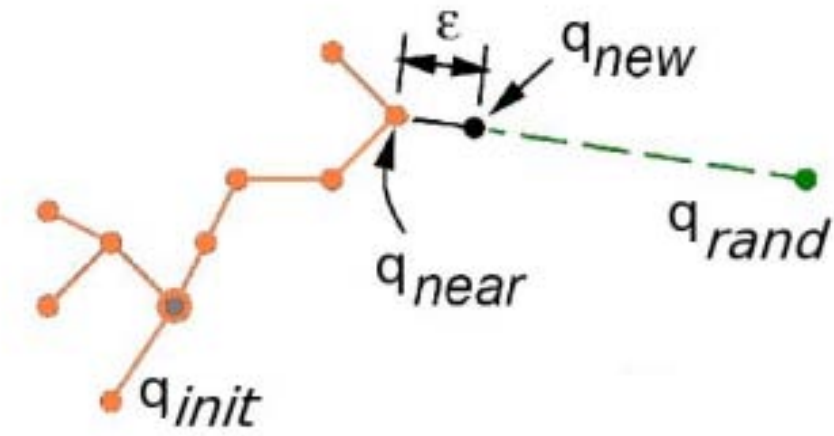


Figure 1.5 — Illustration du principe d'extension du RRT [Flavigné, 2010].

La méthode d'exploration RRT se compose d'une phase de construction d'un arbre couvrant l'espace libre (voir figure 1.6) et d'une phase de requête. La performance de cette méthode vient du fait qu'elle ne nécessite pas de phase de pré-calcul. Une propriété intéressante inhérente de cette approche est que la croissance des arbres est fortement biaisée en faveur des zones inexplorées de l'espace de configuration, l'exploration se fait rapidement. Malgré ces avantages, la RRT comme toute approche peut avoir des lacunes ; elle ne prend pas en compte le coût de la solution lors de la recherche. Ainsi, les chemins qu'elle produise sont parfois loins d'être optimaux.

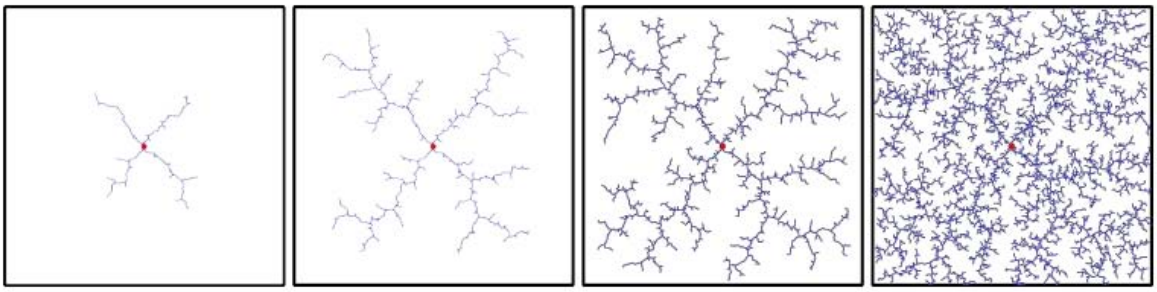


Figure 1.6 — Évolution d'un arbre couvrant l'espace libre par la méthode RRT.

Plusieurs variantes la méthode RRT ont été développées dans l'objectif d'apporter des améliorations relatives à certains aspects ou pour résoudre différents types de problèmes. Nous citons la méthode "RRT bi-directionnel" [LaValle and Kuffner, 1999] dont le principe était de développer deux arbres de diffusion, le premier à partir de la configuration initiale et le deuxième ayant comme racine la configuration finale. Ainsi, la résolution du problème devient plus rapide en probabilité grâce au fait que ces deux arbres s'étendent l'un vers l'autre. Un autre algorithme appelé "RRT connect" [Kuffner and LaValle, 2000] permet aussi une exploration rapide de l'espace des configurations en maximisant l'extension de q_{near} vers q_{rand}. De même pour accélérer l'exploration, dans [Dalibard and Laumond, 2009], un algorithme s'appuyant sur l'analyse en composantes principales de la direction d'expansion de l'arbre est présenté. Ces méthodes probabilistes ont néanmoins des limites. En fait, elles ne permettent pas de détecter l'absence de solution et d'arrêter l'échantillonnage dans ce cas. De plus, il faut noter que dans des situations particulières comme l'existence des passages étroits dans l'environnement, une couverture suffisante de l'espace des configurations pour calculer la solution n'est pas toujours garantie.

1.3.1.3 Les méthodes heuristiques

Les algorithmes génétiques Les algorithmes génétiques (AG) ont été développés dans les années 60 par John Holland. Ils tentent de simuler le processus d'évolution naturelle suivant le modèle darwinien dans un environnement donné et ils ont été appliqués à différents problèmes d'optimisation. Cet algorithme a été largement utilisé pour résoudre les problèmes de planification de mouvement [Tu and Yang, 2003], [Hu et al., 2004], [Cakir, 2015]. Comme illustré à la figure 1.7, son principe consiste

à représenter des solutions aléatoires sous forme d'une population d'individus et de simuler le processus évolutif par la sélection, la reproduction (croisement) et les mutations. Ainsi, les solutions encodées évoluent pour optimiser la fonction de coût. Par conséquent, la population s'améliore et évolue vers une trajectoire quasi optimale. une technique commune pour la planification des trajectoires est d'encoder chaque individu comme une trajectoire complète (ensemble de coordonnées euclidiennes 2D ou 3D).

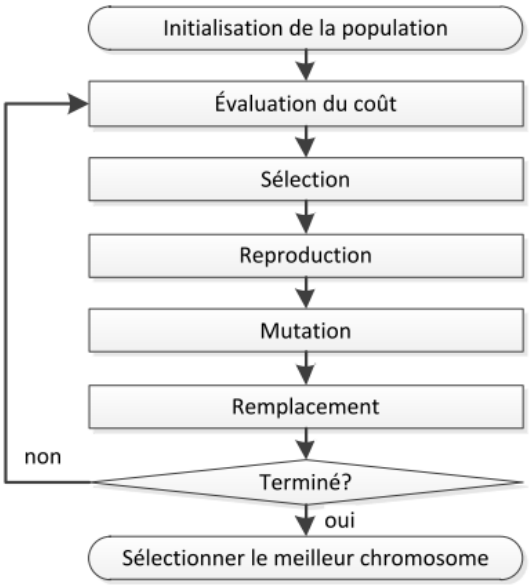


Figure 1.7 — Schéma général d'un algorithme génétique.

Le recuit simulé Le Recuit simulé est un algorithme stochastique développé par Kirkpatrick et al. [Kirkpatrick et al., 1983]. Cette méthode s'inspire du principe physique du recuit des matériaux solides. En fait, le chauffage du matériau à très haute température, puis son refroidissement lent conduisent à une réorganisation plus ordonnée des atomes constituant le matériau, et l'amenant par conséquent vers un état plus stable (énergie plus faible). Le déroulement de l'algorithme est comme suit : une solution possible est encodée dans une particule qui se déplace dans un espace multidimensionnel. Au début du procédé, lorsque la température est élevée, la particule se déplace presque aléatoirement allant même vers une solution avec un plus haut coût afin de favoriser l'exploration. À mesure que le procédé avance et que la température baisse, cet aspect aléatoire diminue et la particule se dirige principalement vers une

solution qui minimise la fonction de coût.

Les avantages de l'algorithme du recuit simulé sont la grande aptitude à éviter les minima locaux et à converger vers le minimum global. En plus, cette technique n'exige pas de condition particulière sur les particularités de la fonction coût. Néanmoins, cet algorithme converge vers une solution approchée et non une solution exacte et aussi la vitesse de convergence est plus faible au voisinage du minimum global [MORETTE, 2009]. Le recuit simulé permet une exploration plus limitée de l'espace de recherche puisqu'il ne se base pas sur l'utilisation d'une population de solutions, contrairement à d'autres algorithmes d'optimisation comme les algorithmes génétiques par exemple. Pour le problème de planification de trajectoires, le recuit simulé a été le plus souvent utilisé pour améliorer ou compléter un autre algorithme (exemple le gradient de potentiel [Masehian and Sedighizadeh, 2007]).

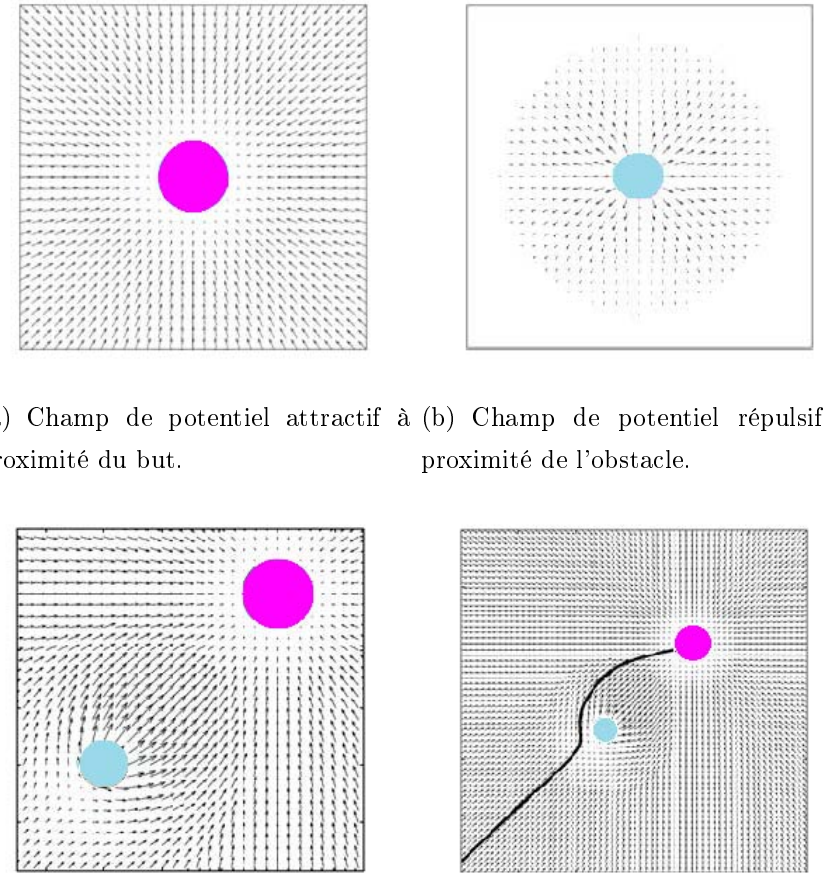
1.3.2 Approches de planification locales

1.3.2.1 Les champs de potentiel

Initialement proposée par Oussama Khatib [Khatib, 1985], l'approche par champs de potentiels a comme idée, la création d'un champ de potentiel artificiel qui guidera les déplacements du robot dans son environnement. Cette approche s'appuie sur le fait qu'elle considère le robot comme une particule plongée dans un champ de potentiel où il est attiré par le but et repoussé par les obstacles $U(q) = U_{att}(q) + U_{rep}(q)$. De ce fait, le modèle de l'environnement est spécifié par une fonction de potentiel qui détermine les forces qui s'exercent dans le système robotique. Le robot calcule d'une manière itérative le mouvement à suivre (Fig. 1.8). Ainsi, une direction vers le but résultant des sommes des différents champs de potentiels sera fixée par la fonction $F(q) = -\nabla U(q)$.

Bien que cette technique se marque par sa simplicité de présentation et de mise en oeuvre et son faible coût de calcul, elle est sujette à plusieurs limites notamment le problème des minima locaux qui provoquent l'immobilité du robot. Par ailleurs, les champs de potentiel peuvent engendrer, dans certaines situations, des mouvements oscillatoires (lors du traversée d'un passage étroit entre les obstacles). De plus, cette technique ne permet pas de détecter un passage entre des obstacles assez proches. Aussi, le robot ne converge pas exactement vers le but si un obstacle se trouve proche de ce dernier. En dépit de ces limites, plusieurs adaptations de cette méthode ont été

présentées. Pour éviter les problèmes liés aux minima locaux, certains proposent de suivre un comportement particulier (suivi des murs ou déplacement aléatoire) dès lors la détection d'une telle situation [Statheros et al., 2006]. D'autres proposent que les champs répulsifs soient calculés non seulement en fonction de la distance aux obstacles mais aussi la vitesse de ces derniers [Huang, 2008].



(a) Champ de potentiel attractif à proximité du but.

(b) Champ de potentiel répulsif à proximité de l'obstacle.

(c) Interactions entre les forces attractives et répulsives.

(d) Chemin final.

Figure 1.8 — Les champs de potentiel [Choset et al., 2005].

1.3.2.2 La fenêtre dynamique

Cette technique, proposée par Dieter Fox et al [Fox et al., 1997], est une méthode d'évitement d'obstacles en temps réel qui travaille dans l'espace des commandes du robot. Cette méthode détermine le domaine des vitesses possibles du robot, c'est à

dire celles qui n'entraînent pas de collisions. Puis, une fois l'espace de recherche est calculé, les commandes envoyées au robot sont le résultat de la maximisation sur ce domaine d'une certaine fonction de coût, qui peut être la minimisation du temps de parcourt, la maximisation de la vitesse ou encore la minimisation de l'énergie dépensée. Ainsi, à partir de la perception locale de l'environnement, cet algorithme permet de sélectionner un couple (v, w) de vitesses de translation et de rotation du robot qui résout les différentes contraintes, dont celle d'éviter les obstacles (Fig. 1.9).

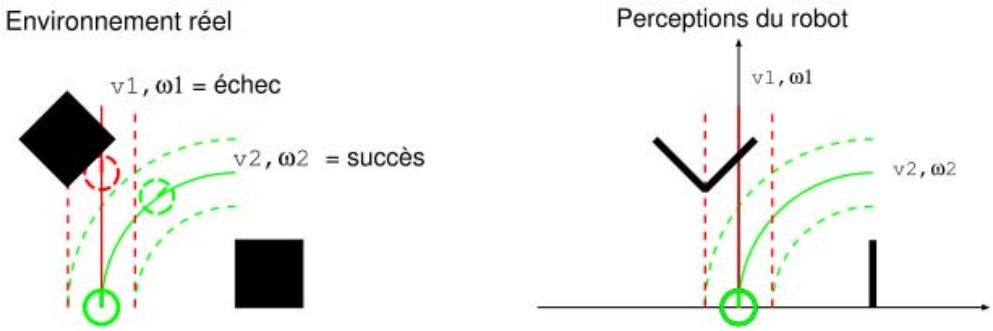


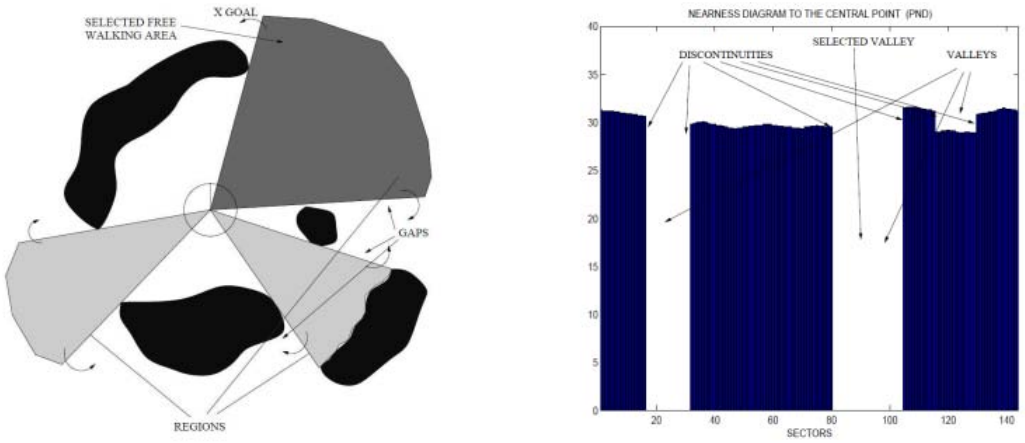
Figure 1.9 — Évitement d'obstacles pour la méthode de la fenêtre dynamique [Filliat, 2013].

Bien que cette méthode permet de prendre en compte les contraintes cinématiques du robot, son implémentation dans un cadre multi-robot s'avère difficile vu le manque de flexibilité. De plus, seule les positions courantes des obstacles étaient prises en compte, mais pas leurs mouvements. Ainsi, la navigation dans un environnement dynamique en utilisant cette approche était fortement compromise. Afin de palier à ce problème, une extension intitulée "Time Varying Dynamic Window" a été proposée dans [Seder and Petrovic, 2007]. Cette dernière calcule, à chaque instant, un ensemble de trajectoires évitant les obstacles puis une vérification de collision à court terme peut être opérée.

1.3.2.3 Les diagrammes de proximité

Proposée par Minguez et Montano [Minguez and Montano, 2000], cette méthode de navigation se base sur un diagramme de proximité des obstacles, actualisé continuellement en fonction du déplacement du robot (Fig. 1.10). Ainsi, la direction du mouvement la plus prometteuse pour atteindre la configuration finale sera sélectionnée selon la distance aux obstacles les plus proches. En effet, le comportement à adopter

peut être soit aller tout droit vers le but, soit passer entre deux obstacles ou bien tout simplement contourner un obstacle. La vitesse est sélectionnée suivant la proximité des obstacles.



(a) Représentation de l'environnement autour du robot.

(b) Caractérisation des vallées disponibles en repérant les discontinuités du graphe de proximité des obstacles.

Figure 1.10 — Navigation par Diagrammes de Proximité [Minguez and Montano, 2000].

Cette approche a été améliorée, en utilisant la technique de 'diviser pour régner', afin de réduire la difficulté du problème de navigation. Au début, la principale limitation de l'approche de navigation par diagrammes de proximité était la portabilité aux différents types de robots, puisqu'elle ne prend pas compte des architectures de forme non circulaire. Par ailleurs, elle ne considère pas les contraintes cinématiques et dynamiques du système robotique étudié. Ensuite, elle a été étendue pour fonctionner sur des robots non-holonomes et avec différentes géométries, tout en incorporant les contraintes cinématiques et dynamiques [Minguez and Montano, 2004], [Minguez and Montano, 2006]. Cependant, la convergence vers le but n'est pas toujours assurée .

1.3.2.4 La logique floue

La logique floue était formalisée en 1965 par Lotfi Zadeh. Elle se base sur la théorie mathématique des ensembles flous qui étend la théorie des ensembles classiques aux ensembles définis de façon imprécise. En effet chaque grandeur physique est converti

en une variable linguistique. Par exemple pour décrire la proximité d'un obstacle, la logique binaire se limite à décrire la situation par "il existe un obstacle ou il n'existe pas ". Alors que, la logique floue permet d'introduire des expressions telles que "l'obstacle est assez près ou l'obstacle est très loin". Ainsi, la planification floue va raisonner sur des variables linguistiques plutôt que physiques. Des règles précises seront appliquées sur ces variables pour générer la commande sous la même forme c'est à dire linguistique et qui sera reconvertie en grandeurs physiques. D'une façon générale, un contrôleur flou se résume en 3 phases comme présenté dans la figure 1.11 : une phase de fuzzification, qui transforme les variables d'entrée en variables floues. Une deuxième phase, qui en se basant sur une base de règles floues et en fonction des variables floues, fournit l'action à accomplir. Et finalement, une phase de défuzzification qui traduit la décision floue issue du moteur d'inférence en une grandeur physique qui sera appliquée au processus étudié.

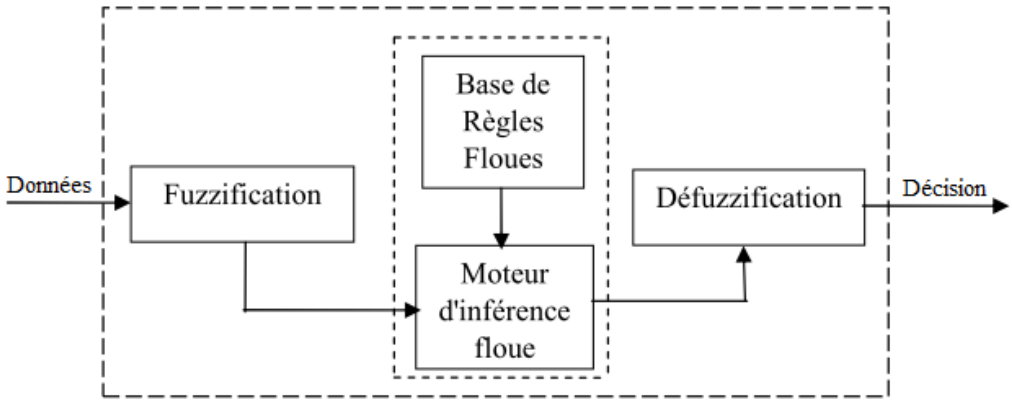


Figure 1.11 — Configuration de base d'un système flou.

Dans l'ensemble, les avantages du contrôle flou dans la conception d'un système de navigation robotique sont : i) la capacité de traitement de l'information incertaine et imprécise, ii) l'exploitation en temps réel, iii) combinaison et coordination faciles des différents comportements, iv) la capacité de développer des stratégies basées sur la notion perception-action, et v) une implémentation aisée. Cependant, les méthodes de navigation floues échouent dans les situations des minima locaux ; elles ont des lacunes de l'auto réglage et organisation et une difficulté de la spécification des règles à partir de la connaissance experte [Hong et al., 2012].

La limitation commune à toutes les méthodes de navigation réactives est qu'elles

ne peuvent pas garantir la convergence globale vers le but, cela est dû au fait qu'elles utilisent une fraction locale de l'information disponible (les informations sensorielles).

1.4 Approches de planification en environnement dynamique

Dans certains problèmes, l'espace de travail peut être dynamique. L'emplacement des obstacles peut changer au fil du temps, ce qui rend le temps un paramètre important dans l'évaluation de la validité du mouvement du robot, contrairement à la planification de chemin. Ainsi, la planification de trajectoire doit respecter les contraintes dynamiques qui découlent de l'environnement (obstacles mobiles) et du système robotique considéré (sa dynamique). Nous distinguons, essentiellement, deux catégories de méthodes de planification de trajectoire : les méthodes s'appuyant sur une trajectoire de référence et celles n'en utilisant pas.

En fait, pour la première catégorie, le planificateur possède une trajectoire de référence calculée dans un modèle de l'environnement en utilisant une méthode globale (sous-section 1.3.1). Celle là est renforcée et mise à jour afin d'assurer un mouvement valide pour le robot. Pour la deuxième catégorie, une méthode de planification locale (sous-section 1.3.2) suffira pour mener le robot à sa position finale. Ci-dessous, nous présentons quelques différentes approches proposées dans la littérature.

1.4.1 Navigation avec trajectoire de référence

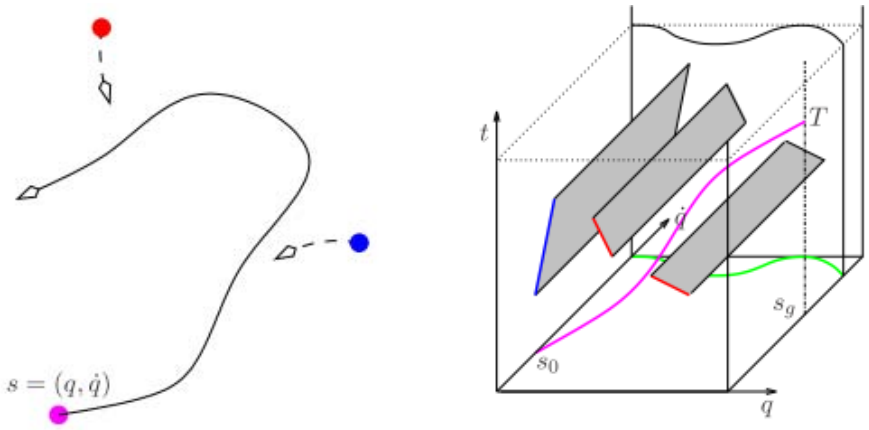
1.4.1.1 Approches basées sur l'espace des états-temps

La notion d'espace des configurations-temps a été introduite par [Erdmann and Lozano-Perez, 1987]. Ce concept a permet de rendre possible la planification d'une trajectoire (courbe de l'espace des configurations-temps) en employant une technique de planification de chemin. En fait, grâce à l'ajout de la dimension temporelle à l'espace des configurations, les contraintes dynamiques de non-collision imposées par les mobiles se transforment en des régions statiques interdites de l'espace des configurations-temps (Fig. 1.12).

Ainsi, plusieurs travaux ont été présentés. Certains proposent des adaptations du

graphe de visibilité [Erdmann and Lozano-Perez, 1987] ou encore des méthodes de décomposition cellulaire [Shih et al., 1990]. Dans la même optique, Fraichard a proposé une méthode basée sur le concept de l'espace des états-temps, en y construisant un graphe dans le but de capturer la topologie de cet espace. Ainsi, le problème de calcul d'une trajectoire solution revient à la recherche d'un plus court chemin dans ce graphe [Fraichard, 1992, Fraichard, 1998].

Toutefois, l'ajout de la dimension temporelle augmente fortement la complexité du processus de planification de mouvement ainsi que le temps de parcours du graphe associé.



(a) Robot ponctuel suivant un chemin donné au milieu d'obstacles mobiles. L'état de ce système est repéré par sa position q et sa vitesse $\dot{q}$.

(b) L'espace des états-temps correspondant $(q, \dot{q}, t)$. La courbe reliant l'état-temps $(s_0, 0)$ à l'état-temps (s_g, T) tout en évitant les régions interdites (traduisant les contraintes imposées par les mobiles (en rouge et en bleu)) est une trajectoire solution.

Figure 1.12 — Exemple d'espace des états-temps. [Fraichard, 2006].

1.4.1.2 Bande élastique de Khatib

Cette méthode a été proposée initialement pour des robots holonomes [Quinlan and Khatib, 1993]. Son principe se base sur l'utilisation d'une trajectoire de référence initialement calculée par un planificateur quelconque et représentée par un ensemble de disques (2D) ou de boules (3D) de l'espace libre. Cette trajectoire est considérée comme un élastique auquel, au cours de l'exécution du mouvement, deux

types de forces virtuelles sont exécutées : des forces répulsives générées par les obstacles et des forces internes d'élasticité ou forces attractives entre bulles consécutives pour maintenir la bande 'tendue'. Ainsi, la taille, la position et même le nombre de ces bulles peuvent s'accroître ou se réduire pour garantir un mouvement sûr jusqu'au but. Le principe de cette méthode est illustré à la figure 1.13. En fait, comme un obstacle se déplace, la position et la taille des disques sont également modifiées (si nécessaire, les bulles sont insérées et supprimées) pour maintenir une trajectoire sans collision.

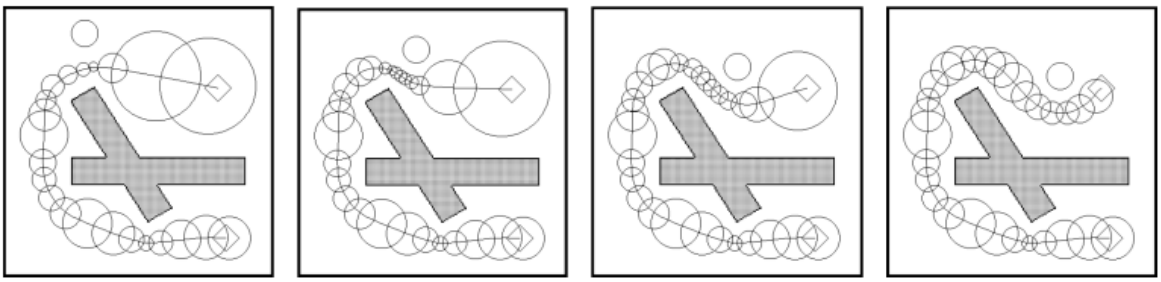


Figure 1.13 — Illustration du principe de la bande élastique [Quinlan and Khatib, 1993].

Plusieurs extensions de la méthode de Bande élastique ont été proposés ultérieurement. Dans [Khatib et al., 1997], Khatib a amélioré cette approche pour être adaptée aux robots non holonomes en utilisant la métrique des trajectoires de Reeds et Shepp. Ainsi, la forme des bulles est donnée par des combinaisons d'arcs de cercle et de lignes droites. Puis, les courbes de Bézier seront employées pour le lissage de la courbe joignant les centres des bulles. La principale limite de cette méthode est que pour certains systèmes non-holonomes complexes, la plus courte distance entre une configuration et les obstacles n'est pas connue.

1.4.1.3 Déformation variationnelle de Lamiraux

Cette méthode suppose l'existence d'une trajectoire sans collision planifiée dans un modèle de l'environnement [Lamiraux et al., 2004]. Ensuite, au cours de l'exécution du mouvement, le robot modifie éventuellement cette trajectoire de référence pour s'écarter des collisions détectées. En fait, l'idée était de perturber les entrées du système de façon à garantir une trajectoire modifiée évitant les obstacles. Pour cela, une fonction de potentiel est définie de façon à ce que le potentiel d'une configuration soit élevé

lorsque le sous-espace occupé par le robot dans cette configuration est proche des obstacles, et qu'il diminue lorsqu'il s'en éloigne. La figure 1.14 illustre la déformation de la trajectoire du robot mobile Hilare 2 tirant une remorque. Chaque trajectoire, le long du processus de déformation, est représentée par une courbe d'abscisse s dans l'espace des configurations, alors que τ représente le temps. Dans cet exemple, le robot est à gauche, et les obstacles (non représentés sur la figure) détectés par un scanner laser entrent en collision avec la trajectoire initiale ($\tau = 0$).

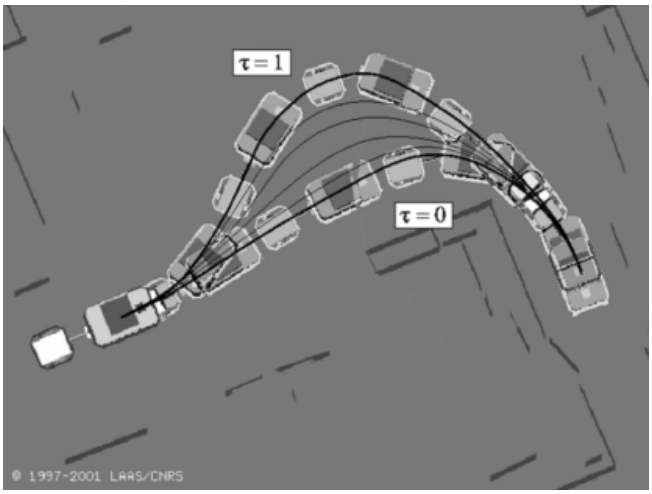


Figure 1.14 — Déformation de la trajectoire du robot mobile Hilare 2 tirant une remorque [Lamiraux et al., 2004].

1.4.1.4 Roadmap élastique

Cette méthode a été proposée pour les robots mobiles de type manipulateurs [Yang and Brock, 2006] . Son principe repose sur la construction d'une roadmap initiale dans l'espace de travail et l'utilisation de l'algorithme A^* pour planifier une trajectoire entre deux configurations initiale et finale. L'étude de la validité d'un mouvement entre deux noeuds de la carte de route est assurée par un contrôleur du système prenant en considération les contraintes cinématiques et dynamiques ainsi que ses contraintes de postures. Lors de l'exécution du mouvement, l'ensemble des noeuds de la roadmap qui se trouvent à proximité des obstacles mobiles sont déplacés en mettant à jour leur connectivité. Si après cette mise à jour certains noeuds restent en collision avec un obstacle, ils seront considérés 'invalides' et en cas où la trajectoire suivie passait par ces derniers, une nouvelle trajectoire sera extraite de la roadmap en respectant les contraintes cinématiques et dynamiques du système robotique considéré.

1.4.1.5 Déformation de trajectoire de Kurniawati et Fraichard

À l'exception de la déformation de Lamiraux où les profils de vitesse et d'accélération peuvent être changés, les approches précédemment cités se caractérisent par le fait que seule la courbe géométrique suivie par le robot est déformée. Néanmoins, les travaux de Lamiraux sont fort coûteux. En dépit des propriétés intéressantes de l'évitement des obstacles et la convergence vers la configuration finale, ces méthodes engendrent des solutions non entièrement satisfaisantes à cause de la non considération de la dimension temporelle d'un environnement dynamique. Ce problème est illustré à la figure 1.15, qui en réponse à l'approche d'un obstacle mobile qui croise le chemin suivi pour le robot, ce dernier est continuellement déformé afin d'éviter cet obstacle ce qui risque de rendre sa faisabilité compromise. En effet, l'étirement du chemin peut rendre son suivi impossible vu les contraintes cinématiques ou celles de type non-holonome (Angle de braquage maximal d'un robot de type voiture par exemple) du robot .

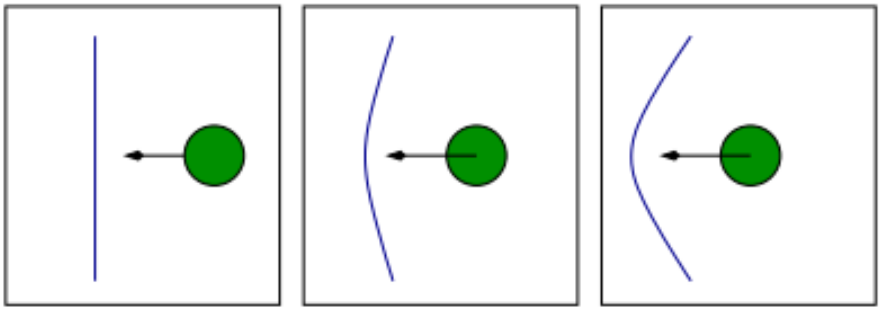


Figure 1.15 — Problème de déformation de chemin [Kurniawati and Fraichard, 2007].

Pour le même exemple, il aurait été judicieux de ralentir à l'approche de l'obstacle et de passer derrière lui (Fig. 1.16). En se basant sur cette observation, Kurniawati et Fraichard ont proposé une approche de déformation de trajectoire prenant compte de la dynamique de l'environnement dans lequel évolue le robot, et ayant la capacité d'adapter la cinématique (vitesse et accélération) de ce dernier en anticipant les mouvements futures des obstacles mobiles [Kurniawati and Fraichard, 2007]. Cependant, cette approche n'était pas en mesure de respecter les contraintes non-holonomes et se limite à traiter le cas d'un simple système de type masse ponctuelle.

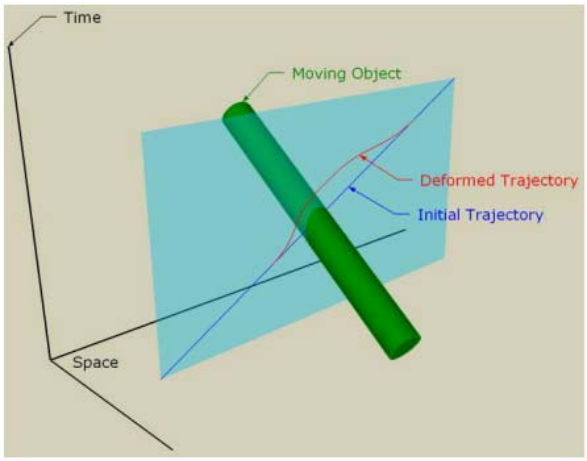


Figure 1.16 — Illustration de la déformation temporelle d'une trajectoire [Kurniawati and Fraichard, 2007].

1.4.1.6 Déformation de trajectoire de Delsart et Fraichard

Comme continuité de l'approche précédente, Delsart et Fraichard ont poursuivi ces travaux afin d'adapter la déformation de trajectoire à des systèmes robotiques sous contraintes cinématiques et dynamiques [Delsart and Fraichard, 2008]. Cette méthode hybride combine une planification globale et un évitement réactif afin d'assurer la sécurité de navigation du robot entre deux configurations initiale et finale (n_0 et n_N).

Ainsi, une trajectoire Γ, définie dans l'espace spatio-temporel $S \times T$ et initialement calculée est fournie sous forme discrétisée ($\Gamma = \{n_0, n_1 ... n_0\}$) en entrée au système robotique où chaque noeud représente un état-temps $n_i = (s_i, t_i)$. Ensuite, au cours de l'exécution du mouvement, un ensemble de forces est exercé sur ces noeuds qui se trouvent, par conséquent, modifiés à la fois dans l'espace d'états du système et dans le temps. Cela est réalisé en se basant sur un modèle prévisionnel du comportement futur des obstacles, déterminé à partir des entrées capteurs et fourni au processus de déformation à chaque pas du temps. Les forces appliquées sur les noeuds sont de deux natures : des forces répulsives (externes) provenant des obstacles et permettant d'assurer la non-collision et des forces élastiques (internes) permettant de tendre la trajectoire et, par conséquent, de minimiser les oscillations.

Le principe général de cette approche est illustré à la figure 1.17. Cet exemple montre un système robotique évoluant dans un environnement obstrué de trois obstacles circulaires B_1, B_2 et B_3. La dimension temporelle étant représentée sur l'axe vertical et

le plan (xy) correspond au présent. Le modèle prévisionnel des mouvements futurs des obstacles supposent qu'ils conserveront leur direction et leur vitesse dans le futur. Les forces externes (F_{ext}) et internes (F_{int}) sont représentées par des vecteurs sur chaque noeud de la trajectoire Γ_k.

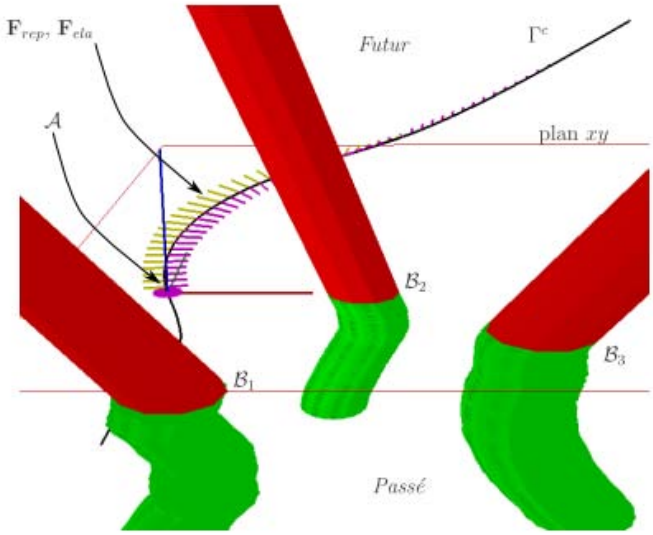


Figure 1.17 — Illustration de la déformation de trajectoire de Delsart et Fraichard [Delsart and Fraichard, 2008].

Vu la nature heuristique de cette approche, la trajectoire déformée n'est pas garantie d'être à la fois sans collision et connectée à chaque pas de temps. Cet échec se présente généralement dans le cas ou la topologie de $S \times T$ change fortement. Comme , par exemple le cas d'une porte qui se ferme ou, autrement dit, lorsque un passage qui était initialement libre se trouve bloqué. Pour cela, les contraintes de la non collision et de la connectivité sont testées après chaque déformation et le processus de déformation de mouvement sera arrêté si l'une de ces contraintes est violée. Dans ce cas, une nouvelle trajectoire doit être déterminée par le planificateur global.

1.4.2 Navigation sans trajectoire de référence

Dans certains cadres applicatifs et pour certains systèmes robotiques, le calcul préalable d'une trajectoire de référence s'avère difficile, voire impossible. Dans un cas pareil de navigation sans trajectoire de référence, l'utilisation d'une méthode d'évitement réactif d'obstacles s'impose.

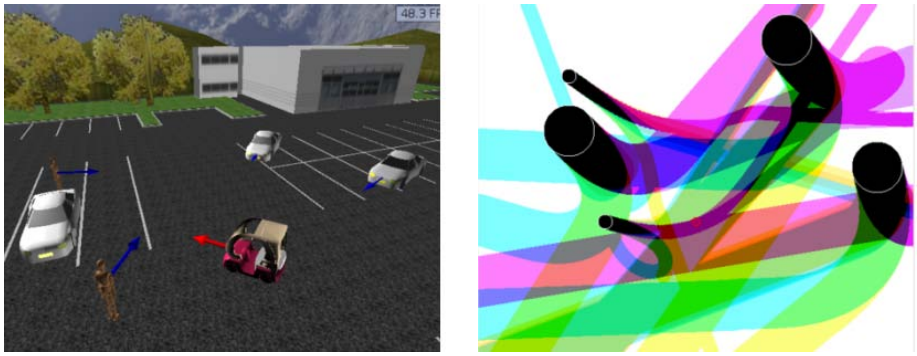
Il est à noter que les méthodes d'évitement réactif d'obstacles sans trajectoire de référence ne sont autre que les méthodes locales de planification (sous-section 1.3.2). Ces méthodes incluent les approches classiques (Les champs de potentiel, la fenêtre dynamique, les diagrammes de proximité,...), les approches évolutionnaires (les algorithmes génétiques, le recuit simulé, les colonies de fourmis,...), les approches basées sur des systèmes d'inférences flous (la logique floue [Faisal et al., 2013], [Chang and Jin, 2013]), etc...

Quoique les approches conventionnelles (classiques) disposent d'une complexité suffisamment faible pour être utilisé en temps réel au cours de l'évolution du robot dans son environnement de travail, cet avantage n'assure cependant en aucun cas ni la convergence vers le but ni la sécurité du système robotique d'où l'apparition des méthodes de planification combinant à la fois les approches classiques et évolutionnaires et également l'apparition des méthodes traitant le problème de sûreté de navigation. Dans ce contexte, nous présenterons, ci-dessous, quelques algorithmes présentés dans la littérature.

1.4.2.1 Méthode de navigation basée sur les états de collisions inévitables

Cette méthode vise à assurer la sécurité du mouvement d'un système robotique évoluant dans un environnement dynamique. La notion d'états de collisions inévitables, proposée par Fraichard, a été fortement développée dans [Fraichard and Hajime, 2003], [Fraichard and Parthasarathi, 2007], [Fraichard and Martinez-Gomez, 2008], [Fraichard and Martinez-Gomez, 2009]. En effet, un état de collision inévitable (ICS) est défini comme étant un état pour lequel, quelle que soit la trajectoire future suivie par le robot, celui-ci rentre en état de collision avec un obstacle.

Ainsi, pour garantir la sûreté du système robotique considéré et celle de son environnement (les autres agents évoluant dans son environnement), il est impératif d'éviter non seulement les états de collision (les CS) mais aussi les états engendrant inévitablement une collision (les ICS). Ce problème est traité en deux phases. La première est consacrée à caractériser les ICS (Fig. 1.18). La deuxième à la création d'un système de navigation permettant d'éviter telles états. Bien que cette méthode permette d'assurer un niveau de sûreté élevée, elle reste fortement dépendante de la fiabilité du modèle prévisionnel du comportement futur des obstacles mobiles considérés.



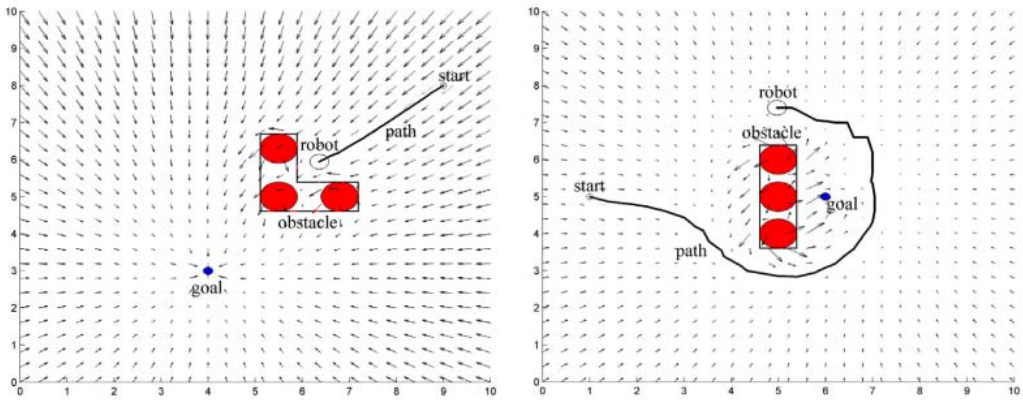
(a) Environnement peuplé d'obstacles fixes et mobiles.

(b) Les états de collisions inévitables correspondants (zones en noir).

Figure 1.18 — Navigation basée sur les états de collisions inévitables [Fraichard, 2013].

1.4.2.2 Champs de potentiel et Recuit simulé

Les travaux de Zhang et al. proposent une approche qui combine les champs de potentiel avec le recuit simulé [Zhang et al., 2004]. Cette méthode traite le problème de la non convergence vers le but, lorsque ce dernier est situé près des obstacles (Goal Non-Reachable with Obstacles Nearby (GNRON)), et le problème des minima locaux (Fig. 1.19). Ainsi, de nouveaux fonctions de potentiel ont été dérivées en considérant l'information de distance des positions initiale et finale pour le problème du GNRON.



(a) Problème du minimum local.

(b) Problème du GNRON.

Figure 1.19 — Limites des méthodes conventionnelles (Exemple : les champs de potentiel [Montiel et al., 2015]).

1.4.2.3 Théorie des écoulements à potentiel de vitesse et Optimisation par essaims particulaires

Hu et al. ont présenté une approche pour diriger un robot dans son environnement, en se basant sur l'optimisation par essaims particulaires (PSO) et la théorie des écoulements à potentiel de vitesse [Hu et al., 2007]. Cette théorie est employée pour guider le système robotique pour échapper aux obstacles. L'optimisation est assurée par l'utilisation du PSO. Bien que cette approche résout le problème des minima locaux, elle ne considère pas les changements instantanés des vitesses d'obstacles.

1.4.2.4 Champs de potentiel et Algorithme d'évolution bactérienne

Montiel et al. ont présenté récemment un nouvel algorithme de navigation robotique, appelé 'Bacterial Potential Field', en utilisant la méthodes des champs de potentiel renforcée par un algorithme d'évolution bactérienne (Bacterial Evolutionary Algorithm (BEA)) [Montiel et al., 2015]. l'utilisation de ce dernier a réduit fortement les limites de l'utilisation des champs de potentiel . Ainsi, l'apport principal de l'algorithme proposé est l'utilisation du BEA, qui a permis de trouver à un faible temps de calcul les valeurs optimales correspondants aux forces attractives et répulsives. Cependant, les contraintes cinématiques et dynamiques du système robotique ne sont pas prises en compte.

1.5 Bilan

Une grande variété de méthodes de planification de mouvement a été proposée dans la littérature. Dans ce chapitre, nous avons présenté un état de l'art non exhaustif de ces méthodes en les classifiant selon la nature de l'environnement considéré (voir Fig. 1.20). Selon la manière d'apercevoir l'environnement, ces méthodes se classifient en deux grandes familles : les approches **globales** qui raisonnent sur la totalité de l'environnement et les approches **locales** qui s'appuient seulement sur le voisinage de l'emplacement courant du robot.

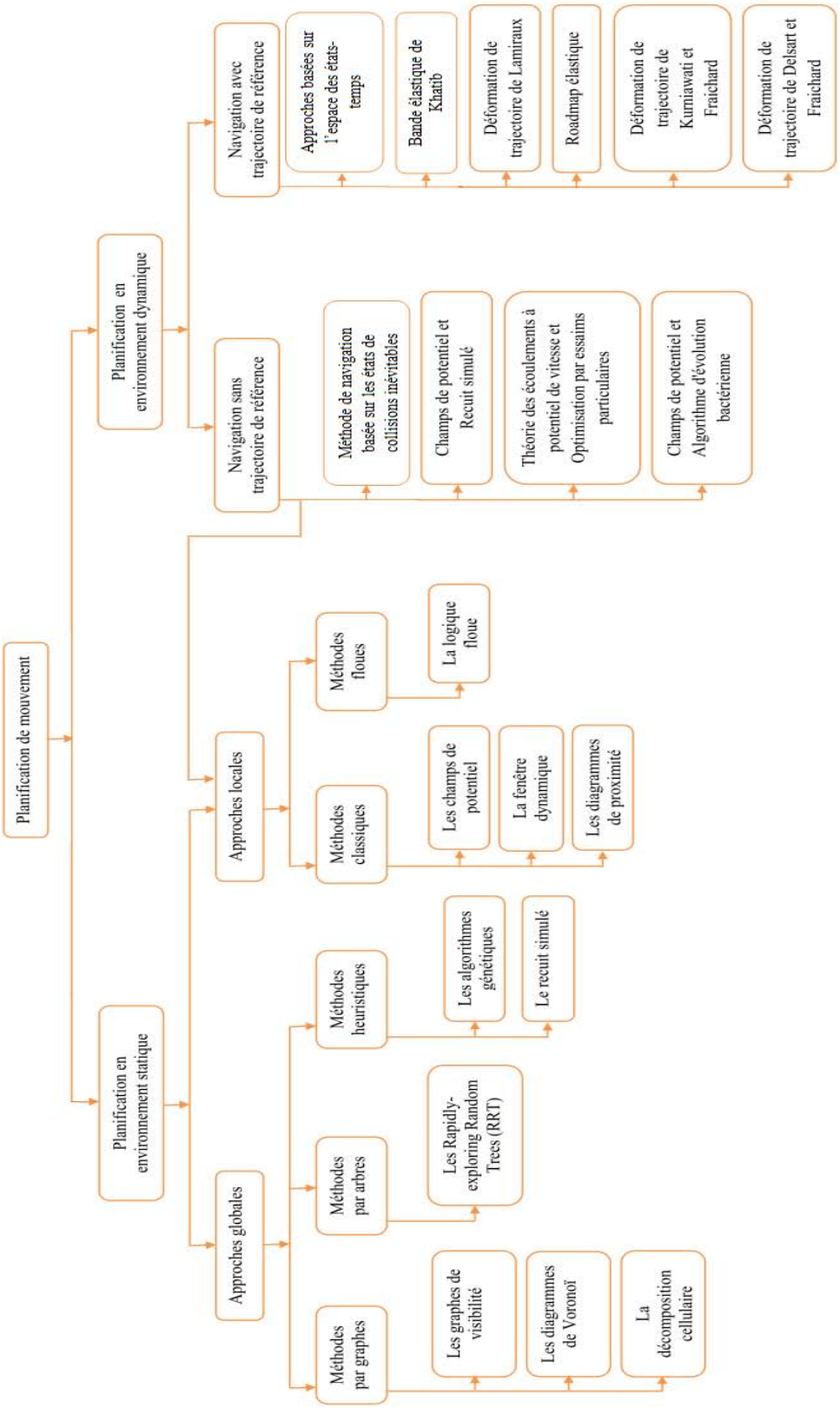


Figure 1.20 — Classification des méthodes de planification de mouvement présentées dans ce chapitre.

En environnement statique, les méthodes avec approche globale sont dites aussi des méthodes '**délibératives**' puisqu'elles permettent de déterminer l'existence ou non d'une solution complète avant que le robot commence son mouvement. Ces méthodes se classifient, à leurs tours, en :

- Des méthodes **classiques**, qui s'appuient sur la capture de la topologie de l'environnement. Nous distinguons essentiellement : celles qui la synthétise dans un graphe comme les graphes de visibilité, les diagrammes de voronoi et la décomposition cellulaire (exacte [Barbehenn and Hutchinson, 1995], approximative [Conte and Zullil, 1995] ou probabiliste [Rosell and Iniguez, 2005]) et celles qui synthétisent le modèle de l'environnement en un arbre ; soit par échantillonnage (les méthodes de construction des réseaux probabilistes, appelées aussi PRM [Kavraki et al., 1996]) ou les méthodes de diffusion incrémentale (RRT).
- Des méthodes **heuristiques**, dites encore '**intelligentes**', qui se sont révélées être effectives notamment en temps de calcul. Cette catégorie de solutions inclue les algorithmes génétiques, le recuit simulé, les réseaux de neurones,...

Quant aux méthodes avec approche locale, qui sont appelées aussi méthodes '**réactives**', vue qu'elles réagissent pour décider du mouvement du robot à effectuer selon les informations du voisinage (vitesses, orientations et distances par rapport aux obstacles présents), elles se classifient aussi en méthodes classiques (les champs de potentiel, la fenêtre dynamique, les diagrammes de proximité,...) et intelligentes (la logique floue par exemple).

Dans un contexte dynamique, ces méthodes de planification locales restent toujours applicables et s'inscrivent sous la famille des méthodes de navigation sans trajectoire de référence qui incluent, en plus, d'autres méthodes qui combinent à la fois les méthodes classiques et intelligentes (les méthodes citées dans sous-section 1.4.2 par exemple) dans l'objectif d'améliorer et de corriger les insuffisances des méthodes conventionnelles.

La navigation avec trajectoire de référence représente la deuxième famille de planification de mouvement en environnement dynamique. Ces méthodes s'appuient sur une des approches globales pour calculer une trajectoire initiale qui sera modifiée au cours de l'exécution du mouvement afin d'éviter les obstacles mobiles. L'évitement d'obstacles peut être également assuré par une méthode avec approche locale. Il est à noter que les méthodes de navigation avec trajectoire de référence garantissent au mieux la

convergence vers le but.

Pour l'évaluation des performances d'un algorithme de planification, plusieurs mesures ont été proposées. Parmi lesquelles, nous distinguons la propriété de complétude. Pour cela trois variantes ont été définies :

Définition 1. Une méthode est dite ***complète*** lorsqu'elle détermine la solution si elle existe, renvoie échec sinon.

Définition 2. Une méthode est dite ***complète en résolution*** lorsqu'elle détermine, à une résolution donnée, la solution si elle existe, renvoie échec sinon. Dans ce dernier cas, il est possible qu'il existe une solution à une résolution plus fine que celle choisie.

Définition 3. Une méthode est dite ***complète en probabilité*** lorsque la probabilité de déterminer l'existence d'une solution tend vers 1 si le nombre d'échantillons générés tend vers l'infini.

Comme le montre le tableau 1.1, la mesure de complétude d'une méthode de planification dépend de la nature de celle ci.

Tableau 1.1 — Complétude des méthodes de planification de mouvement

Nature de la méthode	***Mesure de complétude***
Méthode exacte	Complète
Méthode approximative	Complète en résolution
Méthode probabiliste	Complète en probabilité
Méthode heuristique	Incomplète

Chacune des méthodes citées dans ce chapitre a ses propres avantages et ses propres limites. Nous constatons qu'elles sont des adaptations et/ou des combinaisons des grandes familles de planification résumées dans le tableau 1.2. Aucun algorithme n'est entièrement meilleur que l'autre, sa performance et son apport dépendent de l'application. De plus, dans certains cas d'études, une seule méthode de planification ne suffit pas et il serait plus avantageux d'enchainer différentes techniques de planifications.

Tableau 1.2 — Principales méthodes de planification de mouvement : Avantages et inconvénients

Méthode	**Avantages**	**Inconvénients**
– Les graphes de visibilité – Les diagrammes de Voronoi – Décomposition cellulaire	– Complètes ou Complètes en résolution / probabilité – Des chemins de longueurs minimales	– Temps de calcul élevé
– RRT	– Complètes en probabilité – Facilité de mise en place	– Solutions non optimales – Couverture insuffisante des passages étroits
– Les méthodes heuristiques	– Meilleure exploration de l'espace de recherche – Facilité de mise en place – Efficacité de calcul	– Incomplètes
– Les méthodes floues	– Capacité de traitement de l'information incertaine et imprécise – Temps réel – Implémentation aisé	– Incomplètes – Les minima locaux – Difficulté de la spécification des règles à partir de la connaissance experte
– Les champs de Potentiel – La fenêtre dynamique – Les diagrammes de proximité	– Temps réel – Facilité de mise en place – temps de calcul réduit	– Incomplètes – Solutions non optimales – minima locaux

De plus, nous notons que d'après plusieurs articles de synthèse ([Tang et al., 2012], [A. and Powers, 2013], [Leena and Saju, 2014]) le pourcentage d'utilisation des méthodes heuristiques a augmenté considérablement face à l'utilisation des méthodes conventionnelles (Fig. 1.21). Toutefois, il serait plus judicieux de combiner les deux à la fois pour un résultat meilleur.

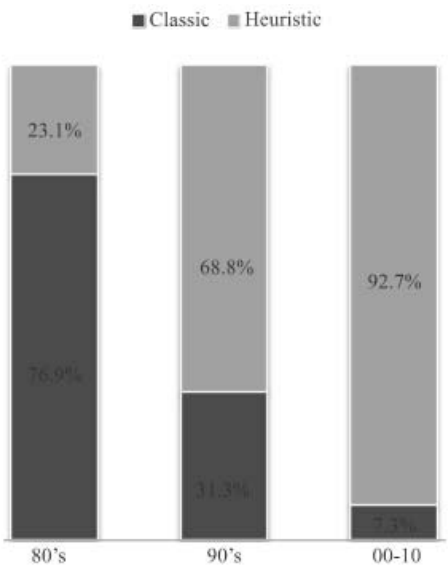


Figure 1.21 — Comparaison de l'utilisation des méthodes conventionnelles vs les méthodes heuristiques [Tang et al., 2012].

1.6 Conclusion

Nous avons présenté dans ce chapitre un état de l'art sur la planification de mouvement pour les robots mobiles. Nous avons également présenté les avantages et les limites de chaque méthode et nous avons constaté qu'aucune approche de planification qui définie une solution qui élimine tous les problèmes n'a été rencontrée. Ainsi, nous apercevons que la planification de mouvement est un domaine de recherche très actif et que de nouvelles méthodes apparaissent régulièrement.

Cette revue de la littérature révèle l'importance de l'utilisation des méthodes heuristiques pour formuler le problème de calcul de mouvement comme un problème d'optimisation dans l'intérêt de satisfaire plusieurs contraintes à la fois. Dans ce travail, nous souhaitons améliorer les performances de la planification automatique du mouvement en proposant une nouvelle méthode de déformation de trajectoire. La génération de cette trajectoire de référence fait partie de notre planificateur. Ainsi, nous présentons dans le chapitre 3 une nouvelle approche de planification de chemin pour un robot mobile non holonome dans un environnement statique. Avant d'introduire cette approche, nous passons dans le chapitre suivant à présenter une synthèse sur les méthodes de génération des chemins lisses utilisés pour la navigation robotique.

CHAPITRE 2

Méthodes de génération des chemins lisses

2.1 Introduction

La planification de chemins pour un robot mobile non-holonome (de type voiture par exemple) a donné lieu à de nombreux travaux. Les premiers consistent à générer des chemins constitués de segments de droite reliant tangentiellement des arcs de cercle de courbure maximale. Ces chemins sont en effet les plus courts, dans des configurations où le robot se déplace uniquement en marche avant (cela a été démontré par Dubins en 1957) ou qu'il fasse des manoeuvres (Reeds et Shepp, 1990).

Toutefois, plusieurs recherches portant sur le contrôle des robots mobiles ont mis en avant l'importance de la continuité de courbure pour obtenir des chemins dont le suivi soit précis. Or, les chemins déjà cités ne vérifient pas cette propriété.

De ce fait, des solutions ont été proposées en utilisant les clothoïdes, alors que d'autres ont opté pour l'introduction des techniques d'interpolation et d'approximation pour modéliser géométriquement les chemins pour les robots mobiles. Parmi ces primitives, nous citons les splines cubiques qui ont été les premières méthodes à être mise au point. Ensuite, les courbes de Bézier ont été introduites, adoptant une conception différente et plus souple. L'évolution s'est poursuivie avec les B-Splines, généralisation des courbes de Bézier, puis avec les NURBS dans les années 1980.

Dans ce chapitre, nous présentons un aperçu sur les différentes techniques de lissage des chemins utilisées en navigation robotique. Nous commençons par présenter les chemins de Dubins. Nous présentons, ensuite, les chemins de Reeds et Shepp. Nous présentons aussi les principales courbes à courbure continue utilisées pour la modélisation

des chemins pour les robots mobiles non holonomes. Pour ce faire, nous introduisons en premier lieu celles se basant sur les clothoïdes. Puis, nous décrivons les splines cubiques, les courbes de Bézier, les courbes B-splines et finalement les courbes NURBS. Nous présentons dans les dernières sections de ce chapitre, un bilan de synthèse des méthodes déjà citées qui résume leurs points forts ainsi que leurs limites.

2.2 Les chemins de Dubins

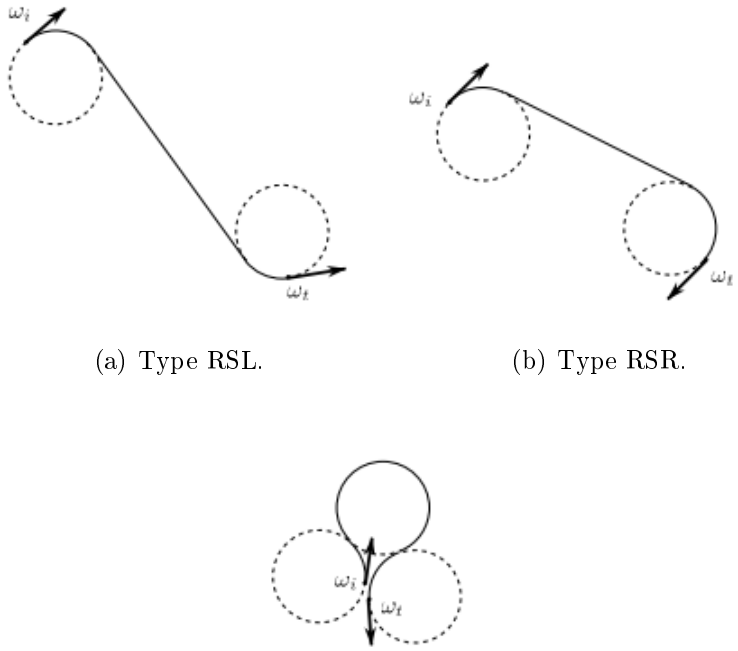
Lester E. Dubins a montré, qu'en absence d'obstacles, un chemin optimal entre deux configurations est une courbe C^1, C^2 par morceaux. Cette courbe est formée d'arcs de cercle reliés tangentiellement par des segments de droites et peut s'écrire sous la forme CSC ou CCC avec C est un arc de cercle de rayon minimal, c'est-à-dire de rayon égale à 1 et S est un segment de droite [Dubins, 1957]. Il est à noter aussi que chaque arc de segment ou de cercle peut être dégénéré c'est-à-dire de longueur nulle.

Le type d'un chemin peut être spécifié de façon plus précise en indiquant son sens de parcours (Figure 2.1). En effet, on note par R (respectivement L), un arc de cercle parcouru dans le sens direct ou sens des aiguilles d'une montre (respectivement sens trigonométrique).

Ainsi, il existe au plus six chemins de Dubins entre deux configurations :

- De type CCC contenant les deux chemins de types LR_uL et RL_uR ou u représente la longueur de l'arc intermédiaire, vérifiant $u > \pi$.
- De type CSC contenant les quatre chemins de types LSL, LSR, RSL et RSR.

La prise en compte de l'existence des obstacles dans le processus de planification de chemins sans manoeuvre a été étudié pour la première fois par Laumond [Laumond, 1987], en considérant un robot de forme circulaire évoluant dans un environnement décrit par des polygones généralisés. En effet, l'algorithme présenté se base sur le calcul de l'espace des centres de girations du robot et la détermination des solutions à partir de la juxtaposition d'arcs de cercle de rayon minimum et de segments de droite.



(a) Type RSL. (b) Type RSR.

(c) Type LRL.

Figure 2.1 — Exemples de plus courts chemins de Dubins.

2.3 Les chemins de Reeds et Shepp

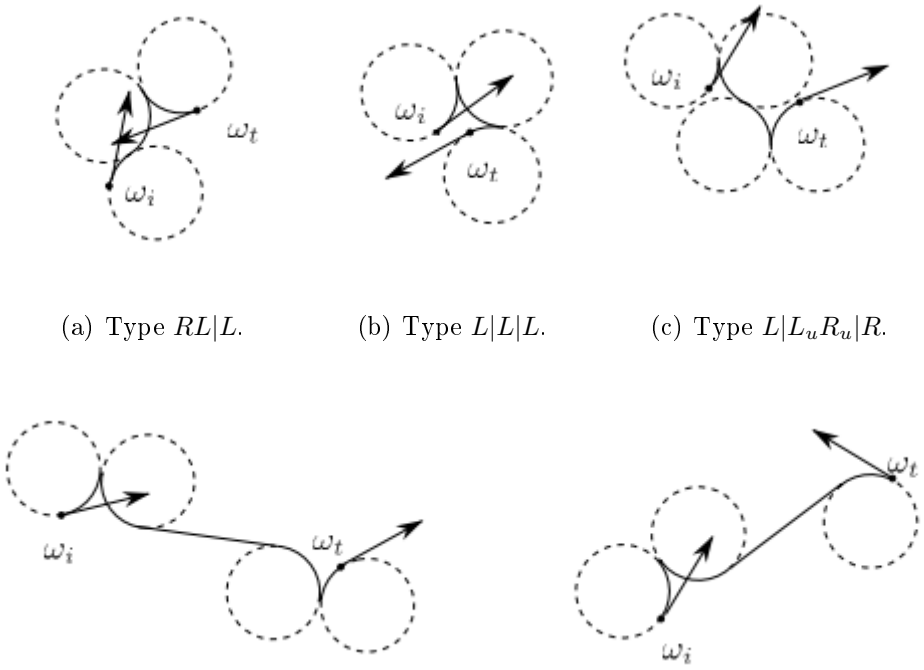
Contrairement aux chemins de Dubins, les chemins de Reeds et Shepp [Reeds and Shepp, 1990] présentent des chemins de longueurs minimales qui **autorisent les manoeuvres**, c'est-à-dire qu'elles contiennent des points de rebroussement où le robot change de la marche avant à la marche arrière. En effet, un plus court chemin est constitué d'au plus cinq arcs, qui sont des segments de droite ou des arcs de cercle unitaire, et inclut au plus deux points de rebroussement. Ces derniers sont notés par le symbole "|" dans le mot représentant le chemin. Ainsi, le type d'un plus court chemin de Reeds et Shepp appartient à l'une des familles suivantes :

- CSC.
- $C|C|C$.
- CCC ou $C|CC$.
- CC_uC_uC ou $C|C_uC_u|C$ avec $u < \pi/2$.

– $C|C_{\pi/2}SC$ ou $CSC_{\pi/2}C$.

– $C|C_{\pi/2}SC_{\pi/2}|C$.

De même que les courbes de Dubins, à chaque arc de cercle est associé un sens de déplacement. Notons que nous ne pouvons pas avoir ni $L|R$ ni $R|L$. La figure 2.2 montre différents exemples de chemins de Reeds et Shepp.



(a) Type $RL|L$. (b) Type $L|L|L$. (c) Type $L|L_uR_u|R$.

(d) Type $L|L_{\pi/2}SR_{\pi/2}|R$. (e) Type $L|L_{\pi/2}SR$.

Figure 2.2 — Exemples de plus courts chemins de Reeds et Shepp.

Les méthodes de calcul de courbes de Dubins et ceux de Reeds et Shepp n'impliquent pas la continuité de courbure. En effet, aux raccordements, la valeur de courbure passe brutalement de $1/R$ pour un arc de cercle de rayon R à zéro pour un segment de droite comme illustré dans la figure 2.3.

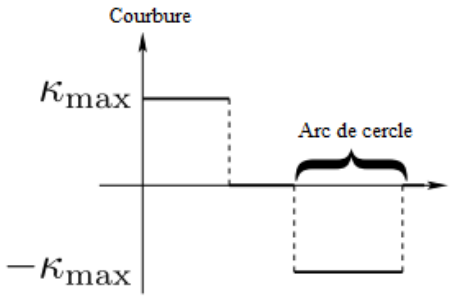


Figure 2.3 — Profil de courbure discontinue d'un chemin de Reeds et Shepp.

2.4 Courbes à courbure continue

2.4.1 Courbes à base de clothoïdes

Afin d'assurer la contrainte de continuité de courbure, qui n'a pas été vérifiée précédemment, les arcs de cercle peuvent être remplacés par des arcs de spirale. Un type de spirale qui est souvent utilisé est la clothoïde. Par définition, une clothoïde ou spirale de Cornu est une courbe dont la courbure est une fonction linéaire de l'abscisse curviligne. La figure 2.4 illustre la forme générale d'une clothoïde.

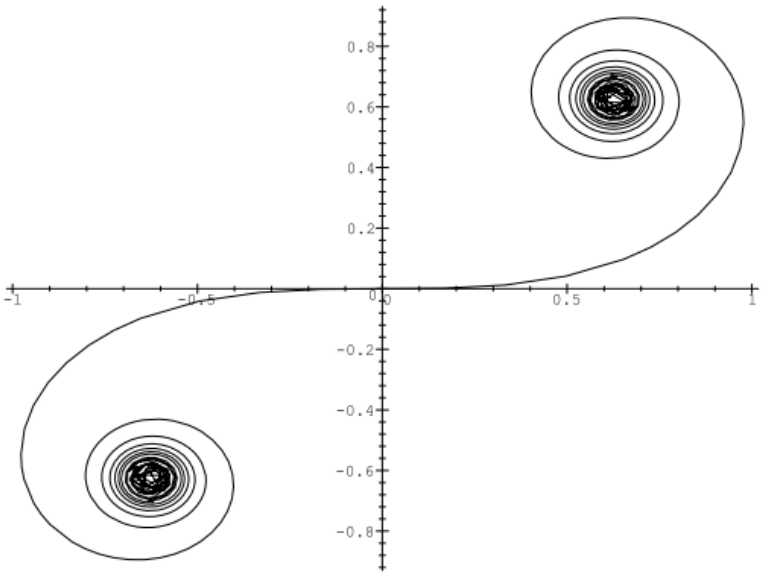


Figure 2.4 — Un exemple de clothoïde.

Parmi les travaux utilisant les clothoïdes, nous citons ceux de Scheuer qui était le premier à avoir intégré, dans ses travaux de thèse, la contrainte de continuité de la courbure dans le problème de planification de chemin pour robot mobile non-holonome [Scheuer, 1998]. Sa méthode consiste à générer des chemins formés par des segments de droite, des arcs de cercle et des arcs de clothoïdes.

Bien que l'intégration des clothoïdes permet d'assurer la continuité de courbure des chemins générés, ces courbes s'expriment à l'aide des intégrales qui sont compliquées à calculer ce qui limite, par conséquent, leurs utilisations dans les applications réelles.

2.4.2 les splines cubiques

L'interpolation par splines a été développée par l'industrie automobile, chez General Motors aux alentours de 1950. L'interpolation par des splines cubiques consiste à trouver pour chaque intervalle $[x_k, x_{k+1}]$ un polynôme $S_k(x)$ du troisième degré de sorte que la fonction d'interpolation résultante sur $[x_0, x_n]$ ainsi que sa première et deuxième dérivée soient continues (Fig. 2.5) :

$$S_k(x) = a_k x^3 + b_k x^2 + c_k x + d_k, \quad x \in [x_k, x_{k+1}] \tag{2.1}$$

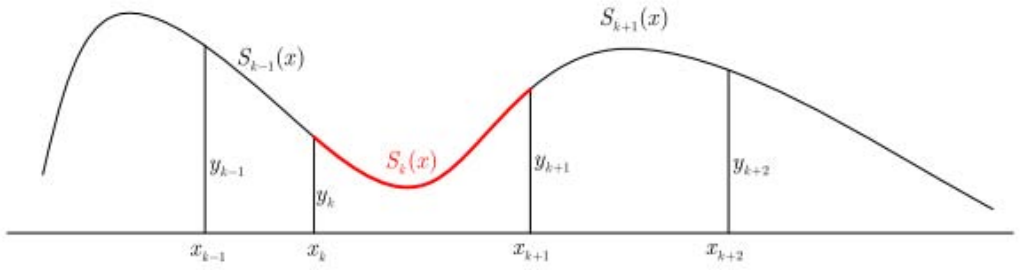


Figure 2.5 — Schéma d'interpolation cubique entre P_k et P_{k+1}.

Comme nous le voyons sur la figure 2.6, les fonctions splines ont l'avantage d'offrir une certaine 'souplesse' (pas de cassure du rayon de courbure) au niveau du tracé de la fonction d'interpolation. Cependant, tous les points sont liés et la modification de l'emplacement d'un de ceux-ci influe tout le système d'équations comme nous pouvons le constater dans les exemples (a) et (b) de la même figure.

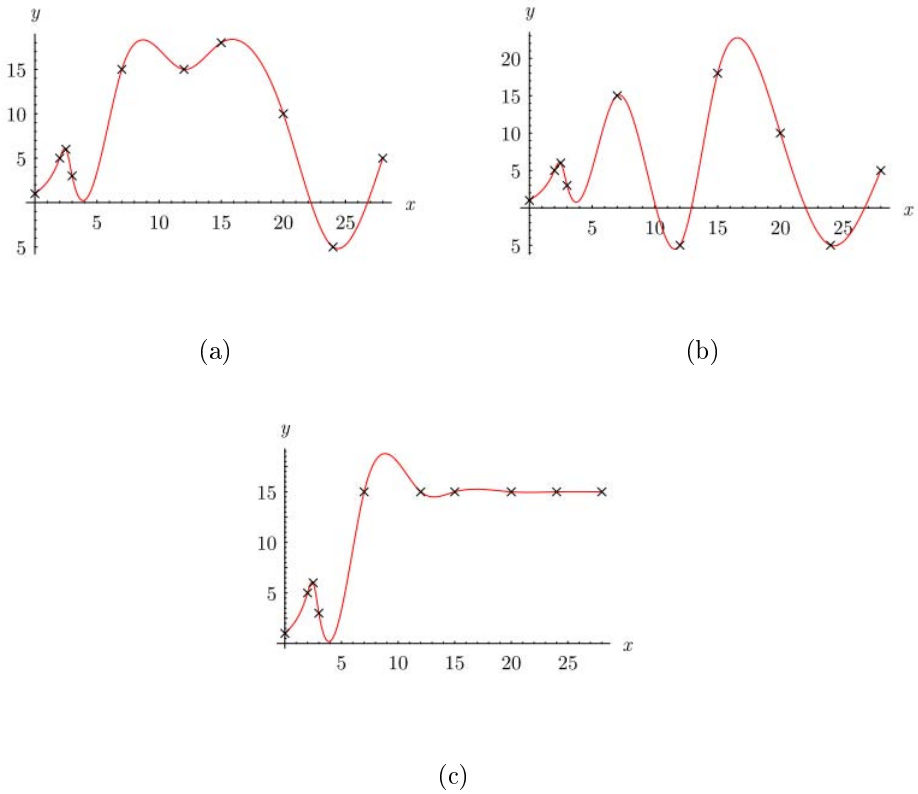


Figure 2.6 — Résultats d'interpolation par splines cubiques.

2.4.3 Les courbes de Béziers

C'est dans le domaine de la CAO que les courbes de Bézier, courbes polynomiales paramétriques, ont été inventées et plus précisément dans l'industrie automobile. En effet, elles étaient inventées dans le but de mieux contrôler la conception (CAO) d'automobile chez la compagnie Renault par Pierre Bézier. L'idée révolutionnaire des courbes de Béziers est l'utilisation de points de contrôle et non de points d'interpolation. Cela veut dire que la courbe ne passe pas par les points donnés mais les approche. Pour $n+1$ points de contrôle $(P_0, \ldots, P_n)$, une courbe de Bézier de degré n est définit par :

$$C(u) = \sum_{i=0}^{n} B_i, n(u) P_i \qquad 0 \leq u \leq 1 \tag{2.2}$$

Où les fonctions de base $B_i, n(u)$ sont les polynômes de Bernstein classiques de degré n, définies par :

$$B_i, n = \frac{n!}{i!(n-i)!} u^i (1-u)^{(n-i)} \tag{2.3}$$

L'algorithme de Casteljau [Piegl and Tiller, 1997] est la manière la plus efficace pour dessiner une courbe de Bézier. En effet, la construction est entièrement géométrique et consiste à tracer la courbe tout en déplaçant le barycentre des points de contrôle. Comme le montre la figure 2.7, la courbe est à l'intérieur de l'enveloppe convexe des points de contrôle (formant le polygone de contrôle). Elle interpole les deux points extrêmes, mais ne passe pas par les autres points qui définissent, cependant, l'allure de la courbe.

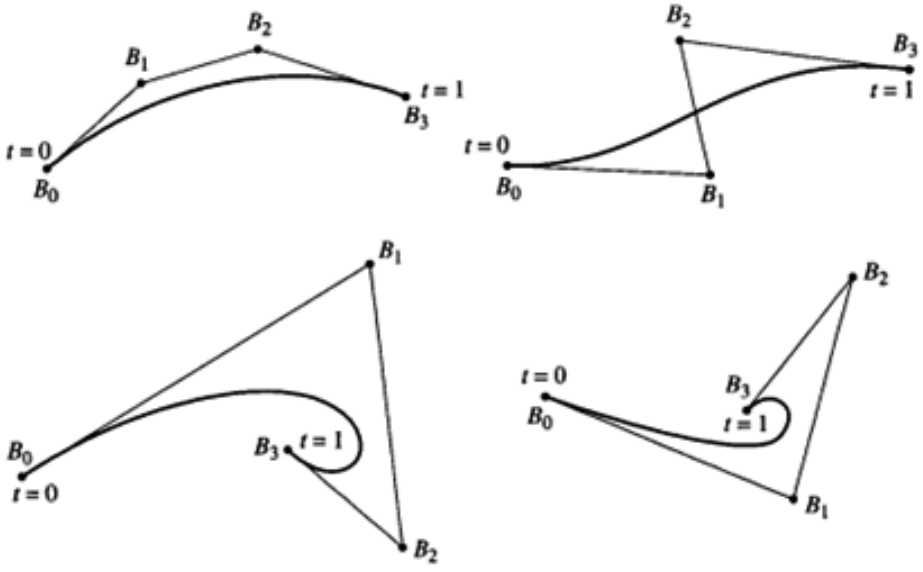


Figure 2.7 — Exemples de courbes de Bézier de degré 3 et avec quatre points de contrôle.

Les courbes de Bézier ont les propriétés suivantes :

- L'enveloppe convexe : la courbe est contenue dans l'enveloppe convexe constitué par les points de contrôle.
- Interpolation des deux points extrémités.
- Une courbe de Bézier est infiniment dérivable (de classe C^∞).
- Le contrôle de la courbe est global : modifier un point de contrôle modifie toute la courbe, et non pas un voisinage du point de contrôle.

– Nécessité d'un degré élevé pour satisfaire un grand nombre de contraintes (n-1 contraintes pour une courbe polynômiale passant par n points de contrôle).

– Les polynômes de degré élevé sont couteux à traiter.

Ainsi, les avantages d'une courbe de Bézier se résument dans le contrôle intuitif, l'inclusion de la courbe dans le polygone convexe et la simplicité de mise en oeuvre. Par contre, les limites sont, en premier lieu, le support global. En effet, la modification d'un point change l'allure générale de la courbe et par conséquent le contrôle n'est pas local. En deuxième lieu, un degré élevé est nécessaire afin de satisfaire un grand nombre de contraintes. En effet, pour une courbe de Bézier passant par n points donnés, il faut un degré égale à $(n-1)$. Cependant, les courbes de degré élevé sont numériquement instables.

2.4.4 Les courbes B-splines

Les courbes constituées d'un seul segment polynomial ou rationnel (comme les cas des Béziers et des Béziers rationnels) sont souvent inadéquates et présentent les inconvénients cités précédemment. Il s'agit donc de concevoir une courbe qui présente tous les avantages des courbes de Bézier mais sans ses inconvénients. Autrement dit, la courbe devra approximer les points de contrôle, présenter les mêmes propriétés que les courbes de Bézier, être simple à manipuler, le degré de la courbe ne devra pas être proportionnel au nombre de points de contrôle mais fixe, la modification d'un point ne doit pas affecter toute la courbe, etc.

La solution est d'utiliser des courbes polynomiales par morceaux. D'où la définition des B-splines comme suit :

$$C(u) = \sum_{i=0}^{n} P_i N_{i,p}(u) \qquad a \ \leq u \tag{2.4}$$

Oú les $\{P_i\}$ sont les points de contrôle (formant un polygone de contrôle). Les $\{N_{i,p}(u)\}$ sont les fonctions de base B-splinaires de degré p définies sur le vecteur nodal :

$$U = (\underbrace{a, ..., a}_{\text{p+1}}, u_{p+1}, ..., u_{m-p+1}, \underbrace{b, ..., b}_{\text{p+1}})$$

Ces fonctions sont définies par la formule récursive de Cox et DeBoor :

$$N_{i,0}(u) = \begin{cases} 1 & \text{si } u_i \ \leq u < u_{i+1} \\ 0 & \text{sinon} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \tag{2.5}$$

Notons que :

- $N_{i,0}$ est une fonction échelon qui est différente de zéro uniquement dans l'intervalle $[u_i, u_{i+1}[$.
- Pour $p > 0$, $N_{i,p}$ est une combinaison linéaire de deux fonctions de base de degré $p - 1$.
- Le calcul d'un ensemble de fonctions de base nécessite la spécification d'un vecteur nodal U et un degré p.

Le calcul d'un point sur la courbe B-spline pour une valeur de u donnée implique les étapes suivantes : La détermination de l'intervalle qui contient u. Ensuite, le calcul des fonctions de base non-nulles et enfin, la multiplication des valeurs des fonctions de base non nulles par les coordonnées des points correspondants. Notons que les fonctions de base B-splinaires ont les propriétés suivantes :

- Support local : $N_{i,p}(u) = 0 \, si \, u \notin [u_i, u_{i+p+1}[$.
- Pour n'importe quel intervalle j, il y a au plus $p + 1$ fonctions parmi les $N_{i,p}$ qui sont non nulles, soit les fonctions : $N_{j-p,p}, \ldots, N_{j,p}$.
- Non négativité : $N_{i,p}(u) \geq 0 \, \forall \, i,p \, et \, u$.
- Partition de l'unité :$\sum_{j=i-p}^{i} Nj, p(u) = 1$.
- Pour un vecteur nodal de la forme $U = \left\{ \underbrace{0, ..., 0}_{p+1}, \underbrace{1, ..., 1}_{p+1} \right\}$, ces fonctions se comportent comme des fonctions de Bernstein de degré p.

Par conséquent, les courbes B-splines ont les propriétés suivantes :

- Si $n = p$ et $U = \{0, ..., 0, 1, ..., 1\}$ alors $C(u)$ est une courbe de Bézier.
- $C(u)$ est une courbe polynômiale par morceaux. Étant donné le degré p, le nombre de points de contrôle $n + 1$ et le nombre de noeuds $m + 1$. Ces trois paramètres sont reliés par : $m = n + p + 1$.
- Les points aux extrémités sont interpolés.
- Propriété de l'enveloppe convexe : la courbe est contenue dans de ses points de contrôle.
- Propriété de modification locale : déplacer Pi change $C(u)$ seulement sur l'intervalle $[u_i, u_{i+p+1}]$.
- Propriété de l'invariance affine.

2.4.5 Les courbes B-splines Rationnelles non uniformes (NURBS)

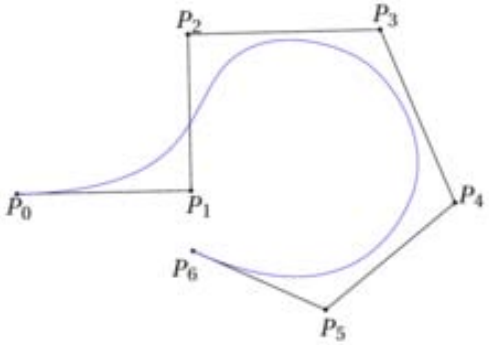
Une courbe NURBS de degré p est définie par :

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i P_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \quad a \ \leq u < b \tag{2.6}$$

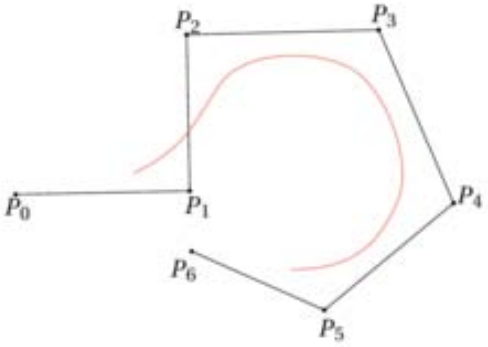
Où les $\{P_i\}$ sont les points de contrôle (formant un polygone de contrôle) et les $\{w_i\}$ sont leurs poids associés. Les $\{N_{i,p}(u)\}$ sont les fonctions de base B-splinaires de degré p définies sur le vecteur nodal U.

Le vecteur nodal est une séquence de valeurs de paramètres qui déterminent où et comment les points de contrôle vont affecter la forme de la courbe. En fait, il divise l'espace paramétrique en des intervalles et à chaque fois que le paramètre entre dans un nouvel intervalle nodal, un nouveau point de contrôle devient actif tandis qu'un plus ancien devient inactif. Ceci garanti donc l'influence locale des points de contrôle. Ce vecteur est défini par une séquence croissante de noeuds ($u_i \leq u_{i+1}$) compris entre 0 et 1. Ces noeuds consécutifs peuvent avoir des valeurs identiques, ceci est défini par la multiplicité d'un noeud, qui permet d'accentuer l'influence d'un point sur la courbe.

Ainsi, pour assurer l'interpolation des deux extrémités du polygone de contrôle il faut fixer la multiplicité du premier et du dernier noeud de U à la valeur $p+1$ (Fig. 2.8). Notons aussi que le degré p , le nombre de points de contrôle $n+1$ et le nombre de noeuds $m+1$ du vecteur nodal sont reliés par : $m = n + p + 1$.



(a) $U = (0, 0, 0, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 1, 1, 1)$.



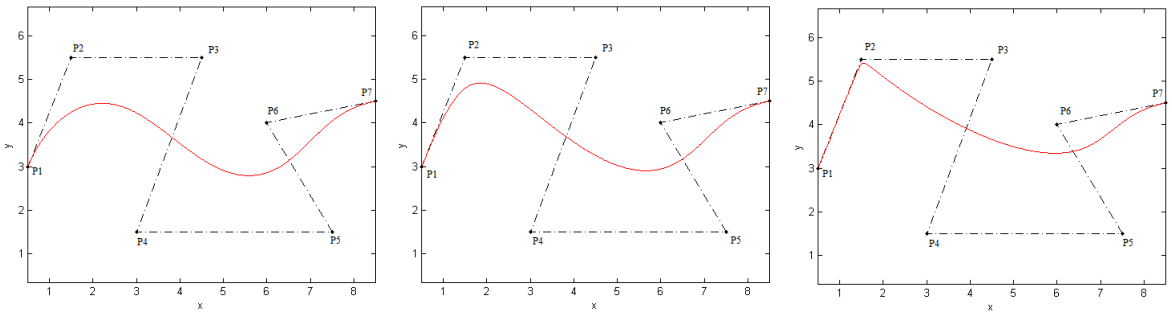
(b) $U = (0, \frac{1}{10}, \frac{1}{5}, \frac{3}{10}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{7}{10}, \frac{4}{5}, \frac{9}{10}, 1)$.

Figure 2.8 — Courbes NURBS possédant le même polygone de contrôle et des vecteurs nodaux différents.

Quant au paramètre poids, Chaque w_i détermine l'influence du point P_i sur la courbe. En fait, l'effet d'augmenter le poids associé à un point tire la courbe vers ce point, alors que lorsque le poids est diminué, la courbe s'éloigne du point (Figure. 2.9).

Si $w_i = 1$ pour tout point, on se retrouve dans le cas des B-splines. Sinon, pour des valeurs différentes de 1, on attribue une plus ou moins grande importance au point de contrôle correspondant :

– $w_i < 1$: donne une courbe moins proche de P_i.



(a) $w_i = 1, i = 1..7$. (b) Même courbe avec $w_2 = 3$. (c) Même courbe avec $w_2 = 30$.

Figure 2.9 — Courbe NURBS avec variation du poids de P_2.

- $w_i > 1$: donne une courbe plus proche de P_i.
- $w_i = 0$: le point P_i n'a plus d'influence.
- $w_i \to \infty$: la courbe passe par P_i.

Les NURBS sont des B-splines qui ont des points de contrôle pondérés. Ceci permet à la courbe d'être attirée vers un ou plusieurs points, comme souhaité. Elles maintiennent l'indépendance du degré (contrairement aux courbes de Bézier où le degré dépend du nombre de points de contrôle) et la propriété de modification locale qui caractérisent les B-splines. L'incorporation des poids des points de contrôle améliore la flexibilité et permet les NURBS de synthétiser différentes formes de courbes, en changeant les points de contrôle, les noeuds et les poids.

La majorité des logiciels de modélisation 3D tels que 3DReshaper, Maya et Rhinoceros3D emploient les courbes et les surfaces NURBS. Ce choix est justifié par le fait qu'elles présentent de nombreux avantages dont le plus important c'est la capacité d'approximer des formes complexes. En fait, un concepteur peut utiliser non seulement la position des points de contrôle, mais aussi la valeur de chaque poids ainsi que les valeurs des composantes du vecteur nodal, pour contrôler la forme de la courbe.

Toutefois, pour une présentation plus détaillée des NURBS, nous renvoyons le lecteur à l'ouvrage de référence [Piegl and Tiller, 1997]. Ci-dessous, nous présentons les propriétés, les plus intéressantes, des courbes NURBS :

- Interpolation des deux points extrémités : $C(0) = P_0$ et $C(1) = P_n$.
- Les courbes NURBS sont invariantes sous une transformation affine (rotation,

translation, homothétie).

- Propriété de l'enveloppe convexe : si $u \in [u_i, u_{i+1}]$ alors $C(u)$ est contenue dans l'enveloppe convexe des points de contrôle $P_{i-p}...P_i$.
- $C(u)$ est infiniment différentiable à l'intérieur d'un intervalle et $p - k$ fois différentiables à un noeud de multiplicité k.
- Approximation locale : si le point P_i ou le poids w_i sont modifiés, la courbe n'est pas affectée que dans l'intervalle $[u_i, u_{i+p+1}]$.
- Une courbe NURBS définie sur un vecteur nodal sans noeuds intérieurs n'est autre qu'une courbe de Bézier rationnelle.
- L'évaluation est raisonnable et numériquement stable.

2.5 Bilan

Comme nous l'avons présenté précédemment, Dubins fut le premier à déterminer, en l'absence d'obstacles, les plus courts chemins entre deux configurations de départ et d'arrivée pour un robot soumis à une contrainte sur le rayon de courbure. Ces travaux ont été suivis en 1990 par les résultats présentés par Reeds et Shepp qui ont proposé un algorithme déterminant le chemin de longueur minimale pour un robot sujet à une contrainte sur le rayon de courbure mais, contrairement au modèle pris par Dubins, ce robot est apte à faire des marches arrière.

Les solutions proposées par Dubins, aussi bien que celle de Reeds et Shepp se caractérisent par le fait qu'étant donné deux configurations du plan, il existe un plus court chemin les joignant qui est formé des segments de droite tangentiellement liés à des arcs de cercle de courbure maximale. Notons que, sur cette base, différents planificateurs de chemins en présence d'obstacles furent implantés.

Cependant, plusieurs recherches s'intéressant au domaine du contrôle des robots mobiles ont mis en vue l'intérêt de la continuité de courbure pour avoir des chemins avec un suivi précis. Cette contrainte n'est pas assurée par les chemins de Dubins ni celles de Reeds et Shepp d'où l'idée d'intégrer les clothoïdes pour le calcul d'un chemin admissible. Les avantages et inconvénients de ces méthodes sont résumés dans le tableau 2.1.

Tableau 2.1 — Tableau comparatif des chemins de Dubins, Reeds et Shepp et ceux utilisant les clothoïdes.

Méthode	**Avantages**	**Inconvénients**
– Les chemins de Dubins	– Simples à mettre en place – Des chemins de longueurs minimales	– Discontinuité de courbure – Planification sans manoeuvre
– Les chemins de Reeds et Shepp	– Simples à mettre en place – Des chemins de longueurs minimales – Adaptés pour robots sans et avec maoeuvre	– Ne se préoccupent pas de l'évitement d'obstacles – Discontinuité de courbure
– Les chemins utilisant les clothoïdes	– Des chemins de longueurs minimales – Continuité de courbure	– Complexité élevée liée au calcul des clothoïdes – Limitation d'utilisation en applications réelles

Vu les inconvénients des méthodes déjà citées, plusieurs travaux ont cherché d'autres solutions en utilisant les splines d'interpolation et d'approximation dans le processus de calcul des chemins et trajectoires pour les robots mobiles non holonomes. Les courbes paramétriques les plus utilisées sont les splines cubiques [Shikin and Plis, 1995] [Egerstedt and Martin, 2010], les courbes de Bézier [Prautzsch et al., 2002], les B-splines et les B-splines rationnelles non uniformes (NURBS) [Piegl and Tiller, 1997]. Chacune de ces splines a ses propres mérites et lacunes, qui sont résumés dans le tableau 2.2.

Parmi les travaux en planification de chemin, avec contrainte sur le rayon de courbure, et qui utilisent les splines nous citons ceux de Li et al [Li et al., 2006] qui emploient les splines cubiques, Ho et al. [Ho and Liu, 2009] pour les courbes de Bézier, Maekawa et al. [Maekawa et al., 2010] pour les B-splines...

Comme nous le pouvons constater, les courbes NURBS s'avèrent les plus avantageuses en raison de leurs propriétés intéressantes tel que le contrôle local et le taux élevé de flexibilité. Ce taux s'explique par la possibilité d'agir sur la courbe pour avoir

la forme désirée en modifiant soit le vecteur nodal, soit les positions des points de contrôle, ou aussi les poids associés à ces points. Ceci permet d'avoir des courbes lisses et ergonomiques sans oublier de souligner le critère de la simplicité de construction et d'implémentation et la faible complexité des algorithmes utilisés.

Tableau 2.2 — Résumé des propriétés des différents splines de degré p

	Contrôle local	***Régularité***	***Stabilité***	***Dérivabilité***
Les Splines	Satisfaisant	C^{p-1}	stabilité d'ordre m, indépendant du nombre de points d'interpolation	Standard, Efficace
Les Courbes de Bézier	Non satisfaisant	C^{p-1}	Pour p+1 points de contrôle, un degré élévé peut violer la stabilité	Standard, Efficace
Les B-Splines	Satisfaisant	C^{p-1}	Stable si le degré p n'est pas élévé	Standard, Efficace
Les NURBS	Excellent	C^{p-1}	Stable si le degré p n'est pas élévé	Standard, Efficace

2.6 Conclusion

Du fait de l'importance de la génération des trajectoires lisses avec des courbures continues dans le processus de contrôle et du suivi des robots mobiles non holonomes, plusieurs solutions ont été proposées dans la littérature pour le calcul des solutions

réalisables.

Dans ce chapitre, nous avons donné un aperçu sur les méthodes proposées pour la résolution de cette problématique en passant par les premières solutions, introduites pour avoir seulement des courbes lisses (en ignorant la continuité de courbure) avec ou sans manoeuvre (courbes de Dubins et courbes de Reeds et Shepp), les courbes à base de clothoïdes, et finalement les différents splines employées pour la modélisation des chemins (les splines cubiques, les courbe de Bézier, les B-Spline et les NURBS).

Nous avons également décrit les courbes NURBS en insistant sur ses avantages par apport aux autres techniques, ce qui nous a guidé à les employer dans nos contributions au domaine de la planification de mouvement, qui vont être détaillées dans les chapitres suivants.

CHAPITRE 3

Navigation autonome dans un environnement statique

3.1 Introduction

La robotique mobile vise, plus précisément, à développer des systèmes capables de se mouvoir d'une manière autonome. Les applications directes sont particulièrement dans les domaines de l'exploration planétaire et de la robotique de service, ce qui justifie l'importance de la planification de mouvement dans ses deux instances : chemin et trajectoire. Nous présentons dans ce chapitre une nouvelle approche pour la planification de chemin en termes de B-Splines rationnelles non uniformes (NURBS) pour les systèmes robotiques non holonomes. Ce chapitre introduit nos motivations, la formulation du problème, l'approche proposée avec ses différentes phases et quelques résultats de simulation.

3.2 Motivations et contribution

Plusieurs travaux de recherche en planification de chemin génèrent des solutions affines par morceaux. Ainsi, le robot suivant ces chemins doit s'arrêter et redémarrer fréquemment pour changer de direction, ce qui provoque un gaspillage d'énergie d'où l'importance de déterminer des solutions lisses. Plusieurs techniques de lissage ont été introduites. La première solution proposée était de générer un chemin optimal formé de segments de droites reliés tangentiellement par des arcs de cercles [Dubins, 1957].

Ces chemins, connus sous le nom de "Chemins de Dubins", n'assurent cependant pas la continuité de courbure ; comme l'a souligné Fraichard et Scheuer qui ont proposé, par la suite, une deuxième solution, appelée "CC Steer", déterminant des chemins lisses avec courbure et dérivée de courbure bornées. Cette méthode fournit une solution constituée de segments de lignes, des arcs circulaires et des arcs de clothoïde [Fraichard and Scheuer, 2004]. Toutefois, les clothoïdes n'ont pas d'équations simples et doivent être approximées par des splines et des polynômes d'ordre élevé, ce qui limite leur utilisation pour les applications robotiques en temps réel.

Les courbes de Bézier, les B-splines et les B-splines rationnelles non uniformes (NURBS pour Non Uniform Rational B-Splines) sont des courbes paramétriques couramment utilisées dans les applications de conception assistée par ordinateur (CAO). Les caractéristiques de ces courbes ont motivé leur utilisation dans des applications de navigation robotique. Certains travaux proposent les splines d'interpolation en tant que méthode de lissage, alors que d'autres utilisent les splines d'approximation. La différence entre ces deux techniques d'ajustement de courbes est illustrée dans la figure 3.1. En fait, le chemin interpolé (courbe en pointillées rouge) est plus long et a des mouvements oscillatoires, plus importants, ce qui augmente, par conséquence, la courbure du chemin. Cette solution n'est pas incluse dans le polygone de contrôle (les segments de droites joignant les points de contrôle) ce qui augmente la probabilité de collision (dans un contexte de planification de chemin, ce polygone n' est autre que le polyligne libre des obstacles). De ce fait, nous optons pour l'approximation plutôt que l'interpolation dans la solution que nous proposons.

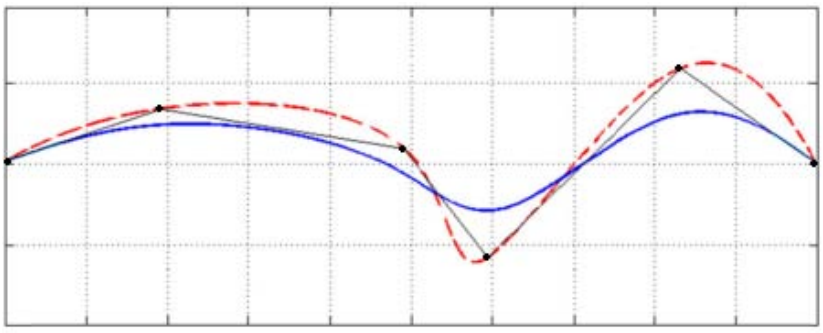


Figure 3.1 — Courbe B-spline d'interpolation (en pointillés rouge) Vs courbe B-spline d'approximation (en bleu).

Dans ce contexte, nous citons la méthode, proposée par Li et al, qui dérive une courbe lisse, en termes de splines cubiques, évitant les obstacles à partir de la donnée

d'une suite de segments sans collision [Li et al., 2006]. Aussi, dans [Ho and Liu, 2009], une approche de planification de chemin lisse en utilisant le diagramme de Voronoi et les courbes de Bézier pour calculer une solution, avec courbure bornée et de longueur minimale, a été présentée. Dans [Wei and Liu, 2010], les auteurs proposent une méthode de planification de chemin avec courbure continue dans un environnement statique en minimisant la longueur et le maximum de valeur de courbure tout au long de ce chemin. Cet algorithme utilise les algorithmes génétiques, pour l'optimisation de ces deux critères, et les fonctions η^3-splines comme technique d'interpolation. Maekawa et al. ont introduit un algorithme utilisant le graphe de visibilité et le recuit simulé pour générer des chemins sans collision en termes de courbes B-spline cubique [Maekawa et al., 2010]. Yang et al. [Yang et al., 2014] ont proposé une solution au problème en utilisant une méthode d'exploration RRT basée sur les splines pour garantir le calcul d'un chemin contraint par une valeur de courbure limite.

Les courbes paramétriques, utilisées dans les travaux cités précédemment, ont des propriétés qui limitent leur utilisation dans les planificateurs de chemin. Par exemple l'absence du contrôle locale pour les courbes de Bézier et le manque de flexibilité pour les courbes B-splines. Ainsi, les courbes NURBS se sont montrées plus intéressantes grâce à leurs propriétés importantes citées dans le chapitre précédent. Nous avons remarqué, également, que l'emploie des courbes NURBS pour la navigation robotique a été très limité. Parmi les travaux existants, nous citons [Singh et al., 2011] et [Xidias and Aspragathos, 2013] qui utilisent les courbes NURBS pour modéliser les solutions qu'ils produisent, mais sans autant donner une paramétrisation particulière ce qui les ramène à des simples B-splines.

Nous notons aussi que, durant la dernière décennie, les algorithmes génétiques (AG) sont largement utilisés pour générer des chemins pour les robots mobiles, grâce à leur forte capacité d'optimisation [Masehian and Sedighizadeh, 2007]. Mais, malgré la qualité de recherche rapide et élevée, les AG ont des limites tel que la convergence prématurée vers un optimum local. Ainsi, de nombreuses méthodes de planification de chemin ont été développées à travers l'adaptation ou la définition de nouveaux opérateurs génétiques [Tu and Yang, 2003, Gemeinder and Gerke, 2003, Sedighi et al., 2004, Yun et al., 2010, Tamilselvi et al., 2012, Tuncer and Yildirim, 2012, Qu et al., 2013, Ahmed and Deb, 2013]. Nous constatons aussi que ces méthodes ne prennent pas en considération les dimensions du robot en le modélisant par un simple point. Bien que

ces méthodes déterminent des chemins en minimisant essentiellement la longueur, la plupart ne parviennent pas à intégrer la continuité et la limite de courbure dans la définition de la fonction objectif à optimiser.

Motivés par la croissance de la demande en planificateurs de chemin plus éminents, nous cherchons à introduire une nouvelle approche modélisant la solution par une courbe NURBS paramétrisée en employant les algorithmes génétiques. Ces derniers représentent l'approche non déterministe la plus utilisée pour résoudre le problème de planification de mouvement [Masehian and Sedighizadeh, 2007], ce qui justifie son choix comme un moteur d'optimisation dans notre implémentation qui sera détaillée ci-dessous.

Cette approche aborde deux questions clés de la planification de chemin : la continuité du chemin et la contrainte de courbure maximale liée aux robots non holonomes. Les principales contributions sont : Premièrement l'adaptabilité de la méthode proposée qui se montre capable de trouver le chemin optimal ou quasi-optimal, en termes de longueur et sécurité à la fois et qui respecte la contrainte de courbure liée au robot considéré, indépendamment de la complexité de la scène étudiée. Deuxièmement, l'emploie des courbes NURBS avec une paramétrisation qui tient compte des contraintes et du résultat attendu en exploitant l'influence de chaque paramètre de ces fonctions d'approximation.

3.3 Approche proposée

3.3.1 Formulation du problème

Dans cette section, nous proposons une nouvelle approche de planification de chemin pour un robot mobile non holonome. Pour ce faire, il faut tout d'abord définir le modèle de l'environnement utilisé i.e la représentation de l'univers dans lequel le robot est amené à évoluer. Cette modélisation consiste, en fait, à traduire en une représentation informatique, un environnement Ω dont les obstacles sont connus tant par leurs dimensions que par leurs localisations.

Dans notre travail, le modèle considéré est une carte $2D$ donnant une représentation discrète de l'environnement en termes de points occupés ou libres. En effet, cette carte est encombrée d'un ensemble d'obstacles statiques de tailles et emplacements arbitraires $O_j, (j = 1, .., M)$, oú O_j est le j^{me} obstacle dans l'environnement d'évolution du robot. Ainsi, l'espace interdit au robot Ω_{obst} est définie comme suit :

$$\Omega_{obst} = \bigcup_{j=1}^{M} O_j \tag{3.1}$$

Par conséquent, nous notons par Ω_{free}, l'espace dans lequel le robot peut se déplacer librement :

$$\Omega_{free} = \Omega \setminus \Omega_{obst} \tag{3.2}$$

Nous supposons un modèle circulaire du robot non holonome R, avec un diamètre d, décrivant ses dimensions (ainsi, nous supposons que, quel que soit la forme du robot, il est circonscrit dans une cercle de diamètre connu) et un rayon de courbure minimale ρ_{min}. La figure 3.2 montre la formulation du problème de planification du chemin en termes de notre approche. Dans cette figure, les zones blanches sont Ω_{free} et celles en noir sont les zones impénétrables Ω_{obst}. L'objectif est alors de déterminer un chemin qui doit être lisse et d'une longueur minimale tout en prenant compte des deux contraintes suivantes : éviter les obstacles et respecter la limite de courbure.

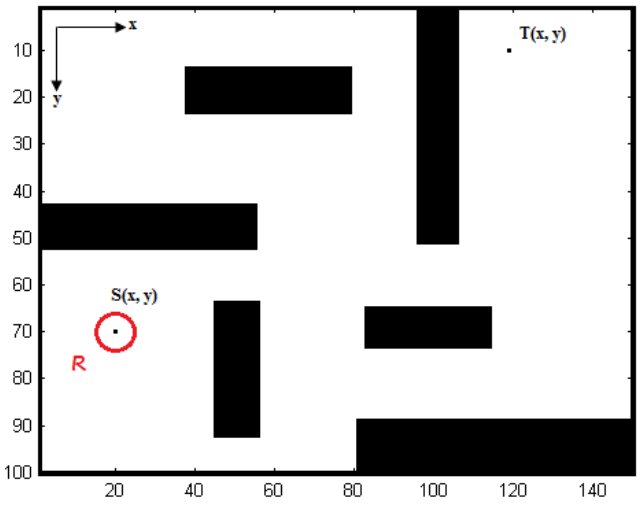


Figure 3.2 — Formulation du problème de planification du chemin en termes de notre approche.

3.3.2 Vue d'ensemble de l'approche

Comme l'illustre la figure 3.3, notre approche est composée de deux grandes phases : la première porte sur le calcul du chemin polyligne. Ce dernier devrait se trouver au milieu de l'espace libre des obstacles, afin de garantir au mieux la sécurité des déplacements du robot et tout en minimisant la longueur. La seconde se concentre sur le processus de lissage en utilisant les courbes NURBS, tout en maintenant la contrainte de la non-collision et en prenant en compte le rayon de courbure minimum du robot.

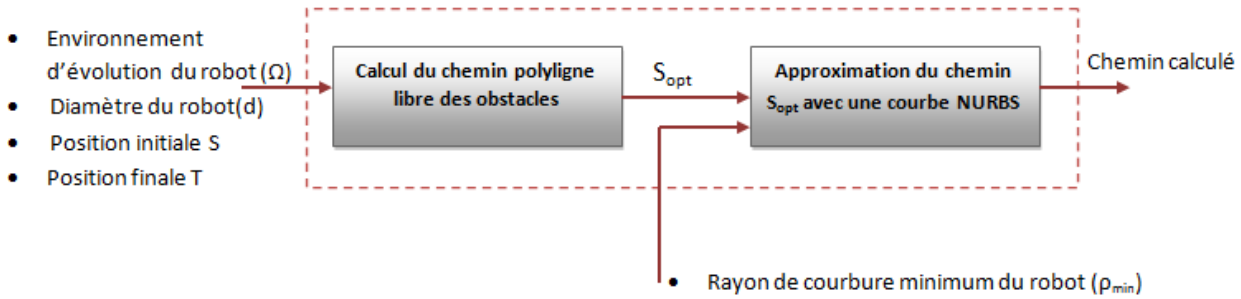


Figure 3.3 — synopsis de l'approche proposée.

Deux solutions utilisant deux différentes méthodes d'optimisation ont été proposées. La première proposition utilise un algorithme génétique pour paramétriser les poids des différents points de contrôle (judicieusement sélectionnés) afin d'exploiter les interpré-

tations géométriques de ce paramètre. Quant à la deuxième solution, elle se base sur la méthode d'exploration locale de Hooke et Jeeves, en agissant non seulement sur le paramètre poids, mais aussi sur les emplacements des différents points de contrôle.

3.3.3 Calcul du chemin polyligne libre des obstacles

Dans cette section, nous présentons les étapes de détermination du chemin polyligne libre des obstacles entre les deux positions initiale et finale du robot. L'idée principale est basée sur la capture de la topologie de l'espace de recherche (qui est l'espace libre des obstacles Ω_{free}) afin de simplifier le problème et le ramener à un problème de calcul de plus court chemin dans un graphe.

Afin d'échapper aux obstacles, ce graphe est construit à partir du squelette de Ω_{free}. En effet, pour augmenter la sécurité des déplacements du robot, ce dernier doit être placé à proximité du milieu de l'espace libre des obstacles. les principales étapes de calcul du chemin polyligne sont énumérées dans l'algorithme 1.

Algorithm 1: Collision-free Polyline Path Generation

Input: Ω, S, T, d

Output: Chemin polyligne libre des obstacles S_{opt}

Begin
- $X = SkeletonExtraction(\Omega_{free}, S, T)$;
- $EP(X) = [\bigcup_i \epsilon_{\theta_i(\overline{A})}(\overline{X})] \cap X$;
- $JP(X) = [\bigcup_i \epsilon_{\theta_i(B)}(X)] \cup [\bigcup_i \epsilon_{\theta_i(C)}(X)]$;
- $CCP(X) = CurveCriticalPoint(X)$;
- $CP(X) = X \backslash (EP(X) \cup JP(X) \cup CCP(X))$;
- $G = GraphConstruction(EP(X), JP(X), CCP(X), CP(X))$;
- $G' = GraphReduction(G, d, \Omega)$;
- **If** *TheSameConnectedComponent(S,T))* **then**
  - $S_{opt} = BellmanFordAlgorithm(G', S, T)$;
- **Else**
  - $S_{opt} = \varnothing$;
- **return** (S_{opt}) ;

End

L'algorithme débute par extraire le squelette de l'espace libre des obstacles Ω_{free} auquel il raccorde les deux positions de départ et d'arrivée (S et T respectivement) ; ce qui donne ,par conséquent, le squelette étendu sur lequel le robot peut se déplacer librement. Ensuite, un processus morphologique est appliqué pour classifier les pixels du squelette X en Points extrémités (End Points (EP)), points de jonction (Junction Points (JP)), points critiques de la courbe (Critical Curve Points (PCC)) et le reste des points de la courbe (Curve Points (CP)). Par la suite, un graphe pondéré est construit en associant $EP(X)$, $JP(X)$ et $CCP(X)$ avec les sommets, et l'ensemble de $CP(X)$ avec les arêtes. Les poids des arêtes ne sont autre que les distances euclidiennes entre les sommets du graphe. Une procédure de réduction de ce graphe est appliquée pour éliminer les arêtes représentants les couloirs inaccessibles par le robot. Et finalement, si S et T appartiennent à la même composante connexe, l'algorithme fait appel à un algorithme de calcul de plus court chemin (algorithme de Bellman Ford dans notre cas) pour renvoyer le chemin polyligne. Dans le cas contraire, la position d'arrivée n'est pas atteignable à partir de la position initiale [Jalel et al., 2013]. Ci-dessous, nous décrivons en détails ces étapes.

3.3.3.1 Extraction du squelette étendu de l'espace libre des obstacles

La squelettisation consiste, par définition, à réduire la forme en un ensemble de courbes, appelées 'squelettes', tout en respectant la topologie de la forme considérée. Cette représentation doit remplir trois conditions :

- Elle doit être aussi fine que possible (1 pixel d'épaisseur au maximum).
- Elle doit respecter la connexité.
- Elle doit être centrée dans la forme qu'elle représente.

La méthode de squelettisation utilisée dans notre approche est fondée sur le calcul d'une fonction discriminante F opérant sur un voisinage de 8 points [Marthon et al., 1979]. En fait, l'idée de base est de calculer, pour chaque point $p(i, j)$, la valeur de $F(i, j)$.

Cette fonction opère sur un voisinage de 8 points. $F(i, j)$ est la somme des vecteurs joignant ce point à tous ceux de son voisinage (noté (X, Y)), puis la longueur de vecteur

par $|X| + |Y|$. Cette longueur est comprise entre 0 et 4.

La valeur de la fonction F pour le point illustré dans la figure 3.4 est calculée comme suit :

$(X, Y) = (-1, 1) + (1, 1) + (1, 0) + (1, -1) = (2, 1)$

$F(i, j) = |X| + |Y| = 3$

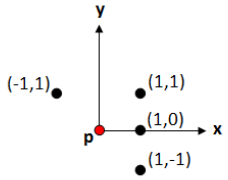
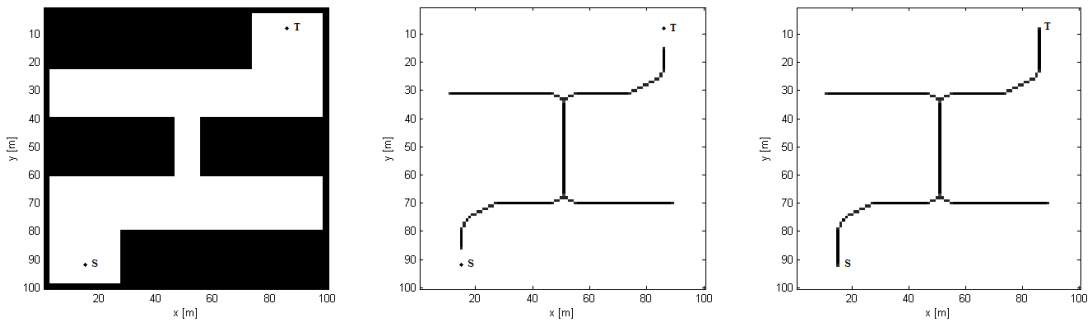


Figure 3.4 — p(i,j) et son voiisinage.

Intuitivement, les points à l'intérieur de la forme à squelettiser sont associés à une petite valeur de $F(i, j)$ et ils seront conservés tandis que ceux en dehors de la forme sont associés à des valeurs plus élevées de $F(i, j)$ et ils seront mis au rebut. La technique de la squelettisation utilisée opère itérativement pour sélectionner les points du squelette. chaque itération comprend deux balayages : Le premier calcule des caractéristiques locales de chaque point : $|X|$, $|Y|$, F et $NVOIS$ (nombre de points adjacents). Quant au deuxième, il décide d'éliminer ou non le point en fonction de ses caractéristiques. En général, les positions initiale et finale (S et T) n'appartiennent pas au squelette déterminé comme montré dans la figure 3.5 (b). Ainsi, ces deux points doivent être y raccordés. Ce raccordement donne lieu au squelette 'étendu' (Figure. 3.5 (c)) sur lequel le robot peut se mouvoir librement.



(a) L'espace des configurations. (b) Squelette initial. (c) Squelette étendu.

Figure 3.5 — Squelettisation de l'espace libre des obstacles.

Plusieurs travaux en planification de chemin s'appuient sur le calcul du squelette de l'espace libre des obstacles. Un algorithme classique est celui des diagrammes de voronoï [Takahashi and Schilling, 1989, Ho and Liu, 2009]. Bien que ce dernier donnera le même résultat, notre choix de la méthode d'extraction de l'axe médian se justifie par le fait qu'outre les propriétés géométriques assurées par tout algorithme de squelettisation, cette méthode rassemble des propriétés algorithmiques intéressantes. En fait, la simplicité du critère d'élimination des points de la forme à squelettiser permet une implantation câblée aisée. De plus, elle présente une faible sensibilité au bruit.

3.3.3.2 Modélisation par un graphe et calcul du plus court chemin

Après avoir calculé le squelette $'X'$ de l'espace libre des obstacles, nous nous intéressons à le convertir en un graphe. Pour ce faire, une analyse du squelette pour classifier ses points s'avère obligatoire. Par convention, nous distinguons 4 types de points :

Définition 1 : Un point d'une courbe numérique de largeur égale à 1 pixel est un point extrémité ($EP(X)$) s'il a un seul pixel de même valeur dans la fenêtre de taille 3×3 qui le contient.

Définition 2 : Un point d'une courbe numérique de largeur égale à 1 pixel est un point de jonction ($JP(X)$) s'il a plus de deux pixels de même valeur parmi ses 8 voisins.

Définition 3 : Un point d'une courbe numérique de largeur égale à 1 pixel est un point de la courbe ($CP(X)$) s'il a exactement deux pixels de même valeur parmi ses 8 voisins.

Définition 4 : Un point d'une courbe numérique de largeur égale à 1 pixel est un point critique ($CCP(X)$) s'il est un point de la courbe ($\in CP(X)$) et qu'il marque un changement significatif de direction.

La méthode employée pour extraire les points extrémités et carrefour (de jonction) est purement morphologique [Di Ruberto, 2004]. En fait, Chaque point extrémité dans une courbe numérique correspond à l'une des huit configurations générées par la rotation de 45° de l'élément structurant A, comme montrée dans la figure 3.6. Ainsi, l'extraction d'un tel point à partir du squelette X est obtenue en appliquant l'opération d'érosion entre le complément de chaque élément structurant définissant un point extrémité, $\overline{A}$ et ses rotations $\theta_i(\overline{A})$, sur le complément de $X, \overline{X}$:

$$EP(X) = [\bigcup_i \epsilon_{\theta_i(\overline{A})}(\overline{X})] \cap X. \tag{3.3}$$

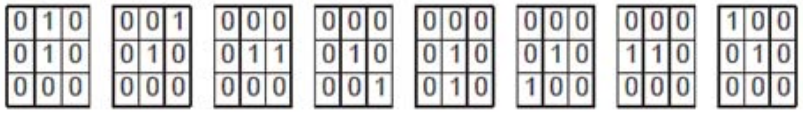


Figure 3.6 — ES (éléments structurants) pour les points extrémités : A et ses rotations $\theta_i(A)$.

Pour un point de jonction, il existe deux configurations fondamentales B et C et leurs sept rotations de 45° (figure 3.7). Par conséquent, l'extraction d'un point de jonction à partir du squelette s'obtient par l'application de l'opérateur d'érosion entre chaque élément structurant (B, C et leurs rotations $\theta_i(B)$ et $\theta_i(C)$) :

$$JP(X) = [\bigcup_i \epsilon_{\theta_i(B)}(X)] \cup [\bigcup_i \epsilon_{\theta_i(C)}(X)]. \tag{3.4}$$

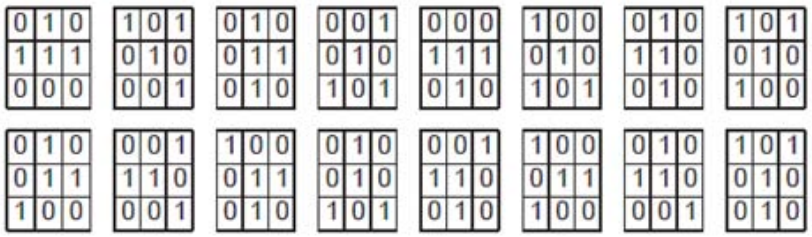


Figure 3.7 — ES pour les points de jonction : B et ses rotations dans la rangée supérieure ; C et ses rotations dans la rangée inférieure.

Tous les points de la courbe sont trivialement obtenus en éliminant les points extrémités et les points de jonction :

$$CP(X) = X \backslash (JP(X) \cup EP(X)) \tag{3.5}$$

Après l'analyse du squelette, ce dernier est converti en un graphe pondéré $G = (V, E)$ qui décrit les informations topologiques de l'espace libre des obstacles. Ce graphe est construit en associant les points extrémités, de jonction et critiques aux sommets et le reste des points (CP(X)) aux arêtes.

$$V = EP(X) \cup JP(X) \cup CCP(X) \tag{3.6}$$

$$E = \{e = (xy) \mid x, y \in V; \forall p_i \in e \wedge p_i \neq \{x, y\} \Rightarrow p_i \in CP(X)\} \tag{3.7}$$

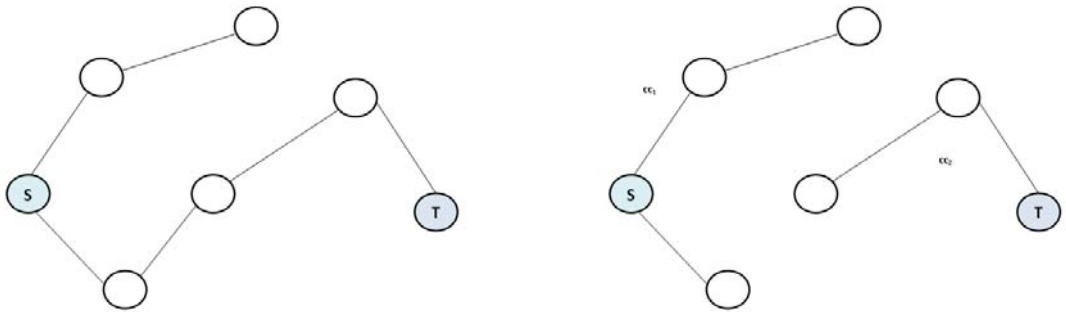
Pour les poids des arêtes, nous considérons une fonction f qui à toute arête $e = (x, y)$ lui associe la valeur de la distance euclidienne entre ses deux extrémités :

$$\begin{array}{rcccl} f & : & E & \rightarrow & R \\ & & e & \mapsto & f(e) = \|y - x\| \end{array}$$

L'étape de construction du graphe est suivie par une procédure de filtrage (réduction). En fait, nous définissons un couloir par un passage délimité par deux obstacles. Une CNS pour que le robot puisse s'engager dans un couloir est que son diamètre d soit inférieur à la plus petite largeur de ce couloir. Ainsi, l'algorithme doit tester pour chaque couloir s'il est suffisamment large pour que le robot puisse le traverser. Si ce n'est pas le cas, l'arête qui le représente dans le graphe G devra être supprimé. Á l'issue de ce filtrage nous obtenons un graphe partiel $G' = (V, E')$ dont chaque arête représente un couloir pouvant être emprunté par le robot.

$$E' = \{e = (x, y) \in E \mid d < l\} \tag{3.8}$$

Grâce aux traitements précédents, nous reformulons notre problème initial comme suit : trouver le plus court chemin joignant le sommet S (la position de départ) au sommet T (la position de d'arrivée) dans le graphe partiel G'. Pour cela, nous pouvons utiliser l'algorithme de Moore-Dijkstra , de Bellman ou encore de Ford-Fulkerson pour les tensions. Cependant, le graphe résultant G' peut être composé de plus qu'une seule composante connexe (figure 3.8). Ainsi, si les deux sommets S et T appartiennent à la même composante connexe cc_i ($i \in [1, .., k]$ avec k est le nombre de composantes connexes de G'), l'algorithme fait appel à une fonction de calcul de plus court chemin (algorithme de Bellman Ford dans notre cas) pour renvoyer le chemin polyligne S_{opt}. Dans le cas contraire, la position d'arrivée n'est pas atteignable à partir de la position initiale et, par conséquent, il n'existe pas de solution pour le scénario étudié.



(a) Graphe initial.

(b) Graphe partiel (formé de deux composantes connexes).

Figure 3.8 — Illustration de la déconnexion du graphe suite à la procédure de filtrage.

3.3.4 Lissage du chemin en termes d'une courbe NURBS et influence du paramètre poids

Le chemin S_{opt}, précédemment trouvé, est formé par une succession de segments de droite (Figure. 3.9 (a)). Pour parcourir un tel chemin, le robot doit se déplacer de façon rectiligne puis effectuer une rotation sur place pour changer de direction et suivre le segment suivant. Cette rotation sur place outre le fait qu'elle n'est pas toujours physiquement possible, fait perdre beaucoup de temps et d'énergie. Pour surmonter cette difficulté, nous cherchons une nouvelle représentation du chemin sous la forme d'une courbe lisse en respectant, outre la contrainte de sécurité, celle de la courbure limite.

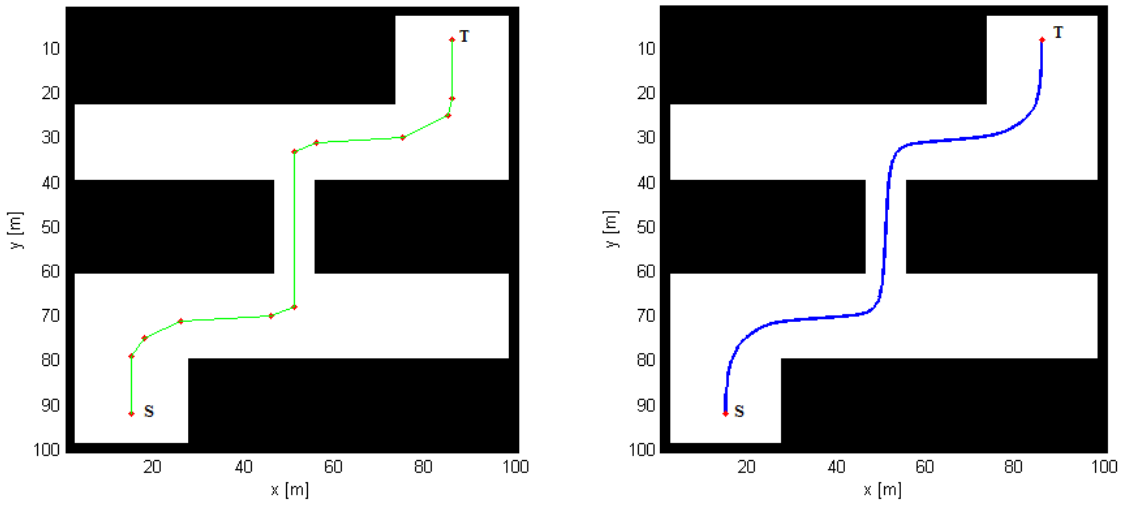
Du fait qu'elles présentent de nombreux avantages, tels que la faculté d'approximer des formes complexes, la simplicité de construction et d'implémentation et la faible complexité des algorithmes utilisés, l'emploi des NURBS est largement répandu et aborde différents domaines comme le design assisté par ordinateur et la biométrie. Ces fonctions d'ajustement permettent la génération des formes lisses et ergonomiques d'où l'idée de les intégrer dans une approche automatique de planification de chemin pour un robot mobile.

Pour générer une courbe NURBS, il faut spécifier l'ensemble de points de contrôle à approximer. Il faut, de même, paramétriser les poids associés à cet ensemble et le vecteur nodal U. Ainsi, à partir du chemin polygonal précédemment obtenu, nous construisons une courbe NURBS dont les points de contrôle P sont les extrémités de

chaque segment de ce chemin polygonal (les points marqués en rouge dans la figure 3.9 (a)). Le vecteur nodal est généré en exploitant l'information de corrélation entre le nuage des points de contrôle [Jalel et al., 2012]. Les poids étant initialisés à 1. Le résultat obtenu est illustré dans la figure 3.9 (b). Cette solution assure la continuité de courbure et la contrainte de la non-collision, mais ne garanti pas celle liée au rayon de courbure minimum du robot.

$$P = \{P_i\}_{i=1..n} = JP(S_{opt}) \cup EP(S_{opt}) \cup CCP(S_{opt}) \tag{3.9}$$

$$w_i = 1, \forall i = 1..n \tag{3.10}$$



(a) Chemin polygonal (S_{opt})).

(b) Lissage du chemin en utilisant une courbe NURBS.

Figure 3.9 — Génération du chemin en termes d'une courbe NURBS.

La courbe NURBS précédemment générée est, en effet, une courbe B-spline, étant donné que tous les points de contrôle ont la même valeur de poids. Bien que, pour cet exemple, la contrainte d'évitement d'obstacles a été maintenue, elle pourrait bien évidemment ne pas l'être. En fait, l'approximation de l'ensemble de points de contrôle peut introduire un écart important par rapport au chemin polygonal. Ainsi, nous allons investiguer l'influence du paramètre poids afin d'assurer les contraintes du système robotique étudié. L'idée est donc d'utiliser la propriété de contrôle local des NURBS.

3.3.4.1 Influence du paramètre poids et contrainte de non-collision

Comme montré dans la section 2.4.5, le poids d'un point de contrôle définit son influence sur la courbe NURBS. En fait, l'augmentation de la valeur du poids d'un point attire la courbe vers ce point. Cette constatation permet, naturellement, de réajuster la courbe en cas de détection de collision. Ainsi et pour assurer la contrainte de sécurité ; il suffit de localiser, en premier lieu, l'ensemble T des points de la courbe qui intersectent les obstacles :

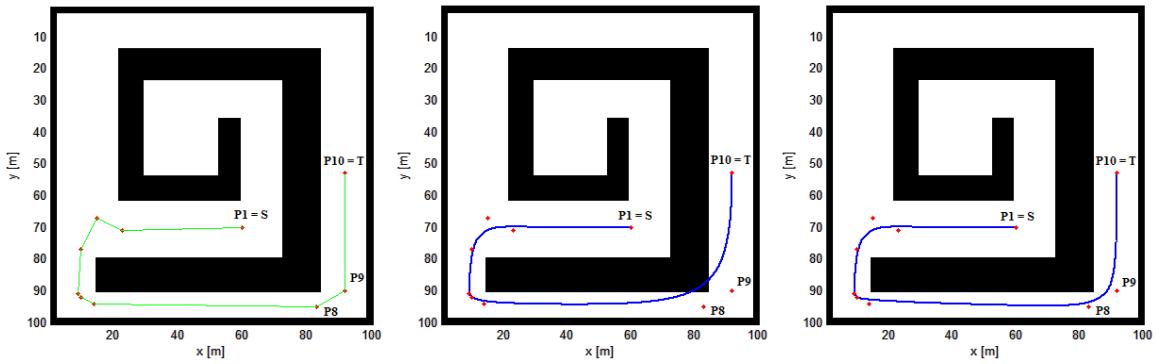
$$T = \{t_j\}_{j=1..n1} = C \cap \Omega_{obst} \tag{3.11}$$

En second lieu, il faut déterminer l'ensemble $T1$ des points de contrôle qui sont les plus proches aux points constituant T :

$$T1 = \{P_j \in P \mid distance(P_j, t_j) = Min(distance(P_i, t_j)), i = 1..n, j = 1..n1\} \tag{3.12}$$

Et finalement, la procédure consiste à augmenter les valeurs des poids des points de $T1$ jusqu'à assurer la non-collision. Le processus d'évitement des obstacles est montré dans La figure 3.10.

Dans cet exemple, après le calcul du chemin polygonal (Figure. 3.10 (a)), l'approximation de l'ensemble des points de contrôle $(P1, .., P10)$ par une courbe NURBS de degré 3 a donné une courbe intersectant les obstacles au voisinage de $P8$ et $P9$ (Figure 3.10 (b)). Le vecteur poids initial est $W = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$. L'augmentation des poids de ces deux points a réussi à assurer la contrainte de non-collision du chemin du robot, comme illustré sur la figure 3.10 (c). Ainsi, après l'exécution du processus d'évitement d'obstacles, le vecteur poids final est $W = (1, 1, 1, 1, 1, 1, 1, 10, 8, 1)$.



(a) Chemin polygonal (S_{opt})). (b) Lissage du chemin en utilisant une courbe NURBS. (c) Réajustement de la courbe NURBS.

Figure 3.10 — Processus d'évitement des collisions.

3.3.4.2 Influence du paramètre poids et contrainte de limite de courbure

L'addition de la courbure en tant que paramètre lors de la génération du chemin optimal exprime naturellement la contrainte de continuité de courbure. Dans la section précédente, nous avons déterminé un chemin lisse libre des obstacles modélisé par une courbe NURBS. Notons que le paramètre de courbure est l'inverse du rayon de courbure du robot, qui est lié aux contraintes de vitesse et d'accélération.

Notons aussi qu'une courbure continue admissible est d'une importance pratique pour la planification et le contrôle des robots mobiles. En fait, un chemin avec une faible variation de courbure est plus approprié pour le suivi et le contrôle qu'un chemin avec un grand écart de courbure ; et ce pour éviter les changements rapides des accélérations radiales. [Wei and Liu, 2010]. Étant une courbe paramétrique, la fonction de courbure d'une NURBS est calculée comme suit [Piegl et al., 2009] :

$$\kappa(u) = \frac{(C'(u) \times C''(u)).B}{|C'(u)|^3} \tag{3.13}$$

$C'(u)$ et $C''(u)$ sont les dérivées première et seconde. B représente la binormale (Le vecteur binormal est le produit vectoriel du vecteur unitaire tangent à la courbe en un point donné et du vecteur orthogonal à ce dernier). La dérivée première d'une courbe NURBS est donnée par :

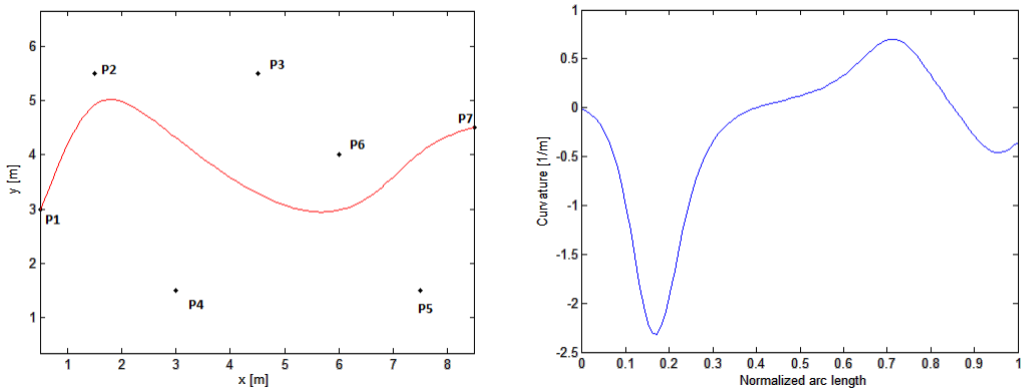
$$C'(u) = \sum_{i=1}^{n+1} R'_{i,p}(u) P_i \tag{3.14}$$

Avec :

$$R'_{i,p}(u) = \frac{w_i N'_{i,p}(u)}{\sum_{i=1}^{n+1} w_i N_{i,p}} - \frac{w_i N_{i,k} \sum_{i=1}^{n+1} w_i N'_{i,k}}{(\sum_{i=1}^{n+1} w_i N_{i,p})^2} \tag{3.15}$$

Une description détaillée du calcul des dérivées d'une courbe NURBS est présentée dans [Piegl and Tiller, 1997]. La figure 3.11 illustre une courbe NURBS approximant 7 points de contrôle et le profil de courbure associé. L'objectif de cette section est de réajuster une courbe NURBS en prenant compte de la contrainte de courbure limite κ_{max} :

$$|\kappa| \leq \kappa_{max} = \frac{1}{\rho_{min}} \tag{3.16}$$



(a) (b)

Figure 3.11 — (a) Une courbe NURBS avec 7 points de contrôle. (b) Profil de courbure associé.

Comme le poids d'un point de contrôle augmente, la courbe s'en rapproche et si le poids est d'une importance suffisante, la courbe va suivre à peu près le point de contrôle considéré. Par conséquent, la valeur de courbure du point de la courbe le plus proche de ce point de contrôle augmente. Ainsi, nous déduisons que le paramètre poids est inversement proportionnel au rayon de courbure comme illustré dans la figure 3.12 où l'augmentation progressive de la valeur du poids du point P_i de 0.25 à 5 conduit à la diminution progressive du rayon de courbure de Q_j.

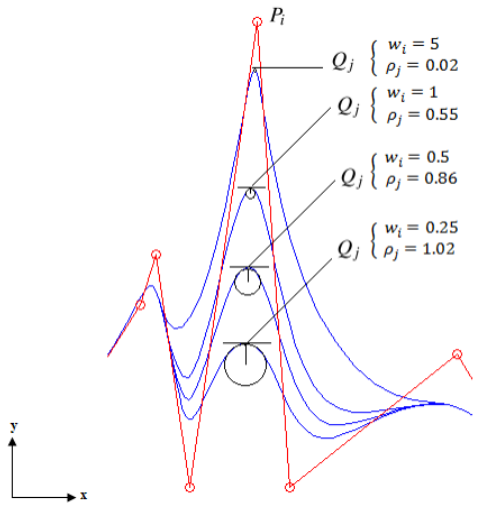
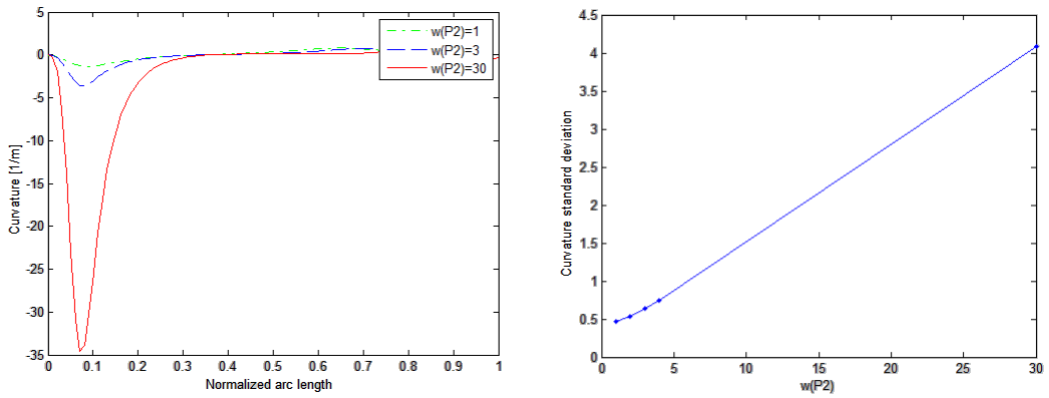


Figure 3.12 — Variation du rayon de courbure du point de Q_i vs variation du poids du point de contrôle le plus proche P_i.

Étant l'inverse du rayon de courbure, la valeur de la courbure augmente avec l'augmentation du poids. La figure 3.13 illustre la proportionnalité entre ces deux paramètres. En fait, nous considérons les courbes NURBS approximant le même ensemble des points de contrôle avec un poids différent pour le point P_2 (Figure 2.9). Comme montré dans la figure 3.13 (a), la variation de la valeur du poids de P_2 a généré l'augmentation de la valeur de courbure du point, de la courbe, le plus proche à ce point de contrôle. Ce qui a augmenté, par conséquent, la mesure de l'écart type des courbures (Figure 3.13 (b)).



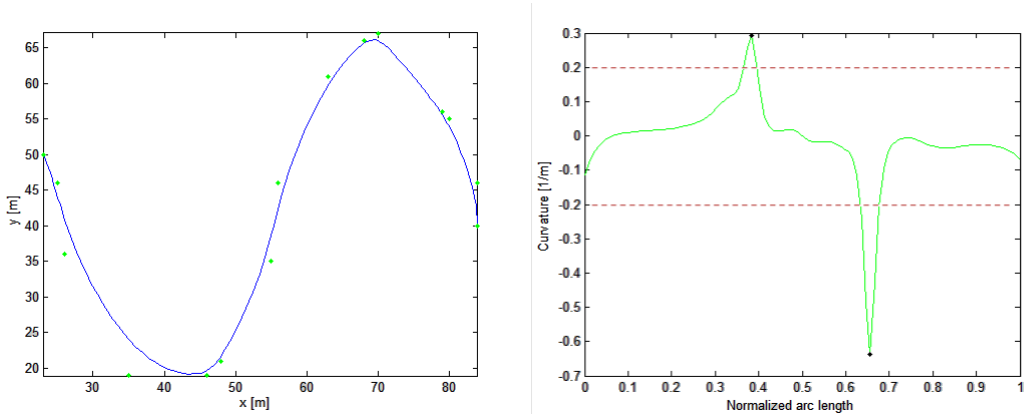
(a) (b)

Figure 3.13 — (a) Profils des courbures des courbes NURBS de la figure 2.9. (b) Variation de l'écart type des courbures en fonction de l'augmentation du poids de $P2$.

Ainsi, pour réajuster une courbe NURBS afin de satisfaire la contrainte de courbure 3.16, la solution serait de diminuer les poids des points de contrôle (ensemble $T2$) les plus proches des points de la courbe qui ne vérifient pas cette contrainte (ensemble T').

$$T' = \{t'_j, j = 1..q || \kappa_{t'_j}| > \kappa_{max}\} \tag{3.17}$$

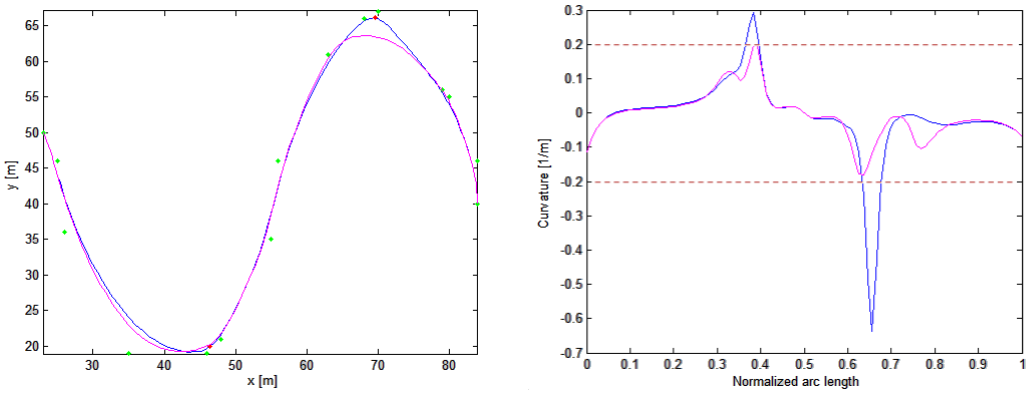
Pour illustrer cette solution, nous proposons l'exemple suivant : La figure 3.14 (a) représente une courbe NURBS de 15 points de contrôle avec des poids unitaires. Le profil de courbure associé est montré dans la figure 3.14 (b). Les deux droites en pointillés représentent la contrainte de courbure ($|\kappa_{max}| = 0.2$). Pour cet exemple, il existe deux points d'inflexion (marqués en noir) qui dépassent cette contrainte.



(a) (b)

Figure 3.14 — (a) Une courbe NURBS avec 15 points de contrôle. (b) Profil de courbure associé.

La figure 3.15 (a) illustre les courbes NURBS avant (bleu) et après (magenta) l'application de la procédure de correction de courbure. Les deux points de la courbe qui violent la contrainte de courbure sont marqués en rouge. Pour réajuster la courbe initiale et la rendre admissible, l'algorithme agit sur les poids des quatre points de contrôle P_5, P_6, P_{10} et P_{11}. La courbe résultante (magenta) possède un profil de courbure acceptable comme montré dans la figure 3.15(b). Le vecteur poids permettant d'avoir ce résultat est $W = (1, 1, 1, 1, 0.5, 0.8, 1, 1, 1, 0.1, 0.1, 1, 1, 1, 1)$.



(a) (b)

Figure 3.15 — (a) Courbes NURBS avant (bleu) et après (magenta) le processus de correction de courbure. (b) Profils de courbures associés.

3.3.5 Utilisation des algorithmes génétiques pour la génération d'un chemin en termes d'une courbe NURBS

L'objectif est d'obtenir une courbe NURBS **réalisable** i.e. sans collision et telle que le rayon de courbure en chaque point de cette courbe soit supérieur à une valeur minimale fixée (ou encore telle que la valeur de courbure en chaque point soit inférieure à une valeur maximale fixée).

Une première idée était de calculer les deux ensembles $T1$ et $T2$, précédemment présentés, et d'y agir en augmentant (diminuant) les poids associés pour garantir la contrainte de non-collision (de courbure) [Jalel et al., 2015b]. Cependant, cette solution manque de complétude étant donné qu'une condition nécessaire pour avoir une solution est que ces deux ensembles n'aient pas des points en commun (vu qu'il est est inutile d'augmenter puis diminuer la valeur du poids). Remarquons que dans un cas pareil, une solution pourra être déterminée en modifiant les poids de certains points au voisinage de ces points appartenant à la fois à $T1$ et $T2$.

Partant de cette constatation et dans le but d'augmenter la complétude de notre méthode, il serait plus judicieux d'avoir la possibilité d'agir sur la totalité des points de contrôle plutôt que sur un ensemble donné. Ceci est fait en résolvant un problème d'optimisation dont les variables sont les poids associés aux points de contrôle. En fait, trouver la bonne paramétrisation d'une courbe NURBS est d'une importance indéniable. Plusieurs algorithmes évolutionnaires ont été employés pour résoudre ce

problème [Adi et al., 2009], [Jing et al., 2009].

Notre approche intègre les algorithmes génétiques (AG) comme moteur de résolution. Ce choix se justifie par la grande robustesse des AG, c'est à dire leur grande capacité à trouver des optimums globaux ou de s'en approcher, et ce indépendamment de l'initialisation.

Formulons, donc, notre problème : soient $P_1, P_2, ..., P_n$ une séquence ordonnée de points de contrôle. Nous voulons générer une courbe NURBS à partir de cet ensemble de points, tout en prenant en compte les contraintes du système liées à l'environnement (évitement des obstacles) et aux caractéristiques du robot (dimensions et rayon de courbure) et tout en optimisant certains critères. Le problème est donc de trouver la bonne paramétrisation des poids associés aux points de contrôle. A cet effet, nous proposons une procédure de paramétrisation des courbes NURBS basée sur un algorithme génétique qui, pour des paramètres de contrôle donnés (nombre d'itérations, taille de population, pourcentages de croisement et de mutation), revoie une solution optimale ou quasi-optimale pour le problème de planification de chemin.

L'algorithme 2 décrit notre adaptation de l'AG pour le problème de planification de chemin, pour les robots mobiles, en termes d'une courbe NURBS. Cet algorithme débute par la création de la population initiale en attribuant des valeurs de poids aléatoires ($w_i \in [0.1; MaxWeight]$) aux différents points de contrôle. Ensuite, il s'exécute jusqu'à un nombre fixe d'itérations.

Un cycle comporte cinq étapes principales : (i) évaluation de la population au moyen d'une fonction fitness multi-objectifs, (ii) sélection des parents, (iii) exécution de l'opération de croisement, (iv) exécution de l'opération de mutation et, enfin, la mise à jour de la population actuelle en supprimant certains individus (les mauvais) pour maintenir la taille de la population inchangée. L'algorithme se termine par sélectionner, à partir de la dernière population, le meilleur individu (chromosome) qui représente le chemin optimal calculé.

Algorithm 2: GA-based NURBS curve parameterization

Input: Ω, P, p, ρ_{min}, d, $MaxWeight$, n_iter, $PopSize$, c_rate, m_rate

Output: Planned path C

```
Begin
    P_w = InitPopulation(P, PopSize, MaxWeight) ;
    it = 1 ;
    While it ≤ n_iter do
        Evaluation(P_w, Ω, ρ_min, d) ;
        Parents = Selection(P_w, c_rate) ;
        CrossoverOp(P_w, Parents) ;
        MutationOp(P_w, m_rate) ;
        UpdatePop(P_w, PopSize) ;
        it = it +1 ;
    C = NURBSCurveGeneration(Best(P_w), p) ;
    return (C) ;
End
```

Avant de décrire en détails ces différentes étapes, nous représentons, ci-dessous, le schéma de codage de l'individu dans cet AG.

3.3.5.1 Représentation de l'individu

La structure d'un chromosome traduit la façon dont une solution est codée. Dans un contexte de planification de chemin, cette structure doit avoir suffisamment d'informations sur la totalité de la solution depuis le point de départ jusqu'à la position finale, afin de pouvoir la représenter. Cette solution, pour notre cas, est une courbe NURBS représentant un chemin. Une telle courbe est définie par un ensemble de n points pondérés comme illustré dans la figure 3.16, où P_1 et P_n sont toujours les positions initiale et finale. Ainsi, chaque gène représente les coordonnées (x_i et y_i) d'un point de contrôle ainsi que son poids w_i. Il est à noter que notre algorithme est autorisé à changer seulement les valeurs des différents poids.

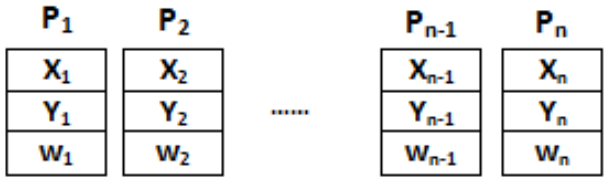


Figure 3.16 — Représentation d'un chromosome.

3.3.5.2 Fonction d'évaluation

Au cours de chaque itération (génération), la population des chemins est évaluée pour quantifier leur degrés d'élitisme qui dépend de la façon dont chaque solution (chemin) est appropriée pour le problème. Par conséquent, la valeur de fitness d'un individu doit être proportionnelle à sa capacité de survie. La plupart des planificateurs de chemins utilisant les AG considèrent la longueur du chemin comme un critère à minimiser et négligent la faisabilité de la solution.

Dans cette étude, nous proposons une fonction d'évaluation précise qui dépend de quatre paramètres : la sécurité, la faisabilité, la longueur et l'écart type de courbures du chemin. Ainsi, la fonction objectif pour un chemin (une courbe) C est définie comme suit :

$$f(C) = \alpha.P_{safety}(C) + \beta.P_{feasibility}(C) + \gamma.\frac{1}{P_{length}(C)} + \delta.P_{\sigma}(C) \tag{3.18}$$

Avec $P_{safety}(C)$ traduit la contrainte de sécurité du chemin en indiquant le nombre de collisions détectées :

$$P_{safety}(C) = |P_{col}| \tag{3.19}$$

Avec :

$$P_{col} = \{p_i \in C, i = 1..k1\} = C \cap \Omega_{obst} \tag{3.20}$$

$P_{feasibility}(C)$ traduit la contrainte de faisabilité du chemin, qui est liée au rayon de courbure minimum du robot, et est déterminée par le nombre points de la courbe qui ne vérifient pas la contrainte de courbure :

$$P_{feasibility}(C) = |P_{cur}| \tag{3.21}$$

Avec :

$$P_{cur} = \{p_j \in C, j = 1..k2 || curvature(p_j)| > \frac{1}{\rho_{min}}\} \tag{3.22}$$

$P_{length}(C)$ est la longueur du chemin, déterminée comme suit :

$$P_{length}(C) = \int_0^1 \|C'(u)\| \, \mathrm{d}u \tag{3.23}$$

$C'(u)$ est la dérivée première de $C(u)$. Finalement $P_\sigma(C)$ désigne l'écart type de courbures tout au long du chemin. Cette valeur mesure la dispersion des valeurs de courbures autour de leur moyenne arithmétique c_{mean} :

$$P_\sigma(C) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(c_i - c_{mean})^2} \tag{3.24}$$

α, β, γ et δ sont des facteurs de pondération. Il est évident, dans notre méthode, que la meilleure solution est celle ayant la valeur fitness la plus petite.

3.3.5.3 Opérateur de sélection

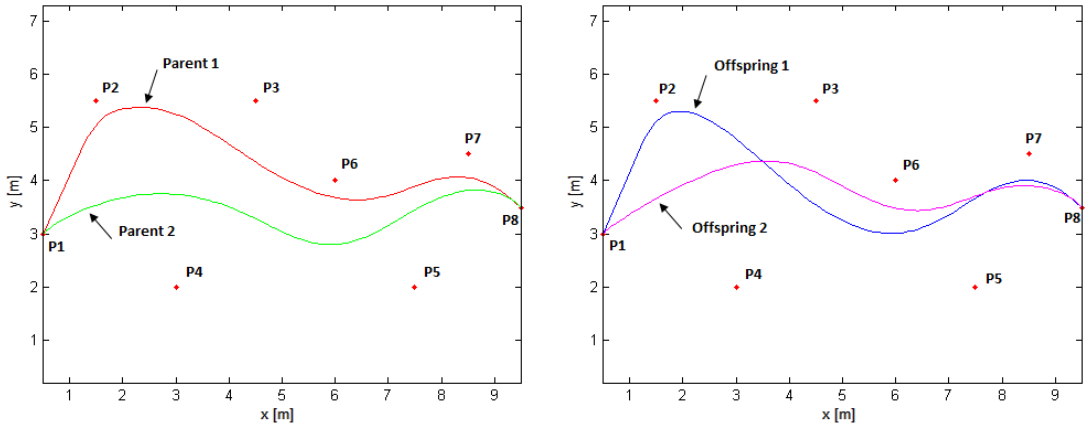
L'opérateur de sélection permet d'identifier les meilleurs individus d'une population afin de créer une nouvelle population après application des opérations de croisement et de mutation. Il est à noter que la probabilité de survie d'un chromosome est directement liée à son efficacité relative au sein de la population. Il existe plusieurs stratégies de sélection telles que la sélection par roulette (wheel), la sélection par rang, la sélection par tournoi, l'élitisme, etc... Dans cette étude, nous employons la sélection par rang.

3.3.5.4 Opérateur de croisement

Le rôle fondamental de l'opérateur de croisement est de copier et recombiner des informations (gènes) des deux chromosomes parents de façon à former deux nouveaux individus fils possédant des caractéristiques issues des deux parents. Les méthodes

conventionnelles de croisement comprennent le croisement avec un seul point de coupure (single-point crossover) et celui avec deux points de coupure (two-point crossover) qui est prouvé plus efficace.

Afin d'accroître la diversité de la population, nous utilisons le croisement avec plusieurs points de coupure (n-point crossover). Le nombre et les positions des sites de croisements sont aléatoires et varient d'une itération à une autre. Par conséquent, le premier individu fils est obtenu en copiant des gènes du $Parent1$ jusqu'au premier point de coupure, puis en complétant avec les gènes du $Parent2$ jusqu'au second point de coupure et ainsi de suite jusqu'au dernier point de coupure. Inversement, nous obtenons le deuxième individu fils. La figure 3.17 illustre cette procédure. Dans cet exemple, nous choisissons trois points de coupures aux positions 2, 5 et 7 (figure 3.17 (a)). Avant l'application de l'opérateur de croisement, le vecteur poids du $Parent1$ est $W = (0.1, 3.4, 4.5, 1.2, 2.3, 3, 4, 2)$. Le vecteur poids du $Parent2$ est $W = (5.2, 0.4, 3.2, 3.6, 5.1, 2, 2, 2)$. Après l'application de l'opérateur de croisement, nous obtenons deux nouvelles courbes NURBS (deux nouveaux individus) comme le montre la figure 3.17 (b). Le vecteur poids du $offspring1$ est $W = (0.1, 3.4, 3.2, 3.6, 5.1, 3, 4, 2)$, le vecteur poids du $offspring2$ est $W = (5.2, 0.4, 4.5, 1.2, 2.3, 2, 2, 2)$.



(a) (b)

Figure 3.17 — Illustration de l'application de l'opérateur de croisement.

3.3.5.5 Opérateur de mutation

L'opération de mutation génère des "erreurs" de recopie, dans l'objectif de créer un nouvel individu qui n'existait pas auparavant. Ce qui permette l'AG d'éviter de conver-

ger vers un optimum local de la fonction. Ainsi, et dans le but d'augmenter la diversité de la population, l'opération de mutation est nécessaire dans les algorithmes génétiques malgré sa faible probabilité. Dans un algorithme génétique classique, c'est souvent la mutation aléatoire qui est employée. Cette dernière consiste à modifier aléatoirement la valeur d'un gène quelconque d'un chromosome quelconque. Cependant cette mutation aléatoire peut causer des chemins infaisables.

Pour éviter cela, nous proposons un nouvel opérateur de mutation qui modifie un gène particulier avec une manière guidée et dans le but d'optimiser la valeur de fitness de l'individu sélectionné pour mutation. Cet opérateur de mutation agit en exploitant l'influence du paramètre poids de la courbe NURBS (comme décrit précédemment). En fait, l'évitement d'obstacle peut être assuré en augmentant le facteur de pondération de certains points de contrôle. De même, la contrainte de courbure peut être assurée en réduisant le poids de quelques points de contrôle.

L'algorithme 3 décrit l'opérateur de mutation proposé. En effet, le processus de mutation commence par vérifier la faisabilité du chemin (individu sélectionné au hasard pour mutation). Ainsi, deux cas peuvent se présenter : Le premier concerne le cas où le chemin est infaisable. Dans un tel cas, le chemin pourrait soit avoir des collisions, soit avoir des valeurs de courbures inadmissibles. Si le chemin intersecte les obstacles ($P_{col} \neq \emptyset$) , alors la procédure consiste à sélectionner aléatoirement un des points de la courbe qui heurtent les obstacles, puis à déterminer le point de contrôle qui lui est le plus proche (en terme de distance euclidienne) et enfin, à augmenter la valeur du poids de ce point de contrôle. Ce traitement a pour conséquence la diminution du nombre de points de courbe qui ne respectent pas la contrainte de non-collision.

Sinon, si l'infaisabilité du chemin découle du fait que certains points de la courbe qui le représente dépassent la limite de courbure, l'algorithme consiste à sélectionner aléatoirement un point de l'ensemble P_{cur} (ensemble des points de la courbes ne respectant pas la contrainte de courbure). Puis, à diminuer le poids de son plus proche point de contrôle, ce qui conduit à réduire le cardinal de P_{cur}. Ainsi, pour un chemin infaisable, l'opérateur de mutation engendre la diminution du nombre de points de la courbe qui sont, soit en collision avec les obstacles soit ne respectant pas la limite de courbure du robot considéré.

Dans le cas où le chemin sélectionné pour la mutation est un chemin faisable, l'algorithme tente de diminuer le maximum des courbures tout au long de ce chemin ce

qui diminue, par conséquent, la mesure de l'écart type de courbures. Il est à noter que, l'opérateur de mutation proposée contribue à améliorer la valeur de fitness de l'individu à muter.

Algorithm 3: Proposed Mutation Operator

Input: Selected path for mutation (Cs)

Output: Mutated path Cs

Begin
  If ($P_{safety}(Cs) = 0$ & $P_{feasibility}(Cs) = 0$) **then**
    $x = SelectMaxCurvature(C_s)$;
    $P_m = NearestPoint(P, x)$;
    $DecreaseWeight(P_m)$;

  Else if ($P_{safety}(Cs) = 0$) **then**
    $x = RandomSelect(P_{cur})$;
    $P_m = NearestPoint(P, x)$;
    $DecreaseWeight(P_m)$;
  Else
    $x = RandomSelect(P_{col})$;
    $P_m = NearestPoint(P, x)$;
    $IncreaseWeight(P_m)$;
  $RegenerateNURBScurve(C_s, P_m)$;
  return (C_s) ;
End

3.3.6 Utilisation de la méthode de Hooke et Jeeves pour la génération d'un chemin en termes d'une courbe NURBS

La méthode de Hooke et Jeeves [Hooke and Jeeves, 1961] est l'une des méthodes d'explorations locales qui consistent, à partir d'une solution initiale, et par transformations successives, à construire une suite de solutions améliorant la fonction objectif. De manière générale, le processus s'achève lorsqu'on ne peut pas améliorer la solution courante. Cette méthode est une méthode géométrique directe d'optimisation (sans calcul des dérivées des contraintes et des critères). Elle se base sur deux étapes principales qui sont la recherche de la direction de descente et le déplacement dans cette direction.

En effet, à partir d'un point de référence (ou initial), l'algorithme effectue une recherche exploratoire par perturbation d'un incrément $+\triangle_i$ d'une variable, les autres variables restant fixes. Si le coût de la fonction à optimiser est meilleur, alors cette nouvelle composante est conservée. Dans le cas inverse, la variable est perturbée en $-\triangle_i$ et conservée en cas d'amélioration de la fonction objectif. Si les deux incréments n'apportent aucune amélioration, la composante reste inchangée. A la fin du processus, toutes les composantes ont été perturbées une fois, et nous obtenons soit un point pour lequel la fonction coût a été améliorée, soit le point de référence lui-même. Dans ce dernier cas, le processus est réitéré avec un pas $\triangle_i$ plus petit (typiquement $+\triangle_i/2$). Dans le cas contraire, le point obtenu et le point de référence sont utilisés pour définir une nouvelle direction de recherche par extrapolation. Un déplacement peut alors être effectué le long de cette direction. Le nouveau point ainsi obtenu devient le point de référence. La procédure est en général stoppée lorsque les incréments $\triangle_i$ deviennent inférieurs à une précision donnée.

Rappelons que notre objectif est d'obtenir une courbe NURBS **réalisable** i.e. sans collision et telle que la valeur de courbure en chaque point soit inférieure à une valeur maximale fixée. Contrairement à l'approche décrite précédemment, qui s'appuie sur l'utilisation d'un algorithme génétique pour trouver la bonne paramétérisation des poids des différents points de contrôle sans changer les positions géométriques de ces derniers, cette deuxième approche se base sur l'utilisation de la méthode de Hooke et Jeeves pour déterminer à la fois les poids et les meilleurs emplacements des points de contrôle [Jalel et al., 2016].

L'algorithme 4 décrit notre adaptation de la méthode de Hooke et Jeeves pour le problème de planification de chemin en termes d'une courbe NURBS.

Algorithm 4: Hooke&Jeeves-based NURBS curve parameterization

Input: Ω, P, p, W, U, c_{max}, d, n_iter, StepSize, MinStepSize

Output: Planned path C

```
1  Begin
2      C_0 = NURBSCurveGeneration(P, p, W, U) ;
3      it = 1 ;
4      terminate = 0 ;
5      While (it ≤ n_iter ∧ ∼ terminate) do
6          (C_0, T1) = CollisionDetectionAndRepair(C_0, Ω) ;
7          T2 = CurvatureProblemDetection(C_0, c_max) ;
8          If T2 ∼= ∅ then
9              If T1 ∩ T2 == ∅ then
10                 (C_0, OK) = WeightsDecreasing(C_0, T2) ;
11                 If ∼ OK then
12                     Pts2beMoved = T2 ;
13                     C_0 = Hooke&JeevesForNodesDisplacing(C_0, Pts2beMoved,
14                     StepSize, MinStepSize, c_max, Ω, f) ;
15
16                 Else
17                     terminate = 1 ;
18
19             Else
20                 Pts2beMoved = T1 ∩ T2 ;
21                 C_0 = Hooke&JeevesForNodesDisplacing(C_0, Pts2beMoved,
22                 StepSize, MinStepSize, c_max, ω, f) ;
23
24         Else
25             terminate = 1 ;
26         it = it +1 ;
27     return (C = C_0) ;
28 End
```

Cet algorithme débute par générer une solution initiale C_0 en attribuant des valeurs de poids positives et aléatoires (ou même tous initialisés à 1) aux différents points de contrôle. Ensuite, l'algorithme opère itérativement tant qu'il n'a pas atteint un nombre maximal d'itérations et qu'aucune solution valide n'est encore déterminée.

Ainsi, l'algorithme commence, en premier lieu, par valider la contrainte de la non collision en attirant la courbe vers les points de contrôle ($T1$) les plus proches des zones de collision. En second lieu, la contrainte de limite de courbure le long du chemin calculé est vérifiée. Soit $T2$ l'ensemble des points de contrôle les plus proches des points de la courbe qui ne vérifient pas cette contrainte. Si cet ensemble est vide alors l'algorithme s'arrête et renvoie C_0 comme solution.

Dans le cas contraire, deux sous cas sont possibles : le premier est lorsque l'ensemble $T2$ n'intersecte pas $T1$. Dans ce cas, l'algorithme procède par réduire les poids associés à ces points sans que ces derniers soient négatifs (pour garantir la propriété de l'enveloppe convexe). Si cette procédure aboutit à réajuster la courbe en respectant les contraintes du système, l'algorithme s'arrête en renvoyant la solution actuelle. Sinon, l'algorithme fait appel à la méthode de Hooke et Jeeves pour déplacer localement ces points (ensemble $T2$) afin d'avoir une solution réalisable. Le deuxième sous cas se présente lorsque les deux ensembles des points critiques $T1$ et $T2$ s'intersectent. De même, dans ce cas notre algorithme utilise la méthode de Hooke et Jeeves pour déplacer les points d'intersection afin de trouver la solution souhaitée.

Pour chacun des points à déplacer, la procédure de Hooke et Jeeves opère comme décrit dans l'organigramme de la figure 3.18. La fonction objectif f employée est celle qui renvoie l'écart type des courbures le long du chemin calculé :

$$f(C) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(c_i - c_{mean})^2} \tag{3.25}$$

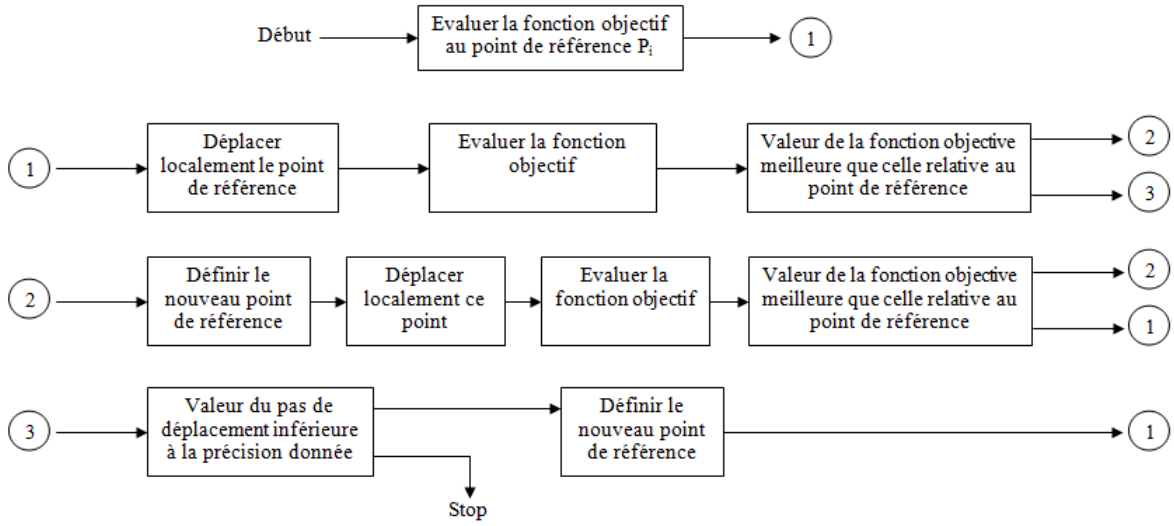


Figure 3.18 — Organigramme descriptif de la méthode de Hooke et Jeeves.

3.4 Résultats de simulation

Notre planificateur de chemins est destiné à déterminer un chemin optimal ou quasi-optimal entre deux positions données. Il doit alors converger vers une solution lorsque la position finale est atteignable. Nous avons implémenté cet algorithme en MATLAB et nous l'avons testé pour évaluer sa performance.

En l'absence d'une base de données contenant des scènes de test (benchmark) pour les algorithmes de planification de chemins, nous avons introduit plusieurs cas d'études ou scénarios afin de montrer l'adaptabilité et l'efficacité de l'approche proposée. Les différents paramètres d'entrée liés à l'environnement et au robot sont résumés dans le tableau 3.1. Ces paramètres sont :

- $N * M$: Dimensions de la scène.
- S : Coordonnées de la position initiale.
- T : Coordonnées de la position finale.
- d : Diamètre du robot cercle.
- ρ_{min} : Rayon de courbure minimum du robot.

Les différentes scènes considérées sont de différentes complexités et inspirées des travaux tirées de la littérature. Ces scénarios sont connus sous les appellations ci-

dessous :

- Map I : Double U-shaped obstacles.
- Map II : Back and forth.
- Map III : Narrow passage.
- Map IV : Dense environment.
- Map V : Maze like environment.

Nous commençons par présenter les résultats de simulation obtenus en utilisant l'algorithme génétique. Puis, nous présentons les résultats obtenus en utilisant l'algorithme 4. Ainsi, les valeurs des différents paramètres de l'AG (taille du chromosome, taille de la population, probabilités de croisement et mutation, nombre de générations) sont résumés dans le tableau 3.2.

Tableau 3.1 — Les paramètres d'entrée liés à l'environnement et au robot pour les différents scénarios.

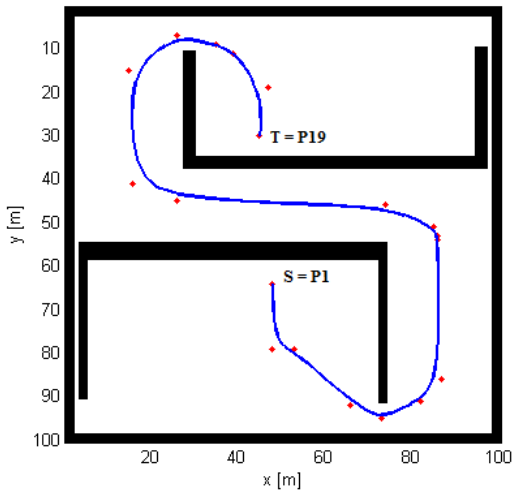
Paramètres	**Map I**	**Map II**	**Map III**	**Map IV**	**Map V**
$N * M$	100*100	100*120	100*110	100*100	150*150
S	(48,64)	(91,93)	(33,86)	(8,36)	(79,83)
T	(45,30)	(9,15)	(78,6)	(73,17)	(78,29)
d	4	2	2	2	5
ρ_{min}	3	1.5	1	0.5	0.8

Tableau 3.2 — Les paramètres d'entrée de l'AG pour les différents scénarios.

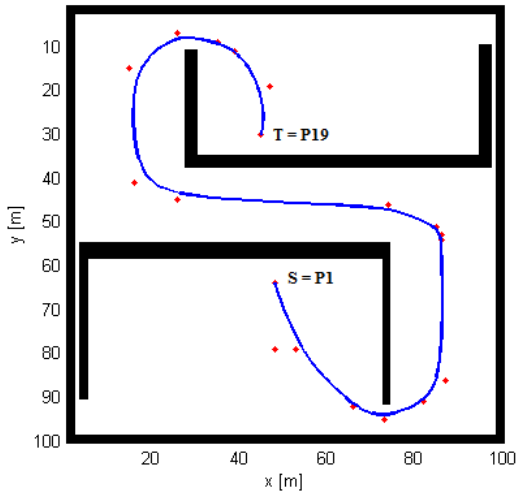
Paramètres	**Map I**	**Map II**	**Map III**	**Map IV**	**Map V**
Taille du chromosome	19	19	18	11	48
Taille de la population	30	20	80	40	20
Probabilité de croisement	0.5	0.4	0.4	0.5	0.5
Probabilité de mutation	0.1	0.05	0.1	0.1	0.1
Nombre de générations	25	50	100	40	100

Les chemins générés, qui sont des courbes NURBS de degré 3, sont illustrés dans les figures 3.19 à 3.22. Pour ces solutions obtenues, différentes valeurs des paramètres

de l'algorithme génétique ont été utilisé. Nous avons varié la taille de la population de 20 à 80, le nombre d'itérations de 25 à 100, la probabilité de croisement de 0.4 à 0.5 et celle de mutation de 0.05 à 0.1.

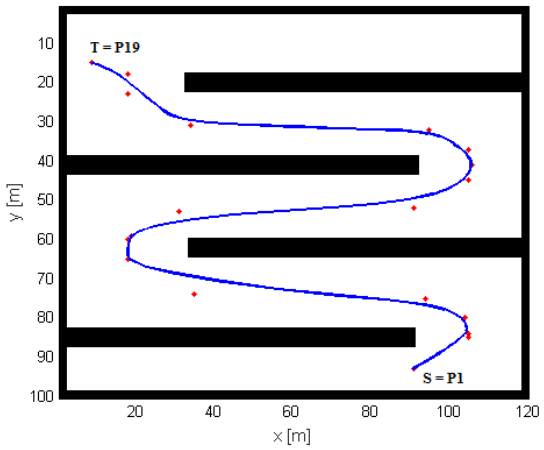


(a) Meilleure solution de la première génération, W = (6.44, 9.61, 8.30, 2.03, 7.52, 7.60, 6.51, 9.15, 1.41, 3.33, 5.53, 3.55, 4.95, 8.66, 9.46, 5.16, 15.05, 10.81, 9.19).

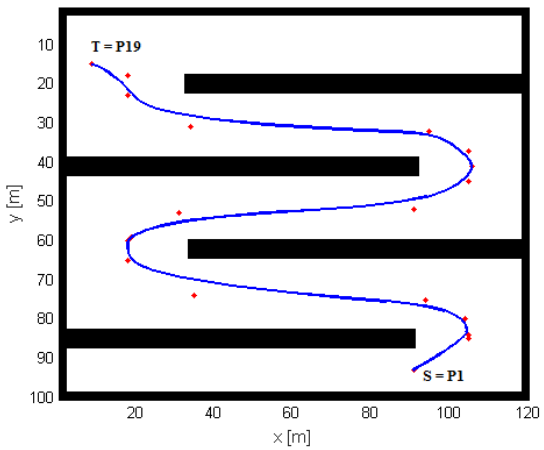


(b) Meilleure solution de la dernière génération, W = (9.78, 0.07, 12.29, 14.47, 12.14, 8.64, 6.51, 9.15, 1.41, 3.33, 5.53, 3.55, 4.95, 8.66, 9.46, 5.16, 15.05, 10.81, 9.19).

Figure 3.19 — Résultats de simulation sous le Map I.

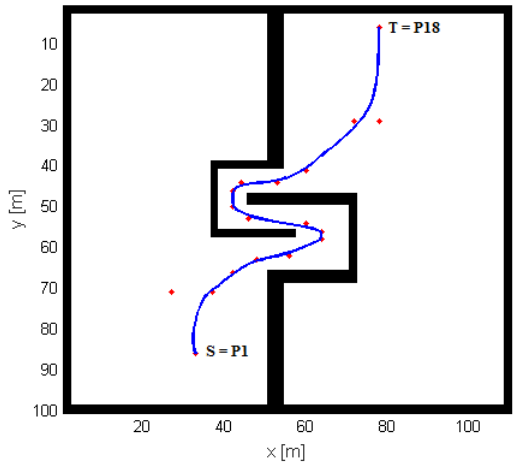


(a) Meilleure solution de la première génération, W = (9.22, 5.49, 14.56, 13.19, 6.62, 2.39, 9.87, 2.50, 4.89, 2.33, 2.51, 3.88, 7.33, 6.52, 13.52, 4.41, 2.60, 8.57, 11.61).

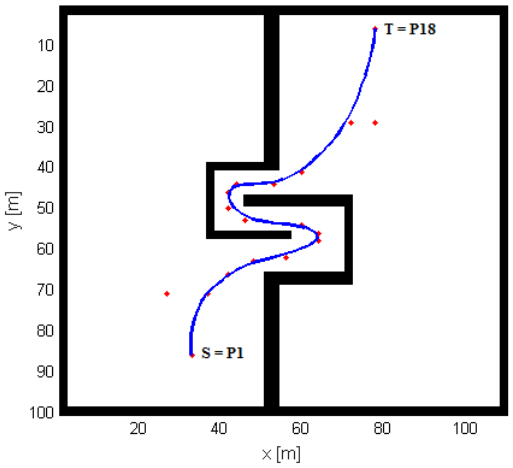


(b) Meilleure solution de la dernière génération, W = (9.22, 5.49, 14.56, 13.19, 6.62, 2.39, 6.18, 6.31, 4.89, 2.02, 1.66, 2.25, 7.33, 4.69, 11.48, 4.31, 13.62, 8.57, 11.61).

Figure 3.20 — Résultats de simulation sous le Map II.

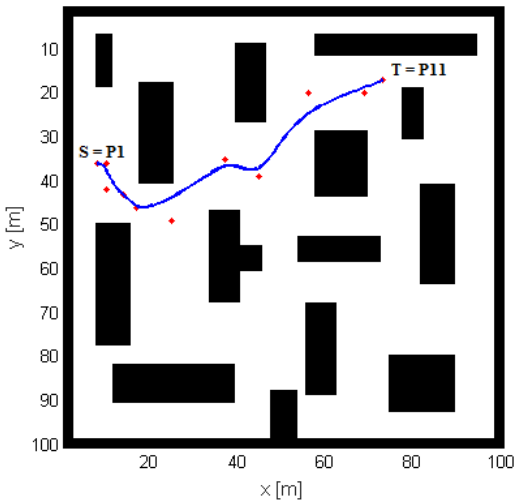


(a) Meilleure solution de la première génération, W = (8.31, 3.13, 13.07, 3.43, 9.11, 3.79, 13.64, 9.65, 2.26, 3.60, 11.87, 13.16, 8.40, 13.56, 14.32, 4.58, 7.59, 6).

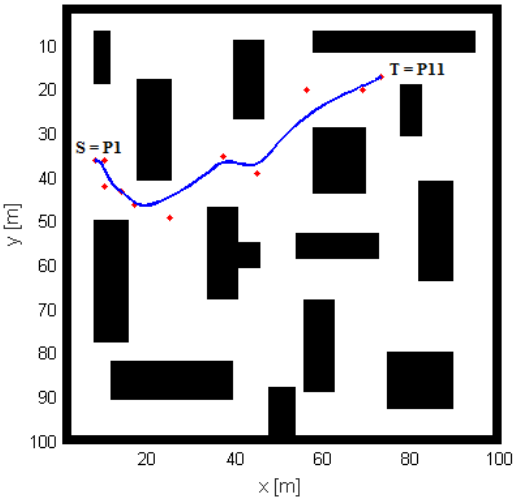


(b) Meilleure solution de la dernière génération, W = (14.93, 3.13, 13.07, 6.60, 8.67, 3.92, 10.20, 12.98, 7.18, 1.78, 1.03, 7.41, 13.82, 12.62, 7.36, 4.21, 1.46, 10.08).

Figure 3.21 — Résultats de simulation sous le Map III.

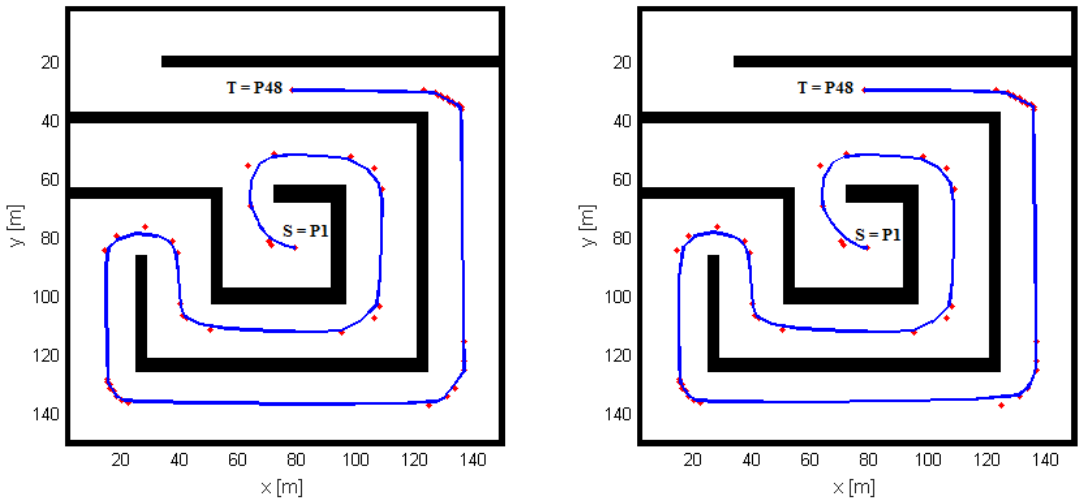


(a) Meilleure solution de la première génération, W = (5.83, 5, 1.64, 5.65, 5.76, 1.25, 6.97, 9.95, 8.67, 6.13, 14.12).



(b) Meilleure solution de la dernière génération, W = (3.40, 5.56, 4.55, 12.87, 14.70, 4.06, 11.94, 7.55, 4.91, 9.34, 14.12).

Figure 3.22 — Résultats de simulation sous le Map IV.



(a) Meilleure solution de la première génération, W = (8.86, 3.67, 8.52, 14.56, 9.04, 14.27, 8.61, 2.82, 10.22, 7.27, 1.86, 13.76, 11.92, 8.23, 11.25, 13.92, 10.08, 7.50, 2.34, 3.70, 14.71, 13.65, 10.83, 2.89, 3.91, 7.69, 10.06, 10.02, 14.05, 11.78, 4.35, 0.73, 13.05, 14.24, 1.22, 4.34, 0.94, 3.21, 7.22, 7.98, 3.29, 2.80, 11.17, 8.26, 0.67, 2.09, 1.28, 13.22).

(b) Meilleure solution de la dernière génération, W = (10.76, 3.46, 2.15, 11.29, 9.04, 14.27, 8.61, 2.82, 8.22, 7.27, 1.86, 13.76, 11.92, 8.23, 11.25, 13.92, 10.08, 7.50, 2.34, 1.98, 3.85, 7.57, 4.13, 7.29, 3.29, 9.63, 9.72, 15.06, 4.17, 0.92, 4.21, 14.17, 8.38, 8.31, 11.43, 9.07, 10.74, 3.39, 6.41, 4.36, 9.02, 8.51, 7.60, 7.02, 7.79, 7.14, 8.54, 10.02).

Figure 3.23 — Résultats de simulation sous le Map V.

Les résultats obtenus sont satisfaisants et respectent les contraintes du problème (voir le tableau 3.3, qui représente la valeur de la fonction d'évaluation, l'écart type de courbure, la longueur du chemin, la courbure maximale le long de ce chemin et le temps mis pour calculer cette solution). Le seul inconvénient étant le temps d'exécution, en fait les algorithmes génétiques sont coûteux en temps de calcul, puisqu'ils manipulent plusieurs solutions simultanément et c'est le calcul de la fonction d'évaluation (fitness) qui est le plus pénalisant. Toutefois, pour ces évaluations, nous avons utilisé un ordinateur de configuration limitée. Une implémentation sur des machines beaucoup plus performantes vont certes améliorer le temps d'exécution.

Tableau 3.3 — Résultats expérimentaux.

Paramètres	**Map I**	**Map II**	**Map III**	**Map IV**	**Map V**
$f(C)$	2.32	3.09	1.35	0.81	5.69
Écart-type de courbures	0.07	0.19	0.22	0.21	0.25
Longueur	229.53	308.06	132.97	79.22	567.13
Max(| courbure |)	0.29	0.65	0.94	0.75	1.1
Temps (s)	216	290	1969	449	545

Le temps de calcul pris par un algorithme génétique croît en fonction du nombre d'itérations (voir Figure 3.24 qui illustre l'augmentation du temps de calcul avec le nombre de générations pour le scénario du Map IV), la taille de la population et le taux de modification des individus (probabilités de croisement et de mutation). Ceci justifie le temps de calcul élevé noté pour le troisième exemple (Map III). En effet, le nombre d'itérations étant égal à 100, la taille de population est de 80 individus avec des probabilités de modification égales à 0.4 pour le croisement et 0.1 pour la mutation.

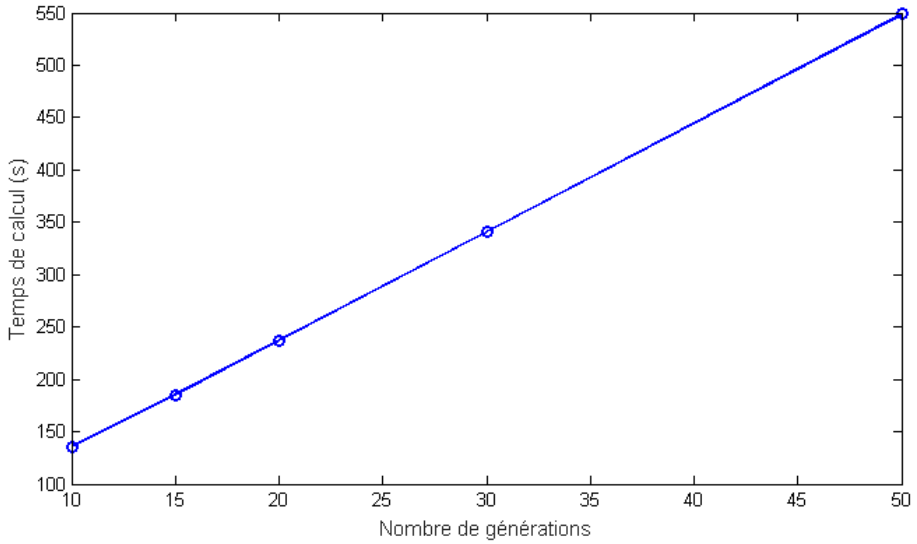


Figure 3.24 — Évolution du temps de calcul en fonction du nombre de générations.

À notre connaissance, il n'existe pas de techniques permettant de fixer, à priori, le nombre d'itérations pour un algorithme génétique. Toutefois, nous remarquons que pour le problème de planification de chemins, il n'est pas nécessaire de trop augmenter le

nombre de générations. En fait, pour les simulations présentées ci-dessus, la convergence vers la solution optimale est assurée à partir des itérations 7, 12, 8, 14 et 18 pour les Maps I, II, III, IV et V respectivement. De ce fait, un nombre de générations de 20 à 50 est suffisamment adéquat pour la résolution de ce problème. De même, une taille de population entre 20 et 40 est largement suffisante.

Nous notons aussi que les chemins obtenus ont une valeur faible de l'écart-type des courbures, ce qui indique l'homogénéité de la distribution de ce paramètre. En effet, plus l'écart-type des courbures est faible, plus les valeurs des courbures sont regroupées autour de la moyenne ce qui implique le critère de continuité de courbure renforçant, par conséquent, la faisabilité du chemin généré.

Contrairement aux méthodes citées dans le tableau 3.4 et qui s'appuient sur les algorithmes génétiques, l'algorithme proposé assure non seulement les propriétés de sécurité et de longueur minimale, mais respecte également la limite de courbure admissible le long de la courbe calculée. L'idée principale est d'exploiter l'effet géométrique du paramètre poids des courbes NURBS lors de la génération du chemin.

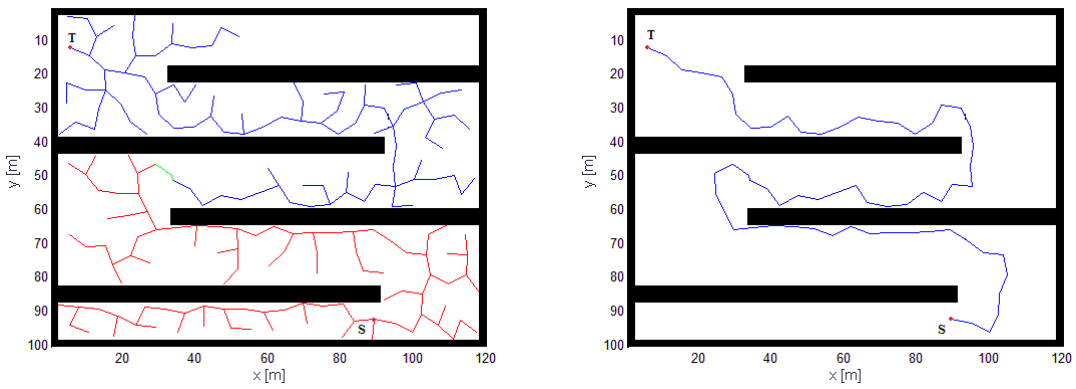
Tableau 3.4 — Caractéristiques de sortie des planificateurs de chemin se basant sur les algorithmes génétiques

Paramètres	**Évitement d'obstacles**	**Longueur minimale**	**Continuité de courbure**	**Limite de courbure**
[Tu and Yang, 2003]	✓	✓	✗	✗
[Gemeinder and Gerke, 2003]	✓	✗	✗	✗
[Sedighi et al., 2004]	✓	✓	✗	✗
[Yun et al., 2010]	✓	✓	✗	✗
[Tamilselvi et al., 2012]	✓	✓	✗	✗
[Tuncer and Yildirim, 2012]	✓	✓	✗	✗
[Qu et al., 2013]	✓	✓	✗	✗
[Ahmed and Deb, 2013]	✓	✓	✗	✗
Approche proposée	✓	✓	✓	✓

Bien que plusieurs efforts se sont déployés afin de développer des algorithmes de planification de chemins, certains problèmes persistent. En effet, aucune méthode n'est parfaite. L'évaluation des performances de ces méthodes s'avère pertinente pour une meilleure compréhension des problèmes constatés. Elle peut contribuer à mieux saisir la dimension qualité pour un planificateur de chemin.

La qualité de l'approche de planification de chemin est évaluée en fonction du succès des expérimentations menées [Caihol, 2015]. Ce **taux de succès** est défini comme le nombre de chemins réalisables (sans collision et respectant la contrainte de courbure) divisé par le nombre d'essais menés. Pour cela, nous avons refait 100 fois les simulations précédentes, tout en modifiant les positions initiales et finales, le diamètre et le rayon de courbure minimum du robot. Le taux de succès de notre algorithme était de 94%.

Ci-dessous, nous présentons le chemin calculé, pour le Map II, par une variante des méthodes à base de Rapidly Exploring Random Tree qui est la 'RRT bidirectionnel' dont le principe consiste à développer deux arbres de diffusion, le premier à partir de la configuration initiale et le deuxième ayant comme racine la configuration finale [Kala, 2014]. Cette méthode calcule un chemin sans collision joignant deux configurations pour un robot assimilé à une masse ponctuelle (en supposant que les obstacles sont augmentés d'une distance de sécurité de façon à prendre en compte la dimension réelle du robot). De ce fait, nous pouvons comparer notre approche face à la 'RRT bidirectionnel' en terme de la longueur du chemin. Comme montré dans la figure 3.25 (b), le résultat obtenu par cette méthode est loin d'être optimal (contrairement à celui obtenu par notre approche (Fig. 3.20)).



(a) Expansion des deux arbres.

(b) Chemin calculé

Figure 3.25 — RRT bidirectionnel : Résultats de simulation sous le Map II.

Pour l'évaluation de la méthode de lissage basée sur la technique de Hooke et Jeeves (algorithme 4), les mêmes simulations ont été menées. Les différents chemins générés sont illustrés dans la figure 3.27. Le tableau 3.5 résume les résultats expérimentaux obtenus.

Pour ces simulations, l'algorithme proposé réussi à produire des solutions réalisables par des simples ajustements (augmentation/diminution) des valeurs des poids associés aux différents points de contrôle. Le temps de calcul était très faible par rapport au temps pris par l'application de l'algorithme génétique (voir figure. 3.26). Par contre, ces chemins ne sont pas plus optimaux en termes de longueur, maximum de courbures et écart-type des courbures.

Tableau 3.5 — Résultats expérimentaux.

Paramètres	**Map I**	**Map II**	**Map III**	**Map IV**	**Map V**
Écart-type de courbures	0.10	0.16	0.20	0.22	0.24
Longueur	230.89	308.84	137.17	80.13	569.51
Max(| courbure |)	0.32	0.58	0.98	0.92	1.02
Temps (s)	14	13	12	27	22

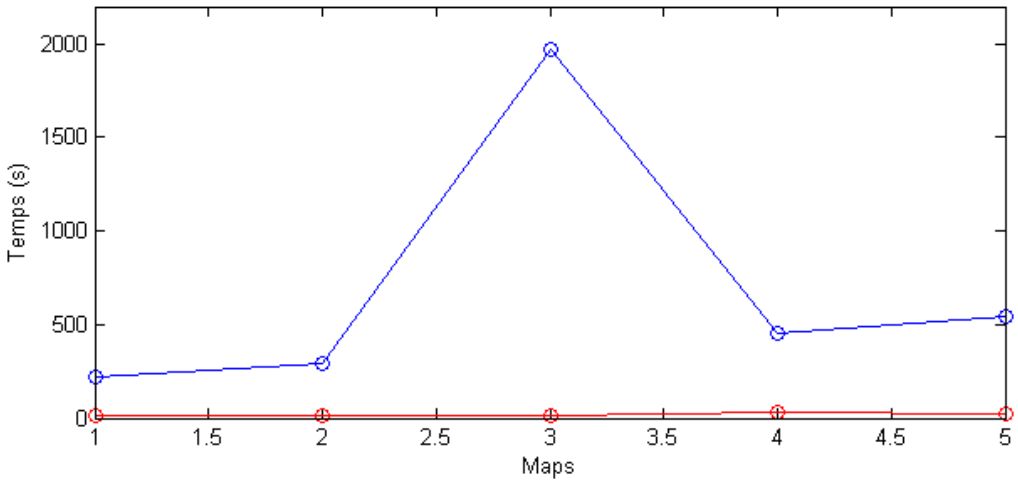
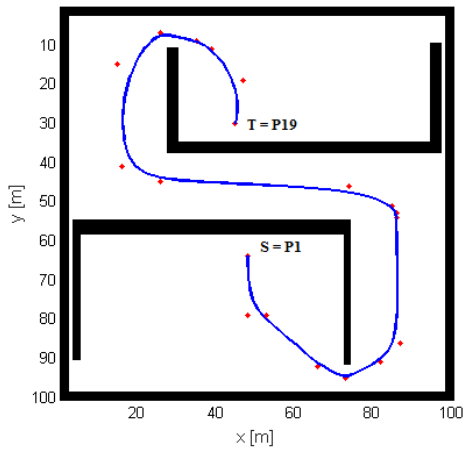
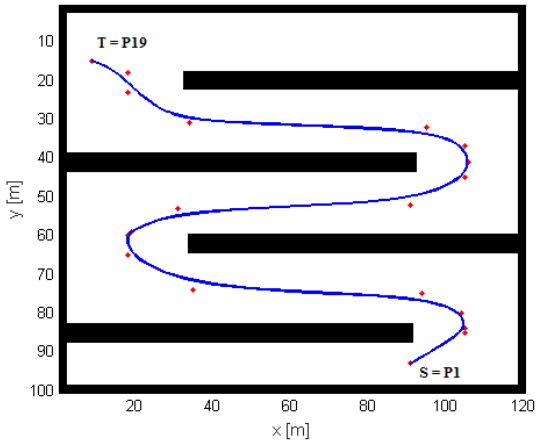


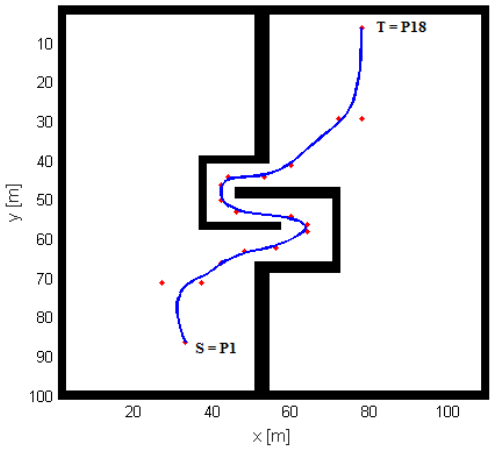
Figure 3.26 — Comparaison entre les temps de calcul pris par l'algorithme génétique (en bleu) et par l'algorithme 4 (en rouge).



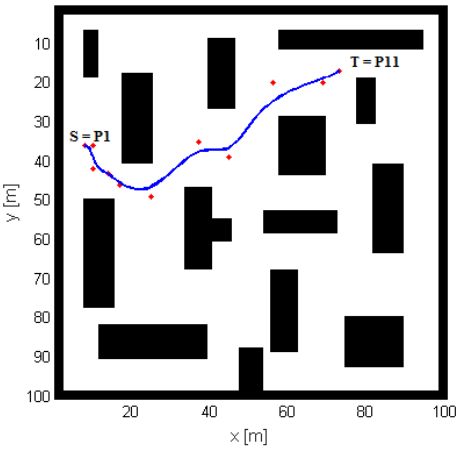
(a) Résultat de simulation sous le Map I.



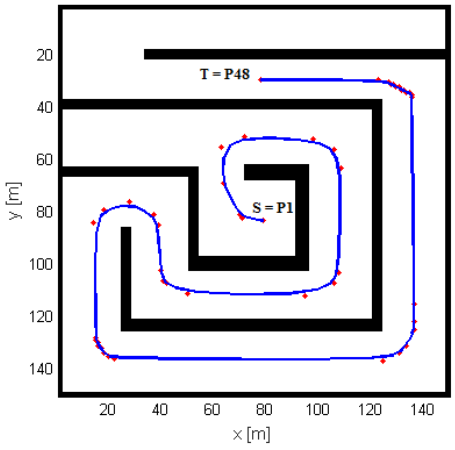
(b) Résultat de simulation sous le Map II.



(c) Résultat de simulation sous le Map III.



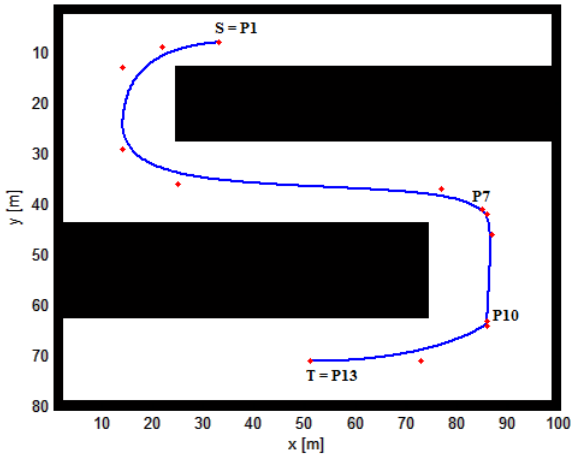
(d) Résultat de simulation sous le Map IV.



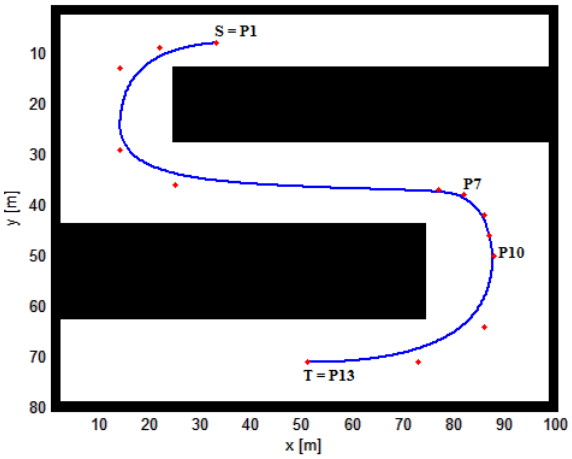
(e) Résultat de simulation sous le Map V.

Figure 3.27 — Résultats de simulation sous le Map V.

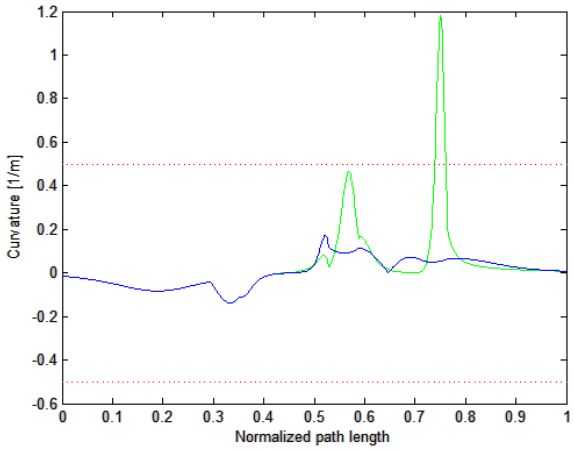
L'exemple ci-dessous montre le succès de l'utilisation de la méthode de Hooke et Jeeves à générer une solution réalisable dans le cas où il n'est pas possible de l'assurer en ajustant seulement les poids des points de contrôle.



(a) Chemin généré avant le processus de correction de courbure ($\sigma(courbures) = 0.16$).



(b) Chemin généré après le processus de correction de courbure ($\sigma(courbures) = 0.06$).



(c) Profils de courbure avant (vert) et après (bleu) le processus de correction de courbure, les lignes en pointillés rouges représentent la contrainte de limite de courbure.

Figure 3.28 — Illustration de l'exécution de la méthode de Hooke et Jeeves.

En effet, pour cet exemple la méthode proposée a permis de fournir, dans un premier lieu, un plus court chemin libre des obstacles (Figure. 3.28(a)). Puis, en second lieu, la solution a été raffinée en modifiant localement le chemin généré précédemment suite au déplacement des deux points de contrôle $P7$ et $P10$ de sorte que la valeur de courbure maximale le long de cette courbe soit inférieure à la valeur limite autorisé par

le robot (Figure. 3.28(b)). La figure 3.28(c) montre le profil de courbure avant et après l'ajustement de la courbe. La valeur de courbure maximale le long du chemin est égale à 0.17 ce qui assure la contrainte de limite de courbure (la valeur maximale admissible étant de 0.5).

Bien que les résultats obtenus par l'algorithme 4 sont moins optimaux que ceux calculés par l'algorithme génétique, le temps de calcul pris par cet algorithme s'avère plus intéressant. De même, le taux de succès de cette approche est de l'ordre de 97%.

3.5 Conclusion

Dans ce chapitre, nous avons détaillé nos contributions concernant la planification de chemin pour les robots mobiles non holonomes. Nous avons développé un planificateur de chemin permettant de générer des chemins libres des obstacles et avec contrainte de limite de courbure.

La principale qualité de ce planificateur consiste à intégrer les courbes NURBS dans le processus de modélisation du chemin en s'appuyant sur la propriété de contrôle local. Ainsi, deux procédures de lissage ont été proposées : La première utilisant un algorithme génétique pour mieux estimer les valeurs des poids des différents points de contrôle. Quant à la deuxième, elle se base non seulement sur l'ajustement des valeurs des poids, mais aussi sur le déplacement de certains de contrôle si nécessaire en utilisant la méthode de Hooke et Jeeves.

Les deux techniques ont donné des taux de succès intéressants. La principale limite de la première technique était le temps de calcul élevé engendré par la manipulation de plusieurs solutions à la fois et pour plusieurs itérations. Quant à la deuxième technique, les solutions fournies étaient moins optimisées mais toujours admissibles et avec un temps de calcul faible. Il est à noter qu'un temps de calcul un peu élevé, pour une planification **offline**, ne constitue pas un vrai soucis vu que le robot ne commencera son mouvement qu'une fois la solution est déterminée. Enfin, tout dépend des spécificités de l'application (l'ordre des priorités de certains critères comme le temps de calcul, l'homogénéité de la solution, la longueur du chemin, etc...).

Dans le chapitre suivant, nous proposons une nouvelle solution pour le problème de planification de trajectoires dans un environnement dynamique.

CHAPITRE 4

Navigation autonome dans un environnement dynamique

4.1 Introduction

Pour la robotique mobile, la mise au point de planificateurs de trajectoires capables d'orienter les choix vers un maximum d'efficacité occupe une importance cruciale. L'un des problèmes majeurs en robotique réside dans la mise en place des robots autonomes. Par autonome, nous entendons que moyennant une spécification de haut niveau de la tâche à exécuter par le robot, celui-ci sera capable de la réaliser sans aucune intervention extérieure.

La mise au point de tels planificateurs est une tâche complexe qui nécessite une modélisation fine et exacte de l'environnement de travail et une définition précise des objectifs à atteindre. En général, les approches de planification de trajectoires se classifient en deux grandes familles : les méthodes dites réactives ou sans trajectoire de référence et les celles qui opèrent avec une trajectoire de référence. Ces dernières méthodes consistent à modifier itérativement, au cours du temps, un mouvement initialement calculé afin de garantir au mieux la convergence vers le but.

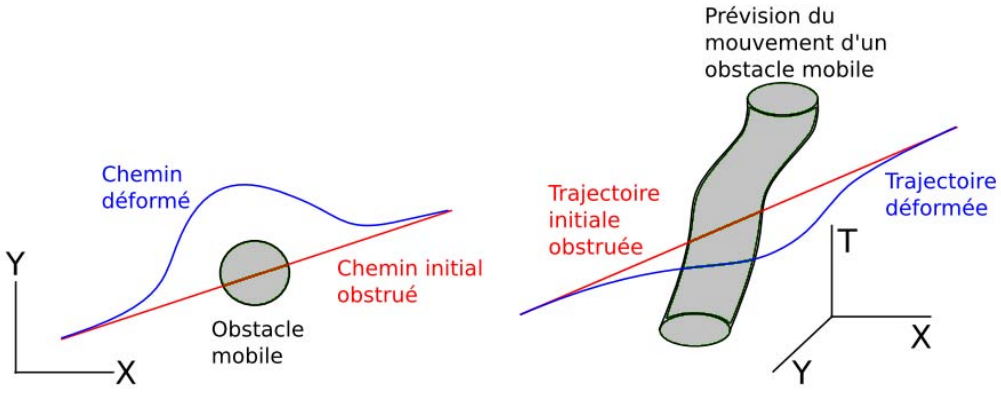
Dans ce chapitre nous présentons une nouvelle approche de déformation de trajectoire permettant à un robot mobile de naviguer en toute sécurité dans un environnement dynamique de façon à s'éloigner des obstacles et à satisfaire les contraintes cinématiques du système. Ce chapitre est organisé comme suit : les motivations et les contributions sont présentées dans la section suivante. Nous enchainons ensuite avec

une formalisation du problème et nous décrivons en détails l'approche proposée. Une étude théorique sur la validité de l'approche est introduite avant de conclure sur le travail apporté.

4.2 Motivations et contribution

Bien que la déformation de chemin consiste à éloigner à chaque instant le chemin suivi par le robot de la position courante des obstacles, la déformation de trajectoire vise à anticiper leurs mouvements en écartant cette trajectoire du modèle prévisionnel des mouvements futurs des objets mobiles. En fait, la trajectoire suivie se décompose elle-même en un chemin (route) suivi par le robot et en une loi de vitesse (cinématique). Ainsi une trajectoire nous indique, à chaque instant, la position du robot sur son chemin. Mathématiquement, une trajectoire est décrite par une courbe paramétrique qui représente le chemin à parcourir dans le temps.

Il est important de remarquer ici que lorsqu'il s'agit d'un environnement dynamique, un planificateur de mouvement est amené à prévoir des déformations de trajectoires pour éviter les éventuelles collisions. À cet effet, le planificateur peut aussi bien déformer le chemin (géométrique) qu'il comptait suivre, que modifier sa vitesse pour éviter un obstacle : pour éviter un animal qui traverserait une route, un conducteur d'automobile pourra soit freiner (soit au contraire accélérer !), soit modifier (légèrement) son chemin (obliquer à droite ou à gauche). les concepts de déformation de chemin et de trajectoire sont illustrés dans la figure 4.1 [Delsart, 2010].



(a) (b)

Figure 4.1 — Déformation de chemin (a) vs. déformation de trajectoire (b).

La revue des travaux de la littérature montre que la technique de déformation de trajectoire n'était pas assez développée. Récemment, Delsart et Fraichard [Delsart, 2010] ont proposé une nouvelle méthode de déformation de trajectoire s'appuyant sur une approche délibérative donnant un mouvement initialement planifié (supposé donné) et un évitement réactif des obstacles au cours du déplacement du robot. Cet évitement réactif a été géré en appliquant des forces répulsives. L'application de ces forces pose un problème de connectivité de la suite des états à atteindre. Pour remédier à ce problème et assurer la faisabilité du mouvement calculé, des forces internes ont été appliquées.

C'est dans cette optique que s'inscrit le travail que nous présentons dans ce chapitre et qui consiste en une nouvelle approche de déformation de trajectoire pour un robot mobile non holonome. Notre méthode vise à étendre l'approche de planification de chemin en termes d'une courbe NURBS en utilisant un algorithme génétique (chapitre 3). L'idée est de concevoir une trajectoire modélisée par une courbe NURBS qui soit flexible. Ceci veut dire que cette approche, en plus de l'exploitation de la propriété du contrôle local (en autorisant la modification des emplacements ainsi que les poids des points de contrôle), elle autorise également l'ajustement de la loi de vitesse considérée tout en respectant les contraintes cinématiques du système robotique étudié.

4.3 Approche proposée de déformation de trajectoire

4.3.1 Formulation du problème et hypothèses

Dans cette section, nous présentons le modèle de l'environnement considéré ainsi que les hypothèses prises en compte pour le calcul de la solution du problème de planification de trajectoire.

Nous considérons que l'environnement est représenté par une carte 2D contenant à la fois des obstacles statiques et dynamiques (Figure. 4.2). Comme dans le chapitre précédent, les obstacles statiques sont de formes rectangulaires et de différentes tailles et emplacements. Quant aux obstacles dynamiques, ils sont supposés circonscrits dans des cercles de différents diamètres.

L'objectif étant de guider le robot pour se déplacer entre les deux positions de départ et d'arrivée dans un environnement changeant tout en respectant, à la fois, les contraintes cinématiques du système robotique et celles qui régissent l'évolution

de l'environnement (contraintes dynamiques). Ainsi, la trajectoire à générer doit éviter tout type d'obstacles, respecter la limite de courbure et respecter les contraintes cinématiques du robot.

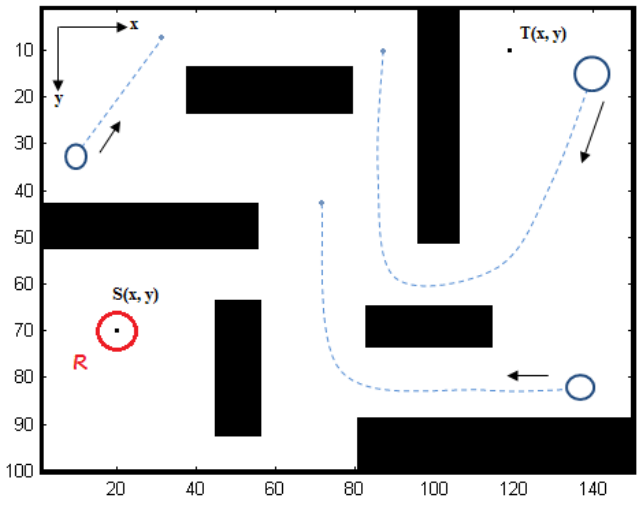


Figure 4.2 — Formulation du problème de planification de trajectoire à t_0 en termes de notre approche.

Dans ce travail, les hypothèses suivantes sont retenues afin de répondre à la problématique de navigation en environnement dynamique :

- Les obstacles mobiles peuvent suivre différentes trajectoires de différentes géométries (non seulement des segments de droites comme la plupart des travaux de la littérature) modélisées aussi par des courbes NURBS.
- Dès qu'un obstacle mobile est détecté par le robot, ce dernier dispose d'un prédicteur **parfait** ($\simeq$ l'environnement est parfaitement connu) et est en mesure de connaître la dimension et la trajectoire suivie par cet objet mobile. En effet, si ce n'est pas le cas, le robot ne sera jamais en mesure de connaître la trajectoire de l'objet mobile et il raisonnera par extrapolation en supposant que cet objet évolue dans la même direction et avec la même vitesse.
- Lors de son déplacement le robot se rend compte de l'existence d'un ou plusieurs objets mobiles s'ils sont bien situés dans son champ de vision.
- Un intervalle de temps largement suffisant est autorisé entre la détection d'obstacles mobiles et la mise en oeuvre des actions nouvellement générées.

4.3.2 Vue d'ensemble de l'approche

Comme l'illustre la figure 4.3, notre approche est basée essentiellement sur deux phases. La première constitue la planification offline d'une trajectoire initiale à $t = t_0$, c'est-à-dire avant que le robot commence son déplacement. La deuxième concerne les mises à jour éventuelles qui commencent avec le début du mouvement du robot (planification online).

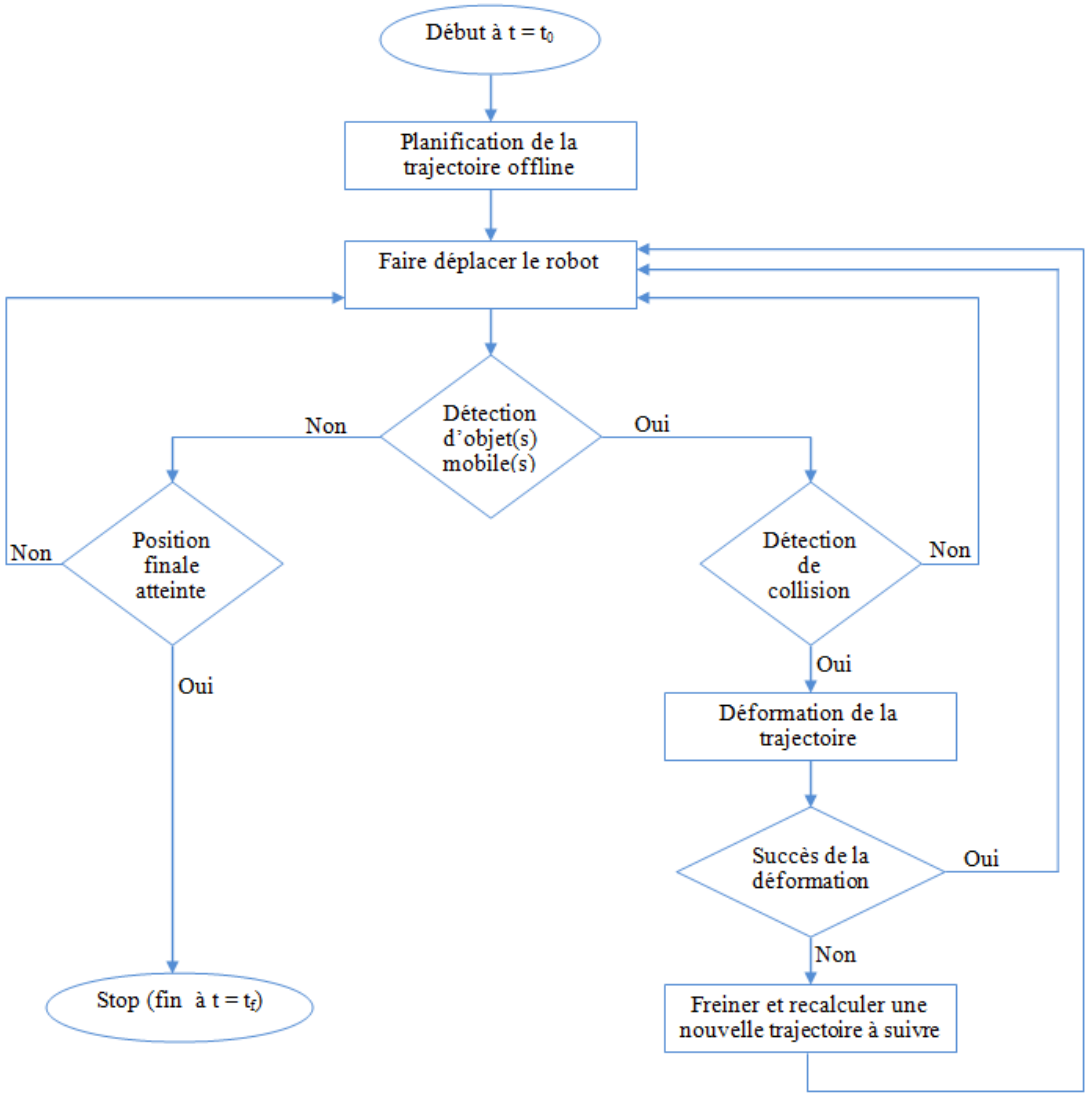


Figure 4.3 — Organigramme de l'approche proposée de navigation en environnement dynamique.

En effet, pour chaque déplacement, le robot met à jour sa perception de l'environnement, dans lequel il évolue, en se basant sur les informations acquises à partir de son champ de vision qui est définit à partir des capteurs dont il est équipé. Ainsi, dès qu'un ou plusieurs objets mobiles sont détectés, le robot agit par la technique de déformation de trajectoire s'il estime l'existence de collisions. Sinon, il continue à se déplacer comme déjà planifié. Un algorithme génétique est mis en place pour le module de déformation de trajectoire qui sera détaillé ultérieurement.

Comme tout algorithme de nature heuristique, un algorithme génétique peut échouer à trouver une solution. Pour cela, si le module de déformation de la trajectoire n'aboutit pas à faire naviguer le robot en toute sécurité, le système n'a qu'à freiner et recalculer une nouvelle trajectoire à suivre pour atteindre son but à partir de sa position actuelle.

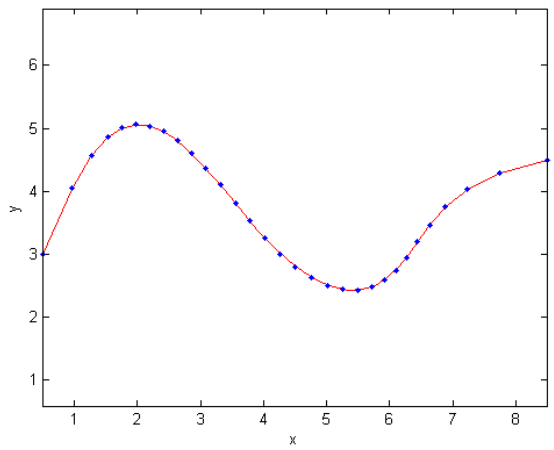
4.3.3 Planification Offline

La différence majeure entre une trajectoire et un chemin réside dans la dimension temporelle. En effet, ce dernier nous renseigne seulement sur les lieux géométriques par lesquels le robot doit passer, sans préciser la manière avec laquelle il va y passer. Ainsi, le calcul d'une trajectoire admissible s'avère obligatoire dès que l'environnement dans lequel évolue le robot est jugé dynamique même s'il est, toutefois, possible de planifier une trajectoire dans un milieu statique s'il est souhaitable de minimiser le temps de parcours par exemple.

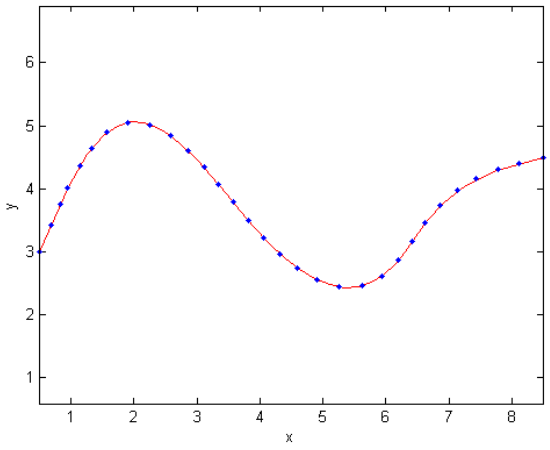
Notre planificateur utilise une trajectoire de référence calculée à $t = t_0$, ce module de planification offline est constitué comme suit : Initialement, un chemin en termes d'une courbe NURBS est calculé (comme expliqué dans le chapitre 3). Avant que le robot commence sa mission, il faut lui doter d'une loi de vitesse qui respecte sa cinématique (vitesse et accélérations maximales). Cette loi de vitesse va nous permettre de repérer sa position pour chaque instant t pour le suivi et le calcul d'éventuelles collisions avec les obstacles mobiles. De ce fait, il faut reparamétriser la courbe NURBS en fonction du temps. Par exemple, si le robot va effectuer ses déplacements avec une vitesse constante, il faut faire une reparamétrisation selon l'abscisse curviligne.

La figure 4.4 illustre deux différentes reparamétrisations d'une courbe NURBS. La première (figure 4.4 (b)) représente celle seoln l'abscisse curviligne. Dans ce cas, le mo-

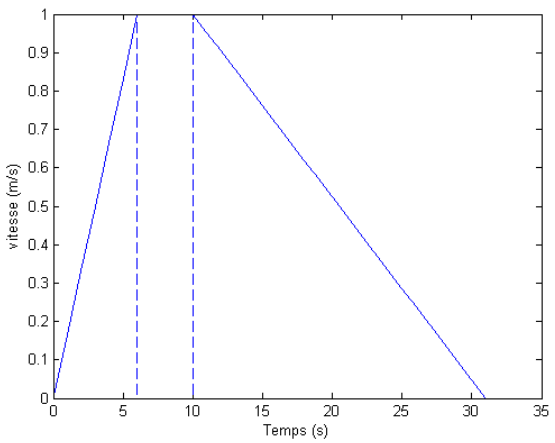
bile est amené à se déplacer à égale distance d'une position à une autre afin d'atteindre sa destination (déplacement avec vitesse constante). Le deuxième scénario (figure 4.4 (d)) concerne la reparamétrisation de la courbe pour un mobile se déplaçant selon une loi de vitesse donnée qui consiste en une vitesse croissante suivie d'une vitesse constante et enfin une vitesse décroissante. Les points en bleu marquent les lieux géométriques du mobile lors de son évolution dans son environnement.



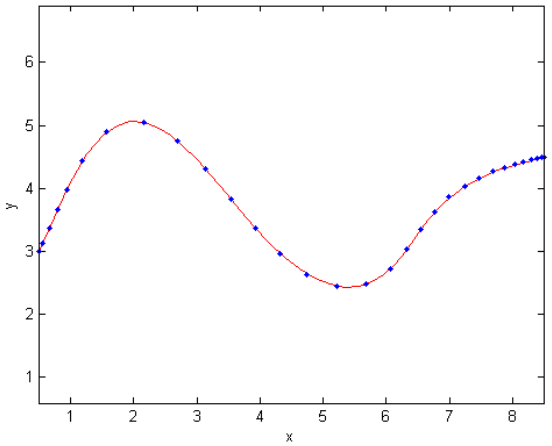
(a) Paramétrisation initiale de la courbe NURBS.



(b) Reparamétrisation de la courbe selon l'abscisse curviligne.



(c) Loi de vitesse donnée.



(d) Reparamétrisation de la courbe selon la loi de vitesse.

Figure 4.4 — Différentes reparamétrisations d'une courbe NURBS.

Pour récapituler, dans ce module, nous commençons à planifier la trajectoire initiale (de référence) du robot ainsi que les différentes trajectoires des différents obstacles. Le

calcul de chacune de ces trajectoires ne tient pas compte des autres. Autrement dit, aucune procédure d'évitement de collisions entre ces différents objets n'est mise en place à ce niveau. Nous notons aussi que les obstacles mobiles sont considérés comme des agents passifs c'est-à-dire qu'ils vont suivre leurs trajectoires telles qu'elles sont calculées.

Un scénario d'une planification offline est présenté dans la figure 4.5, qui illustre une scène comportant un robot ayant pour mission de naviguer de la position initiale S à sa destination finale T, en présence de trois obstacles mobiles M1, M2 et M3. Toutes les trajectoires étant définies en évitant seulement les obstacles statiques. La procédure d'évitement des collisions entre le robot et ces obstacles mobiles sera invoquée au fur et à mesure que le robot commence sa mission, ce qui fait l'objet de la section suivante.

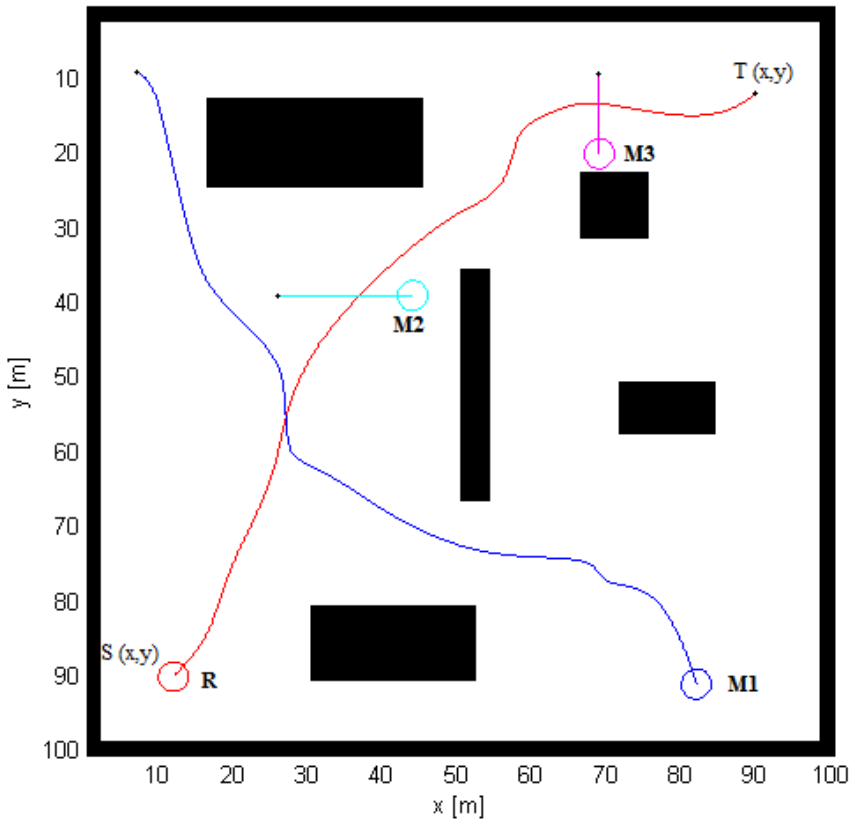


Figure 4.5 — Planification offline à t_0 .

4.3.4 Planification Online

Une fois la trajectoire initiale est déterminée, elle est transmise au robot. Lors de son déplacement, la partie du mouvement restant à être exécutée est déformée continuellement en réponse aux informations sur l'environnement récupérées par les capteurs. Le robot peut ainsi modifier son parcours en fonction des déplacements d'obstacles appartenant à son champ de vision définit par un cercle d'un certain diamètre comme montré dans la figure 4.6 qui illustre les domaines couverts par les capteurs du robots à différents instants de son évolution dans l'environnement.

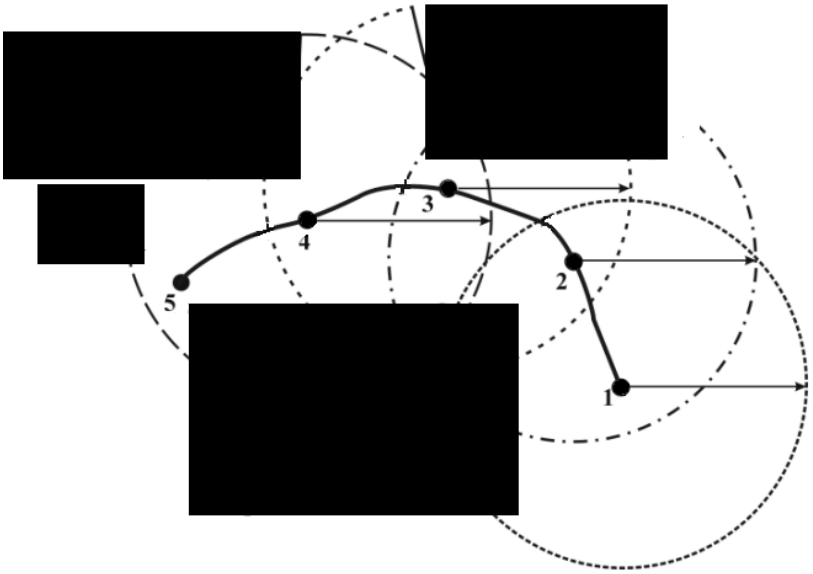


Figure 4.6 — Représentation schématique du champ de vision du robot au cours de son déplacement.

4.3.4.1 Procédure de détection des collisions

La procédure de détection des collisions est décrite dans l'algorithme 5. En fait, cet algorithme estime que le robot est susceptible de heurter un obstacle mobile s'il existe un instant $t_k < t_f$ (t_f étant l'instant d'arrivée du robot à sa destination), tel que la distance entre la position du centre du robot et celle du centre de l'objet mobile soit inférieure à la somme des rayons des deux cercles représentatifs de ces deux objets.

Comme le montre l'organigramme de la figure 4.3, dès que le robot s'aperçoive de l'existence d'un ou plusieurs objets mobiles se déplaçant dans le domaine de l'environnement délimité par son champ de vision, il fait appel à la procédure d'estimation de

collisions probables. Ainsi, si aucun risque n'est dégagé, le robot continue tranquillement son parcours initial. Dans le cas contraire, l'algorithme réagit par une technique de déformation de mouvement pour modifier le reste à être exécuté de sa trajectoire actuelle. En effet, trois scénarios sont possibles afin d'éviter la ou les collisions prévues : soit en changeant la loi de vitesse suivie par le robot (en augmentant ou diminuant la vitesse), soit en changeant seulement le chemin géométrique sans toucher à sa loi de vitesse, soit en modifiant à la fois la géométrie et la vitesse du système robotique.

Algorithm 5: Collision Detection Algorithm

Input: Ω, Planned trajectory of the robot, Trajectory of the mobile obstacle, r_{robot}, r_{obst}, $t_{current}$, $\triangle t$, t_f

Output: Decision of Collision-free or Collision-position

```
1  Begin
2      for (t = t_current : △t : t_f) do
3          Compute robot center position ;
4          Compute obstacle center position ;
5          If (distance(Pos_robot, Pos_obst) < r_robot + r_obst) then
6              return (Collision-position) ;
7
8          Else
9              return (Collision-free) ;
10 End
```

4.3.4.2 Mécanisme de déformation de trajectoire

Pour déformer une trajectoire donnée, nous pouvons agir sur sa composante géométrique (courbe), la cinématique du mobile parcourant cette trajectoire ou bien les deux à la fois. Ci-dessous, nous détaillerons les voies possibles du mécanisme de déformation de trajectoire que nous proposons :

1. Processus de déformation du chemin géométrique : Nous notons que pour chaque déformation de mouvement incluant la déformation du chemin géométrique par lequel le robot doit passer, la génération de la courbe modélisant le reste du chemin à être exécuté doit assurer la continuité de courbure, la limite de courbure liée

au rayon de courbure du robot ainsi que la contrainte de sécurité des déplacements.

Bien évidemment, si la déformation de la trajectoire actuelle du robot inclut la déformation du chemin , ce dernier doit commencer à partir d'un point appartenant au chemin initial, qui présente en fait le premier point de contrôle de la courbe à générer. Ce chemin est, à son tour, une courbe NURBS qui nécessite une définition particulière des points de contrôle. En fait, soient $\{P_1^{'}, P_2^{'}, P_3^{'}, ..., P_m^{'}\}$ l'ensemble des points à approximer avec $P_m^{'}$ la position de la destination du robot.

Comme expliqué précédemment, $P_1^{'}$ est un point appartenant au chemin initial et qui sera déterminé aléatoirement en s'assurant qu'il se situe entre la position actuelle du robot et avant la position de la première intersection avec l'obstacle mobile détecté. Le deuxième point se calcule en respectant la contrainte de continuité de courbure entre les deux courbes. Pour cela, il faut en premier lieu sélectionner le point de contrôle du chemin initial le plus proche du point $P_1^{'}$. En second lieu, il faut le déplacer dans la direction garantissant la continuité de courbure entre les deux courbes. Pour cela, nous avons utilisé la méthode de Hooke et Jeeves. Le déplacement du point est effectué dans le sens qui minimise la valeur de l'angle θ entre la tangente au chemin initial au point $P_1^{'}$ (vecteur $\overrightarrow{v1}$) et le vecteur $\overrightarrow{v2}$ dont les extrémités sont $P_1^{'}$ et le point du chemin initial qui lui est le plus proche (point P_3 pour l'exemple de la figure 4.7) :

$$\theta = arccos\left(\frac{\overrightarrow{v1}.\overrightarrow{v2}}{\| \overrightarrow{v1} \| . \| \overrightarrow{v2} \|}\right) \tag{4.1}$$

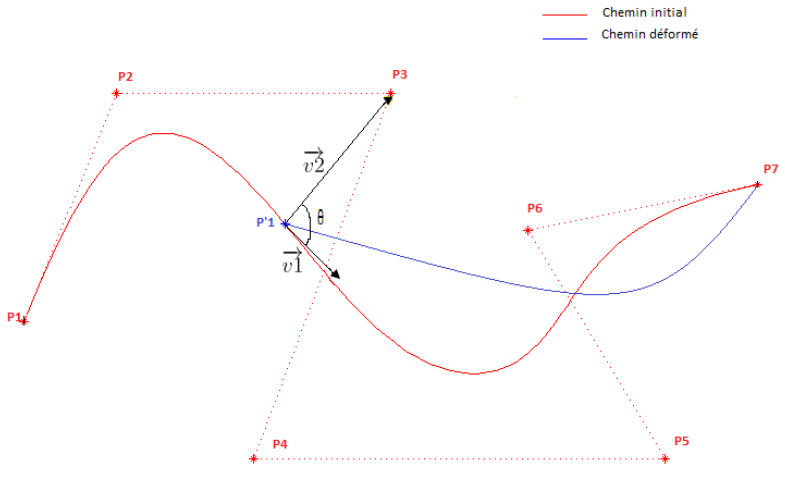


Figure 4.7 — Illustration de l'angle thêta.

Le reste des points de contrôle du nouveau chemin ne sont autre que la suite des points initiaux situés après le point sélectionné pour le calcul de $P_2^{'}$. Ces points seront perturbés aléatoirement (sauf le dernier qui représente la position à atteindre) en les déplaçant localement dans l'une des huit directions montrées dans la figure 4.8. La figure 4.9 illustre la procédure de déformation de chemin avec contrainte de continuité de courbure. Il est à noter que la vérification de la faisabilité et la sécurité des solutions incluant une déformation du chemin géométrique se fait au niveau de l'étape d'évaluation (fonction fitness) de l'algorithme génétique mis en place pour le module de déformation de trajectoire.

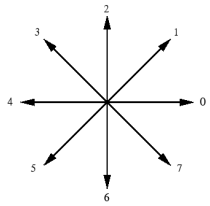


Figure 4.8 — Directions de déplacement des points de contrôle.

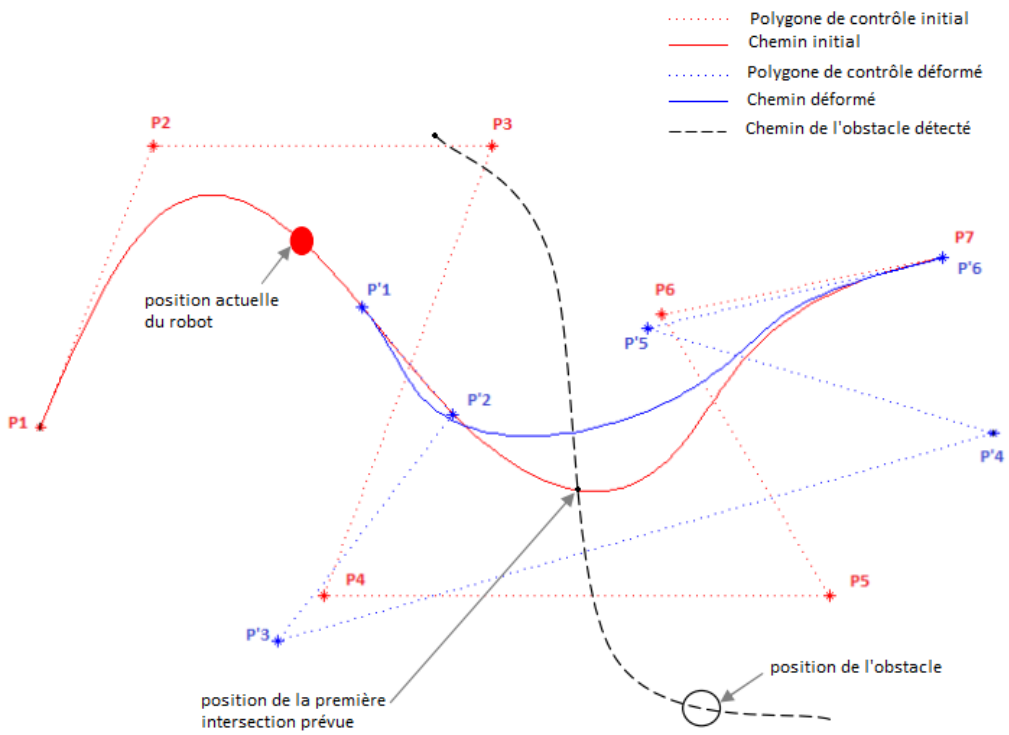


Figure 4.9 — Illustration de la procédure de déformation de chemin avec continuité de courbure.

2. Processus d'adaptation de la cinématique du robot : Une autre alternative pour éviter les éventuelles collisions est d'ajuster la vitesse du robot tout en respectant les contraintes cinématiques du système robotique considéré (vitesse et accélération maximales).

Ainsi, deux cas sont envisageables : Le premier serait de décélérer, sans changer de chemin géométrique, puis accélérer afin d'atteindre la valeur de vitesse initialement calculée (figure 4.10). Cette solution traduit l'intention de laisser passer le ou les objets mobiles avant que le robot atteigne les lieux géométriques des collisions prévues.

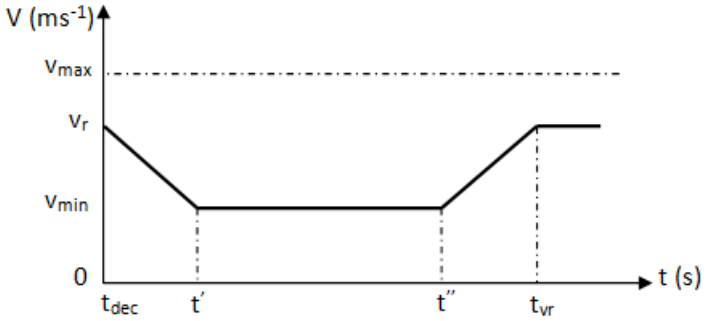


Figure 4.10 — Ajustement de la vitesse du robot : Premier cas possible.

Comme le montre la figure 4.11, la deuxième solution possible serait d'accélérer, afin de passer par les zones critiques avant que le ou les obstacles mobiles y soient présents, puis décélérer pour revenir à la valeur de vitesse initialement calculée.

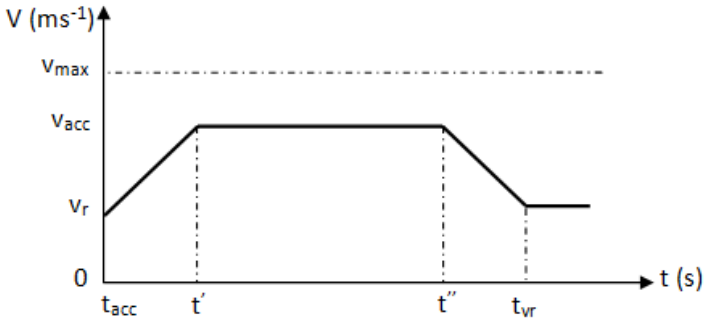


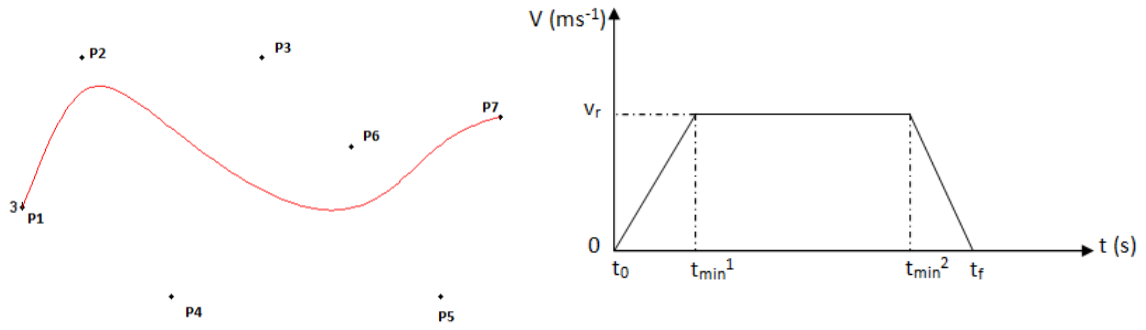
Figure 4.11 — Ajustement de la vitesse du robot : Deuxième cas possible.

3. Processus de déformation du chemin géométrique avec ajustement de la vitesse du robot : Cette solution consiste à modifier toutes les composantes de la trajectoire i.e. la courbe et la vitesse de parcours à la fois. Autrement dit, il s'agit de générer une nouvelle trajectoire à partir d'un état donné de la trajectoire actuelle tout en respectant à la fois les contraintes cinématiques et géométriques (continuité de courbure entre les deux chemins).

4.3.4.3 Mise en place de l'algorithme de déformation de trajectoire

Pour gérer le comportement du robot lors de son évolution dans son environnement, nous proposons un algorithme génétique pour le guider à faire le choix le plus convenable à sa situation. Les étapes de l'algorithme utilisé pour la déformation de mouvement sont les mêmes que celles énoncées dans l'algorithme 2. Le schéma de codage des individus ainsi que les différentes composantes de cet algorithme sont détaillées ci-dessous :

Représentation de l'individu : Un individu est une trajectoire définie par un chemin géométrique et une loi de vitesse comme l'illustre la figure 4.12 où le robot doit se déplacer du point P1 au point P7 en suivant la loi de vitesse linéaire donnée.



(a) Chemin géométrique (courbe NURBS) (b) Loi de vitesse

Figure 4.12 — Composition d'une trajectoire.

Ainsi, une trajectoire sera modélisée par un enregistrement contenant deux champs. Le premier décrit le chemin géométrique (la courbe NURBS) et sera représenté par un

ensemble de points pondérés d'oú chaque gène géométrique représente les coordonnées x et y d'un point de contrôle ainsi que son poids. Le deuxième champ décrit la loi de vitesse associée au chemin géométrique. En se référant à la figure 4.12, cette composante sera représentée par un vecteur contenant les quatre paramètres V_r, t^1_{min}, t^2_{min} et t_f. Une telle représentation permet de déduire l'ensemble des états-temps du mobile c'est à dire les positions du mobile aux différents instants.

Initialisation de la population : Comme énoncé précédemment, lorsque notre système robotique envisage une déformation de trajectoire, la solution consiste à : soit changer sa cinématique, soit la géométrie du chemin, soit les deux à la fois. En partant de cette hypothèse, la population initiale doit inclure tous ces types de solutions candidates.

Ainsi, nous proposons des pourcentages égaux pour chaque type de solution. En effet, le un tiers de la taille de la population est consacré aux candidats se basant sur la déformation de chemin détaillé précédemment. Le même nombre de candidats assurant une modification de la loi de vitesse est généré aléatoirement, tout en respectant les valeurs de vitesse et d'accélération maximales du robot. Et finalement, le reste des candidats consiste à générer des solutions modifiant conjointement le chemin géométrique et la loi de vitesse. Notons que la taille de la population pour la phase de planification online ne doit pas être trop importante pour éviter d'engendrer un temps de calcul énorme.

Fonction d'évaluation : La fonction de coût utilisée pour évaluer la performance de chaque solution candidate Γ prend en considération trois contraintes :

Contrainte de sécurité $C_{safety}(\Gamma)$ *:* Cette contrainte assure que le robot soit maintenu toujours loin des obstacles (statiques et dynamiques).

Contrainte de faisabilité $C_{feasibility}(\Gamma)$ *:* Cette contrainte assure que la trajectoire calculée est réalisable par le robot (le rayon de courbure en chaque point de passage doit être supérieur au rayon de courbure minimum du robot, les valeurs de vitesse et d'accélération doivent être des valeurs autorisées).

Contrainte temporelle $C_{time}(\Gamma)$ *:* Cette contrainte assure que le temps de suivi de la trajectoire soit minimal.

$$f(\Gamma) = \alpha.C_{safety}(\Gamma) + \beta.C_{feasibility}(\Gamma) + \gamma.C_{time}(\Gamma) \tag{4.2}$$

α, β et γ sont des facteurs de pondération et la meilleure solution est celle ayant la valeur fitness la plus petite.

Opérateurs génétiques : La procédure de sélection des individus prometteurs pour effectuer l'opération de croisement se base sur la valeur de la fonction coût. Ainsi, dans cet algorithme, nous utilisons la sélection par rang.

Pour la procédure de croisement, nous optons pour l'utilisation du croisement en 1 seul point (voir figure 4.13). Ce choix est fait dans l'optique de minimiser le temps de calcul. Étant donnée que les individus codent à la fois le chemin (points de contrôle, poids) et la cinématique (loi de vitesse), les gènes sont de deux types. De ce fait, nous adaptons notre algorithme génétique de sorte qu'il ne mélange pas ces types.

Pour apporter l'aléa nécessaire à une exploration efficace de l'espace de recherche et pour éviter les extréma locaux, l'opérateur de mutation s'avère indispensable. Pour cela, nous appliquons la mutation aléatoire avec une certaine probabilité pour modifier aléatoirement tantôt un gène géométrique, tantôt un gène cinématique.

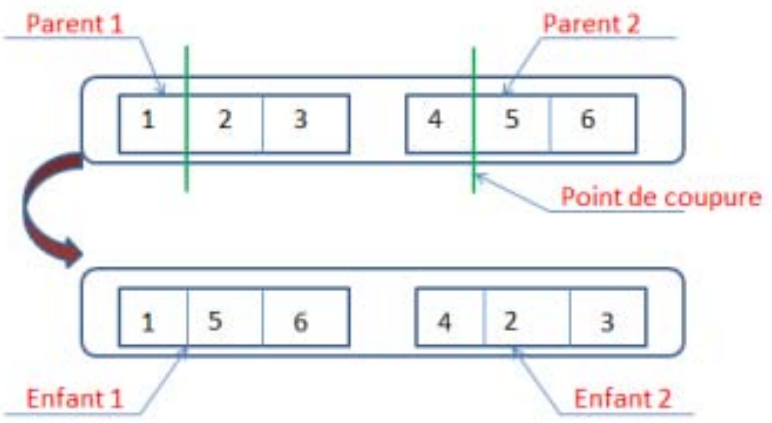


Figure 4.13 — Illustration du principe de croisement en 1 point.

4.4 Validation de l'approche proposée

Pour valider la faisabilité et l'intérêt de la méthode de déformation de trajectoire proposée, nous avons comparé notre approche avec quelques travaux existants (présentés dans le chapitre 1). Cette comparaison est faite sur la base des critères suivants :

- Réévaluation de l'espace de travail : Ce critère implique la refaite des calculs par le robot pour analyser tout l'espace de travail afin de prendre de nouvelles décisions en calculant de nouveau la trajectoire permettant d'atteindre la position but. En général, cette procédure est déclenchée au cas où l'environnement change fortement (par exemple une porte qui se ferme soudainement, un obstacle qui passe avec une vitesse très importante,...) et que le robot est incapable de continuer sa navigation en suivant sa trajectoire actuelle ou même de la modifier convenablement. De même, cette réévaluation est indispensable dans le cas où la méthode de déformation de mouvement est de nature heuristique.

- Limite de courbure : Ce critère assure que la valeur de courbure en chaque point de la trajectoire suivie par le robot ne dépasse pas une certaine valeur maximale liée au caractère non-holonome de ce dernier.

- Risque de collisions : Ce critère reflète la susceptibilité de l'approche à échouer à faire guider le robot pour naviguer à l'abri des obstacles.

- Longueur minimale : Ce critère indique si la méthode de déformation de trajectoire évite ou non de produire des solutions avec des longueurs non optimisées.

- Risque de perte de connectivité : Ce critère renseigne sur la sensibilité de la méthode utilisée à produire des mouvements discontinus.

- Adaptation de la cinématique du robot : Ce critère indique si l'approche de déformation de trajectoire agit seulement sur le chemin géométrique ou bien elle intègre la possibilité d'adaptation de la vitesse du robot dans sa stratégie de contournement d'obstacles.

- Temps de calcul : Ce critère permet d'évaluer le temps de calcul mis pour l'exécution du module de déformation de trajectoire.

Le tableau 4.1 montre l'adaptabilité de chaque approche citée vis-à-vis de ces différents critères. Nous notons que cette comparaison a montré que nos résultats sont assez concurrentiels. En effet, pour notre méthode, la réévaluation de l'environnement de travail est contextuelle, ce qui veut dire que cette alternative est possible dans un contexte bien défini. En fait, vu la nature heuristique du module de déformation de trajectoire, cet algorithme peut échouer à trouver une solution admissible ce qui nécessite, par la suite, un freinage du système robotique qui doit recalculer une nouvelle trajectoire à partir de sa position actuelle. Par conséquent, le risque de collisions est assez faible. De plus, notre méthode garantie un mouvement continu et contraint par une limite de courbure découlant de la nature du système robotique, tout en offrant la possibilité de déformer aussi bien le chemin géométrique que la loi de vitesse. Quant à la longueur parcourue, les résultats sont toujours satisfaisants vu que le chemin géométrique de référence est calculé en prenant en compte cette contrainte et même lors des éventuelles déformations, les variations ne sont pas assez importantes puisque le déplacement des points de contrôle se fait toujours dans un voisinage bien précis.

Tableau 4.1 — Tableau comparatif des méthodes de déformation de mouvement.

Méthode	**Réévaluation de l'espace de travail**	**limite de courbure**	**Risque de collisions**	**Longueur minimale**	**Risque de perte de connectivité**	**Adaptation de la cinématique du robot**	**Temps de calcul**
Bande élastique de Khatib	Non indiqué	Peu assurée	Très fréquent	Non assurée	Très fréquent	Non prise en compte	Coûteux
Déformation variationnelle de Lamiraux	Non indiqué	Assurée	Faible	Non assurée	Peu fréquent	Prise en compte	Fort coûteux
Roadmap élastique	Contextuelle	Non assurée	Moyen	Assurée	Peu fréquent	Non prise en compte	Coûteux
Déformation de trajectoire de Kurniawati et Fraichard	Contextuelle	Non assurée	Faible	Non assurée	Fréquent	Prise en compte	Faible
Déformation de trajectoire de Delsart et Fraichard	Contextuelle	Assurée	Faible	Non assurée	Fréquent	Prise en compte	Faible
Approche Proposée	Contextuelle	Assurée	Faible	Satisfaisante	Pas de risque	Prise en compte	Moyen

4.5 Conclusion

Dans ce chapitre, nous avons présenté une nouvelle approche de navigation en environnement dynamique pour les robots mobiles non-holonomes. Cette approche est capable de guider le robot pour se déplacer d'une manière autonome en évitant les collisions et tout en respectant les contraintes cinématiques. Pour garantir la convergence vers le but, notre méthode s'inscrit sous la famille d'approches de navigation avec une trajectoire de référence. Ce générateur de trajectoires peut agir, à la fois, sur le chemin géométrique aussi bien que la loi de vitesse du système robotique.

L'emploi des courbes NURBS comme primitives de modélisation nous a permis d'avoir des trajectoires continues respectant une certaine limite de courbure. Ces mêmes critères ont été assurés lors des éventuelles déformations subies par ces trajectoires grâce à la propriété du contrôle local de ces courbes. Nous notons aussi que l'utilisation de l'algorithme génétique, lors du processus de prise de décision, permet une large exploitation de l'espace de recherche ce qui permet, par la suite, d'orienter le robot vers le meilleur choix.

En la comparant à d'autres travaux portant sur la navigation autonome avec une trajectoire de référence, en un environnement changeant , notre méthode donne des résultats prometteurs face aux différents critères de comparaison pris en compte.

Conclusion générale et Perspectives

Les travaux présentés dans ce manuscrit portent sur la planification de mouvement pour les robots mobiles non-holonomes dans des environnements quelconques statiques ou dynamiques. Ce problème fondamental en navigation robotique consiste à déterminer le mouvement que le système robotique doit effectuer afin d'atteindre un objectif fixé à priori. Nos contributions dans cette problématique touchent les deux aspects du problème à savoir la planification de chemin et la planification de trajectoire.

Ainsi, dans le cadre de la planification de chemin, nous avons développé un planificateur de chemins permettant de générer des chemins libres des obstacles et avec contrainte de courbure maximale, modélisés par des courbes NURBS. Ce planificateur se base sur deux phases : La première s'appuie sur la technique de squelettisation et la théorie des graphes afin de déterminer le plus court chemin entre deux configurations données. Quant à la deuxième, elle s'occupe de la procédure de lissage du chemin calculé tout en maintenant la contrainte de la non-collision et en prenant en compte le rayon de courbure minimum du robot.

En s'appuyant sur l'importance de l'influence du paramètre poids des courbes NURBS vis-à-vis des contraintes de sécurité et de limite de courbure, deux solutions utilisant deux différentes méthodes d'optimisation ont été proposées. la première solution utilise un algorithme génétique pour la paramétrisation des poids des différents points de contrôle qui sont judicieusement sélectionnés. Ainsi, une fonction coût bien précise a été employée. Cette fonction est définie de sorte à prendre en compte quatre paramètres à la fois qui sont la sécurité, la faisabilité, la longueur et l'écart type de courbures du chemin considéré. En plus, une nouvelle adaptation de l'opérateur de

mutation a été proposée afin de diminuer le taux des chemins infaisables engendré par la mutation aléatoire. La deuxième solution intègre la méthode d'exploration locale de Hooke et Jeeves en agissant non seulement sur le paramètre poids mais aussi sur les emplacements des différents points de contrôle.

Les principales qualités de ce planificateur sont : En premier lieu, l'adaptabilité de la méthode proposée qui se montre capable de trouver le chemin optimal ou quasi-optimal, en termes de longueur et sécurité à la fois et qui respecte la contrainte de courbure liée au robot considéré, indépendamment de la complexité de la scène étudiée. En deuxième lieu, l'emploie des courbes NURBS avec une paramétrisation qui tient compte des contraintes et du résultat attendu en exploitant l'influence de chaque paramètre de ces fonctions d'approximation.

Les résultats mis en évidence dans ces travaux nous ont permis d'étendre cette même idée pour le contexte d'un environnement dynamique. Ainsi, dans le cadre de la planification de trajectoire, nous avons introduit une nouvelle approche de déformation de trajectoire permettant à un robot mobile de naviguer dans un environnement changeant de façon à s'éloigner des objets mobiles et de satisfaire les contraintes cinématiques et non holonomes (contrainte du rayon de courbure minimum) du système robotique considéré. L'idée est de concevoir une courbe NURBS flexible en fonction de la dynamique de l'environnement.

Ce planificateur se base sur deux modules : Celui du calcul de la trajectoire de référence (module de planification offline) et celui de la planification online qui assure l'adaptation de la trajectoire courante avec les changements détectés au cours des déplacements du robot. Cette adaptation autorise à la fois la déformation géométrique du mouvement ainsi que celle cinématique en adaptant la vitesse du mobile en fonction de la situation rencontrée. Un algorithme génétique est mis en oeuvre pour ce module de déformation de trajectoire afin d'assurer une meilleur exploration de l'espace de recherche.

Il est à noter que l'utilisation des courbes NURBS, en tant que primitives de modélisation, permet d'avoir des trajectoires continues avec une contrainte de limite de courbure. Ces mêmes critères sont assurés lors des éventuelles déformations subies par ces trajectoires grâce à la propriété du contrôle local de ces courbes. Ainsi, notre méthode garantit des résultats prometteurs face à la convergence vers le but et au respect des contraintes cinématiques liées au robot et celles dynamiques se référant à l'envi-

ronnement de travail. Par conséquent, notre première perspective à court terme serait de tester et valider cette approche dans des environnements simulés numériquement.

Les résultats présents dans cette thèse sont encourageants. Néanmoins, nos approches de planification de mouvement n'étaient pas évaluées par des expérimentations réelles. De ce fait, une deuxième perspective à court terme aussi serait de les intégrer sur un robot réel pour des éventuels tests avec des scénarios distincts.

Vu que la robotique mobile en général et la planification de mouvement en particulier représente un sujet d'actualité, il y'a toujours matière à des recherches plus approfondies. Par conséquent, il serait intéressant, dans un premier temps, de prendre en compte les incertitudes des données provenant des imperfections liées aux capteurs et à la nature de l'environnement lui même. Une orientation possible serait d'utiliser la théorie des fonctions de croyance (TFC) qui est basée sur un fondement mathématique solide, permettant de donner des solutions acceptables en termes de modélisation et de complexité liées aux problèmes de la fusion de données ce qui permet, par conséquent, de gérer les situations d'ignorance et d'incertain. L'objectif est donc d'adapter notre méthode de prise de décision en navigation robotique de sorte à ce qu'elle soit valide en environnement dynamique et incertain et tout en se basant sur la TFC.

Une autre perspective envisageable serait d'étudier et de développer le modèle de prédiction du mouvement futur des obstacles mobiles. En fait, le modèle actuellement utilisé suppose l'existence d'un prédicteur parfait ce qui n'est pas toujours évident de le mettre en place.

Une extension envisageable et très intéressante serait d'étendre nos travaux pour le contexte multi-robots. L'idée serait de gérer une scène avec plusieurs robots qui s'y déplacent afin de joindre certains objectifs fixés à priori. Deux cadres d'étude sont possibles : le coopératif et le non coopératif. L'objectif serait, ainsi, de concevoir une approche de déformation de trajectoire généralisée en fonction de la nature des agents présents dans la scène.

Bibliographie

[A. and Powers, 2013] A., A. and Powers, D. (2013). Review of classical and heuristic-based navigation and path planning approaches. *International Journal of Advancements in Computing Technology*, 5(14) :1–14.

[Adi et al., 2009] Adi, D., Shamsuddin, S., and Ali, A. (2009). Particle swarm optimization for nurbs curve fitting. In *Sixth International Conference on Computer Graphics, Imaging and Visualization : New Advances and Trends(CGIV)*, pages 259–263.

[Ahmed and Deb, 2013] Ahmed, F. and Deb, K. (2013). Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms. *Soft Comput.*, 17 :1283–1299.

[Barbehenn and Hutchinson, 1995] Barbehenn, M. and Hutchinson, S. (1995). Toward an exact incremental geometric robot motion planner. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 39–44.

[Caihol, 2015] Caihol, S. (2015). *Planification intéractive de trajectoire en réalité virtuelle sur la base de données géométriques, topologiques et sémantiques.* Institut National Polytechnique de Toulouse.

[Cakir, 2015] Cakir, M. (2015). 2d path planning of uavs with genetic algorithm in a constrained environment. In *Modeling, Simulation, and Applied Optimization (ICMSAO), 2015 6th International Conference on*, pages 1–5.

[Chang and Jin, 2013] Chang, H. and Jin, T. (2013). *Command Fusion Based Fuzzy Controller Design for Moving Obstacle Avoidance of Mobile Robot*, volume 235 of *Lecture Notes in Electrical Engineering.*

[Choset et al., 2005] Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., Burgard, W., Kavraki, L. E., and Thrun, S. (2005). *Principles of Robot Motion : Theory, Algorithms, and Implementations.* MIT Press, Cambridge, MA.

[Conte and Zullil, 1995] Conte, G. and Zullil, R. (1995). Hierarchical path planning in a multi-robot environment with a simple navigation function. *IEEE Trans. On Systems, Man, and Cybernetics*, 25(4) :651–654.

[Dalibard and Laumond, 2009] Dalibard, S. and Laumond, J. (2009). Control of probabilistic diffusion in motion planning. *Algorithmic Foundation of Robotics VIII*, pages 467–481.

[Delsart, 2010] Delsart, V. (2010). *Navigation autonome en environnement dynamique : Une approche par déformation de trajectoire.* Université de Grenoble.

[Delsart and Fraichard, 2008] Delsart, V. and Fraichard, T. (2008). Navigating dynamic environments using trajectory deformation. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 226–233.

[Di Ruberto, 2004] Di Ruberto, C. (2004). Recognition of shapes by attributed skeletal graphs. *Pattern Recognition*, 37 :21–31.

[Dubins, 1957] Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3) :497–516.

[Egerstedt and Martin, 2010] Egerstedt, M. and Martin, C. (2010). *Control Theoretic Splines : Optimal Control, Statistics, and Path Planning.* Princeton University Press, New Jersey, USA.

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

[Faisal et al., 2013] Faisal, M., Hedjar, R., Al Sulaiman, M., and Al-Mutib, K. (2013). Fuzzy logic navigation and obstacle avoidance by a mobile robot in an unknown dynamic environment. *International Journal of Advanced Robotic Systems*, 10 :1–7.

[Filliat, 2013] Filliat, D. (3 octobre 2013). *Robotique mobile.* Ecole nationale supérieur de techniques avancées Paris Tech.

[Flavigné, 2010] Flavigné, D. (2010). *Planification de mouvement interactive : coopération Humain-Machine et l'animation.* Université de Toulouse.

[Fox et al., 1997] Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Journal on Robotics and Automation*, 4.

[Fraichard, 1992] Fraichard, T. (1992). *Thèse de doctorat : Planification de mouvement pour mobile non-holonome en espace de travail dynamique.* Institut National Polytechnique de Grenoble.

[Fraichard, 1998] Fraichard, T. (1998). Trajectory planning in a dynamic workspace : a state-time space approach. *Advanced Robotics*, 13 :75–94.

[Fraichard, 2006] Fraichard, T. (2006). *Contributions à la planification de mouvement.* Mémoire d'Habilitation à Diriger des Recherches, Institut National Polytechnique de Grenoble.

[Fraichard, 2013] Fraichard, T. (April 2013). *Research Activity Summary.* Institut National Polytechnique de Grenoble.

[Fraichard and Hajime, 2003] Fraichard, T. and Hajime, A. (2003). Inevitable collision states a step towards safer robots. In *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, pages 388–393.

[Fraichard and Martinez-Gomez, 2008] Fraichard, T. and Martinez-Gomez, L. (2008). An efficient andgeneric 2d inevitable collision state-checker. In *IEEE-RSJ Int. Conf. on Intelligent Robots and Systems.*

[Fraichard and Martinez-Gomez, 2009] Fraichard, T. and Martinez-Gomez, L. (2009). Collision avoidance in dynamic environments : an ics-based solution and its comparative evaluation. In *Proc. of the Intl Conf. on Robotics and Automation.*

[Fraichard and Parthasarathi, 2007] Fraichard, T. and Parthasarathi, R. (2007). An inevitable collision state checker for a car-like vehicle. In *Proc. of the Intl Conf. on Robotics and Automation.*

[Fraichard and Scheuer, 2004] Fraichard, T. and Scheuer, A. (2004). From reeds and shepp's to continuous-curvature paths. *IEEE Transactions on Robotics*, 20 :1025–1035.

[Gemeinder and Gerke, 2003] Gemeinder, M. and Gerke, M. (2003). Ga-based path planning for mobile robot systems employing an active search algorithm. *Appl. Soft Comput.*, 3 :149–158.

[GUECHI, 2010] GUECHI, E. (2010). *Suivi de trajectoires d'un robot mobile non holonome : approche par modèle flou de Takagi-Sugeno et prise en compte des retards.* Université de Valenciennes et du Hainaut Cambrésis.

[Ho and Liu, 2009] Ho, Y. and Liu, J. (2009). Collision-free curvature-bounded smooth path planning using composite bezier curve based on voronoi diagram. In *CIRA, IEEE*, pages 463–468.

[Hong et al., 2012] Hong, T.-S., Nakhaeinia, D., and Karasfi, B. (2012). Application of fuzzy logic in mobile robot navigation. *Fuzzy Logic - Controls, Concepts, Theories and Applications*, pages 21–36.

[Hooke and Jeeves, 1961] Hooke, R. and Jeeves, T. (1961). "direct search" solution of numerical and statistical problems. *Journal of the Association for Computing Machinery (ACM)*, 8(2) :212–229.

[Hu et al., 2007] Hu, C., Wu, X., Liang, Q., and Wang, Y. (2007). Autonomous robot path planning based on swarm intelligence and stream functions. In *7th International Conference, ICES*, pages 277–284.

[Hu et al., 2004] Hu, Y., Yang, S. X., zhong Xu, L., and Meng, M.-H. (2004). A knowledge based genetic algorithm for path planning in unstructured mobile robot environments. In *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*, pages 767–772.

[Huang, 2008] Huang, L. (2008). Velocity planning for a mobile robot to track a moving target - a potential field approach. *Robotics and Autonomous Systems*, 57 :55–63.

[Jalel et al., 2013] Jalel, S., Marthon, P., and Hamouda, A. (2013). Optimum path planning for Mobile Robots in Static Environments using Graph Modelling and NURBS Curves. In *WSEAS International Conference on Signal Processing, Robotics and Automation, February 20-22, 2013 Cambridge, UK*, pages 216–221.

[Jalel et al., 2015a] Jalel, S., Marthon, P., and Hamouda, A. (2015a). NURBS Based Multi-objective Path Planning. In *7th Mexican Conference, MCPR 2015, June 24-27, 2015 Mexico City, Mexico*, pages 190–199. Springer.

[Jalel et al., 2015b] Jalel, S., Marthon, P., and Hamouda, A. (2015b). NURBS based robot navigation. In *Twentieth International Symposium on Artificial Life and Robotics, AROB 20th 2015, January 21-23, 2015 B-Con Plaza, Beppu, Japan*, pages 377–382.

[Jalel et al., 2015c] Jalel, S., Marthon, P., and Hamouda, A. (2015c). Optimized NURBS Curves Modelling Using Genetic Algorithm for Mobile Robot Navigation. In *16th International Conference on Computer Analysis of Images and Patterns, CAIP 2015, 2-4 September, 2015 Valletta, Malta*, pages 534–545. Springer.

[Jalel et al., 2016] Jalel, S., Marthon, P., and Hamouda, A. (2016). A new path generation algorithm based on accurate nurbs curves. *Int J Adv Robot Syst*, 13 :75 :1–14.

[Jalel et al., 2012] Jalel, S., Naouai, M., Hamouda, A., and Jebabli, M. (2012). Nurbs parameterization : A new method of parameterization using the correlation relationship between nodes. In *MCPR, Springer, Lecture Notes in Computer Science, 7329*, pages 216–225.

[Jing et al., 2009] Jing, Z., Shaowei, F., and Hanguo, C. (2009). Optimized nurbs curve and surface modelling using simulated evolution algorithm. In *Second International Workshop on Computer Science and Engineering (WCSE)*, pages 435–439.

[Kala, 2014] Kala, R. (2014). *Code for Robot Path Planning using Bidirectional Rapidly-exploring Random Trees.* Indian Institute of Information Technology Allahabad, Available at : http ://rkala.in/codes.html.

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

[Khatib et al., 1997] Khatib, M., Jaouni, H., Chatila, R., and Laumond, J. (1997). Dynamic path modification for car-like nonholonomic mobile robots. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 4, pages 2920–2925 vol.4.

[Khatib, 1985] Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 500–505.

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. (1983). Optimization by simulated annealing. *Science*, New Series 220 (4598) :671–680.

[Kuffner and LaValle, 2000] Kuffner, J. and LaValle, S. M. (2000). Rrt-connect : An efficient approach to single-query path planning. In *IEEE International Conference on In Robotics and Automation*, pages 995–1001.

[Kurniawati and Fraichard, 2007] Kurniawati, H. and Fraichard, T. (2007). From path to trajectory deformation. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 159–164.

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

[Latombe, 1991] Latombe, J. (1991). *Robot motion planning*. The Kluwer international series in engineering and computer science, Kluwer Academic Publishers.

[Laumond, 1987] Laumond, J. (1987). Finding collision-free smooth trajectories for a non-holonomic mobile robot. In *Int. Joint Conf. on Artificial Intelligence*, pages 1120–1123.

[Lavalle, 1998] Lavalle, S. M. (1998). Rapidly-exploring random trees : A new tool for path planning. Technical report.

[LaValle, 2006] LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press.

[LaValle and Kuffner, 1999] LaValle, S. M. and Kuffner, J. (1999). Randomized kinodynamic planning. In *IEEE International Conference on In Robotics and Automation*, pages 473–479.

[Leena and Saju, 2014] Leena, N. and Saju, K. (2014). A survey on path planning techniques for autonomous mobile robots. *IOSR Journal of Mechanical and Civil Engineering*, pages 76–79.

[Levitt and Lawton, 1990] Levitt, T. and Lawton, D. T. (1990). Qualitative navigation for mobile robots. *Journal Artificial Intelligence*, 44 :305–360.

[Li et al., 2006] Li, Z., Meek, D., and Walton, D. (2006). A smooth, obstacle-avoiding curve. *Computers and Graphics*, 30(4) :581–587.

[Lozano-Pérez, 1983] Lozano-Pérez, T. (1983). Spatial planning : A configuration space approach. *IEEE Transactions on Computers*, C-32 :108–120.

[Maekawa et al., 2010] Maekawa, T., Noda, T., Tamura, S., Ozaki, T., and Machida, K. (2010). Curvature continuous path generation for autonomous vehicle using b-spline curves. *Computer-Aided Design*, 42(4) :350–359.

[Marthon et al., 1979] Marthon, P., Bruel, A., and Biguet, G. (1979). Squelettisation par calcul d'une fonction discriminante sur un voisinage de 8 points. In *2 ème Congrès AFCET-IRIA, Reconnaissance des formes et intelligence artificielle*, pages 107–114.

[Masehian and Sedighizadeh, 2007] Masehian, E. and Sedighizadeh, D. (2007). Classic and heuristic approaches in robot motion planning - a chronological review. *International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, 5 :228–233.

[Minguez and Montano, 2000] Minguez, J. and Montano, L. (2000). Nearness diagram navigation (nd) : a new real time collision avoidance approach. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 3, pages 2094–2100 vol.3.

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

[Minguez and Montano, 2006] Minguez, J. and Montano, L. (2006). Abstracting vehicle shape and kinematic constraints from obstacle avoidance methods. *Autonomous Robots*, 20 :43–59.

[Montiel et al., 2015] Montiel, O., Orozco-Rosas, U., and Sepúlveda, R. (2015). Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. *Expert Systems with Applications*, 42(12) :5177 – 5191.

[MORETTE, 2009] MORETTE, N. (2009). *Contribution à la Navigation de robots mobiles : approche par modèle direct et commande prédictive*. Univesité d'Orléans.

[Nilsson, 1969] Nilsson, N. J. (1969). A mobile automaton : An application of artificial intelligence techniques. In *1st International Conference on Artificial Intelligence*, pages 509–520.

[Piegl et al., 2009] Piegl, L., Rajab, K., Smarodzinava, V., and Valavanis, K. (2009). Using a biarc filter to compute curvature extremes of nurbs curves. *Engineering with Computers*, 25(4) :379–387.

[Piegl and Tiller, 1997] Piegl, L. and Tiller, W. (1997). *The NURBS Book*. S.l. :Springer. 2ème edition.

[Prautzsch et al., 2002] Prautzsch, H., Boehm, W., and Paluszny, M. (2002). *Bézier and B-Spline Techniques*. Springer. Berlin, Germany.

[Qu et al., 2013] Qu, H., Xing, K., and Alexander, T. (2013). An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots. *Neurocomputing*, 120 :509–517.

[Quinlan and Khatib, 1993] Quinlan, S. and Khatib, O. (1993). Elastic bands : connecting path planning and control. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 802–807 vol.2.

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

[Rosell and Iniguez, 2005] Rosell, J. and Iniguez, P. (2005). Path planning using harmonic functions and probabilistic cell decomposition. In *Int. Conf. on Robotics and Automation*, pages 1803–1808.

[Scheuer, 1998] Scheuer, A. (1998). *Planification de chemins à courbure continue pour robot mobile non holonome*. Institut National Polytechnique de Grenoble.

[Seder and Petrovic, 2007] Seder, M. and Petrovic, I. (2007). Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. In *International Conference on Robotics and Automation*.

[Sedighi et al., 2004] Sedighi, K., Ashenayi, K., Manikas, T., Wainwright, R., and Tai, H. (2004). Autonomous local path planning for a mobile robot using a genetic algorithm. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1338–1345.

[Shih et al., 1990] Shih, C., Lee, T., and Gruver, W. (1990). Motion planning with time-varying polyhedral obstacles based on graph search and mathematical programming. In *IEEE Int. Conf. on Robotics and Automation*, pages 331–337.

[Shikin and Plis, 1995] Shikin, E. V. and Plis, A. (1995). *Handbook on Splines for the User*. CRC Press, Florida, USA.

[Siegwart and Nourbakhsh, 2004] Siegwart, R. and Nourbakhsh, I. R. (2004). *Introduction to autonomous mobile robots*. MIT press.

[Singh et al., 2011] Singh, A., Aggarwal, A., Vashisht, M., and Siddavatam, R. (2011). Robot motion planning in a dynamic environment using offset non-uniform rational b-splines (nurbs). In *ICIT, IEEE*, pages 312–317.

[Statheros et al., 2006] Statheros, T., Defoort, M., Khola, S., McDonald-Maier, K., Howells, W., Kokosy, A., P. J., Perruquetti, W., and Floquet, T. (2006). Automated control and guidance system (acos) : An overview. In *International Conference on Recent Advances in Soft Computing (RASC)*.

[Takahashi and Schilling, 1989] Takahashi, O. and Schilling, R. (1989). Motion planning in a plane using generalized voronoi diagrams. *IEEE Transactions on Robotics and Automation*, 5(2) :143–150.

[Tamilselvi et al., 2012] Tamilselvi, D., Mercy Shalinie, S., Fathima Thasneem, A., and Gomathi Sundari, S. (2012). Optimal path selection for mobile robot navigation using genetic algorithm in an indoor environment. In *Springer, Lecture Notes in Computer Science, 7135*, pages 263–269.

[Tang et al., 2012] Tang, S. H., Khaksar, W., Ismail, N. B., and Ariffin, M. K. A. (2012). A review on robot motion planning approaches. *Pertanika Journal of Science and Technology*, 20(1) :15–29.

[Tu and Yang, 2003] Tu, J. and Yang, S. X. (2003). Genetic algorithm based path planning for a mobile robot. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, pages 1221–1226.

[Tuncer and Yildirim, 2012] Tuncer, A. and Yildirim, A. (2012). Dynamic path planning of mobile robots with improved genetic algorithm. *Computers & Electrical Engineering*, 38 :1564–1572.

[Wei and Liu, 2010] Wei, J. and Liu, J. (2010). Generating minimax-curvature and shorter η3-spline path using multi-objective variable-length genetic algorithm. In *ICNSC, IEEE*, pages 319–324.

[Xidias and Aspragathos, 2013] Xidias, E. and Aspragathos, N. (2013). Continuous curvature constrained shortest path for a car-like robot using s-roadmaps. In *MED, IEEE*, pages 13–18.

[Yang et al., 2014] Yang, K., Moon, S., Yoo, S., Kang, J., Lett Doh, N., Kim, H., and Joo, S. (2014). Spline-based rrt path planner for non-holonomic robots. *Journal of Intelligent and Robotic Systems*, 73 :763–782.

[Yang and Brock, 2006] Yang, Y. and Brock, O. (2006). Elastic roadmaps : Globally task-consistent motion for autonomous mobile manipulation in dynamic environments. In *Robotics : Science and Systems II, August 16-19, 2006. University of Pennsylvania, Philadelphia, Pennsylvania, USA*.

[Yun et al., 2010] Yun, S., Ganapathy, V., and Chong, L. (2010). Improved genetic algorithms based optimum path planning for mobile robot. In *11th International Conference on Control, Automation, Robotics and Vision(ICARCV)*, pages 1565–1570.

[Zhang et al., 2004] Zhang, P.-Y., Lu, T.-S., and Song, L.-B. (2004). Soccer robot path planning based on the artificial potential field approach with simulated annealing. *Robotica*, 22 :563–566.