UNIVERSITÉ PARIS 8 - VINCENNES-SAINT-DENIS
U.F.R. 6

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ PARIS 8
Spécialité Informatique

soutenue le 10 décembre 2004

par

Nicolas JOUANDEAU

Algorithmique de la planification de mouvement probabiliste pour un robot mobile

Jury

Patrick GREUSSAY	Président	UNIVERSITÉ PARIS8
Arab ALI CHÉRIF	Directeur	UNIVERSITÉ PARIS8
Raja CHATILA	Rapporteur	LAAS-CNRS
Yacine AMIRAT	Rapporteur	UNIVERSITÉ PARIS12
René ZAPATA	Examinateur	LIRMM-CNRS
Jacqueline SIGNORINI	Examinateur	UNIVERSITÉ PARIS8

Remerciements

Je tiens à remercier vivement tous les membres du jury :

Patrick Greussay pour m'avoir accueilli au sein du laboratoire d'Intelligence Artificielle pour effectuer cette thèse et pour avoir accepté de présider le jury.

Arab Ali Chérif pour sa confiance et son soutien. Je tiens à exprimer toute ma reconnaissance à mon directeur de thèse, pour avoir su orienter mes aspirations en me laissant une grande liberté.

Raja Chatila et Yacine Amirat, de m'avoir fait l'honneur d'accepter d'être rapporteurs de ma thèse, de participer au jury et de s'être interessés à mes travaux.

Jacqueline Signorini et René Zapata, d'avoir accepté d'examiner mes travaux.

Les travaux présentés dans ce mémoire ne seraient pas ce qu'ils sont, sans la présence, le soutien et la patience, sans les discussions diverses et enrichissantes ayant contribués au support de cette thèse, ayant contribués à l'amélioration de mes travaux, l'ensemble des personnes ayant partagés avec moi cette période, et plus particulièrement, Erik Adelbert, Anaïs Atencia, Farès Belhadj, Vincent Boyer, Louisa El Gharbi, Bashir Menaï, Nicolas Neveu, Anna Pappa, Nathalie Pibaleau et Dominique Sobzcyk.

Résumé

Nous étudions dans ce mémoire la planification de mouvement probabiliste incrémentale. Nos contributions se concrétisent par un nouvel algorithme de construction des arbres aléatoires d'exploration rapide et par une nouvelle décomposition de l'espace des configurations en cellules irrégulières, basée sur la propriété de visibilité pour un robot mobile. Le mémoire s'articule autour de quatre chapitres.

Le chapitre 1 présente l'ensemble des méthodes de planification probabiliste et la formalisation du problème de planification de mouvement.

Le chapitre 2 présente un nouvel algorithme de construction des arbres aléatoires optimisant la phase d'expansion dans l'espace de recherche. Nos travaux débutent par une présentation des algorithmes existants associée à une analyse des relations entre les composants opérationnels communs à l'ensemble de ses algorithmes. Nous proposons de minimiser les appels au détecteur de collision par la synthèse d'une phase d'expansion limitée au premier élément valide. Notre algorithme minimise également le nombre de résolutions numériques liées à la modélisation des déplacements du mobile, dans une collecte d'informations. La diminution des temps de calcul est fonction de la proximité des nœuds aux obstacles dans l'espace de recherche et de la dimension de l'ensemble des commandes associées au mobile. La complétude des méthodes probabilistes étant garantie pour un temps d'exécution infini, l'optimisation de la phase d'expansion proposée dans notre algorithme fait, de nos travaux, une des composantes principales des algorithmes de construction des arbres aléatoires.

Le chapitre 3 présente notre décomposition de l'espace des configurations fondée sur la notion d'accessibilité du mobile afin d'intégrer simultanément les contraintes géométriques définies par les obstacles et les contraintes dynamiques du mobile. Partant des principales approches d'échantillonnage de l'espace de recherche utilisées dans les méthodes probabilistes, l'analyse des propriétés associées à ces échantillonnages nous conduit à proposer une décomposition de l'espace en cellules irrégulières adaptée à la notion d'accessibilité. Après une définition de cette décomposition, ces algorithmes de construction sont évalués comparativement à un échantillonnage uniforme de l'espace de recherche. Cette nouvelle décomposition nous permet de varier l'échantillonnage en fonction des propriétés de couverture définies. Les travaux présentés sont testés dans des simulations incluant un mobile de type voiture évoluant dans un environnement $2D$ statique .

Le chapitre 4 présente une conclusion de l'ensemble de nos contributions et les perspectives faisant suite à nos travaux.

Table des matières

Table des figures

Liste des algorithmes

Chapitre 1

Introduction

Nous proposons une méthode de planification de mouvement[1] probabiliste[2] incrémentale pour un robot mobile[3]. Les travaux présentés dans cette thèse s'intéressent plus particulièrement aux conséquences de la géométrie de l'espace de recherche et à l'exploration probabiliste de cet espace dans le cadre du problème de la planification de mouvement.

[1] La planification de mouvement regroupe l'ensemble des mécanismes décisionnels nécessaires à l'exécution d'actions dans un espace d'évolution. Dans ce contexte, la planification de mouvement s'adresse sans distinction aux robots mobiles et aux robots manipulateurs.

[2] Les méthodes de planification non-déterministes sont dites à complétude probabiliste. Résoudre un problème de planification de mouvement consiste à explorer l'espace de recherche afin de trouver une solution. La spécificité des méthodes probabilistes peut se résumer à un parcours aléatoire de l'espace de recherche, réduisant ainsi la complexité de la résolution.

[3] Les robots mobiles sont historiquement initiés autour de l'intelligence artificielle avec le robot *Shakey*, premier robot mobile, développé par le *Stanford Research Institute* entre 1966 et 1972 ; muni d'une caméra, de capteurs de contact et deux roues, il est le premier symbole de la classe des robots mobiles ; différenciée de la classe des robots manipulateurs par la taille de son champ d'action. Le robot mobile a la charge d'adapter son comportement à l'environnement dans lequel il se meut.

1.1 Formalisation du problème

La planification de mouvement consiste à trouver une séquence d'actions permettant à un mobile de relier une position finale à partir d'une position initiale. Cette séquence d'actions peut se définir comme une succession de points de passage appartenant à la trajectoire du mobile considéré soumis à des contraintes géométriques (*i.e.* la forme du mobile et des obstacles) ; si la vitesse de l'angle entre les roues et l'axe du mobile est bornée, la trajectoire est à courbure continue[4].

1.1.1 Espace des configurations

L'espace de travail W permet de définir l'ensemble des positions atteignables par le mobile $\mathcal{M}$; W peut contenir un ensemble de n obstacles $\mathcal{O} = \{\mathcal{o}_1, \mathcal{o}_2, \ldots, \mathcal{o}_n\}$; $\mathcal{M}$ est physiquement présent dans W. Une position est atteignable si $\mathcal{M}$ n'est pas en collision avec un obstacle $\mathcal{o}_i \in \mathcal{O}$; si W est un espace $2D$, $\mathcal{M}$ agit dans le plan ; si W est un espace $3D$, $\mathcal{M}$ agit dans l'espace. Dans le cas d'un mobile non ponctuel, les mouvements de $\mathcal{M}$ dans W sont définissables par comparaison entre la taille de $\mathcal{M}$ et celle des espaces libres entre les obstacles. Ainsi certains espaces libres de W ne sont accessibles que pour certaines positions angulaires de $\mathcal{M}$. Pour uniformiser la description de l'espace de travail, W est transformé en *Espace des configurations* (noté C) [LP83]. Cette transformation en *Espace des configurations* permet de substituer la recherche d'une trajectoire de $\mathcal{M}$ dans W par la recherche de la trajectoire d'un point dans C.

$\mathcal{M}$ étant un robot à n degrés de liberté[5], une configuration est définie par un vecteur $q = \{q_1, q_2, \ldots, q_n\}$ composé de l'ensemble des n valeurs des degrés de liberté de $\mathcal{M}$. C est défini par l'ensemble des valeurs possibles des n variables de q. Les problèmes de planification de mouvement dans W sont isomorphes aux problèmes de recherche dans un espace de dimension n. L'espace des configurations C se divise en deux sous-ensembles complémentaires dans C : C_{libre} et C_{obs}. C_{libre}

[4]La continuité de la courbure facilite le suivi de la trajectoire. La présence de discontinuités, dans le cas de chemins, implique un arrêt du mobile et une rotation sur place. La planification de chemin pose donc un problème de faisabilité. Chaque discontinuité implique des glissements et peut être source de mouvements irréalisables pour le mobile [Pru96].

[5]Un degré de liberté (ou de commande) correspond à un paramètre géométrique de position, linéaire ou angulaire. Pour un bras articulé, le nombre de degrés de liberté est défini par le nombre de ses articulations munies d'un actionneur. Pour un robot mobile de type voiture se déplaçant dans le plan, le nombre de degrés de liberté est égal à 3 : un degré en x, un degré en y et un degré en θ (rotation autour d'un point de $\mathcal{M}$).

est l'ensemble des configurations dites libres (*i.e.* en dehors des obstacles et sans contact). C_{obs} est l'ensemble des configurations en collision (*i.e.* en collision ou en contact) avec les obstacles. Les images des obstacles de W dans C sont appelées *C-obstacles*.

Afin d'intégrer la notion de contraintes dynamiques dans la planification de mouvement, B.R. Donald *et al.* [DXCR93] proposent d'utiliser l'espace des états X. Un état x, appartenant à X, est un ensemble de paramètres définissant les composantes d'une configuration et la dérivée (première ou seconde) par rapport au temps de certaines de ces composantes. Les déplacements du mobile considéré sont définis par un système d'équations différentielles. Les solutions de ce système d'équations définissent des déplacements conformes aux contraintes. La recherche de solutions dans X permet la planification de mouvement[6] pour un mobile soumis à des contraintes dynamiques.

1.1.2 Déplacements d'un robot

Un robot mobile $\mathcal{M}$ est un système mécanique composé d'un ou plusieurs corps reliés par des articulations. Il est défini par un système d'équations caractérisant sa forme et ses déplacements[7]. La planification de ses mouvements doit intégrer :

[6] La planification d'un robot mobile de type voiture [Lat91, LJTM94] est classiquement calculée en deux étapes : une étape d'approximation holonome et une étape d'introduction des contraintes non holonomes ; si la première étape échoue, le problème est sans solution. Cette première étape est fondée sur les chemins de *Reeds et Shepp* [RS90]. Ces chemins sont optimaux en longueur en l'absence d'obstacle. En présence d'obstacles, les chemins générés sont localement optimaux. La deuxième étape modifie successivement par morceaux le chemin généré pour respecter les contraintes de non holonomie. Les contraintes sont divisées entre holonomie en première étape et non holonomie en deuxième étape. L'espace des états X permet de formaliser toutes les contraintes dans un seul espace. La notion d'holonomie occupe une place importante en robotique. Pour J.P. Laumond, "Les systèmes non holonomes sont caractérisés par des contraintes cinématiques prenant la forme d'équations qui portent sur les dérivées des paramètres de configurations : le système est non holonome lorsque ces équations différentielles ne sont pas intégrables" [LDE+01]. La contrainte de roulement sans glissement est une contrainte non holonome. Cette contrainte implique l'inadéquation possible entre un chemin de C et son exécution dans W. Pourtant toute configuration de C accessible par un mobile soumis à des contraintes holonomes l'est aussi par un mobile identique soumis à des contraintes non holonomes. La non holonomie est donc caractérisée par une contrainte sur $\mathcal{M}$ sans restriction sur la dimension de C.

[7] Pour Galilée, le mouvement est indépendant du mobile et de l'environnement dans lequel il se produit et s'inscrit à l'intérieur du phénomène de l'écoulement du temps comme un simple changement de position. Pour Aristote, le temps est une conséquence du mouvement. Cette affirmation est justifiée par l'absurdité de la considération du temps dans un monde immobile. Le temps est de ce fait le reflet de la réalité qu'est le mouvement. Dans ce cas, l'accélération est le taux de variation

des contraintes géométriques (dues aux géométries de $\mathcal{M}$ et de W), des contraintes cinématiques (dues aux limitations de $\mathcal{M}$, indépendantes de leurs causes dans W), des contraintes dynamiques (dues aux forces auxquelles $\mathcal{M}$ est soumis), des contraintes de commande (dues aux limites des actionneurs de $\mathcal{M}$), des incertitudes de perception (dues aux capteurs de $\mathcal{M}$) et des incertitudes d'exécution (dues aux actionneurs de $\mathcal{M}$).

Nous proposons de formuler le problème de la planification de mouvement par :

Résoudre le problème de planification de mouvement, pour un mobile $\mathcal{M}$ évoluant dans un espace W contenant un ensemble d'obstacles $\mathcal{O}$, avec une position initiale p_d et un ensemble de positions finales P_f, équivaut à définir, s'il existe, un ensemble ordonné G de positions p reliant p_d à un élément de P_f. Cet ensemble doit vérifier les contraintes caractérisant la forme et les déplacements de $\mathcal{M}$.

À partir de cette définition, chaque configuration est produite par la sélection d'une configuration de G et par l'intégration d'une commande sur un intervalle de temps δt. Le choix de l'intervalle de temps δt définit la distance entre deux configurations successives appartenant à G. De ce fait, δt discrétise les espaces des configurations. Le passage d'une configuration à une autre est réalisé par application d'un vecteur de commandes de $\mathcal{M}$. J. Barraquand *et al.* [BL93] discrétisent la commande d'un robot non holonome par un vecteur de commande à deux dimensions définissant la vitesse linéaire v et l'angle de braquage ϕ. La discrétisation des commandes v et ϕ prend des valeurs dans l'ensemble défini par $\{-v_0, v_0\} \times \{-\phi_{max}, 0, \phi_{max}\}$. L'intégration d'une des six commandes possibles est associée à une valeur de δt unitaire et une valeur de v_0 égale au double du pas de discrétisation de C. La discrétisation des commandes simplifie la modélisation de la partie commande du mobile et permet d'énumérer méthodiquement toutes les configurations accessibles à partir d'une configuration donnée. Dans le cas continu, chaque couple de configurations est relié par l'utilisation d'une méthode dite locale[8].

---

de la vitesse par rapport à la distance parcourue. Pour Galilée, le temps est le premier phénomène et le mouvement une succession de positions dans le temps. Au $XVII^{\text{e}}$ siècle, la définition de Galilée est une véritable révolution dans la conception du temps et du mouvement. Elle introduit le concept d'accélération, définit comme le taux de variation de la vitesse par rapport au temps.

[8]Une méthode locale de planification (ou méthode de guidage) a pour objectif de relier tout couple de configurations, dans un espace continu, situées dans un voisinage, par une trajectoire admissible. Il n'existe pas de méthode locale générique complète pour tout système. En l'absence d'obstacles, elle se divise en deux familles : les méthodes exactes divisées entre familles de systèmes et les méthodes approximatives pour un système général. Les méthodes exactes sont

1.1.3 Complexité

La complexité[9] du problème de planification de mouvement est formalisée dans le problème du déménageur de piano. J.T. Schwartz *et al.* [SS83] montrent la complexité déterministe-polynômiale de ce problème. J.F. Canny [Can87] montre la classe *PEspace*[10] du problème pour un nombre de degrés de liberté non borné. Il décrit un algorithme de complexité en temps de résolution $O(m^n \log m)$. La complexité en temps d'un problème de planification de mouvement en présence d'obstacles est donc exponentielle en la dimension de l'espace des configurations[11].

différenciées pour les systèmes nilpotents (*i.e.* commandables sans dérive) [LS91], pour les systèmes chaînés (*i.e.* différentiels à deux entrées sinusoïdales, constantes par morceau ou polynômiales) [MRS95], pour les systèmes plats (*i.e.* différentiels composées de variables indépendantes résultat d'un nombre fini de dérivées) [MS93]. Les méthodes approximatives donnent des solutions approchées et sont sans garantie de convergence vers une solution. Ces méthodes produisent des commandes sous-optimales. Les méthodes de commande optimale ne sont définies que pour des modèles simplifiés de robot mobile soumis à des contraintes cinématiques à angle de braquage borné se déplaçant en marche avant uniquement ou se déplaçant en marche avant et arrière. L'ouvrage de J.P. Laumond [LDE+01] propose une synthèse de ces techniques.

[9] La complexité d'un algorithme (ou d'une méthode de résolution) est un critère d'évaluation et de comparaison des algorithmes. Pour remédier aux différences entre ordinateurs, la définition de la complexité s'accompagne d'un modèle abstrait d'ordinateur, visant à définir l'unité élémentaire d'espace mémoire et l'unité élémentaire opératoire. L'unité élémentaire d'espace mémoire définit les variables occupant une unité de stockage en mémoire. L'unité élémentaire opératoire définit les opérations exécutées en une unité de temps. La complexité s'exprime donc en espace ou en temps. La complexité en espace (respect. en temps) d'un algorithme est le nombre d'unités élémentaires d'espace (respect. opératoires) requis pour son exécution. La résolution d'un algorithme dépendant des données assignées à la définition du problème, la complexité est au pire ou moyenne. La complexité au pire donne la borne supérieure de la complexité. La complexité en moyenne donne la moyenne du nombre d'unités élémentaires utilisées. La complexité d'un problème est définie par la complexité de sa résolution. Elle est notée $O(f(m))$ si le nombre d'unités élémentaires nécessaires (en temps ou en espace) suit asymptotiquement les variations définies par la fonction $f(m)$. m définit dans ce cas la nature des données initiales du problème. Si $f(m)$ est un polynôme de degré constant indépendant de m, la complexité est polynômiale. Une complexité non polynômiale est exponentielle.

[10] Les problèmes se divisent en trois classes, allant des plus simples aux plus difficiles : la classe *P* pour les problèmes polynomiaux déterministes en temps, la classe *NP* pour les problèmes polynomiaux non-déterministes en temps et la classe *PEspace* pour les problèmes polynomiaux non-déterministes en espace. Un problème est *P* s'il est associé à un algorithme de longueur polynomiale en la taille de ses données. Un problème est *NP* s'il n'a pas d'algorithme de ce type, mais la validité d'une solution à ce problème est vérifiée par un algorithme de ce type. Un problème polynomial non-déterministe n'est pas difficile en soit puisque sa résolution peut se limiter à l'énumération de toutes les configurations possibles et au choix de la meilleure de toutes ces configurations. La problématique d'une telle résolution se situe dans le nombre de ces configurations et des chemins reliant ces configurations.

[11] La complexité est fonction de la complexité combinatoire des décompositions possibles de l'espace des configurations sans collision. L'espace libre est défini par un ensemble de demi-

Pour permettre la commande de robots hautement dimensionnés[12] (manipulateurs à 31 degrés de liberté et couple de mobiles à 3 degrés de liberté), J. Barraquand et J.C. Latombe [BL90] proposent d'utiliser une méthode probabiliste de planification de mouvement. Ce principe est repris dans de nombreuses méthodes [BL91, Bar91, Kav95, KLMR95, SSLO96, Sve97, AL94, LaV98]. L'attrait pour ces méthodes est justifié par leur relation à la complexité : la complexité combinatoire inhérente au problème de planification de mouvement est contournée par le biais de choix aléatoires. L'aléatoire inclus dans le déroulement de ces méthodes se retrouve dans les solutions calculées. Deux exécutions consécutives (partant de données identiques) peuvent produire deux solutions[13] différentes (si les données du problème admettent plusieurs solutions). Ces méthodes sont non-déterministes ; elles sont probabilistement complètes. Pour un problème donné, la probabilité de trouver une solution tend vers 1 quand le temps d'exécution tend vers l'infini. La vitesse de convergence vers la solution dépend des choix aléatoires effectués en cours d'exécution. Ainsi le temps de résolution et l'espace mémoire dépendent simultanément de l'exécution et des données initiales. Dans le cas de mobiles hautement dimensionnés (en degrés de liberté ou en contraintes), ces méthodes possèdent une complexité en moyenne inférieure à la complexité des méthodes déterministes [Lat99].

---

plans en $2D$, par un ensemble d'hyper-plans en 3D et par un ensemble d'hyper-plans de dimension d en $(d-1)D$. Les séquences de Davenport-Schinzel sont utiles pour l'analyse de tels ensembles [SA95].

[12]Nous définissons par espaces hautement dimensionnés, les espaces dont la combinatoire, induite par leur dimension, implique une géométrie implicite. La dimension de l'espace de recherche définit la combinatoire du problème considéré. Pour les espaces de dimension inférieure à 4, une décomposition géométrique exhaustive de l'espace permet de définir les solutions du problème. À partir de la quatrième dimension, l'espace des configurations devient un espace de recherche à géométrie implicite. La dimension de l'espace de recherche est définie par combinaison de la dimension de l'espace de travail et de la dimension du mobile. La notion d'espace de recherche de grande dimension est initiée par J. Barraquand *et al.* [BL90] avec un manipulateur à 31 degrés de liberté. S.M. LaValle [LaV04] présente un tel espace comme doté d'une centaine de dimensions.

[13]En géométrie algorithmique, un algorithme est randomisé s'il utilise des choix aléatoires sans modifier son résultat. Cette différence est dûe à la dimension de l'espace de recherche et à l'ensemble des solutions admissibles. La dimension de l'espace de recherche implique une recherche non-exhaustive. La difficulté associée à la découverte d'une solution suppose la suffisance d'une solution sans garantie d'optimalité. Les algorithmes randomisés et les méthodes probabilistes s'appuient pourtant sur une analyse commune des propriétés combinatoires des problèmes visant à écarter les cas extrêmes des cas possibles. Une étude de F. Lamiraux *et al.* [LL96] estime la complexité en moyenne d'une méthode probabiliste dans le cas d'un mobile polygonal en mouvement dans le plan. Le lecteur est invité à se reporter à l'ouvrage de J.D. Boissonnat *et al.* [eMY95] pour plus de détails.

1.2 Méthodes probabilistes

De par leur caractère probabiliste, ces méthodes génèrent des solutions souvent sous-optimales[14]. Les solutions peuvent être améliorées par une méthode de lissage des chemins [LSL$^+$98].

1.2.1 Méthode RPP

La méthode *Randomized Path Planning* (*RPP*) a été proposée par J. Barraquand *et al.* [BL91, Bar91]. Son principe se développe à partir de la méthode de descente de gradient proposée par O. Khatib [Kha85].

Initialement, la descente de gradient [Kha85] pour la planification de mouvement est un parcours du champ de potentiel « en meilleur d'abord ». Elle propose une solution pour les mouvements d'un bras articulé réduit à un élément. Les mouvements sont effectués sans contrainte de dynamique et de temps dans C. Les mouvements du bras sont assimilés aux déplacements d'une particule chargée soumise à deux types de champ magnétique : un champ attracteur de la configuration initiale vers la configuration finale ; des champs répulsifs associés à chaque obstacle. Le champ magnétique résultant guide naturellement la particule vers la configuration objectif. Le chemin-solution est obtenu par déplacement successif de la particule dans le champ magnétique. La superposition des deux types de champs peut provoquer des minima locaux, susceptibles d'emprisonner la particule pendant sa descente, et d'annuler les possibilités de résolution du problème. En conséquence, cette méthode ne garantit pas toujours de solution.

Pour rétablir la complétude, la méthode *RPP* [BL91, Bar91] propose d'échapper

[14]Le rapport à l'optimisation d'un problème consiste à définir les grandeurs à minimiser. Le nombre de ces grandeurs peut rendre certains cas impossibles. Le problème demeure non pas dans la définition de l'optimalité mais dans la recherche d'une solution optimale sans savoir si elle existe. Son existence étant supposée, la programmation dynamique propose d'obtenir la solution optimale d'un problème à partir des solutions optimales de ses sous-problèmes. C'est le principe d'optimalité de Bellman [LaV04]. Dans le cas d'un problème de planification de mouvement, une décomposition en sous-problèmes par introduction progressive des contraintes peut rendre plus difficile (voir impossible) les résolutions successives associées [LSL$^+$98]. Une décomposition en sous-problèmes par une succession de représentations locales peut provoquer des minima locaux (pouvant annuler les possibilités de résolution) [Kha85]. La difficulté de la résolution du problème de planification de mouvement pour un mobile hautement dimensionné justifie ainsi la suffisance d'une résolution potentiellement sous-optimale.

des minima locaux par un déplacement aléatoire. Ce déplacement aléatoire est un mouvement *Brownien*[15]. Il est défini par un nombre s de mouvements élémentaires indépendants d'amplitude Δt. Le déplacement aléatoire a une durée totale de $T = s\Delta t$. Pour chaque pas, Δi est la projection de Δt sur le $i^{ème}$ axe de configuration. Δi est proportionnelle aux variations définies sur la dimension i. L'amplitude $D_i(T)$ résultante des s variations Δi est la différence entre la configuration à l'instant t et la configuration initiale. $D_i(T)$ est proportionnelle à $s\Delta i^2$ et est appelée variance. La variance et l'écart type sont respectivement proportionnels à t et à $\sqrt{t}$. Un tel mouvement obéit à une loi normale (ou loi de Gauss), d'espérance nulle et d'écart type $\sqrt{t}$. La méthode ***RPP*** ne construit pas explicitement C_{obs} et C_{libre}. Cette absence de représentation explicite rend les réflexions impossibles. Pour conserver les mouvements aléatoires dans C_{libre}, chaque mouvement en collision est simplement oublié au bénéfice du mouvement sans collision suivant. La valeur de t nécessaire pour échapper d'un minimum local est fonction de la valeur attractive de ce minimum.

1.2.2 Réseaux Probabilistes

Les *Méthodes de Réseaux Probabilistes* (*MRP*) ont été proposées simultanément par L.E. Kavraki *et al.* [Kav95, KLMR95] sous l'intitulé *Probabilistic RoadMap* (*PRM*) et par P. Svestka *et al.* [Sve97] sous l'intitulé *Probabilistic Path Planner* (*PPP*). Le principe de résolution de ces méthodes divise la planification de mouvement en deux étapes successives : une étape de construction d'un graphe appelée *phase d'apprentissage* et une étape de construction d'un chemin basé sur ce graphe appelée *phase de recherche*.

Pendant la *phase d'apprentissage*, le graphe est construit dans l'espace C_{libre} où chaque nœud est choisi aléatoirement selon une distribution uniforme dans C_{libre}. La distribution uniforme est motivée par la nécessité d'explorer tout l'espace. Elle est obtenue par des tirages aléatoires dans C, associés à un détecteur de collision. Les nœuds sont reliés par une méthode locale associant, à deux configurations, un chemin sans collision respectant les contraintes du mobile. Les chemins validés (*i.e.* sans collision, continûment ou par discrétisation[16] dans C) ne sont pas obli-

[15]Le mouvement *Brownien* est également appelé processus de Wiener. L'amplitude des déplacements successifs suit une loi gaussienne. Chaque déplacement est indépendant des déplacements précédents. En présence d'obstacles, les déplacements doivent rester dans C_{libre}. En cas de collision, l'approche classique est de réfléchir, sur les obstacles, les mouvements menant à des collisions.

[16]Pour un chemin discrétisé en une séquence de configurations, la détection de collision se

gatoirement conservés par le planificateur local ; un arc entre deux configurations dénote l'existence d'un chemin sans préciser obligatoirement la succession des commandes associées à son suivi.

Pendant la *phase de recherche*, le graphe est utilisé pour relier deux configurations q_{init} et q_{obj} quelconques de C_{libre}. La recherche d'un chemin consiste à ajouter les deux nœuds correspondants à q_{init} et q_{obj} dans le graphe et à rechercher un chemin les connectant. Cette phase peut être répétée avec deux configurations quelconques tant que le mobile et les obstacles restent inchangés. La *phase d'apprentissage* est parfois considérée comme un pré-traitement. Dans ce cas, le graphe G résultant de la *phase d'apprentissage* peut être réutilisé[17] pour différentes requêtes de planification tant que l'environnement ne change pas. Dans le cadre des déplacements de plusieurs mobiles dans un même environnement [SO98], les graphes de la *phase d'apprentissage* sont superposables dans un « *super-graph* » appelé « *flat super-graph* » pour des mobiles identiques et appelé « *multi-level super-graph* » pour des mobiles différents. La recherche d'une solution dans le « *super-graph* » permet de pallier au problème d'une résolution découplée pour chacun des mobiles.

1.2.3 Fil d'Ariane

La méthode *du Fil d'Ariane* a été proposée par J.M. Ahuactzin-Larios [AL94]. Son principe repose sur la décomposition de la trajectoire dans l'espace des commandes. Habituellement une succession de configurations, la trajectoire devient une succession de commandes. Pour un robot mobile $\mathcal{M}$, la trajectoire est une séquence d'accélérations et de rotations. Cette séquence, appelée plan, est une succession de commandes du type « accélérer-décélérer » et « tourner », de la forme $(a_1, \theta_1,\ a_2, \theta_2, \ldots, a_n, \theta_n)$. Pour un système de k commandes possibles ($k = 2$ dans le cas du robot mobile $\mathcal{M}$), le vecteur q, de dimension $m < k$, définit une suite de commandes distinctes. La configuration finale devient le résultat de l'application de n vecteurs q à partir de la configuration initiale (avec $n-1$ configura-

résume au test de l'exclusion de C_{obs} pour chacune de ces configurations. Les chemins sont donc validés par approximation.

[17] Les méthodes *PRM* soulignent trois aspects du problème de planification dans C, pour lesquelles la construction du graphe G de la *phase d'apprentissage* diffère :

- Répondre à tous les problèmes de planification dans C ;
- Répondre aux requêtes de connexion à partir d'une configuration initiale ;
- Répondre aux requêtes de planification entre deux configurations.

tions intermédiaires). Cette méthode suit en parallèle deux algorithmes : *Explore* dont la fonction est la collecte des informations relatives à l'espace de travail et *Search* dont la fonction est la recherche de trajectoires vers la configuration finale. Ces deux algorithmes construisent incrémentalement un arbre G en partant de la configuration initiale. La fonction *Explore* sélectionne une configuration libre (appelée balise) q_e accessible à partir d'une configuration de G et la plus distante possible de G (pour minimiser le nombre n de vecteurs q) ; si la fonction *Search* relie q_e avec la configuration finale, le chemin-solution est la concaténation des chemins utilisés.

Cette méthode garantit la complétude à une résolution ϵ près. Si la fonction *Explore* ne peut pas sélectionner une nouvelle configuration à une distance au moins égale à ϵ de toutes les configurations de G, alors G recouvre l'espace des configurations ; si aucune solution n'est proposée pour un recouvrement fixé, aucune solution n'existe à ϵ près.

1.2.4 Méthode RRT

La méthode *des arbres aléatoires d'exploration rapide* (*RRT*) a été proposée par S.M. LaValle [LaV98] sous l'intitulé *Rapidly-exploring Random Tree*. Son principe repose sur la construction d'un arbre G dans l'espace de recherche $\mathcal{S}$ considéré[18]. À partir de la position initiale, la construction de l'arbre est réalisée par intégrations successives de commandes visant à rapprocher le mobile d'un élément e choisi aléatoirement dans l'espace de recherche à chaque itération. Pour éviter les cycles, deux éléments de G ne peuvent être identiques. La répartition uniforme des tirages des éléments e dans $\mathcal{S}$ garantit la convergence de l'algorithme vers une solution. Cette répartition uniforme implique également une plus grande probabilité de répartition des tirages vers les zones inexplorées de $\mathcal{S}$. La notion de proximité entre deux éléments est définie par une distance métrique. La construction de l'arbre est réalisée par itération de trois phases : génération aléatoire d'un élément e_{rand} de $\mathcal{S}$, sélection de l'élément e_{prox} de G le plus proche de e_{rand} et génération d'un nouvel élément e_{new} par application d'une commande optimale à partir de e_{prox} visant à minimiser la distance entre e_{new} et e_{rand}.

[18] Le choix de la nature de l'espace de recherche dépend du modèle de résolution choisi. Les principales variantes sont l'espace des configurations C [LP83], l'espace des états X [DXCR93] et l'espace des états-temps ST [Fra93]. C est destiné à la planification de mouvement d'un mobile dans un environnement statique. X ajoute la prise en compte de contraintes dynamiques. ST ajoute la possibilité d'un environnement dynamique.

La sélection de l'élément e_{prox} est réalisable par des méthodes de partitionnement de l'espace $\mathcal{S}$ [dBSvKO00]. A. Atramentov *et al.* [AL02] proposent un algorithme de recherche de plus proche voisin basé sur les *KdTree*[19]. À chaque nouvelle insertion dans G, l'arbre progresse dans l'environnement. La progression résultant de l'ajout de chaque e_{new} est définie par l'intervalle de temps utilisé dans le système différentiel associé au mobile. L'objectif définissant la position finale recherchée est un ensemble d'éléments de $\mathcal{S}$ pour augmenter la convergence de l'algorithme vers la solution dans le cas de mobile soumis à des contraintes non-géométriques (*i.e.* cinématiques, dynamiques et de commande). La commande du mobile est définie par une évaluation itérative de toutes les commandes applicables à partir de l'élément q_{prox} ; sur ce principe de générer-tester appliqué à la commande, cette méthode ne nécessite pas de méthode de planification locale. La distance métrique définit les poids de chaque dimension dans la relation de proximité entre deux éléments de $\mathcal{S}$. La commande est implicitement définie par cette relation de proximité. L'insertion d'un nouvel élément e_{new} est conditionnée par un détecteur de collision. Aucun pré-calcul n'est nécessaire[20].

1.3 Contributions

L'objectif de cette thèse est l'étude des relations inhérentes aux principaux composants d'une planification probabiliste incrémentale. L'étude est centrée sur les méthodes du type *RRT* appliquées à la robotique mobile. Ce travail se destine génériquement à tous les espaces hautement dimensionnés, définissant les mouvements d'un mobile soumis à des contraintes dynamiques. La réalisation d'un planificateur de mouvement probabiliste suppose l'assemblage d'un ensemble de composants responsable d'opérations spécialisées ; les principaux composants de cet ensemble, issus de la géométrie algorithmique, sont la détection de collision et la recherche de plus proche voisin ; des composants issus de la robotique, tels qu'un système de commande, un modèle différentiel et une méthode de planification locale, viennent s'ajouter à ces composants géométriques pour définir une

[19] Un *KdTree* est un arbre binaire particulier : chaque nœud est associé à une région rectangulaire d'une des k dimensions de l'espace. Les nœuds sont alternativement classés vis-à-vis d'une des k dimensions. Un *KdTree* équilibré composé de n éléments de $\mathbb{R}^k$ peut se construire en temps relatif à $O(n \log n)$ et en espace relatif à $O(\log n)$.

[20] La nature incrémentale de l'algorithme associe cette méthode à la résolution d'un problème unique. Dans le cadre de la méthode *PRM* (§ 1.2.2), la réalisation de la *phase d'apprentissage* dans un pré-calcul permet d'utiliser le graphe pré-calculé pour plusieurs requêtes dans un même environnement pour un même mobile. La méthode *RRT* intégrant dans son algorithme position initiale et finale, le graphe G n'est utile que pour une unique requête.

méthode entière de planification de mouvement. Le développement de ces composants communs pour la construction incrémentale d'un graphe reste une opération sensible à une fonction d'évaluation de la progression du graphe dans l'espace implicitement défini. Nous concentrons, dans un premier temps, nos travaux sur l'étude des interactions entre ces différents composants dans un but d'optimisation de leurs interactions dans l'algorithmique de la planification de mouvement. Les méthodes de décomposition cellulaire de l'espace sont garantes d'une propriété de complétude sous condition d'un pas de discrétisation des contraintes géométriques. Ces méthodes s'adressent à des mobiles génériques mais dans des espaces sans grande dimension [JC02]. Basée sur une analyse géométrique récursive, ces méthodes traduisent l'espace en une succession de cellules régulières. Ces décompositions s'abstenant des considérations relatives à la dynamique des mobiles, nous étudions l'influence des échantillonnages pour définir une décomposition irrégulière basée sur la notion d'accessibilité, adaptée à la planification de mouvement probabiliste itérative [Jou03, JC03, JC04a, JC04b].

Le chapitre 2 définit les principes associés à une résolution du type *RRT* en détaillant les principales approches existantes. Nous présentons un algorithme, optimisant les appels au détecteur de collision et au système différentiel définissant les déplacements du mobile. L'étude de l'ensemble des variantes de la méthode *RRT* est suivie d'une analyse des relations entre les composantes opérationnelles de l'algorithme commun à toutes ses variantes ; la présentation des résultats d'implémentation conclut sur une réduction du temps de construction de G.

Le chapitre 3 présente les principales variantes existantes pour un échantillonnage de l'espace de recherche ; nous proposons une décomposition des espaces libres basée sur la notion d'accessibilité des configurations de G. Cette décomposition nous permet de définir une enveloppe dynamique de l'échantillonnage de C, dont la progression est définie par l'expansion de G. Nous proposons conjointement l'ensemble des algorithmes de construction de cette décomposition. Après une évaluation des temps de construction pour un mobile ponctuel, nous proposons le détail des algorithmes nécessaires à l'utilisation de notre décomposition en cellules irrégulières des espaces libres ; leur utilisation appliquée à une résolution du type *RRT* est évaluée par des simulations dans un passage étroit et dans un champ d'obstacles.

Le chapitre 4 présente une conclusion de l'ensemble de nos contributions et les perspectives faisant suite à nos travaux.

Chapitre 2

Arbres aléatoires d'exploration

2.1 Principe des *RRT*

Dans sa forme originale, la méthode ***RRT*** est un arbre $G = (V, E)$ avec V l'ensemble des éléments de l'espace de recherche[1] de l'arbre et E l'ensemble des arêtes de l'arbre. À partir d'une configuration initiale q_{init}, l'objectif est d'énoncer une suite de commandes, permettant au mobile $\mathcal{M}$, d'explorer[2] l'espace des configurations C. Pour résoudre ce problème, la méthode ***RRT*** recherche une solution par construction d'un arbre dont le nœud-racine est la configuration q_{init}. Les nœuds de l'arbre sont des configurations admissibles de $\mathcal{M}$. Les arcs de l'arbre sont les commandes à appliquer pour passer d'une configuration à une autre. La méthode ***RRT*** est une recherche incrémentale aléatoire des commandes permettant une exploration uniforme de l'espace. Elle répète successivement une boucle composée de trois phases : génération d'une configuration q_{rand}, sélection d'une configuration q_{prox}, génération d'une nouvelle configuration q_{new}.

Lors d'une phase de génération d'une configuration q_{rand}, une fonction aléatoire sélectionne un élément de l'espace des configurations. La phase suivante sélectionne l'élément de G, le plus proche nœud de q_{rand} ; cet élément est appelé q_{prox}.

[1] Cet espace de recherche est ici assimilé à l'espace des configurations par simplicité.

[2] La formulation initiale de l'algorithme de la méthode ***RRT*** ne mentionne pas l'objectif de l'exploration de C, lequel est le plus souvent de rejoindre une configuration finale q_{obj}. Cette présentation est probablement volontairement généraliste, pour permettre la formulation de problèmes plus divers tels que la recherche d'objet dans un environnement [TLM03], le ralliement d'un ensemble de points de passages ou la poursuite d'un mobile [SLS02].

Cette sélection est fondée sur la définition d'une distance métrique ρ. La phase de génération d'une nouvelle configuration q_{new} est l'application d'une commande visant à rapprocher q_{prox} de q_{rand}. La nouvelle configuration q_{new} est créée par intégration des contraintes du mobile $\mathcal{M}$ pendant un intervalle de temps fixé à partir de la configuration q_{prox}.

2.1.1 Construction du graphe de la méthode *RRT*

Initialement, la méthode ***RRT*** considère le problème de planification de mouvement pour un mobile quelconque[3]. L'espace est de dimension arbitraire et dénué d'obstacle. L'algorithme 1 présente la construction élémentaire du graphe de la méthode ***RRT***.

Les trois étapes de construction sont ici conduites dans les fonctions ① , ② et ③ (ALG. 1). `confAleatoire` assure une répartition uniforme des tirages aléatoires dans C et garantit ainsi une exploration uniforme de C.

`confLaPlusProche` sélectionne la configuration de G la plus proche[4] de q_{rand}. Cette relation de proximité est définie par la distance métrique ρ (ALG. 2).

`nouvelleConf` définit une nouvelle configuration q_{new} à partir de q_{prox} en direction de q_{rand}. Le mobile étant ici considéré comme holonome, il est toujours possible d'appliquer une commande visant à déplacer q_{prox} en direction de q_{rand}. L'amplitude de ce déplacement est définie par la valeur Δt. La fonction `ajouterConf` ajoute q_{new} dans la liste des sommets de G et la fonction `ajouterArc` ajoute un arc reliant q_{prox} et q_{new}.

[3]Les possibles contraintes associées au mobile ne sont pas mentionnées dans l'algorithme original. Les modifications à apporter pour l'ajout de contraintes (telles que la non holonomie) sont considérées comme mineures dans la formulation de la construction de G. La précision de l'intégration des contraintes est principalement fonction de la méthode de planification locale choisie.

[4]La recherche de plus proches voisins en dimension arbitraire est optimisée avec les ***KdTree*** (ou arbre K-d) dans le cadre des méthodes probabilistes ***PRM*** et ***RRT*** [AL02]. La diminution du temps de recherche d'un plus proche voisin permet l'utilisation d'une distance métrique plus complexe.

```
consRrt(q_init, k, Δt, C)
  | init(q_init, G) ;                                          ①
  | pour i ← 1 à k
      q_rand ← confAleatoire(C) ;
      q_prox ← confLaPlusProche(q_rand, G) ;                   ②
      q_new ← nouvelleConf(q_prox, q_rand, Δt) ;               ③
      ajouterConf(q_new, G) ;
      ajouterArc(q_prox, q_new, G) ;
  | retourner G ;
```

ALG. 1: Construction élémentaire du graphe de la méthode *RRT*.

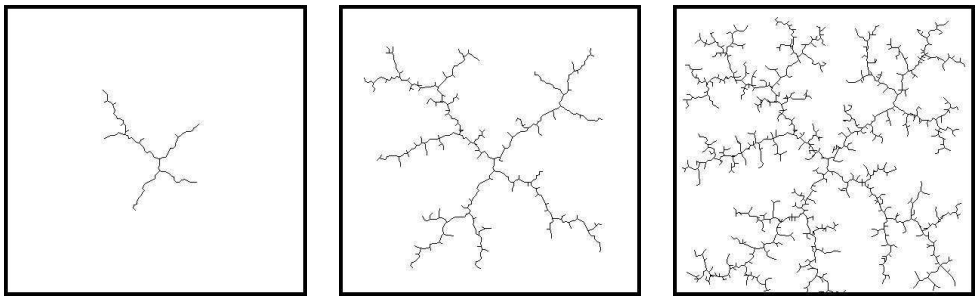
```
confLaPlusProche(q_rand, G)
  | d ← +∞ ;
  | pour chaque q ∈ G
      si ρ(q, q_rand) < d
        | q_prox ← q ;
        | d ← ρ(q, q_rand) ;
  | retourner q_prox ;
```

ALG. 2: Recherche du plus proche voisin.

Les figures 2.1 *(a)*, *(b)* et *(c)* illustrent l'expansion de G après respectivement 100, 500 et 1500 tirages. Les tirages aléatoires sont uniformément répartis dans un espace carré. q_{init} est au centre de la fenêtre. Le mobile est ponctuel soumis à des contraintes holonomes ; C est un espace à deux dimensions.

2.1.2 Arbre ou graphe

À chaque étape de génération d'une nouvelle configuration q_{new}, la méthode *RRT* ajoute une configuration par propagation d'une configuration q_{prox} de G. Aucune restriction n'est posée sur q_{new} vis-à-vis de l'ensemble des configurations



(a) *(b)* *(c)*

FIG. 2.1 – Expansion du graphe de la méthode *RRT* dans un espace carré sans obstacle.

Le graphe G tend à couvrir itérativement l'ensemble de l'espace après 100 *tirages aléatoires en (a), après* 500 *en (b) et après* 1500 *en (c).*

de G. De ce fait, q_{new} peut être identique à une configuration q_{exist} (précédemment insérée dans G) pour chacun de ces paramètres. Ainsi il est possible de construire un graphe avec ou sans cycle. Soit $Card$ le cardinal d'un ensemble, si $Card(V) = Card(E) - 1$, le graphe est sans cycle. Pour éviter l'empilement de mouvements identiques, chaque nœud q_{prox} n'est étendu en direction de q_{rand} pour créer q_{new} que s'il ne possède pas déjà un nœud-fils considéré comme identique. Si q_{prox} est extensible en direction de q_{rand}, un nouvel arc reliant q_{prox} à q_{new} est inséré dans E. Si $Card(V) \leq Card(E)$, le graphe contient au moins un cycle. q_{new} est supprimée et un arc reliant q_{prox} à q_{exist} est inséré dans E.

Dans sa résolution et notamment dans les espaces de recherche de dimension supérieure à 3, l'optimalité du chemin-résultat n'est pas une priorité. La création de cycles[5] a pour conséquence une diminution du nombre d'expansions de G dans les zones inexplorées.

2.1.3 Expansion naturelle

La répartition aléatoire des tirages guidant les expansions oriente naturellement la croissance de G vers les régions les plus larges de l'espace. Cette affirmation

[5]La création de cycles permettrait cependant de dresser une liste des solutions possibles à des situations d'interblocage. Cette réflexion étant plus topologique que géométrique, la méthode *RRT* est préférée sans cycle [LaV98].

est vérifiée par la construction d'un diagramme de Voronoï[6] associant, à chaque nouveau nœud de G, une cellule de Voronoï. À chaque itération de la méthode *RRT*, la probabilité de localisation du prochain tirage aléatoire est plus importante en direction des plus grandes cellules du diagramme de Voronoï défini par l'ensemble des tirages aléatoires précédents. Soit C_k la distribution des k tirages aléatoires dans l'espace des configurations C ; C_k converge en probabilité vers C sous condition de l'uniformité de la répartition des tirages aléatoires dans C [LK00].

La triangulation de Delaunay étant le dual du diagramme de Voronoï, un exemple d'expansion du graphe associé à la méthode *RRT* est présenté en figure 2.2. Les graphes représentés en *(a)*, *(b)*, *(c)* sont respectivement les résultats de 25, 275 et 775 expansions dont les triangulations de Delaunay sont représentées en *(a')*, *(b')* et *(c')*. L'espace est de forme carré à deux dimensions et sans obstacle. Le graphe G suit l'algorithme de construction précédemment défini (ALG. 1). La loi de répartition des tirages aléatoires est uniforme. A chaque itération, l'ajout d'un nouveau point conduit à la construction d'une nouvelle triangulation. La configuration initiale, fixant le point de départ de l'expansion du graphe G est représentée par un cercle en *(a)*, *(b)* et *(c)*. Dans la figure 2.2, la configuration initiale est au centre de C. La figure 2.3 fait état de l'évolution des nouvelles configurations de G au fil des itérations. Les axes sont gradués en échelles logarithmiques. L'axe des abscisses représente le nombre de configurations contenues dans le graphe et l'axe des ordonnées représente le pourcentage de la surface totale S. Le graphique relatif aux aires présente les variations de l'aire moyenne, de l'aire minimale et de l'aire maximale. L'aire moyenne est la moyenne des aires des triangles. L'aire minimale (*respect.* maximale) est l'aire du plus petit (*respect.* du plus grand) triangle. Le graphique relatif aux écart-types présente les variations de l'écart-type, de l'écart-type minimum et de l'écart-type maximum. La configuration initiale divise l'espace en quatre triangles d'aire 0.25, d'écart-type nul. L'aire moyenne des triangles décroît linéairement avec le nombre de configurations. Sur les deux graphiques (FIG. 2.3), les positions des figures *(a)*, *(b)* et *(c)* de la figure 2.2 sont

[6] Le calcul du diagramme de Voronoï est un des problèmes célèbres de la géométrie algorithmique. Son intérêt s'explique par la remarquable diversité de ses propriétés. Il permet notamment de résoudre en temps optimal d'autres problèmes comme le calcul des plus proches voisins. Pour un ensemble de points, le diagramme de Voronoï est formé de segments de droite. Pour un ensemble d'obstacles, le diagramme de Voronoï est formé de segments de droites et d'arcs de paraboles. Il est calculable pour des ensembles de points [O'R94], d'obstacles segments (de façon incrémentale) [BDS$^+$92] et pour un ensemble d'obstacles polygonaux convexes [AGSS89]. En représentant les obstacles par l'ensemble des points situés à leurs frontières (résultat possible d'un échantillonnage de surface par balayage d'un laser), il est appelé diagramme de Voronoï approché. Dans le cas d'obstacles polygonaux, il est appelé diagramme de Voronoï généralisé. Si les obstacles sont représentés par des courbes (obstacles généralisés), il n'existe actuellement aucun algorithme pour calculer le diagramme de Voronoï généralisé correspondant [ICK$^+$99].

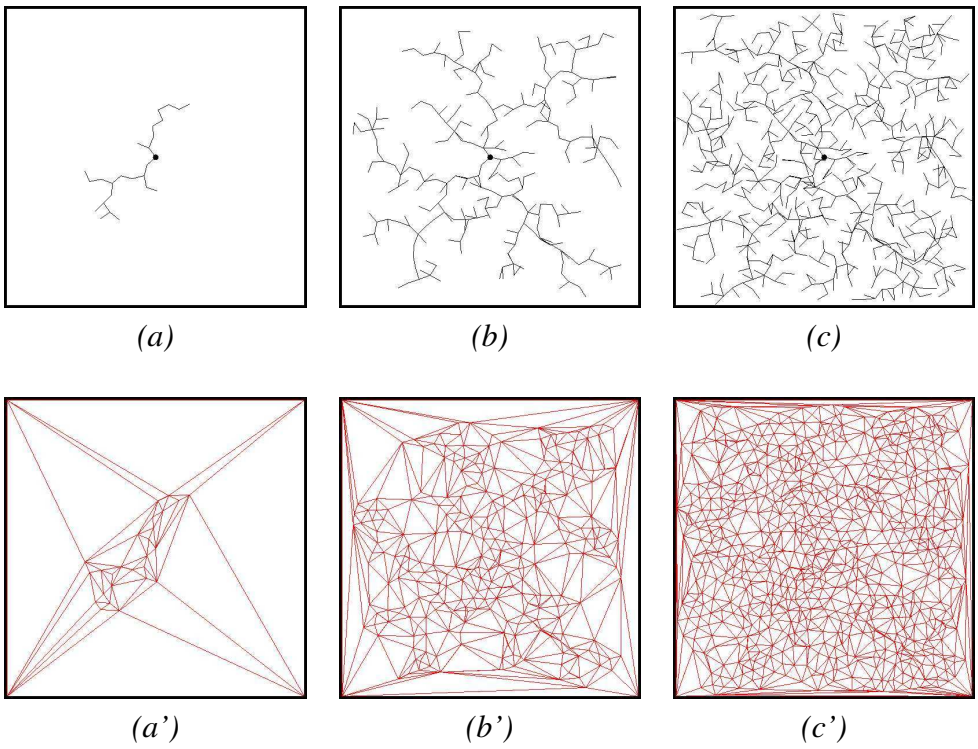


FIG. 2.2 – Analyse de la répartition des tirages aléatoires.

Les figures (a), (b) et (c) montrent trois états de la progression du graphe G après respectivement 25, 275 *et* 775 *itérations. Les figures (a'), (b') et (c') montrent les triangulations de Delaunay correspondantes.*

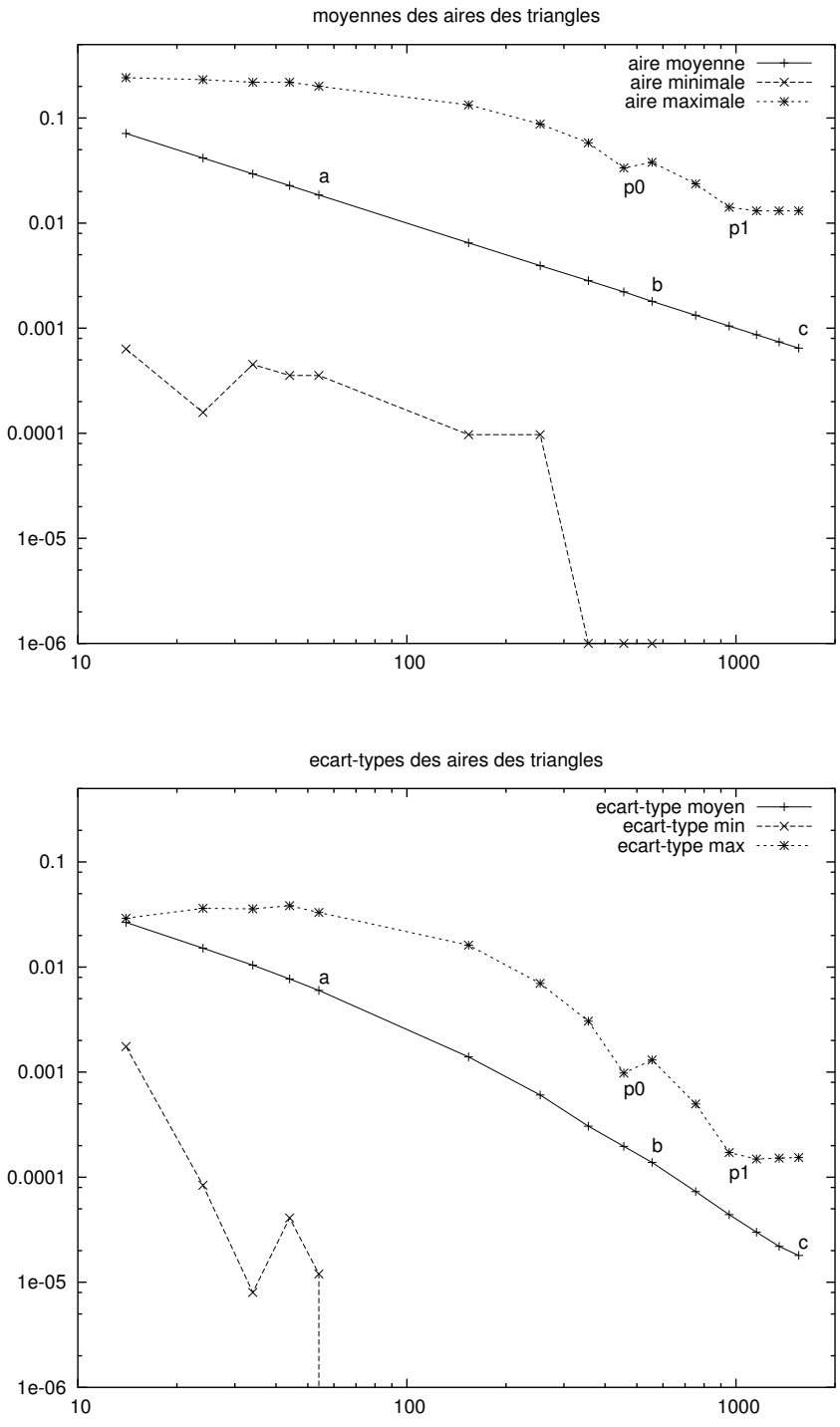


FIG. 2.3 – Moyennes et écart-types de la répartition des tirages aléatoires.

Les graphiques présentent l'évolution de la moyenne et de l'écart type des aires des triangles présentés dans la figures 2.2.

placées près des courbes des aires moyennes et des écart-types. Les variations des aires (ou écarts-types) maximum et minimum peuvent croître ou décroître selon leur positionnement relatif à la valeur moyenne en diminution permanente. De par l'échelle logarithmique, la position des variations minimales (d'aires et d'écarts-types) vis-à-vis des moyennes montre la quasi-égalité entre valeur moyenne et valeur minimale. La position des variations maximales montre à l'inverse la présence de triangles bien plus grands que la valeur moyenne avant un seuil de densité (8.15 fois plus grands avant 353 configurations[7]). Après ce seuil de 353 configurations, le rapport entre plus grand triangle et moyenne progresse par paliers. Deux paliers ($p0$ et $p1$) sont placés sur les courbes des moyennes et des écart-types (FIG. 2.3). Ce rapport se stabilise autour de 2 à partir de $p1$. La position de la configuration initiale est sans influence sur les statistiques relatives à l'expansion. La probabilité d'expansion est, par construction, favorisée en direction des triangles d'aires maximales.

2.1.4 Condition de terminaison

La formulation complète d'une requête de planification de trajectoire pour un mobile comprend une configuration-objectif. Cette configuration-objectif est instanciée en q_{obj} ou un ensemble de configurations C_{obj}. La restriction de la recherche à une unique configuration-objectif est pénalisante pour des mobiles soumis à des contraintes dynamiques ou non-holonomes. Pour améliorer la convergence vers l'objectif, les résolutions du type ***RRT*** utilisent une configuration q_{obj} dont les composantes ne sont pas toutes fixes ; Le problème de planification revient à trouver un chemin reliant q_{init} à un des éléments de l'ensemble C_{obj}. Partant de q_{init}, le graphe G cherche à atteindre une configuration q_{obj}. Cette recherche est réalisée par ajouts successifs d'une nouvelle configuration q_{new} dans l'arbre G. La variable k définit le nombre d'itérations jugées nécessaires à la résolution du problème. Si ce nombre d'itérations est insuffisant, il est possible de poursuivre la recherche en réalisant à nouveau k itérations à partir de l'arbre précédemment généré. La construction de G se termine dès que $q_{obj} \cap G \neq \emptyset$.

[7]Pour une sphère, un point intérieur est situé à une distance du centre strictement inférieure à son rayon. Un empilement de sphères est la réunion de sphères de même rayon et sans point intérieur commun. La densité d'un empilement de sphères est la proportion de l'espace occupée par ces sphères. Chaque configuration est à une distance d d'une configuration précédente par intégration de la fonction *nouvelleConf* (ALG. 1 ③). L'assimilation de chaque configuration à une sphère de rayon d permet de calculer un nombre de configurations n associé à une densité d'empilement de valeur 1 avec $n = S/(\pi d^2)$.

2.2 Expression des contraintes

2.2.1 Kinodynamique

La planification de mouvement est par essence la résolution d'un calcul de trajectoires admissibles[8] sans collision. Ce problème est de nature géométrique. Les planificateurs de mouvement complet [LSL$^+$98] s'accommodent d'une première phase simplificatrice, calculant des chemins purement géométriques, sans considérer les commandes nécessaires à l'exécution de ces chemins, puis d'une seconde phase de reformulation du problème par insertion de contraintes supplémentaires associées aux commandes de $\mathcal{M}$. Cette décomposition peut conduire à une résolution simplifiée de par la distinction des contraintes. Elle rend cependant plus difficile la seconde résolution, par contrainte de cette seconde résolution dans l'espace de la première. Dans un cas extrême, la première résolution rend impossible la satisfaction de la seconde résolution.

La généricité de l'approche *RRT* permet de considérer les contraintes cinématiques et dynamiques sans distinction. La planification de mouvement ne considère de ce fait plus uniquement les contraintes de position mais aussi celles de commande. Cette considération unifiée des contraintes permet un rapprochement des limites mécaniques inhérentes au mobile considéré. L'espace de recherche $\mathcal{S}$ est alors indifféremment espace des configurations C ou espace des états X. La prise en compte unifiée des contraintes cinématiques et dynamiques est appelée *kinodynamique* [DXCR93]. Elle impose une relation entre chaque configuration q et sa dérivée temporelle $\dot{q}$. Cette relation est définie par la fonction F avec :

$$F(q, \dot{q}) = 0$$

Dans une optique de modélisation opérationnelle, les robots mobiles sont très souvent modélisés par un système non holonome [LSL$^+$98]. J.P. Laumond définit

[8] Un chemin est admissible s'il vérifie les contraintes de déplacement du mobile. Une trajectoire est admissible si elle vérifie les contraintes de déplacement et de commande du mobile. Un chemin est donc l'image d'une trajectoire dans C [LDE$^+$01]. Cette dissociation entre domaine de déplacement et domaine de commande a pour effet d'annexer les résultats des planificateurs de mouvement aux méthodes de guidage. Elle pose ainsi la question de la légitimité de la planification purement géométrique de chemins, en dehors des considérations de temps et de dynamique. Les chemins ainsi générés sont composés de discontinuités, nécessitant des points d'arrêt au cours du déplacement du mobile. Ce problème de discontinuité est résolu par l'insertion de courbes à courbure continue telles que les courbes de Béziers [Kha96] et les clothoïdes [Fle96]. En fonction des contraintes considérées, la planification de mouvement est un calcul de chemins ou de trajectoires.

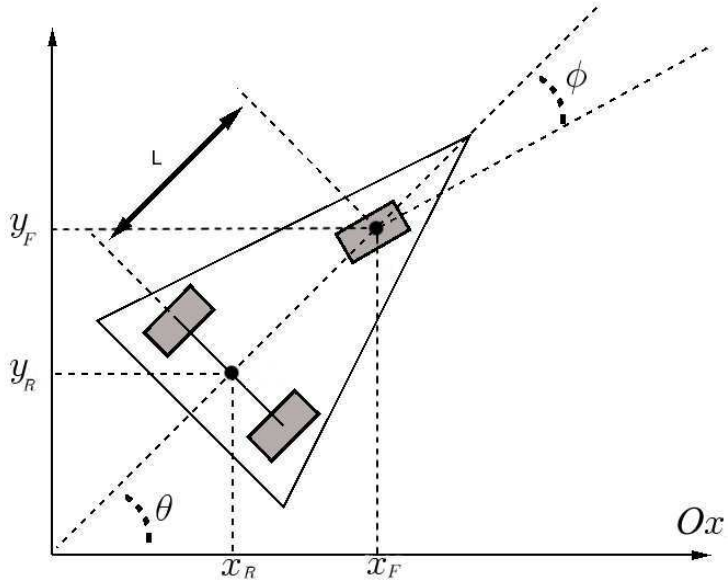


FIG. 2.4 – Mobile de type voiture.

Le mobile de type voiture est modélisé par un vecteur $3D$ (x, y, θ) pour un déplacement à vitesse constante avec pour commande l'angle des roues ϕ.

un mobile non holonome par un système dont « les équations différentielles ne sont pas intégrables » [LDE+01]. Il est de ce fait impossible de supprimer $\dot{q}$ dans l'équation précédente. La figure 2.4 illustre l'application de contraintes non holonomes sur un mobile de type voiture : Les points P_F et P_R sont définis dans le plan du mobile, ici représenté par un triangle. Le couple de coordonnées (x_R, y_R) définit la position du milieu de l'axe des roues arrières ; le point P_F est à la verticale du point d'application des forces de la roue avant sur le sol ; θ est l'angle des roues arrières par rapport à l'axe des abscisses Ox. Il définit également l'angle du mobile ; ϕ est l'angle de la roue avant vis-à-vis de l'angle du mobile[9]. Les roues arrières sont fixes.

Les déplacements du mobile sont supposés sans glissement. La contrainte de dé-

[9]Dans le cas d'un mobile de type voiture composé de deux roues directrices, P_F est au milieu des roues avant et l'angle ϕ est l'angle d'une roue virtuelle au milieu de l'axe des roues avant. L'épure de Jeantaud (ou d'Ackermann pour les anglo-saxons) impose aux quatre roues de se déplacer par rapport à un même centre de rotation pour minimiser les forces de frottement avec le sol. En réalité, toutes les roues se déplacent sur des courbes dont les centres de rotation diffèrent et la position du centre de rotation se déplace à chaque variation de vitesse.

placement sur les angles des roues du mobile se traduit par :

$$tan(\theta) = \frac{\dot{y}}{\dot{x}}$$

Cette contrainte d'orientation équivaut à considérer le système différentiel défini par les équations suivantes (avec v la vitesse linéaire du mobile) :

$$\begin{cases} \dot{x} = v\ cos(\theta) \\ \dot{y} = v\ sin(\theta) \\ \dot{\theta} = \frac{v}{L}\ tan(\phi) \end{cases}$$

Pour satisfaire les contraintes usuelles des mobiles de type voiture, il reste à ajouter une borne supérieure à ϕ (correspondant à l'angle de braquage maximum) et une borne supérieure à $\dot{\phi}$ (correspondant à la vitesse limite des variations de l'angle des roues). Le système complet est donc :

$$\begin{cases} \dot{x} = v\ cos(\theta) \\ \dot{y} = v\ sin(\theta) \\ \dot{\theta} = \frac{v}{L}\ tan(\phi) \\ |\phi| < \phi_{max} \\ |\dot{\phi}| < \dot{\phi}_{max} \end{cases}$$

Le modèle associé à ce système reste « sous-réaliste » mais il a cependant été utilisé[10] dans de nombreux travaux de recherche, concernant la planification de mouvement, résumés dans [LDE+01].

Dans sa formulation originale [DXCR93], la *kinodynamique* est étudiée dans le cadre du principe fondamental de la dynamique appliquée à une particule de masse m dans $\mathbb{R}^d$ avec $(d = \{2, 3\})$. Cette particule est dans un état initial $S = (s, \dot{s})$ et recherche un état final $G = (g, \dot{g})$. Les mouvements sont contrôlés par application de forces ou de commandes ; forces et commandes étant équivalentes pour une masse ponctuelle. Le système de contrôle, composé de d forces orientées par les axes de $\mathbb{R}^d$, est noté $U = (u_x, u_y)$ pour $d = 2$. À chaque instant, un état est

[10] L'intérêt pour ce modèle est justifié par l'énumération exhaustive de familles de chemins et par l'étude des chemins de longueur minimum. En 1957, L.E. Dubins caractérise les chemins d'une particule en marche avant. En 1990, J.A. Reeds et R.A. Shepp ajoutent la marche arrière. Les premiers planificateurs de mouvement sont établis sur la base de ces familles de chemins avec une garantie de complétude.

caractérisé par un vecteur de dimension $2d$: (x, y, v_x, v_y). Ainsi, les mouvements d'une particule de masse m sont régis par le système différentiel suivant :

$$\begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{v}_x = \frac{u_x}{m} \\ \dot{v}_y = \frac{u_y}{m} \end{cases}$$

Dans le cas du robot mobile non holonome comme dans celui de la particule, le système est dirigé par application de commandes. Aussi l'introduction d'un paramètre temporel permet de traiter naturellement l'influence des commandes sur la dynamique du système.

2.2.2 Commandabilité

La commandabilité d'un robot $\mathcal{M}$ définit l'ensemble des configurations accessibles de son voisinage. $\mathcal{M}$ est dit commandable si toutes les configurations de son voisinage sont accessibles. La définition de la commandabilité est une composante essentielle de la planification de trajectoire. Elle induit l'existence d'un chemin entre deux configurations en l'absence d'obstacle. En présence d'obstacles, il est nécessaire de définir la commandabilité en temps petit. $\mathcal{M}$ est dit commandable en temps petit si toutes les configurations de son voisinage sont accessibles avant un temps fixé.

Sous l'hypothèse de commandabilité en temps petit, l'existence d'un chemin pour un système contraint est démontrée par l'existence d'un chemin pour un système non contraint. La commandabilité en temps petit est conditionnée par la possibilité d'exécuter des manœuvres [Lau86].

En utilisant un modèle dynamique dans X, les contrôles associés deviennent insuffisants pour garantir l'accessibilité du voisinage d'un état en temps petit. À chaque instant, $\mathcal{M}$ possède une vitesse de déplacement v. Dans les cas où v est grand, les configurations correspondant à des points de rebroussement ne sont pas réalisables : dans ce cas, le point de rebroussement réel est situé en dehors du voisinage considéré ; l'existence d'un chemin dans C ne garantit pas l'existence d'un chemin dans X.

Dans X, l'utilisation des distances métriques peut être pénalisante par l'interprétation de leur évaluation des distances reliant deux configurations (ou deux états).

L'utilisation d'une distance métrique est cependant nécessaire[11]. À chaque expansion de G, la sélection de la commande à appliquer est réalisée par évaluation des configurations résultantes des commandes applicables. `nouvelleConf` détaille ce processus de sélection des commandes (ALG. 3).

```
nouvelleConf(q_prox, q_obj, G)
  | d_min ← +∞ ;
  | pour chaque u ∈ U
  |     q ← integration(q_prox, u, Δt) ;                    ①
  |     d ← ρ(q, q_obj) ;                                   ②
  |     si d < d_min
  |         | q_ret ← q ;
  |         | d_min ← d ;
  | retourner q_ret ;                                       ③
```

ALG. 3: Évaluation des commandes avec une distance métrique.

Pour définir la commande rapprochant q_{prox} de q_{obj}, chacune des commandes de U génère une nouvelle configuration q évaluée par sa valeur de distance métrique avec q_{obj} (ALG. 3 ① et ②). La configuration q_{ret} de distance métrique minimale est retenue (ALG. 3 ③).

La présence de distances identiques pour deux éléments distincts d'un élément de référence est appelée défaut (ou dysfonctionnement) de distance métrique. Ces défauts impliquent l'exclusion de certaines commandes, modifiant ainsi le domaine d'accessibilité des états dans X [LK00]. L'utilisation d'heuristiques associées aux distances métriques permet de modifier la valeur de distance associée à une configuration.

[11] Sans fonction de coût, une estimation de la distance reliant deux configurations semble illusoire.

2.2.3 Espace de recherche et obstacles

Dans un environnement statique, les obstacles sont une contrainte géométrique. La présence d'obstacles dans l'environnement divise C en deux[12] sous-ensembles : C_{libre}[13] l'ensemble des configurations dans lequel le mobile n'est ni en contact ni en collision avec un des obstacles, et C_{obs} l'ensemble des configurations, complémentaire de C_{libre} dans C. Dans son espace de travail W, le mobile $\mathcal{M}$ est représenté par une configuration q définissant un volume ; Dans l'espace des configurations C, $\mathcal{M}$ est représenté par q définissant une position. q est de dimension n, composé de variables identifiant les degrés de liberté de $\mathcal{M}$, avec $q = \{q_1, q_2, ..., q_n\}$. $\mathcal{O}$ est l'ensemble des obstacles de W. C_{libre} est défini par :

$$C_{libre} = \bigcup_{i=1}^{n} (q_i | q_i \cap \mathcal{O} = \emptyset)$$

Chaque obstacle de $\mathcal{O}$ étant défini par un ensemble de points, le passage de W à C est possible par la réalisation d'une somme de Minkowski [LP83]. Pour deux ensembles de points A et B du plan, et avec n le nombre de points de A, la somme de Minkowski entre A et B est définie par l'union de A et de n copies de B centrées sur chacun des éléments de A :

$$A \oplus B = \{x + y \mid x \in A, y \in B\}$$

Si A est un point p, $A \oplus B$ est équivalent à B localisé en p.

Pour un mobile circulaire soumis à des contraintes géométriques dans le plan, $\mathcal{M}$ est identifié à chaque instant par deux coordonnées (x, y). La forme de C est indépendante de la position de $\mathcal{M}$ (FIG. 2.5). La dimension de C est égale à la dimension de W. $\mathcal{M}$ est assimilé à un cercle c de centre s et de rayon r. Chacun des obstacles $\mathcal{o}_i$ de W devient un obstacle $\mathcal{o}_i^+$ de C avec :

$$c \oplus \mathcal{o}_i = \bigcup_{x \in c} (x \oplus \mathcal{o}_i)$$

[12] Considérant comme impossible le chevauchement d'un obstacle de $\mathcal{O}$ avec le mobile $\mathcal{M}$, les configurations de C sont classées selon trois catégories : les configurations interdites, dans lesquelles un ou plusieurs obstacles de $\mathcal{O}$ chevauchent $\mathcal{M}$; les configurations libres, dans lesquelles aucun des obstacles de $\mathcal{O}$ ne chevauche $\mathcal{M}$; les configurations de contact, dans lesquelles un ou plusieurs obstacles de $\mathcal{O}$ sont en contact avec $\mathcal{M}$. Ces trois ensembles sont respectivement appelés *C-espace* des obstacles, espace libre et surface de contact. Il est courant de regrouper les configurations en collision et en contact dans un même ensemble [WB00] appelé C_{obs}.

[13] Dans la littérature, C_{libre} est appelé C_{libre}, C_{free} ou CS_{free} (*Configuration Space*).

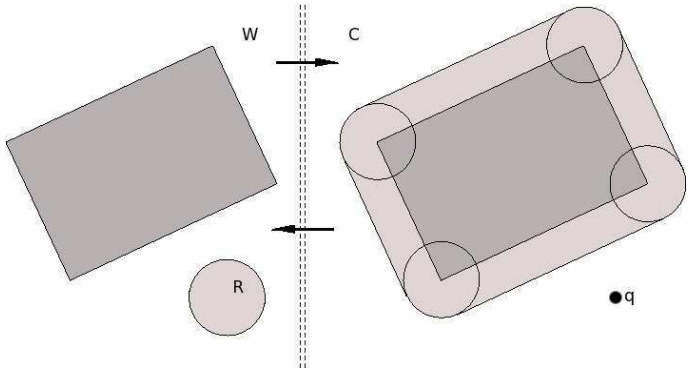


FIG. 2.5 – Espace des configurations pour un robot circulaire.

La forme de C est indépendante de la position de $\mathcal{M}$ dans W.

Les mouvements de $\mathcal{M}$ dans W se substituent aux mouvements de q de position s dans C. Les mouvements du robot dans l'espace de travail sont ainsi simplifiés par les mouvements d'un point s repérant $\mathcal{M}$ dans C (FIG. 2.5).

Pour le mobile de type voiture (§ 2.2.1), la forme de C varie en fonction de l'orientation de $\mathcal{M}$. Pour W de dimension 2 ($\mathcal{M}$ se déplace dans le plan), C est de dimension 3 (FIG. 2.6). En fixant un pas de variation à la valeur de $\Delta\theta = |\theta_{max} - \theta_{min}|/n$, C se divise en n tranches correspondant à l'image de C dans les n positions angulaires de $\mathcal{M}$ possibles (FIG. 2.6). Ainsi les mouvements de rotation correspondent à un changement de tranche et les translations correspondent à une translation du point P_R dans une même tranche. Dans ce cas, $\mathcal{M}$ est de forme polygonale P. P n'étant plus symétrique par rapport à P_R, il est nécessaire de définir $(-P)$ l'image de P par la symétrie de centre P_R. Le volume correspondant à C_{obs} est défini par :

$$P \oplus \mathcal{O}_i = \mathcal{O}_i \oplus (-P)$$

Pour deux polygones respectivement définis par p et q segments, avec $n = p + q$, la complexité du calcul de leur somme de Minkowski est de l'ordre de $O(n)$ pour deux polygones convexes, $O(n\ \log(n))$ si un seul des polygones est concave et $O(n^2\ \log(n))$ pour deux polygones concaves [O'R94].

A chaque mouvement, le point représentant le mobile $\mathcal{M}$ dans C est libre de se déplacer dans C_{libre}. Toute translation de C_{libre} dans C_{obs} est un mouvement invalide.

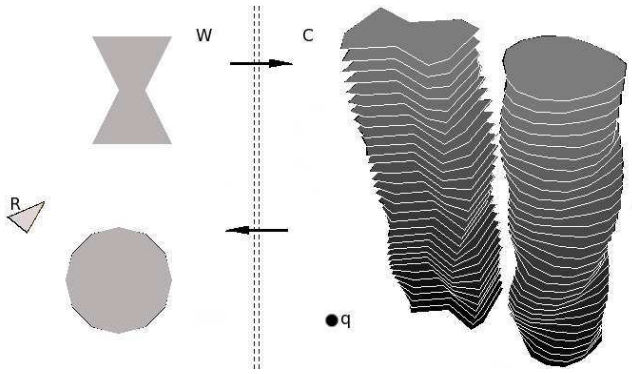


FIG. 2.6 – Espace des configurations pour un robot polygonal.

La forme de C dépend de l'orientation de $\mathcal{M}$ dans W. Pour un espace W plan, C est divisé en n tranches ; chaque tranche est l'image d'une position angulaire de $\mathcal{M}$ dans W.

La considération de C_{obs} dans C suggère, dans l'espace des états X, la considération de l'espace des états en collision X_{obs}. X_{obs} étant par définition un espace implicite, sa définition se limite à la projection des états dans l'espace des configurations :

$$q \in X_{obs} \leftrightarrow q_a \in C_{obs}, q = (q_a, \dot{q}_a)$$

Si les obstacles de $\mathcal{O}$ sont en position fixe, il est possible de pré-calculer C_{obs} pour un mobile. Ce pré-calcul impose cependant la définition d'un pas de variation angulaire.

En présence d'obstacles, l'ajout d'une configuration q_{new} dans le graphe défini par une méthode du type ***RRT*** et l'ajout d'un arc reliant les configurations q_{prox} et q_{new}, noté $\overrightarrow{V}(q_{new},\ q_{prox})$, sont réalisés sous les deux conditions suivantes :

$$\begin{cases} q_{new} \in C_{libre} \\ \overrightarrow{V}(q_{new},\ q_{prox}) \cap C_{obs} = \emptyset \end{cases}$$

La distribution aléatoire des tirages et la fonction de sélection du plus proche voisin guident l'expansion du graphe G dans C_{libre}. La distribution est optimale si elle rapproche de façon optimale, à chaque itération, le graphe G de la solution recherchée. Une telle distribution est, dans C, fonction du modèle du mobile considéré et des obstacles de l'espace de travail. De façon analogue à l'espace des configurations, l'espace des états X se divise en deux parties, X_{libre} et X_{obs}. Un état

$x = \{q_1, \ldots, q_m, \ldots, q_n\}$ est libre si les m composantes définissant la configuration q associée est une configuration de C_{libre}. De par la nature implicite de X_{obs}, l'uniformité de la distribution est considérée comme garantie de convergence du graphe de la méthode ***RRT*** vers une solution dans C comme dans X [LK00].

```
nouvelleConf(q_prox, q_obj, M, G, C)                                  ①
    d_min ← ρ(q_prox, q_rand) ;                                       ②
    succes ← FAUX ;
    pour chaque u ∈ U
        q ← integration(q_prox, u, Δt) ;
        si estSansCollision(q, M, C)                                  ③
            d ← distance(q, q_obj) ;                                  ④
            si d < d_min
                q_ret ← q ;
                d_min ← d ;
                succes ← VRAI ;
    si succes = VRAI                                                  ⑤
        retourner q_ret ;
    retourner ∅ ;                                                     ⑥
```

ALG. 4: Évaluation des commandes en présence d'obstacles.

En présence d'obstacles, la détection de collision est intégrée dès la génération des configurations accessibles à partir de q_{prox} [CL01]. L'évaluation des configurations q_{new} par la distance métrique est précédée de la détection de collision (ALG. 4 ③). Le mobile $\mathcal{M}$ et l'espace des configurations C sont insérés dans les paramètres de `nouvelleConf` (ALG. 4 ①). Pour les configurations sans collision, la distance à l'objectif est calculée 4 ④). La valeur minimale de ces distances est conservée dans d_{min}. Pour chaque configuration sans collision, q_{ret} reçoit la configuration générée par application de la commande u_i et la variable $succes$ reçoit $VRAI$. Après évaluation de toutes les configurations accessibles, si la variable $succes$ est à $FAUX$, alors aucune configuration n'est solution (ALG. 4⑤) ; dans le cas contraire, q_{ret} est la configuration solution (ALG. 4⑥).

Cette présentation de la sélection de l'élément accessible à partir de q_{prox} commence par l'initialisation de la variable d_{min} ; initialisée par un appel à la distance métrique(ALG. 4 ②). Nous préférons une initialisation par une valeur statique $+\infty$, identique à la formulation précédente (ALG. 3) ; une telle initialisation ré-

duit le nombre d'appels à la distance métrique du nombre de configurations étendues (*i.e.* sélectionnée en tant que plus proche configuration pour expansion en direction de q_{rand}) dans G.

2.2.4 Chemins de l'espace de recherche

Dans le cas d'un mobile de forme polygonale, dont les mouvements correspondent à un modèle de type voiture ou de type particule ponctuelle, le passage d'une tranche d'orientation a vers une tranche d'orientation b suggère le calcul du volume balayé entre a et b [LP81, LP83]. PA et PG sont respectivement les polygones approximatifs et généralisés correspondant au volume balayé par le déplacement du mobile.

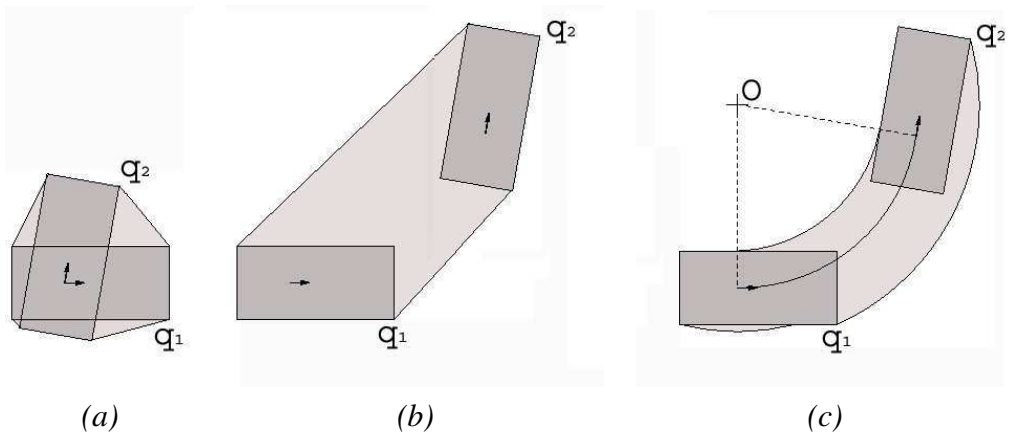
Le volume PA d'un chemin peut être approximé par une succession de segments de droite [LP81, LP83]. Dans le plan, l'espace des configurations C est $\mathbb{R}^2 \times S$ et contient des configurations représentées par un triplet (x, y, θ). Un chemin vertical est une rotation en position fixe, permettant au mobile de relier une configuration $q_2(x, y, \theta_2)$ à partir de $q_1(x, y, \theta_1)$ avec $|\theta_1 - \theta_2| = \Delta\theta$ (FIG. 2.7 *(a)*, q_1 et q_2 en gris foncé et le volume PA, ajouté à q_1 et q_2, en gris clair). La figure 2.7 *(b)* présente le volume PA sans contrainte de rotation en position fixe. L'espace des configurations est découpé en tranches d'orientation à un $\Delta\theta$ près[14]. Un chemin horizontal est une translation à angle fixe, permettant au mobile de relier une configuration (x_2, y_2, θ) à partir de (x_1, y_1, θ). Les chemins verticaux sont réalisables si aucune contrainte n'est fixée sur les variations de θ. Les chemins horizontaux sont réalisables dans la tranche de l'angle de départ θ_1. Un chemin de C est donc approximé par une succession de chemins horizontaux et verticaux. C est décomposé en tranches de variations d'orientation $\delta\theta$. Sans phase de pré-calcul, cette décomposition équivaut à une approximation par succession d'enveloppes convexes des configurations voisines de $\delta\theta$. Pour un chemin polygonal composé d'une suite de configurations $Q = \{q_0, q_1, ..., q_n\}$, PA correspond à la succession des enveloppes convexes des couples (q_i, q_{i+1}).

Le calcul du volume PG pour un mobile rectangulaire de type voiture est proposé dans [Ler98, SLL98]. Le mobile se déplace sur des chemins composés de segments et d'arcs de cercle[15]. Dans le cas de segments, ce calcul est trivial. Dans le

[14] La variation entre deux angles $\Delta\theta$ définit la discrétisation angulaire des chemins polygonaux. Cette valeur définit a priori l'approximation de la résolution. Elle peut conduire à l'absence de solution. Le choix d'un $\Delta\theta$ grand privilégie la rapidité de résolution aux dépens de la complétude.

[15] Ces chemins sont appelés chemins de *Reeds et Shepp* [RS90]. Le mobile se déplace en marche

cas d'arcs de cercle, PG est défini par des arcs de cercle et des points de liaison : les arcs de cercles correspondent aux rotations des sommets du rectangle et les points de liaison sont les intersections entre arc de cercle et segment ou entre segment et segment . La généralisation de ce calcul est réalisée dans un diagramme de description des contours résultants du déplacement sur un arc de cercle. Le centre de rotation est soit à l'intérieur du rectangle, soit à l'extérieur du rectangle. La position du centre de rotation donne lieu à deux diagrammes distincts. La figure 2.7 *(c)* présente la projection du volume PG ; elle correspond au passage de la configuration q_1 à la configuration q_2, pour un centre de rotation O.



(a) *(b)* *(c)*

FIG. 2.7 – Exemple de volumes balayés pour un mobile rectangulaire.

(a) est une rotation en position fixe. (b) est une rotation avec changement de position. (c) est un déplacement sur l'arc de cercle de centre O.

Pour un mobile soumis à un système différentiel, les mouvements ne correspondent pas obligatoirement à des arcs de cercle. Le volume balayé est donc approximé par PA. Cette approximation est réalisée à un intervalle de temps Δt près, correspondant à l'intervalle de temps d'intégration du système différentiel considéré.

avant ou en marche arrière à vitesse fixe. Ils définissent le chemin optimal entre deux configurations par une famille de 48 chemins paramétrés. De par la vitesse constante fixée, ces chemins sont optimaux en temps et en distance.

2.3 Expression nécessaire à la résolution

La méthode de construction du graphe G, définie dans une méthode du type *RRT*, est directement liée à la mise en équation de cinq expressions, nécessaires à la bonne formulation du problème : un détecteur de collision, un ensemble de commandes, un modèle différentiel du mobile, une distance métrique et une stratégie de planification locale.

2.3.1 Détecteur de collision

Le détecteur de collision a pour fonction de valider l'évolution du mobile au cours de sa progression dans l'environnement. La réalisation d'un détecteur de collision est associée à la résolution de deux problèmes connexes : la détection exacte et l'optimisation de cette détection. Les algorithmes de détection exacte se basent sur une décomposition en polygones convexes [Ler98, LSL$^+$98]. L'optimisation de cette détection est divisée en deux approches : la recherche de proximité et la détection approximative. La recherche de proximité est classiquement résolue par des méthodes de partitionnement de l'espace (*grilles régulières*, *segment-trees*, *quadtrees*, *BSP-trees*[16]) [dBSvKO00]. La détection approximative repose sur la définition des obstacles en volumes simplifiés [LGLM99], approximant ainsi les obstacles considérés (*AABB*, *OBB*, *kDOPs*[17]).

Le détecteur de collision différencie les configurations de l'espace libre C_{libre} des configurations C_{obs} ; Une configuration appartenant au domaine défini par C_{obs} est considérée comme non admissible. Dans le cas contraire, la configuration est dans C_{libre} et est considérée comme admissible. Au cours de la construction du graphe G, chaque nouvelle configuration q_{new} est définie par intégration d'une commande u à partir d'une configuration q_{prox}. Par validation d'une succession de configurations, le détecteur de collision assure la construction d'une trajectoire admissible.

[16] Un arbre *B*SP, abréviation de « Binary Space Partition », est une structure couramment utilisée pour définir les faces visibles d'un ensemble de polygones à partir d'un angle de vue.

[17] La détection approximative regroupe un ensemble d'algorithmes basés sur une recherche hiérarchique de volumes englobant (*BVHs* pour « Bouding volumes hierarchies ») ; En guise de référence, nous rappelons les noms de quelques fondamentaux regroupés dans cet ensemble : les *AABB* pour « Axis-Align Bounding Boxes », *OBB* pour « Oriented Object Boxes » et *kDOPs* pour « Discretely-oriented polytopes ».

2.3.2 Ensemble de commandes

Pour le mobile considéré, il est nécessaire d'énumérer l'ensemble des commandes applicables. Ces commandes sont définies dans le modèle du mobile. Elles s'expriment dans un vecteur U. Chacune de ces composantes, notées u_1 à u_n, est associée à un domaine de définition borné[18]. Le système ainsi défini, appelé *système de commande affine*, est de la forme :

$$\begin{cases} \dot{q} = X_0(q) + u_1 X_1(q) + \cdots + u_n X_n(q) \\ \forall u_i \in U, u_{i\ min} \leq u_i \leq u_{i\ max} \end{cases}$$

avec q une configuration de C, $X_0, \ldots, X_n$ les champs de vecteurs associés à q, $u_{i\ min}$ et $u_{i\ max}$ les limites physiques des actionneurs. Dans le cas d'un mobile holonome associant une commande à un déplacement de distance d, le vecteur u est de dimension 2 avec $-\pi \leq u_1 \leq \pi$ et $|u_2| \leq d$.

2.3.3 Modèle différentiel du mobile et résolution numérique

Le modèle différentiel définit l'évolution du mobile pour un intervalle de temps Δt choisi. Il s'exprime par une ou plusieurs équations différentielles du premier ordre. Chaque équation différentielle définit une relation entre une fonction recherchée et sa dérivée. Leur résolution est l'identification d'une telle fonction, satisfaisant la relation précédemment définie. Un modèle différentiel composé d'une équation différentielle ordinaire est de la forme :

$$\dot{q} = f(q, \Delta t)$$

avec f la fonction d'intégration numérique de q et Δt, q une configuration du système et $\dot{q}$ la dérivée de q dans l'intervalle de temps Δt. Pour f définie par :

$$f : x \longrightarrow -Kx$$

la fonction recherchée est $x = -e^{Kt}$. L'ordre de la résolution numérique définit la précision de la valeur résultante $\dot{q}$. Cette précision variant en fonction de la

[18] Les composantes de U sont soit continues soit discrètes. Ce choix dépend de la stratégie de commande choisie. Lors de l'exécution de la trajectoire, calculée par intégration d'une série de commandes théoriques, le critère le plus fiable pour $\mathcal{M}$ reste la trajectoire géométrique résultante. Il semble donc naturel de réaliser lors de l'exécution un asservissement des commandes de $\mathcal{M}$ sans tenir compte des commandes théoriques calculées.

valeur initiale q, l'utilisation d'une méthode à pas variable permet de définir le compromis entre ordre d'intégration et précision des calculs [WBK97].

2.3.4 Distance métrique

La distance métrique a pour fonction la transformation de l'espace des configurations en espace métrique. Dans le cadre des méthodes du type ***RRT***, la distance métrique est une fonction à valeurs réelles de la forme :

$$d : X \times X \longrightarrow [0; +\infty[$$

Elle spécifie la valeur de la distance $d(q_a; q_b)$ entre deux configurations q_a et q_b. Cette distance permet de sélectionner une configuration q_{prox} de G à partir de la configuration q_{rand}. Le rôle de la distance métrique est crucial pour l'expansion de G dans l'espace de recherche. Elle permet de mesurer la proximité entre une configuration et l'objectif, permettant ainsi une évaluation de la progression dans l'espace de recherche. Cette fonction est garantie sous l'hypothèse d'une relation entre proximité et atteignabilité : les configurations les plus proches doivent être également les plus faciles à atteindre. Dans un espace comportant plus de deux dimensions, la définition d'une distance métrique n'est pas triviale. S'il existe une trajectoire entre deux configurations, une distance métrique est considérée comme bonne quand la distance mesure cette trajectoire ; s'il existe plusieurs trajectoires possibles, la distance associée est la plus petite de ces trajectoires. Une distance métrique appropriée fournit donc la longueur de la trajectoire minimale reliant deux configurations[19]. Dans le cas d'un mobile soumis à des contraintes dynamiques, le calcul d'une telle distance métrique peut devenir coûteux en temps. Une solution triviale est de choisir la distance euclidienne en tant que distance métrique et de réduire les valeurs des configurations en coordonnées cartésiennes.

Le choix d'une distance métrique a des conséquences non négligeables sur la connexité des configurations de C. Il définit le prochain nœud-objectif du planificateur local. La planification pour les mobiles non-holonomes propose l'utilisation d'une métrique associant la proximité à la dextérité requise pour l'exécution des mouvements [LSL$^+$98]. La distance métrique doit être choisie en accord avec le comportement du planificateur local, pour limiter les échecs de ce dernier. Son

[19]La notion de distance métrique appropriée est fonction du contexte de la planification de mouvement. Dans une optique de mouvements minimisant les possibilités de collision, une distance métrique traduisant le nombre de chemins possibles nous semble préférable.

calcul, étant utilisé à hauteur du nombre d'éléments de U, doit être rapide. Ce calcul est fonction de la nature des paramètres de C : pour une configuration q identifiée par i paramètres linéaires et j paramètres angulaires, c_k représente le $k^{ème}$paramètre linéaire et α_k représente le $k^{ème}$paramètre angulaire (*respect.* q' identifiée par c' et α'). Les distances métriques les plus utilisées sont :

– La distance euclidienne [AW96, HLM97, KKL96] ;

$$d(q,q') = \left(\sum_{k=0}^{i} (c_k - c'_k)^2 + nf^2 \sum_{k=0}^{j} (\alpha_k - \alpha'_k)^2 \right)^{\frac{1}{2}}$$

avec nf le facteur de normalisation égal au maximum des variations entre les valeurs de chacun des c_k ;

– La distance euclidienne pondérée [KGW96, LK01] ;

$$d(q,q') = \left(s \sum_{k=0}^{i} (c_k - c'_k)^2 + nf^2(1-s) \sum_{k=0}^{j} (\alpha_k - \alpha'_k)^2 \right)^{\frac{1}{2}}$$

avec s une valeur fixée de l'intervalle $[0;1]$;

– La distance de Manhattan [AL94] ;

$$d(q,q') = \sum_{k=0}^{i} |c_k - c'_k| + nf \sum_{k=0}^{j} |\alpha_k - \alpha'_k|$$

– La longueur de la trace associée au volume balayé [Xav97], est calculée par approximation d'un maillage de triangles obtenus par discrétisation des paramètres c_k et α_k.

Une simplification possible du calcul de la distance entre deux configurations de C est de réaliser ce calcul dans W. Après comparaison de différentes métriques avec la méthode *PRM*, la métrique euclidienne pondérée donne les meilleurs résultats dans la plupart des cas [ABD+98b]. Plus l'espace est encombré, plus les valeurs de pondération sur les positions doivent être importantes.

2.3.5 Stratégie de planification locale

Les deux opérations les plus sollicitées par les méthodes probabilistes sont le calcul de proximité (défini par la distance métrique § 2.3.4) et la stratégie de planification locale. Cette stratégie constitue la partie décisionnelle de l'expansion

du graphe de la méthode *RRT* dans l'espace X. Elle définit le comportement du planificateur local. Ayant deux configurations q_{prox} et q_{rand}, elle définit la ou les commandes à appliquer depuis q_{prox}. Son rôle est de déterminer, pour chaque élément q, le vecteur de commande u à appliquer à chaque instant. Le planificateur local peut :

– Rapprocher simplement q_{prox} de q_{rand}. Le planificateur local réalise une unique expansion du graphe défini par la méthode *RRT* ;
– Connecter q_{prox} et q_{rand} par un chemin sans collision[20]. Le planificateur réalise autant d'expansions que nécessaire pour relier les deux configurations par une suite de configurations en respectant les contraintes liées au mobile et aux obstacles. Après un nombre fixé d'itérations sans succès, le planificateur local peut être considéré en échec. Cet échec indique soit l'absence de solution entre q_{prox} et q_{rand} soit la nécessité d'une recherche approfondie.

Dans les deux cas, l'objectif de chaque expansion est de rapprocher q_{prox} de q_{rand}. Partant de q_{prox}, l'application d'une commande u produit q_{new}. Étant donné ces deux configurations, le problème est de trouver la commande u de U minimisant la distance métrique entre q_{new} et q_{rand}. Une telle commande est considérée comme commande optimale et le chemin résultant appelé chemin optimal.

Dans le cas d'un mobile de type voiture se déplaçant à vitesse constante, tout chemin sans collision est localement défini par un ensemble de plus courts chemins [LSL$^+$98], approximation dont la valeur exacte des distances des chemins correspondants est également calculable [Ven96]. Ce domaine, défini par un feuilletage[21], permet de définir les plus courts chemins en présence d'obstacles [Jao97]. Dans le cas du mobile, de forme ponctuelle ou polygonale, évoluant à vitesse variable (*i.e. dans l'espace des états*), la notion de voisinage devient insuffisante pour assurer l'existence d'une séquence de contrôle permettant de relier deux états dans un temps donné. La sélection d'une commande optimale reliant deux configurations est garantie par évaluation de toutes les commandes applicables [LaV98]. Disposant d'une métrique, le développement de recherches heuristiques[22] est une

[20]Ce mode de planification locale est cependant moins utilisé dans le cadre des méthodes du type *RRT*. Il nécessite que les deux configurations soient dans l'espace des configures libres $Clibre$.

[21]Un feuilletage est un ensemble de feuilles de la variété différentielle M de dimension n sur laquelle un point évolue en vérifiant les contraintes dynamiques imposées. Les feuilles sont des ensembles disjoints deux à deux.

[22]Une heuristique est une règle, introduisant une information dans la prise de décision. Cette règle a l'obligation d'être bénéfique pour la majorité des éléments de l'espace de recherche. Le choix de la distance euclidienne comme distance métrique est un bon exemple. Elle n'est pas toujours une bonne évaluation de la direction à suivre pour relier une position mais elle reste utile dans un espace de grande dimension. La qualité d'une heuristique s'exprime dans la sélection d'une configuration parmi l'ensemble des configurations, sous le nom de facteur de branchement effectif. En planification de mouvement, la sélection d'un unique nœud par itération conduit à un

solution courante[23]. Une solution triviale est d'évaluer toutes les commandes possibles à partir de q_{prox}. D'autres solutions à la planification locale sont données par la programmation mathématique. L'espace du problème est défini par C_{libre}. Le planificateur local considère C_{libre} comme un espace continu ou discret.

Dans le cas continu, les planificateurs locaux les plus utilisés sont :

- **La ligne droite** [HLM97] ; Pour deux configurations q_1 et q_2, il suffit de vérifier l'exclusion de q_1 et q_2 du domaine C_{obs} et de l'absence d'intersection entre la ligne reliant ces deux configurations et C_{obs} [24] ;
- **La rotation en s** [ABD+98b] ; Ce planificateur local différençie l'évolution des variables de position et d'orientation entre la configuration initiale q_1 et la configuration finale q_2 ; Il définit deux configurations intermédiaires q' et q'' telles que :

$$q_1 = (x_1, y_1, z_1, \alpha_1, \beta_1, \gamma_1)$$
$$q_2 = (x_2, y_2, z_2, \alpha_2, \beta_2, \gamma_2)$$
$$q' = (s(x_2 - x_1), s(y_2 - y_1), s(z_2 - z_1), \alpha_1, \beta_1, \gamma_1)$$
$$q'' = (s(x_2 - x_1), s(y_2 - y_1), s(z_2 - z_1), \alpha_2, \beta_2, \gamma_2)$$

  Les chemins entre les configurations q_1, q', q'' et q_2 sont des lignes droites dans C ; $s = 0.5$ produit un volume balayé inférieur à celui de la ligne droite entre q_1 et q_2 et augmente ainsi les chances de réussite de cette méthode ;
- **Les chemins de Reeds et Shepp** [RS90] ; Ils définissent les plus courts chemins pour des déplacements en marche avant et en marche arrière[25] en l'absence d'obstacle[26] ; Ces chemins sont composés de segments de droite et d'arcs de cercle[27].

---

nombre d'échecs important pour un facteur de branchement effectif toujours égal à 1.

[23] Les recherches aveugles sont naturellement exclues, de par la définition d'une distance métrique.

[24] Dans un espace discret, la ligne droite est discrétisée en une succession de configurations avec des valeurs d'incrémentation dépendant des variables de configuration dans [KSLO94]. Dans ce cas, $q(t) = q_1 + t(q_2 - q_1)$. Dans un espace continu, elle est utilisée sans discrétisation [LK99].

[25] Les chemins de Dubins définissent les plus courts chemins pour des déplacements en marche avant uniquement en l'absence d'obstacle. Les chemins de *Reeds et Shepp* s'inscrivent dans la continuité des travaux de Dubins.

[26] Le problème des plus courts chemins en présence d'obstacles est ouvert. Les obstacles ont des formes convexes dont les côtés sont des segments de droite ou des arcs de rayon 1. Un algorithme de complexité $O(n^2 \log n)$ [BL96] sélectionne des trajectoires dont la courbure est inférieure à 1. Ces trajectoires sont contraintes dans leur forme et leur longueur.

[27] Le lecteur est invité à se reporter à l'ouvrage dirigé par J.P. Laumond [LDE+01] pour une synthèse de ces travaux.

Dans le cas discret, C_{libre} subit une discrétisation régulière des domaines de définition des variables de configurations. Le problème de la planification est posé par l'association de cet espace à une configuration q_1 initiale et une configuration q_2 finale. La notion de localité implique la présence des configurations dans leurs voisinages respectifs. Le temps de résolution diminue (*respect.* augmente) avec l'augmentation de la proximité (*respect.* l'éloignement) des configurations q_1 et q_2. Dans ce cas, les stratégies les plus utilisées dans les planificateurs locaux sont :

- **Le chemin de Manhattan** [AL94]. A l'ordre 1, il correspond à une succession de déplacements alternés de chacun des degrés de liberté. A l'ordre n, il est le résultat de la concaténation de n chemins de Manhattan d'ordre 1. Cette considération des déplacements successifs de chacun des degrés de liberté est équivalent à des déplacements successifs dans une dimension par déplacement ;
- **Le parcours en meilleur d'abord** (best-first) [Lat91][28]. Son principe est d'étendre le nœud ayant la meilleure évaluation en premier. Cette évaluation est uniquement fonction du nœud courant et de son successeur. A chaque itération, l'algorithme évalue tous les successeurs directs de la configuration en cours q. Le meilleur de ces successeurs est élu comme configuration en cours pour l'évaluation subséquente de ses successeurs[29]. Le choix de la distance métrique a une influence importante sur la résolution de cet algorithme ;
- **Le parcours A*** [HC95]. C'est un cas particulier de la recherche *en meilleur d'abord*[30]. Cette heuristique combine, pour l'évaluation des nœuds, une mesure $g(n)$ de la distance entre nœud courant et nœud évalué (le nœud évalué résultant de commandes appliquées au nœud courant) et une mesure $h(n)$ de la distance entre nœud évalué et nœud final. A chaque itération, tous les nœuds fils du nœud évalué sont insérés dans une liste de nœuds ouverts. Le meilleur des nœuds de cette liste est développé à chaque itération. Un nœud ne peut pas être développé deux fois. La première solution trouvée par la méthode A* est la meilleure[31]. N.M. Amato et al [ABD+98b] proposent **A*-clearance** (à chaque itération, le nœud le plus loin des obstacles est sélectionné.) et **A*-distance** (à chaque itération, le nœud le plus proche de q_{obj} est sélectionné). Si le parcours

[28] J.C. Latombe présente en premier la recherche *en profondeur d'abord*, ayant pour effet de suivre une descente de potentiel tant que la configuration finale n'est pas atteinte. Cette descente présente cependant l'inconvénient majeur de minima locaux pouvant entraver la progression du mobile vers l'objectif.

[29] Si d est le pas de discrétisation des axes de C et n est la dimension de C, la complexité en temps de la recherche en meilleur d'abord est de l'ordre de $O(m\, r^m \log\, r)$.

[30] L'algorithme de recherche A* combine la recherche en coût uniforme et la recherche gloutonne.

[31] Le défaut de A* est de stocker tous les nœuds (ici configurations) développables. En définissant une profondeur de parcours, la recherche s'exécute en temps fini. Le problème est de ne pas sous-estimer la profondeur nécessaire.

de C_{libre} rencontre un bord (une limite de C_{libre} avec un C-obstacle), A* a pour effet de sélectionner les nœuds le long des obstacles. C'est un effet indésirable de A*.

– **le parcours IDA*** (Iterative Deepening A*) [Hsu00]. C'est une recherche A* avec des coupes définies par une fonction de seuil. Ce seuil est redéfini à chaque itération par la valeur du plus petit coût des nœuds de l'itération précédente. Il permet de diminuer l'espace mémoire nécessaire (*i.e.* la liste des nœuds à développer) en conservant l'optimalité[32] de A* ;
– **Le parcours A*-multirésolution** [AL97b, AL97a]. Le pas de discrétisation ayant une conséquence sur la complétude, ce parcours de l'espace repose sur une décomposition en octree. Cette variation favorise le passage dans les plus grandes cellules de l'octree. Le pas de discrétisation de la recherche est défini par la taille de la cellule courante. Il diminue à proximité des obstacles et augmente dans les grands espaces libres. La racine de la recherche reste le nœud de départ et la distance entre deux nœuds (père et fils) varie en fonction de la position des nœuds dans C_{libre}.

2.4 Exemple

Nous proposons dans ce paragraphe la description d'un exemple de résolution d'un problème, par construction du graphe G de la méthode ***RRT*** (FIG. 2.8). L'environnement est un espace $2D$ composé d'obstacles représentés en gris. L'espace est représenté par le repère fixe $(R_0, \overrightarrow{i}, \overrightarrow{j})$. Le robot mobile $\mathcal{M}$ est modélisé par un cercle de centre s et de rayon r. Une configuration q de $\mathcal{M}$ est un vecteur de dimension trois (x, y, θ) de $\mathbb{R}^2 \times S$, où x et y sont les coordonnées de s dans R_0 et θ est l'angle entre l'axe de vecteur unitaire $\overrightarrow{i}$ et l'axe directeur de $\mathcal{M}$. Les déplacements de $\mathcal{M}$ sont réalisés sans glissement. Ils sont de deux types : mouvement de rotation d'une configuration d'angle θ_1 vers une configuration d'angle θ_2 ; mouvement de translation d'une configuration de position (x_1, y_1) vers une configuration de position (x_2, y_2).

Dans le cas d'une rotation :

$$q_2 = (x_1, y_1, \theta_1 + d\theta),\ tq\ -\pi < d\theta \leq \pi$$

[32]Une deuxième variante de A* appelée SMA* (Simplified Memory-Bounded A*) permet de définir l'occupation mémoire maximale de la recherche. Quand l'espace fixé est atteint, seuls les nœuds ayant le plus faible coût sont conservés. La complétude de cette variante est garantie si l'espace mémoire est suffisant pour contenir le chemin reliant configurations initiale et finale.

Pour tous les mouvements de translation, une intensité d'avancement d est fixée. Dans le cas d'une translation :

$$q_2 = (x_1 + dcos\theta_1, y_1 + dsin\theta_1, \theta_1)$$

C étant l'espace des configurations, C_{obs} est défini par la somme de Minkowski (§ 2.2.3) entre chacun des obstacles et du cercle de rayon s associé à $\mathcal{M}$. La forme circulaire de $\mathcal{M}$ produit une somme identique, avec tous les obstacles de $\mathcal{O}$, pour toutes les configurations de $\mathcal{M}$. La stratégie de planification locale choisie est la ligne droite dans C (§ 2.3.5). Le détecteur de collision détermine la validité des configurations par exclusion de la configuration de départ q_1, de la configuration d'arrivée q_2 et de l'absence d'intersection entre le segment $[q_1, q_2]$ et le domaine C_{obs}. Les commandes sont soit des translations, soit des rotations. Partant d'une configuration $q_1(x_1, y_1, \theta_1)$, la configuration $q_2(x_2, y_2, \theta_2)$ à une distance d de q_1 est définie par :

$$\theta_2 = atan2(\frac{y_2 - y_1}{x_2 - x_1})$$

Cette configuration q_2 est atteinte par application :
– D'une rotation d'angle $\theta_2 - \theta_1$;
– Poursuivie d'une translation d'intensité d.
Dans ce cas, le mobile, représenté par un point dans C, évolue dans un espace $2D$. Ces déplacements sont définis par :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} cos\ u_2 \\ sin\ u_2 \end{pmatrix} u_1$$

Les deux commandes u_1 et u_2 sont respectivement la vitesse linéaire et l'angle du chemin décrit par le mobile entre deux configurations consécutives. Nous distinguons ici l'application des deux commandes u_1 et u_2 par :
– À vitesse constante, la valeur de u_1 est fixe. La solution recherchée est à vitesse constante (*i.e.* à u_1 constante) ;
– u_2 est, conformément à la notion d'ensemble de commandes (FIG. 2.8) précédemment définie (§ 2.3), choisie dans l'ensemble des commandes applicables à partir de q_1, avec $\theta = \theta_2 - \theta_1$ et $k \in$ IN :

$$(\theta, \theta - \frac{\pi}{2k}, \theta + \frac{\pi}{2k}, \theta - \frac{2\pi}{2k}, \theta + \frac{2\pi}{2k}, \ldots, \theta - \frac{(k-1)\pi}{2k}, \theta + \frac{(k-1)\pi}{2k}, \theta + \frac{\pi}{2}, \theta - \frac{\pi}{2})$$

L'élément le plus proche de q_{rand} est défini par évaluation de la distance métrique euclidienne.

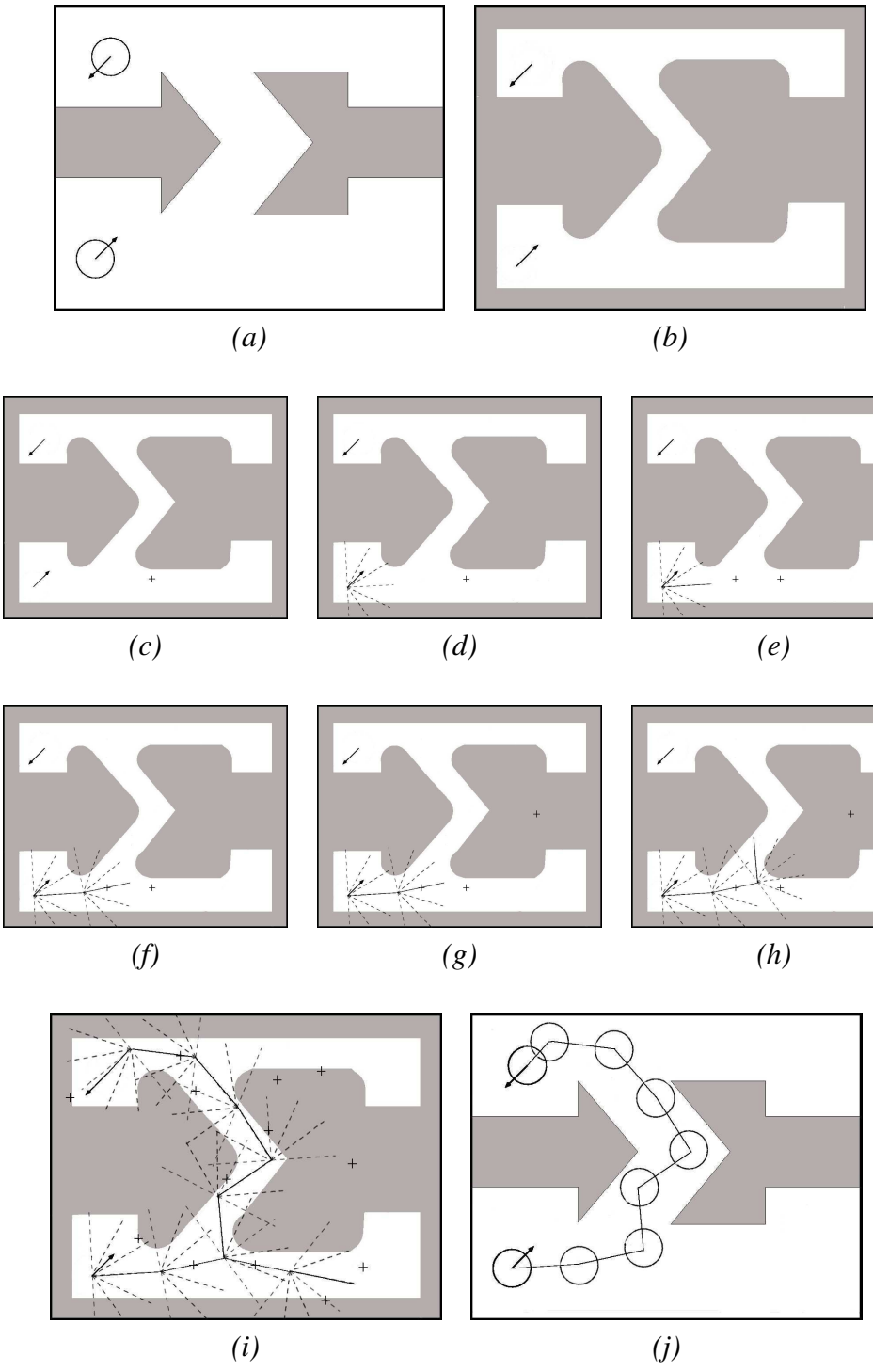


FIG. 2.8 – Résolution d'un problème de planification pour un robot holonome circulaire par évaluation des commandes applicables de U.

Le graphe G est construit par application de la commande générant l'élément le plus proche de q_{rand} ; chaque itération est construite par évaluation d'une commande parmi $2k + 1$ commandes (FIG. 2.8).

Pour assurer l'absence de cycle dans G, l'insertion d'un nœud q_{new} vérifie :

$$\forall q \in G \, , \, q_{new} \neq q$$

À chaque ajout d'une nouvelle configuration dans G, il est possible de vérifier une condition d'arrêt : s'il est possible de rejoindre q_{obj} à partir de q_{new}, alors l'arbre G est une solution du problème.

Sur la figure 2.8 *(a)*, la configuration initiale q_{init} est en bas à gauche et la configuration à atteindre q_{obj} est en haut à gauche. $\mathcal{M}$ est initialement orienté vers le haut tourné vers la droite. Les obstacles sont deux polygones concaves. La figure 2.8 *(b)* montre l'espace C_{obs} correspondant en gris. q_{init} est insérée dans G.

Un point est aléatoirement choisi dans C (FIG. 2.8 *(c)*). Ce point implique la sélection de q_{prox}, la configuration la plus proche de ce point dans G. Pour la première configuration q_{new}, q_{prox} est q_{init}. Par la suite, q_{prox} sera soit q_{init}, soit une configuration q précédemment insérée dans G. La liste des successeurs potentiels de q_{prox} est identifiée par un éventail en pointillés (FIG. 2.8 *(d)*).

Une nouvelle configuration q_{new} est sélectionnée à partir de q_{prox} en direction de ce point. Le segment $[q_{prox}, \ q_{new}]$ sélectionné est en dehors de C_{obs}. Il est inséré dans G et un nouveau point est aléatoirement choisi dans C (FIG. 2.8 *(e)*). Ce nouveau point conduit à la sélection d'un nouveau q_{prox}, l'évaluation d'un nouvel ensemble de successeurs et l'insertion d'un segment dans G (FIG. 2.8 *(f)*). L'évaluation exhaustive de tous les successeurs permet de maintenir l'insertion d'un nouveau segment dans G ; cette insertion est fonction du pas de discrétisation de U (FIG. 2.8 *(g)* et *(h)*).

Après n itérations (FIG. 2.8*(i)*), q_{new} est située à une distance inférieure à d de q_{obj}. Le segment $[q_{new}, q_{obj}]$ étant sans intersection avec C_{obs}, q_{obj} est insérée dans G. G est une solution du problème (FIG. 2.8 *(j)*).

2.5 Variations

2.5.1 Bidirectionnelle

La méthode *RRT* associée à une recherche bidirectionnelle (abrégée *bi-RRT*) est inspirée des techniques classiques de recherche de séquences d'éléments dans un espace dont l'élément initial et final (*i.e.* q_{init} et q_{obj}) sont connus dès le début de la recherche. Son principe repose sur la construction simultanée de deux arbres (appelés G_{init} et G_{obj}) dont le premier croît à partir de q_{init} et le second à partir de q_{obj}. Les deux arbres se développent tant qu'aucune connexion n'est établie entre-eux.

```
consBiRrt(q_init, q_obj, k, Δt, C)
  | init(G_init, q_init) ;
  | init(G_obj, q_obj) ;
  | pour i ← 1 à k
      q_rand ← confAleatoire(C) ;
      r_init ← expansG(q_rand, Δt, G_init) ;                    ①
      r_obj ← expansG(q_rand, Δt, G_obj) ;                      ②
      si r_init = CONNECTE
        | si r_obj = CONNECTE
            sol ← consChem(q_rand, G_init, G_obj) ;             ③
            retourner sol ;
  | retourner PAS_DE_SOLUTION ;
```

ALG. 5: Conjugaison de l'expansion de deux graphes de la méthode *RRT*.

À chaque itération, une nouvelle configuration q_{rand} est aléatoirement générée. L'algorithme tente d'étendre chacun des arbres en direction de cette configuration (ALG. 5 ① et ②). La valeur retournée par `expansG`$(q_{rand}, \Delta t, G)$ varie en fonction du déroulement de chacune de ces expansions. La valeur retournée est :

- `CONNECTE` si q_{rand} est directement connectable à G, c'est-à-dire sans expansion supplémentaire et sans collision ;
- `SUCCES` si q_{rand} n'est pas directement connectable à G mais une expansion à partir d'un nœud de G en direction de q_{rand} est réalisée avec succès ;
- `ECHEC` si q_{rand} n'est pas directement connectable et aucune expansion à partir d'un nœud de G en direction de q_{rand} n'est possible.

Si deux expansions consécutives, la première à partir de G_{init} (*respect.* G_{obj}) et la seconde à partir de G_{obj} (*respect.* G_{init}), retournent la valeur `CONNECTE`, la solution est le chemin reliant les deux arbres en la configuration q_{rand} (ALG. 5 ③).

```
expansG(q_cible, Δt, G)
  q_prox ← confLaPlusProche(q_cible, G) ;                          ①
  si ρ(q_prox, q_cible) < d_max                                    ②
      si connect(q_prox, q_cible)                                  ③
          ajouterConfArc(q_prox, q_cible, G) ;
          retourner CONNECTE ;
      q_new ← nouvelleConf(q_prox, q_cible, Δt) ;                  ④
      si connect(q_prox, q_new)                                    ⑤
          ajouterConfArc(q_prox, q_new, G) ;                       ⑥
          retourner SUCCES ;
  retourner ECHEC ;
```

ALG. 6: Expansion d'un graphe de la méthode *bi-RRT*.

La convergence des deux graphes G_{init} et G_{obj} vers une même configuration (identique en tout paramètres) diminue avec l'augmentation du nombre de composantes dans une configuration (*i.e.* avec l'augmentation de la dimension de C) et avec l'augmentation du nombre des contraintes sur ces composantes. Pour deux graphes G_{init} et G_{obj}, la convergence de leurs croissances est fonction de l'évolution des graphes dans l'espace libre. À chaque itération, q_{rand} est la nouvelle configuration de convergence proposée. Pour chaque configuration q de C, V_q définit l'ensemble des configurations situées dans le voisinage de q ; ces configurations sont atteignables par une suite de manœuvres à partir de la position q. Pour un mobile soumis à des contraintes dynamiques, la planification d'un chemin local permettant de relier q_{rand} nécessite la recherche d'une configuration dans le voisinage V_q de chacune des configurations de G. L'utilisation d'un planificateur local de type *ligne droite* (§2.3.5) ne permet pas de déterminer cette suite de manœuvres. La croissance de deux arbres distincts implique une primitive supplémentaire pour accélérer la détection de leur convergence. La génération d'une configuration q_{rand} implique deux expansions en direction de q_{rand} : une expansion partant de G_{init} et une expansion partant de G_{obj}. Pour chacune de ces expansions, q_{rand} est appelée q_{cible}. Une expansion en direction d'une configuration q_{cible} commence par la sélection du plus proche voisin de q_{cible} dans l'arbre considéré (ALG. 6 ①). Pour pallier à la problématique de la convergence des deux arbres,

une valeur de proximité d_{max} définit l'intervalle des contraintes à satisfaire[33]. La distance métrique ρ définit l'écart entre deux configurations. Si la valeur définie par $\rho(q_{prox}, q_{cible})$ est inférieure à d_{max}[34] (ALG. 6 ②), alors les deux configurations sont considérées comme connectables (pour un mobile en l'absence d'obstacle). `Connect`(q_{prox}, q_{cible}) vérifie l'absence de collision sur le chemin reliant q_{prox} et q_{cible} dans C. Cette vérification est uniquement géométrique (ALG. 6 ③ et ⑤). Si la distance entre q_{prox} et q_{cible} est supérieure à d_{max}, une nouvelle configuration q_{new} est générée par intégration des contraintes différentielles pendant l'intervalle de temps Δt (ALG. 6 ④). La fonction `ajouterConfArc` (ALG. 6 ⑥) ajoute une nouvelle configuration dans G et un arc vers sa configuration parent.

Une recherche bidirectionnelle est justifiée par l'hypothèse de la position du point de rencontre des deux arbres : position supposée à mis-parcours de l'espace des configurations séparant q_{init} et q_{obj}. La version élémentaire de la méthode ***RRT*** (ALG. 1) construit un arbre de dimension d ; la version bidirectionnelle de la méthode ***RRT*** construit deux arbres de dimension $d/2$, réduisant ainsi la complexité en temps de la résolution [RN03][35]. La construction d'un arbre partant de q_{obj} requiert une fonction inverse d'intégration des contraintes permettant de retrouver les configurations précédentes possibles d'une configuration. La formulation du problème reste identique. Partant d'une configuration q_{init}, l'objectif est d'énoncer une suite de commandes permettant au mobile $\mathcal{M}$ d'explorer l'espace des configurations C. La résolution est ici bornée à k itérations. L'intervalle de temps séparant deux configurations est défini par Δt.

2.5.2 Bidirectionnelle connectée

RRT-Connect [KL00] est une variation de la méthode *bi-RRT* ayant pour objectif d'augmenter la convergence des deux arbres, augmentant par la même occasion la convergence de l'algorithme bidirectionnel vers la solution. Cette variante a pour double objectif de :

[33] La violation de certaines contraintes introduit des discontinuités dans l'intégration de ces contraintes sur le parcours défini par la trajectoire. Il est possible de diminuer ces discontinuités par l'introduction de perturbations particulières (correspondant à des propriétés géométriques) dans l'historique des commandes [CFL04].

[34] ρ peut également être une fonction avec une pondération spécifique correspondant à la terminaison recherchée.

[35] La diminution de la complexité en temps est cependant garantie pour un parcours en largeur d'abord. La méthode ***RRT*** n'étant pas fondée sur un tel parcours, cette diminution de complexité n'est pas garantie.

– Assurer une résolution rapide pour les problèmes dit « simples » (Dans un espace sans obstacle, la progression du graphe doit être plus rapide que dans un espace composé d'obstacles) ;
– Maintenir la propriété probabiliste de convergence (L'emploi d'une heuristique modifiant la probabilité de convergence vers la solution, modifie l'évolution de la distribution des configurations dans C. La modification de la distribution des tirages aléatoires peut provoquer l'apparition de minima locaux, ralentissant la convergence de l'algorithme vers la solution.).

Les deux graphes précédemment appelés G_{init} et G_{obj} sont ici appelés G_a et G_b. G_a (*respect.* G_b) se substitue alternativement à G_{init} (*respect.* à G_{obj}) et à G_{obj} (*respect.* à G_{init}). La solution proposée par la variante ***RRT-Connect*** est de réaliser autant de phases d'expansion que possible (***i.e.*** sans collision) vers q_{new} dans G_b après une connexion avec succès vers q_{new} dans G_a. La configuration q_{cible} devient configuration de convergence q_{co}.

connectG$(q, \Delta t, G)$
  $r \leftarrow$ SUCCES ;
  tant que r = SUCCES
    $r \leftarrow$ expansG$(q, \Delta t, G)$;
  retourner r ;

ALG. 7: Connexion du graphe G courant dans la variante ***RRT-Connect***.

Une connexion d'un arbre à une configuration q réalise autant d'expansions que possible dans la direction de q (ALG. 7).

```
consRrtConnect(q_init, q_obj, k, Δt, C)
  init(q_init, G_a) ;
  init(q_obj, G_b) ;
    pour i ← 1 à k
      q_rand ← confAleatoire(C) ;
      r ← expansG(q_rand, Δt, G_a) ;                    ①
      si r ≠ ECHEC                                      ②
        si r = CONNECTE
          q_co ← q_rand ;                               ③
        sinon
          q_co ← q_new ;                                ④
        si connectG(q_co, Δt, G_b) = CONNECTE           ⑤
          sol ← consChem(q_co, G_a, G_b) ;
          retourner sol ;
      swapT(G_a, G_b) ;                                 ⑥
  retourner PAS_DE_SOLUTION ;
```

ALG. 8: Expansion attractive des deux graphes G_a et G_b dans la variante ***RRT-Connect***.

L'expansion attractive des deux graphes (ALG. 8) est garantie par l'alternance des fonctions `expansG` et `connectG`. À chaque itération, la fonction d'expansion `expansG` (ALG. 6) est utilisée pour ajouter une nouvelle configuration dans G_a en direction de q_{rand} (ALG. 8 ①). Si cette expansion retourne une valeur différente de `ECHEC` (*i.e.* égale à `SUCCES` ou `CONNECTE`, ALG. 8 ②), la configuration q_{co} devient la configuration de convergence de G_b vers G_a. q_{co} est soit q_{rand} soit q_{new} (ALG. 8 ③ et ④). Si l'expansion de G_b en direction de q_{co} (ALG. 8 ⑤) se solde par une connexion, la solution est la suite des éléments, reliant G_a et G_b, en q_{co}. Dans le cas contraire, G_a devient G_b et G_b devient G_a (ALG. 8 ⑥).

2.5.3 Probabilité de violation de contraintes

Cette variante, appelée *CVP* pour *Constraint Violation Probability*, a pour vocation de réduire la dépendance de la méthode *RRT* à la distance métrique [CSL01, CL01, Che01]. Son principe de fonctionnement est de collecter des informations au cours de la construction du graphe G.

Au cours de la croissance de G, la méthode ***RRT*** utilise la distance métrique pour approximer le coût des déplacements entre configurations. Ainsi la distance métrique doit refléter la dextérité requise (*i.e.* les manœuvres inhérentes au chemin reliant deux configurations) pour le suivi du chemin reliant les configurations ; l'inverse est appelé dysfonctionnement de la distance métrique. Pour un espace C construit à partir de valeurs de positions et de valeurs d'orientation, la définition d'une telle métrique n'est pas triviale. La croissance de G en direction des espaces inexplorés est garantie par l'adéquation de la distance métrique (elle-même associée à la répartition uniforme des tirages aléatoires, § 2.1.1). Le dysfonctionnement de la distance métrique a trois conséquences :

– La sélection d'éléments considérés les plus proches mais géométriquement non plus proches. La sélection d'une configuration n'étant pas le plus proche voisin de q_{rand} dans G peut provoquer des expansions répétées en direction de régions précédemment explorées ;
– La sélection répétée d'une même configuration pour une même expansion. Pour garantir l'absence de cycle, une configuration ne peut pas être étendue deux fois à l'aide de la même commande ; Ce type de dysfonctionnement implique une augmentation du nombre des expansions achevées par un échec ;
– L'éloignement des nouvelles configurations de la distribution uniforme. Cette conséquence est conditionnée par l'utilisation d'un planificateur local avec parcours itératif tel que A* (§ 2.3.5). Dans ce cas, la meilleure des expansions est définie à chaque itération par la distance métrique. Le parcours A* peut, d'évaluation en évaluation, éloigner le graphe de la solution recherchée.

La distance métrique permet de diriger les expansions des nœuds de G dans C. Pour diminuer l'influence de la distance métrique sur la croissance de G, cette variante propose d'ajouter deux critères d'évaluation des configurations :

– **La collecte d'informations d'exploration** ; l'ajout d'une nouvelle configuration q_{new} dans G résulte de l'application d'une commande[36] à partir d'une configuration q_{prox} de G. Pour chaque configuration de G, la liste des commandes précédemment appliquées est maintenue. Cette liste permet de retirer les commandes précédemment appliquées de la liste des commandes possibles (§ 2.3.2). Ainsi à chaque sélection d'une configuration q_{prox}, la stratégie de

[36] En fait, q_{new} peut résulter de l'application d'une ou plusieurs commandes selon :
– Le planificateur local choisi (§ 2.3.5) ;
– L'intervalle de temps Δt définissant la précision de l'intégration numérique des équations différentielles du mobile (§ 2.3.3) ;
– L'intervalle de temps (également appelé Δt) entre deux configurations dans G (§ 2.1.1).

commande sélectionne la meilleure commande parmi la liste des commandes disponibles. Cette exclusion des commandes précédemment appliquées induit la sélection de commandes localement non-optimales au regard de la distance métrique et la progression de G en direction des régions inexplorées. Chaque nœud de G est associé à un ensemble de commandes disponibles (*i.e.* non précédemment appliquées). Les configurations sans commande disponible ne sont pas sélectionnables lors de la recherche de plus proche voisin de q_{rand} dans G. Cette information permet d'éviter les appels répétés à des commandes menant à des collisions. La nouvelle recherche de plus proche voisin (ALG. 9) implique deux niveaux de sélection des éléments : $d_{min'}$ la distance entre q et q_{rand} et d_{min} la même distance dont l'affectation est conditionnée par la valeur de *CVP* (donc admissible sous condition de faible probabilité de collision). La variable r est le résultat d'un tirage aléatoire uniforme (ALG. 9 ②) et concrétise le choix associé à la probabilité de collision *CVP* . Si r est supérieur à *CVP*, la distance d est comparée à la distance minimum actuelle d_{min} (ALG. 9 ③). Pour chaque q, la distance d est comparée à la distance minimum actuelle par défaut $d_{min'}$ (ALG. 9 ①). Pour déterminer le plus proche voisin de q_{rand}, q_{best} (s'il existe) est préféré à $q_{best'}$. Si un plus proche voisin q_{best} (à une distance par d_{min}) existe (ALG. 9 ④), il est le plus proche voisin de q_{rand}. Sinon (ALG. 9 ⑤), $q_{best'}$ (défini sans utilisation de *CVP*) est le plus proche voisin de q_{rand} ;

– **La propagation d'informations de violation des contraintes** ; Pour relier deux configurations distantes de k intervalles de temps Δt, il existe une séquence de k commandes. À partir d'une configuration q, S est l'ensemble des séquences de k commandes. Pour une configuration q, la probabilité de violation des contraintes est la probabilité de collision sur le chemin correspondant à la séquence de longueur $k\Delta t$. En appliquant toutes les séquences de S à partir de q, cette probabilité, appelée *CVP*, est le nombre de séquences menant à une collision divisé par le nombre de séquences applicables. Pour une valeur de *CVP* égale à 1, une configuration mène inévitablement à une collision. Cette variante propose donc d'adapter la sélection du plus proche voisin avec la valeur de *CVP* pour chaque configuration. Les configurations ayant une valeur de *CVP* faible (*respect.* forte) ont une priorité plus importante (*respect.* plus faible). Pour une configuration modifiable par un ensemble de commandes discrétisées en n commandes distinctes, le calcul exact de *CVP* nécessite $\frac{(n^k-1)}{2} - 1$ intégrations différentielles. Pour éviter le calcul exact de *CVP*, chaque commande d'une configuration q menant à une collision implique une augmentation de $\frac{1}{n}$ pour la configuration q et d'une augmentation de $\frac{1}{n^i}$ pour le $i^{ème}$parent de q. Cette valeur est retropropagée sur k générations de nœuds parents.

confLaPlusProche($q_{rand}, \Delta t, G$)
  $d_{min} \leftarrow +\infty$;
  $d_{min'} \leftarrow +\infty$;
  pour chaque $q \in G$
    si $\exists u$ NON_SELECTIONNE
      $d \leftarrow \rho(q, q_{rand})$;
      si $d < d_{min'}$ ①
        $d_{min'} \leftarrow d$;
        $q_{best'} \leftarrow q$;
      $r \leftarrow$ valeur aléatoire dans $[0, 1]$; ②
      si ($r > CVP$ et $d < d_{min}$) ③
        $d_{min} \leftarrow d$;
        $q_{best} \leftarrow q$;
  si $d \neq +\infty$ ④
    retourner q_{best} ;
  retourner $q_{best'}$; ⑤

ALG. 9: Recherche du plus proche voisin dans la variante *CVP*.

2.5.4 Résolution complète

RC-RRT [Che01, CL02] est une variante de la méthode *RRT* ayant pour objectif de garantir la complétude de la résolution[37]. Pour un intervalle de temps Δt et pour un seuil de violation de contraintes d_{max}, le graphe d'accessibilité G' définit les connexions possibles entre plusieurs configurations de C. Une connexion de G' est définie :

– Par intégration des contraintes différentielles au cours de plusieurs intervalles de temps Δt(*i.e.* par une succession de connexions[38] dans G). Une étude théo-

[37]La complétude d'une méthode de planification probabiliste implique la complétude de la discrétisation et la complétude de la résolution. La résolution du problème nécessite une première approximation par discrétisation du temps et des commandes possibles. Ces deux approximations constituent la complétude de la discrétisation. La solution recherchée correspond alors à cette formulation approximée du problème. Le problème du choix d'une discrétisation adéquate peut se résoudre par une diminution du pas de discrétisation au cours du temps [BL93]. La complétude de la résolution est la garantie du succès de la résolution en un temps fini si une solution existe. Elle est conditionnée par une répartition uniforme des tirages aléatoires dans C.

[38]G' devient une simplification des connexions de G, à laquelle les cycles sont ajoutés.

rique [CL02] définit les relations entre G' et G et propose un ensemble de conditions nécessaires à la complétude de la résolution ;
– Par une relation à la distance métrique ; G' devient une approximation de G, à laquelle les relations de voisinage et les cycles sont ajoutés.

Si toutes les configurations accessibles à partir de q_{init} sont dans G' et si aucune solution n'existe dans G', alors le problème est sans solution pour les valeurs de Δt et d_{max} utilisées.

2.5.5 Bidirectionnelle parallèle

parallèle bi-RRT [CP02] est une variation parallèle de la méthode bi-*RRT* ayant pour objectif de diminuer le temps de résolution et de diminuer la longueur du chemin résultat. Dans une résolution parallèle « fictive » [39], la loi d'Amdahl définit le temps de résolution d'un problème comme réductible à la fraction séquentielle inhérente de son algorithme. Conformément à cette loi, la méthode *RRT* est parallélisable selon deux modes :
– Le mode *OR parallel* pour lequel plusieurs processeurs sont associés à la résolution du même problème. Chaque processeur contribue à une solution différente. Chaque processeur exécute le même algorithme et le premier processeur possédant une solution arrête la recherche dans les autres processeurs par diffusion. L'objectif est de réduire le temps de résolution ;
– Le mode *embarrassingly parallel*[40] pour lequel plusieurs processeurs sont associés à la même résolution du même problème[41]. Chaque processeur contribue à la même solution.

Dans sa forme originale, l'algorithme de la méthode *RRT* répète principalement deux opérations (ALG. 1 ⑤ et ⑥) réalisables en parallèle. Ces deux opérations

[39] L'attribut « fictive » souligne le principe de la décomposition des temps d'exécution et le mode de fonctionnement de l'ordinateur parallèle défini dans la loi d'Amdahl. Toute instruction prend un temps unitaire identique pour une donnée de taille identique, indépendamment de sa position dans l'ordinateur. Les temps induits par le partage des données sur l'ordinateur parallèle sont exclus des calculs de diminution du temps d'exécution.

[40] Les problèmes nécessitant des calculs indépendants sur différents processeurs associés à des phases de communication épisodiques avec une partie centrale sont appelés *embarrassingly parallel* [AD99]

[41] Le développement d'une telle approche n'est concevable que dans une résolution probabiliste. Dans le cas d'une résolution déterministe, chaque processeur exécuterait la même instruction pour produire en parallèle la même solution.

se résument en une phase de connexion associée à une détection de collision. Précédée à chaque itération de la génération d'une configuration aléatoire q_{rand}, chacune de ces phases de connexion est indépendante[42]. Cette indépendance des phases de connexion, constituantes centrales de la méthode *RRT*, implique une bonne extensibilité[43] en parallèle.

Le mode *OR parallel* bi-RRT (ALG. 10)

```
consOrParalleleBiRrt(q_init, q_obj, k, Δt, C)
  | init(q_init, G_init) ;
  | init(q_obj, G_obj) ;
  | pour i ← 1 à k
      si message de terminaison recu                          ①
        | retourner PAS_DE_SOLUTION ;
      q_rand ← confAleatoire(C) ;
      r_init ← expansG(q_rand, Δt, G_init) ;
      r_obj ← expansG(q_rand, Δt, G_obj) ;
      si r_init = CONNECTE
        | si r_obj = CONNECTE
            diffusion du message de terminaison ;             ②
            sol ← consChem(q_rand, G_init, G_obj) ;           ③
            retourner sol ;
  | retourner PAS_DE_SOLUTION ;
```

ALG. 10: Construction d'un *OR parallel bi-RRT*.

Chaque processeur exécute l'algorithme indépendamment, avec sa propre représentation des données. Soit $P_1(tp)$ la probabilité pour un processeur de trouver une solution avant le temps tp. Pour un ordinateur composé de n processeurs, la probabilité qu'aucun des n processeurs ne trouve une solution avant le temps t est $P^n_{aucun}(t) = (1 - P_1(t))^n$. Cette relation implique une diminution théorique du temps de résolution[44]. Dès qu'un processeur possède une solution, il envoie

[42]Dans le cadre de la méthode probabiliste *PRM*, une estimation des proportions du temps d'exécution [ABD+98a] propose 2 à 3% pour les générations de configurations aléatoires et 97 à 98% pour leurs connexions.

[43]L'extensibilité est la capacité de modulation de charge d'un ordinateur parallèle

[44]La diminution du temps de résolution est l'objectif le plus courant de la planification de mou-

un message de terminaison par diffusion(ALG 10 ②). L'exécution de chaque processeur est cadencée sur le test *message de terminaison recu* (ALG 10 ①). Pour assurer l'arrêt de tous les processeurs dès la découverte d'une solution, ce test est associé à une synchronisation sur barrière. Cette synchronisation garantit l'état d'avancement de tous les programmes dans tous les processeurs. Seul un programme retourne le chemin correspondant à la solution du problème (ALG 10 ③).

Le mode *embarrassingly parallel* bi-RRT (ALG. 11)

```
init(q_init, G_a) ;                                                    ①
init(q_obj, G_b) ;                                                     ②
fin ← FAUX ;                                                           ③

consEmbarrassinglyParalleleBiRrt(q_init, q_obj, k, Δt, C)
    tant que fin ≠ VRAI                                                ④
        q_rand ← confAleatoire(C) ;
        r ← paralleleExpansG(q_rand, Δt, G_a) ;                        ⑤
        si r ≠ ECHEC
            si r = CONNECTE
                q_co ← q_rand ;
            sinon
                q_co ← q_new ;
            si paralleleExpansG(G_b, q_co, Δt) = CONNECTE              ⑥
                fin ← VRAI ;
                sol ← consChem(G_a, G_b, q_co) ;
                retourner sol ;
        swapT(G_a, G_b) ;                                              ⑦
    retourner PAS_DE_SOLUTION ;
```

ALG. 11: Construction d'un *embarrassingly parallel bi-RRT*.

La variable *fin* et les données associées à G_a et G_b sont dans une mémoire partagée sur un modèle d'ordinateur parallèle *PRAM-CREW*[45]. Leur initialisation n'est

vements appliquée à la programmation parallèle. La majorité des approches utilise des représentations géométriques simples pour résoudre des problèmes de planification dans des environnements statiques [Hen96, Hen97]

[45]Parallel Random Access Memory - Concurrent Read Exclusive Write.

pas réalisée en parallèle (ALG. 11 ① , ② et ③). Comme le modèle d'ordinateur choisi le spécifie, les données associées à ces structures sont accessibles à tout instant, par tous les processeurs, en lecture et à chaque instant, par un seul processeur, en écriture. `consEmbarrassinglyParalleleBiRrt` est exécutée simultanément sur plusieurs processeurs. La boucle d'itération (ALG. 11 ④) est (contrairement aux algorithmes précédents) contrôlée par une variable *fin*. Cette variable est mise à jour par le processeur à l'origine de la solution[46]. La fonction `paralleleExpansG` (ALG. 11 ⑤ et ⑥) est une variation de `expansG` précédemment définie (ALG. 6). La seule modification de son fonctionnement est l'exécution exclusive de la fonction `ajouterConfArc` (ALG. 6 ⑥).

2.6 Contributions

Après une analyse des relations inhérentes aux composantes opérationnelles communes à toutes les variantes de la méthode du type *RRT*, nous proposons un algorithme diminuant le temps de construction de G. Le temps de construction de G est évalué dans une succession de simulations en environnement statique.

2.6.1 Relations entre les étapes de la construction de G

L'ensemble des variations de la méthode *RRT* présentées dans ce chapitre s'appuie sur la séquence d'opérations suivante :

1. Génération de q_{rand} ;
2. Sélection de q_{prox} dans G ;
3. Génération de l'ensemble S des éléments successeurs de q_{prox} par intégration de chacune des commandes de U à partir de q_{prox} ;
4. Détection de collision pour chacun des éléments de S ;
5. Sélection de l'élément q_{new}, plus proche voisin de q_{rand} parmi les éléments de S sans collision.

[46]Pour une présentation similaire aux algorithmes précédents, la variable k est partagée et une écriture exclusive est réalisée à chacune des itérations de chaque processeur. La variable `fin` ici proposée évite ce comptage fastidieux, correspondant à k écritures d'une variable partagée. Le nombre d'itérations de chaque processeur pouvant être différent, un dénombrement approximatif de ces itérations est possible par multiplication du nombre de processeurs par le nombre d'itérations d'un processeur fixé et de laisser à ce processeur la charge de la mise à jour de la variable `fin`.

Les relations inhérentes à une construction adéquate de G dans C sont traduites par :
- La déviation des tirages aléatoires[47] dans les variantes *bi-RRT* et *RRT-Connect* (§2.5.1 et 2.5.2) ;
- La sélection adaptée d'éléments q_{prox} à la probabilité de collision dans la variante *CVP* et l'intégration de la détection de collision dès la génération de q_{prox} (§2.5.3) ;
- L'adaptation de S à l'accessibilité du voisinage de q_{prox} dans la variante *RC-RRT* (§2.5.4) ;
- L'exécution en parallèle des opérations de construction de n graphes distincts dans la variante *OR parallel bi-RRT* (§2.5.5) ;
- L'exécution en parallèle des opérations de construction d'un graphe partagé dans la variante *embarrassingly parallel bi-RRT* (§2.5.5).

La construction de G correspond à la répétition de cette séquence d'opérations. La détection de collision discrimine les deux issues possibles d'une telle séquence :
- L'insertion de q_{new} dans G (*i.e.* en l'absence de collision) ;
- Le rejet de q_{new} (*i.e.* en présence d'une collision).

Le rejet de q_{new} induit une probabilité d'expansion relative à son voisinage ; plus une configuration est proche d'un obstacle, plus sa probabilité d'expansion dans une direction aléatoire est faible ; au dessus d'un seuil de distance aux obstacles, la probabilité d'expansion d'une configuration est égale à 1. La variante *OR parallèle bi-RRT* souligne le problème de la définition de la profondeur de la recherche d'une solution[48].

[47] Les variantes de RRT-Connect sont appelées *RRT-ExtCon*, *RRT-ConCon*, *RRT-ExtExt* ; Elles modifient la stratégie de construction d'un des deux arbres de la méthode *bi-RRT* par priorité alternatives des extensions et des connexions (*RRT-ExtCon*), par priorité aux connexions (*RRT-ConCon*) ou aux extensions (*RRT-ExtExt*) [LK00].

[48] Pour un problème de planification de mouvement p, doté d'une solution s accessible par une succession de n intégrations à partir de q_{init}, il s'agit de maximiser la probabilité de trouver une solution ; il est possible de réaliser a recherches à profondeur b variable. Selon le concept « d'action rationnelle », la réponse de classe $P3$ à l'adaptation de la recherche en-ligne peut être résolue par la définition d'une formule exprimant le coût d'une recherche en termes « d'effets locaux » et de « propagations » [Rus02]. Cette problématique devrait trouver une solution dans la mise en œuvre

Dans le cas d'un espace composé d'un unique passage étroit de largeur proche de la largeur de $\mathcal{M}$ dans W, l'utilisation des configurations le plus souvent en collision est nécessaire à la résolution d'un problème visant à relier deux configurations séparées par ce passage étroit. La faible probabilité de telles configurations est de ce fait une entrave à la stabilité de la complétude des méthodes du type *RRT*.

L'intégration de la détection de collision dès l'opération de sélection de l'élément q_{new} transforme l'opération de sélection du plus proche voisin de q_{rand} dans S en une sélection du plus proche voisin de q_{rand} dans $S \cap C_{libre}$. Bien que l'opération de détection de collision soit coûteuse en calculs, le calcul et l'évaluation de la distance métrique lui sont subordonnés.

```
consRrt(q_init, k, Δt, C)
  | init(q_init, G) ;
  | pour i ← 1 à k
      q_rand ← confAleatoire(C) ;                                  ①
      q_prox ← confLaPlusProche(q_rand, G) ;                       ②
      d_min ← ρ(q_prox, q_rand) ;                                  ③
      succes ← FAUX ;
      pour chaque u ∈ U                                            ④
        | q ← integration(q_prox, u, Δt) ;                         ⑤
        | si estSansCollision(q, q_prox, M, C)                     ⑥
            d ← ρ(q, q_rand) ;                                     ⑦
            si d < d_min
              | q_new ← q ;
              | d_min ← d ;
              | succes ← VRAI ;
      si succes = VRAI
        | ajouterConfArc(q_prox, q_new, G) ;                       ⑨
  | retourner G ;
```

ALG. 12: Construction de G en présence d'obstacles.

d'un programme décidant d'utiliser une ou plusieurs (*i.e.* a en nombre) variantes *CVP* à profondeur k (*i.e.* ici appelée b), avec k variant en fonction de données statistiques relatives à la progression de G.

En présence d'obstacles, les 5 opérations de la construction de G sont regroupées dans les étapes suivantes :

1. Génération aléatoire d'une configuration q_{rand} dans C (ALG. 12 ①) ;
2. Sélection du plus proche voisin de q_{rand} dans G (ALG. 12 ②) ;
3. Génération de l'ensemble S des éléments successeurs de q_{prox} par intégration de chacune des commandes de U à partir de q_{prox} (ALG. 12 ④ et ⑤) ;
4. Détection de collision pour la trajectoire locale reliant q_{prox} et son $i^{ème}$successeur (ALG. 12 ⑥) ;
5. En l'absence de collision, le $i^{ème}$successeur de q_{prox} est évalué par la distance métrique (ALG. 12 ⑦) ; la valeur de distance minimale est conservée et la configuration associée est q_{new}.

L'initialisation de la distance minimale par une valeur statique à la place d'un appel à la distance métrique (ALG. 12 ③) permet une diminution du nombre de calculs de distance métrique (§2.2.3).

À chaque itération de la $2^{ème}$ opération (*i.e.* sélection du plus proche voisin de q_{rand}), les éléments de G ont une probabilité P_s d'être sélectionnés. Dès qu'un nœud est étendu (*i.e.* possède un successeur par intégration numérique), sa probabilité P_s diminue. Cette diminution implique une attention particulière à l'expansion des nœuds ; elle renforce l'importance des expansions sans future collision. Cette diminution pose également la question de la légitimité de la rétropropagation d'informations relatives à l'exploration de G dans C. La collecte d'informations propres à la configuration en cours est intéressante (§2.5.3) ; la probabilité d'une configuration q, dont les $2k+1$ successeurs (§2.4) sont en collision, doit être modifiée à 0. De ce fait, chaque nœud de G stocke les commandes associées à ses successeurs existants.

2.6.2 Sélection du premier élément de C_{libre}

Nous proposons de réduire la recherche au premier élément de C_{libre} par inversion de la relation entre détection de collision et distance métrique ; subordonné à la distance métrique, le premier appel au détecteur de collision validé est la solution. Cette inversion induit :

– Une réduction du nombre d'appels au détecteur de collision proportionnelle à la nature et à la dimension de U ; elle a pour vocation de rapprocher le nombre d'appels au détecteur de collision et le nombre de configurations q_{new} ajoutées avec succès dans G.
– Une équiprobabilité d'expansion des nœuds indépendamment de leurs relations aux obstacles ;

Cette inversion se concrétise dans la substitution des opérations 4 et 5 de la construction de G par 2 nouvelles opérations ; l'objectif est d'adapter la phase d'expansion aux éventuelles collisions. La construction de G devient :

1. Génération aléatoire d'une configuration q_{rand} dans C (ALG. 12 ①) ;
2. Sélection de q_{prox}, le plus proche voisin de q_{rand} dans G (ALG. 12 ②) ;
3. Génération de l'ensemble des successeurs de q_{prox} par intégration de chacune des commandes de U à partir de q_{prox} (ALG. 12 ④ et ⑤) ; chaque successeur est associé à la distance le séparant de q_{rand} pour former un couple appelé s ; S est l'ensemble de ses couples ;
4. Tri des éléments de S par ordre croissant de distance ;
5. Sélection du premier élément de S dans C_{libre}.

À chaque itération, S est initialisé à $\emptyset$ (ALG. 13 ①). Chacune des commandes u de U est testée par intégration à partir de q_{prox} (ALG. 13 ②). L'intégration d'une commande u conduit à la création d'un nouvel élément q associé à une valeur de distance le séparant de q_{prox}. Les couples $(q, \rho(q, q_{rand}))$ sont ajoutés à S (ALG. 13 ③). Après énumération de tous les successeurs de q_{prox} (*i.e.* par application de toutes les commandes de U), les couples de S sont triés (ALG. 13 ④) par ordre croissant des valeurs de distance (*i.e.* $\rho(q, q_{rand})$). Les éléments triés de S sont évalués dans l'ordre ainsi défini ; `iemeCouple` retourne le $n^{ème}$couple de S (ALG. 13 ⑤) ; `premierElementDuCouple` retourne la configuration associée au couple s (ALG. 13 ⑥) ; cette configuration est la configuration q_{new} candidate à une insertion dans G ; les éventuelles collisions sur la trajectoire reliant q_{new} à q_{prox} sont testées par un appel au détecteur de collision (ALG. 13 ⑦) ; le premier élément q définissant un chemin reliant q_{prox} sans collision met fin au parcours des éléments de S (ALG. 13 ⑧) ; l'évaluation des configurations de S par le détecteur de collision est restreinte au premier élément définissant une trajectoire dans C_{libre}.

```
consRrt(q_init, k, Δt, C)
  | init(q_init, G) ;
  | pour i ← 1 à k
      q_rand ← confAleatoire(C) ;
      q_prox ← confLaPlusProche(q_rand, G) ;
      S ← ∅ ;                                                    ①
      pour chaque u ∈ U                                          ②
        | q ← integration(q_prox, u, Δt) ;
        | d ← ρ(q, q_rand) ;
        | S ← S + {(q, d)} ;                                     ③
      trier(S, d) ;                                              ④
      n ← 0 ;
      succes ← FAUX ;
      tant que succes = FAUX et n < Card(S)
        | s ← iemeCouple(n, S) ;                                 ⑤
        | q_new ← premierElementDuCouple(s) ;                    ⑥
        | si estSansCollision(q_new, q_prox, M, C)               ⑦
            ajouterConfArc(q_prox, q_new, G) ;                   ⑧
            succes ← VRAI ;
        | n ← n + 1 ;
  | retourner G ;
```

ALG. 13: Construction de G minimisant les appels au détecteur de collision.

Pour éviter la sélection répétée d'un même nœud de G dont les successeurs directs sont en collision, il est nécessaire de désactiver ce nœud. L'ajout d'informations supplémentaires au niveau des nœuds de G permet la rétropropagation de ce processus. L'algorithme 14 présente, en ce sens, une amélioration de l'algorithme 13.

Pour collecter les informations relatives à la progression de G, chaque configuration est associée à un ensemble de commandes disponibles : à chaque itération, une configuration q_{prox} de G est sélectionnée pour être étendue en direction de q_{rand} ; Uq_{prox} est l'ensemble des commandes applicables de U non précédemment appliquées à q_{prox} ; chaque commande d'un nœud q_{prox} testée (*i.e.* menant à une collision ou à une nouvelle configuration q_{new}) est retirée de Uq_{prox} ; une configuration, telle que $Uq_{prox} = \emptyset$, ne peut pas être retenue par la $2^{ème}$ opération de sélection du plus proche voisin de q_{rand} dans G ; les nœuds associés à ces configurations sont désactivés(ALG. 14 ④). Dans ce cas, S devient un ensemble

T de triplets *successeur-distance-commande*.

```
consRrt(q_init, k, Δt, C)
  | init(q_init, G) ;
  | pour i ← 1 à k
      q_rand ← confAleatoire(C) ;
      q_prox ← confLaPlusProcheValide(q_rand, G) ;
      n ← 0 ;
      si Eq_prox = CREATION
        | Eq_prox ← EXPANSION ;
        | tant que n < Card(Uq_prox)
            u ← commande(Tq_prox[ n ]) ;
            q ← integration(q_prox, u, Δt) ;                    ①
            d ← ρ(q, q_rand) ;
            Tq_prox[ n ] ← (q, d, u) ;
            n ← n + 1 ;
      tant que n < Card(Uq_prox)
        | q ← successeur(Tq_prox[ n ]) ;
        | d ← ρ(q, q_rand) ;                                    ②
        | Tq_prox[ n ] ← (q, d, u) ;
        | n ← n + 1 ;
      trier(Tq_prox, d) ;
      n ← 0 ;
      succes ← FAUX ;
      tant que succes = FAUX et n < Card(Tq_prox)
        | q_new ← successeur(Tq_prox[ n ]) ;
        | Tq_prox ← Tq_prox − {Tq_prox[ n ]} ;                   ③
        | si estSansCollision(q_new, q_prox, M, C)
            ajouterConfArc(q_prox, q_new, G) ;
            succes ← VRAI ;
        | n ← n + 1 ;
      si Tq_prox = ∅
        | Eq_prox ← DESACTIVE ;                                 ④
  | retourner G ;
```

ALG. 14: Construction de G avec collecte d'informations.

`successeur`$(Tq_{prox}[\ n\ ])$ retourne le $n^{ème}$ successeur de l'ensemble Tq_{prox} (*respect.* `commande` pour la $n^{ème}$ commande) ; pour éviter le calcul répété des

successeurs à chaque sélection de q_{prox}, chaque configuration q est associée à Eq et à Tq ; Eq définit l'état de la configuration q : $CREATION$ pour toutes les configurations insérées dans G, $EXPANSION$ après une sélection comme configuration q_{prox}, $DESACTIVE$ après le test de tous ses successeurs ; Tq est l'ensemble des triplets *successeur-distance-commande* de la configuration q ; à la création de chaque configuration q, Tq contient la liste des commandes u, sans successeur ni distance, (dans ce cas, $Eq = CREATION$) ; tous les successeurs de q sont calculés et insérés dans Tq à la première sélection de q (ALG. 14 ①) ; les distances séparant les successeurs de q et q_{rand} sont calculées à chaque sélection d'une nouvelle configuration q_{rand} (ALG. 14 ②) ; un ensemble Tq est maintenu par configuration de q dans G ; les trois sous-ensembles de Tq notés Sq, Dq et Uq sont respectivement les ensembles des successeurs non testés, des distances de q_{rand} et des commandes applicables de q ; après sélection de tous les successeurs de q, Eq prend la valeur $DESACTIVE$ et `confLaPlusProcheValide` ne peut plus sélectionner cette configuration q dans G. Chaque test d'une configuration q successeur de q_{prox} implique le retrait d'un triplet de Uq_{prox} (ALG. 14 ③).

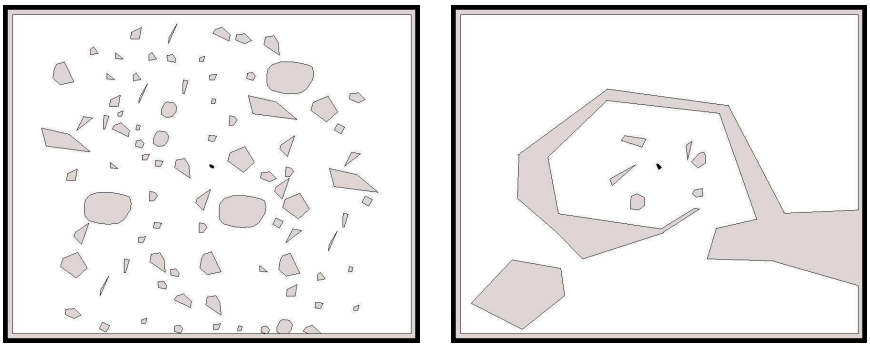
2.6.3 Résultats

Les figures 2.9 *(a)* et *(b)* présentent les deux environnements choisis pour comparer notre algorithme (appelé ***RRT_a***) avec l'algorithme classique (appelé ***RRT***, ALG. 12). Ces deux environnements sont composés d'obstacles quelconques. Pour évaluer le coût de la détection de collision, nous utilisons un troisième environnement sans obstacles, appelé *vide*.

Les déplacements de $\mathcal{M}$ sont définis par un système de dimension 5 proposé par P. Cheng [CL01] :

$$\begin{cases} \dot{x} = s\cos(\theta) - v\sin(\theta) \\ \dot{y} = s\sin(\theta) + v\cos(\theta) \\ \dot{\theta} = r \\ \dot{r} = (Fa - Rb)/I \\ \dot{v} = -sr + (F + R)/m \end{cases}$$

avec m la masse, F le torseur cinématique avant, R le torseur cinématique arrière, a la distance entre le centre de masse et l'avant de $\mathcal{M}$, b la distance entre le centre de masse et l'arrière de $\mathcal{M}$, I le moment d'inertie, s la vitesse (constante), θ l'angle de $\mathcal{M}$ dans le repère global. La commande u définit l'angle des roues de $\mathcal{M}$. La résolution numérique choisie est la méthode de *Runge-Kutta* d'ordre 4. La détection de collision utilise un partitionnement de l'espace en *quadtree*, associé à une



(a) *(b)*

FIG. 2.9 – Deux environnements utilisés pour tester notre algorithme.

Dans un champ d'obstacles (a) ou dans un labyrinthe (b), $\mathcal{M}$ est placé au centre de la scène, avec un angle d'orientation θ aléatoire. Ces deux environnements sont clos. Les obstacles sont en gris.

détection de collision approximative en volume *AABB*. La détection de collision exacte correspond à une somme de Minkowsky. À chaque déplacement, la liste des obstacles susceptibles d'être en collision avec le mobile de configuration q, est utilisée pour calculer les C-obstacles desquels q doit être exclue. L'espace des configurations n'est pas maintenu en mémoire. Le mobile $\mathcal{M}$ est placé au centre de la scène, avec un angle d'orientation θ aléatoire. L'échantillonnage de C est pseudo-aléatoire uniforme dans C.

Les tests sont réalisés avec un nombre d'itérations variant entre 2000 et 20000 ; les moyennes sont établies sur 10 tests. Pour évaluer les temps d'expansion des arbres correspondant à chaque algorithme, nous avons choisi de placer l'objectif en dehors des environnements. Les graphes G construits sont de ce fait composés du maximum de nœuds admissibles dans C_{libre}. Pour évaluer les conséquences de la taille de U, nous utilisons un ensemble U_1 composé de 3 commandes ($u = -0.08 + 0.08k$ avec $k \in [0 : 2]$) et un ensemble U_2 composé de 9 commandes ($u = -0.08 + 0.02k$ avec $k \in [0 : 8]$). Des exemples de graphes résultant de 2000 et 12000 itérations sont présentés en annexe A.

Les résultats du tableau 2.1 présentent une validation expérimentale de la diminution du temps[49] de construction de G ; diminution réalisée par notre algorithme.

[49]Ces temps de calcul sont réalisés avec un processeur Intel Pentium4 3Ghz avec 1GO DDR

TAB. 2.1 – Résultats comparatifs des algorithmes de construction de G.

		RRT $(2k)$	*RRTa* $(2k)$	*RRT* $(20k)$	*RRTa* $(20k)$
2.9 *(a)*					
	U_1	0.71	0.35	16.50	12.99
	U_2	1.98	0.74	30.63	17.29
2.9 *(b)*					
	U_1	0.40	0.25	11.31	7.83
	U_2	1.14	0.43	22.45	12.28
vide					
	U_1	0.33	0.24	19.22	16.74
	U_2	0.70	0.33	23.67	19.90

Les temps sont donnés en secondes. Les colonnes anotées $2K$ (*respect.* $20k$) donnent les temps moyens de construction de G pour 2000 itérations (*respect.* 20000 itérations). Pour un système de contrôle composé de 3 commandes, l'algorithme *RRTa* est en moyenne 1.47 fois plus rapide que l'algorithme *RRT* classique ; pour un système de contrôle composé de 9 commandes, l'algorithme *RRTa* est en moyenne 1.84 fois plus rapide que l'algorithme *RRT* classique. Le maintien de ce gain indifféremment des environnements choisis et de la dimension de U utilisée montre la robustesse de notre algorithme.

2.7 Conclusion

L'algorithme de construction de G diminue le temps nécessaire à l'exécution d'un nombre fixé d'expansions ; l'augmentation de la dimension de U implique une augmentation des intégrations numériques de ces commandes sur chaque nœud sélectionné lors de la recherche de plus proche voisin ; le maintien en mémoire des résultats de ces intégrations diminue également le nombre de ces appels.

Les travaux présentés sont applicables dans toutes les variantes de la méthode *RRT* présentées dans ce chapitre. Notre algorithme rappelle un principe pratique de la géométrie algorithmique selon lequel il faut minimiser le nombre d'appels au détecteur de collision ; ce principe est également appliqué à la résolution numérique

SDRAM sous Linux (5976 bogomips).

du système différentiel définissant les successeurs d'un nœud de G. La diminution du temps de résolution devient proportionnelle :

- au pas de discrétisation de l'ensemble des commandes applicables au mobile ;
- à la position des nœuds de G dans C. G regroupe des nœuds étendus (*i.e.* avec 1 ou plusieurs successeurs) et des nœuds à étendre (*i.e.* sans successeur) ; Parmi l'ensemble des nœuds de G, la probabilité d'être sélectionné (par la distance métrique) est plus forte pour les nœuds sans successeur. Si un nœud ne possède que des successeurs valides, la méthode d'extension la plus rapide choisit un de ces nœuds et le valide par un appel au détecteur de collision. En utilisant notre algorithme, plus un nœud possède de successeurs et plus la probabilité d'accélération de la phase d'expansion de ce nœud augmente.

Chapitre 3

Échantillonnage sous contraintes

La recherche de positions libres et adéquates constitue une des étapes de la construction de G dans les méthodes ***RRT*** et ***PRM***. Pour V. Boor *et al.* [BOS01], les deux étapes consommatrices en temps de calcul de la construction de G dans les méthodes ***PRM***, sont des problèmes de nature essentiellement géométrique : la génération d'un élément dans l'espace libre et l'appel au planificateur local. Le générateur de cette première étape doit produire un élément dans C_{libre} adéquat à la progression de G. Le planificateur local associé à un détecteur de collision recherche une connexion. Le temps dépensé pour la construction de G est alors défini par :

$$T = n_t T_t + n_a T_a$$

avec n_t le nombre d'éléments générés et testés, n_a le nombre d'éléments ajoutés avec succès dans G, T_t le temps requis pour le test d'un élément et T_a le temps requis pour l'ajout d'un élément dans G. Dans les méthodes ***PRM***, T_t est négligeable devant T_a ; les auteurs suggèrent de diminuer n_a et d'augmenter n_t. La réduction de n_a est réalisée par la sélection de nœuds visant à améliorer l'adéquation entre la connexité de G et la connexité de W. Néanmoins cette sélection s'accompagne dans la pratique d'une diminution de n_t. Dans les méthodes ***RRT***, c'est le contraire : T_a est négligeable devant T_t. Il faut donc diminuer n_t et augmenter n_a. La sélection de nœuds visant à améliorer l'adéquation entre G et W reste un objectif valide[1] pour les méthodes ***RRT***.

[1]La conservation de cette optimisation est inhérente aux natures de l'espace de recherche C et de n_t. L'exploration des éléments de C peut se faire : (a) par énumération ordonnée ; (b) par énumération aléatoire ; (c) par énumération déterministe ; (d) par énumération heuristique. Si (d) est dirigée par une heuristique triviale, elle se résume en (a), (b) ou (c). Si (d) est dirigée par une heuristique non triviale, son temps d'exécution est supérieur à (a), (b) et (c) (§ 1.1.3). L'objectif de

Dans le cadre des méthodes probabilistes, le parcours de C repose sur l'échantillonnage de l'espace de recherche. D'un point de vue général, les variations d'échantillonnage de C tentent d'assurer (indépendamment de la forme et des contraintes de déplacement du mobile utilisé) :
– La négociation des passages étroits ;
– La génération des configurations nécessaires et suffisantes à la résolution du problème.
Ces deux objectifs induisent l'extraction des propriétés géométriques de l'espace de recherche.

3.1 Méthodes existantes

3.1.1 Aléatoire uniforme

Un échantillonnage aléatoire uniforme de l'espace de recherche génère des positions par choix aléatoires. Une position étant définie par n paramètres indépendants, générer aléatoirement une position équivaut à réaliser n tirages aléatoires indépendants dans les intervalles de variation de ces n paramètres.

Dans le cas de la méthode *PRM*, les positions engendrées aléatoirement sont des éléments potentiels du graphe G. Pendant la phase *d'apprentissage*, une position appartenant à l'espace libre est retenue. Une position n'appartenant pas à l'espace libre est rejetée et une nouvelle position aléatoire doit être générée. Les positions ainsi générées sont uniformément réparties dans l'espace libre. Le temps de génération d'un échantillon fini de positions aléatoires valides est cependant fonction de la proportion entre espace libre et espace occupé (par l'image des obstacles). Les passages étroits sont d'autant plus difficiles à capturer qu'ils sont petits en comparaison des espaces libres et de la totalité de l'espace de recherche.

Dans le cas de la méthode *RRT*, les positions engendrées aléatoirement guident la croissance du graphe G. Une position aléatoire e_{rand} implique la sélection du plus proche élément e_{prox} de G et la création d'un nouvel élément e_{new} rapprochant

l'exploration de C étant la mise en avant d'une solution, (a) est en moyenne plus longue en temps que (b) et (c) ; (b) et (c) sont les deux énumérations les plus rapides dans la résolution du problème de planification pour un espace hautement dimensionné. Étant les plus rapides (à condition de ne pas connaître la solution avant de commencer la résolution du problème), l'augmentation de n_t se résume à une diminution de la différence $n_t - n_a$. Le principe est donc de mieux générer les nœuds pour éviter de les tester inutilement (*i*.e. sans ajout par la suite).

e_{prox} de e_{rand}. Les positions e_{rand} ne sont pas insérées dans G. Le constat sur les espaces étroits est identique aux méthodes *PRM*. Un échantillonnage uniforme aléatoire est utilisé dans le cadre de la méthode *RRT* [LK99] pour la planification de mouvement de mobiles kinodynamiques diversifiés (la dimension de X variant de 4 à 12) dans un environnement statique encombré de nombreux obstacles pouvant créer des passages étroits.

Dans une optique d'optimisation du graphe G résultant d'un pré-calcul dans la méthode *PRM*, L. E. Kavraki *et al.* [KLMR95] proposent un schéma d'échantillonnage divisé en quatre étapes :
– Génération aléatoire d'un ensemble d'éléments dans l'espace libre ;
– Tentative de connexion entre tous les éléments situés dans un voisinage fixé ; chaque connexion est réalisée par un planificateur local dit « simple » (tel que la *ligne droite* dans C (§ 2.3.5)) ;
– Génération aléatoire autour de l'ensemble G' des éléments faiblement connectés de G et nouvelle tentative de connexion à G par un planificateur local « simple » pour chaque élément de G' ;
– Tentative de connexion entre tous les éléments situés dans un voisinage fixé ; chaque connexion est réalisée par un planificateur local dit « complexe » (tel que le *parcours en meilleur d'abord* dans C (§ 2.3.5)).
À l'échantillonnage uniforme est ajouté un échantillonnage autour des éléments faiblement connectés. L'objectif est d'assurer une bonne couverture du graphe dans C_{libre}. La dernière étape étant la plus lente, elle est en pratique remplacée par une itération des trois premières étapes.

3.1.2 Sur les obstacles

Par réflexion sur les obstacles
Th. Horsch *et al.* [HST94] proposent, dans le cadre de la méthode *PRM*, de clore la phase *d'apprentissage* par une étape de connexion des sous-ensembles de G. Après la phase *d'apprentissage*, s'il existe deux nœuds q_a et q_b non joignables dans G (*i.e.* succession d'arcs de G), G est la réunion d'au moins deux « sous-graphes ». Le principe de cette variante est de relier les « sous-graphes » de G par la sélection d'un nœud q_0 d'un des sous-graphes et par la progression dans C par réflexion aléatoire sur les obstacles. Dans G, G_0 est le « sous-graphe » contenant le moins d'éléments, G_1 est le « sous-graphe » contenant le plus d'éléments. q_0 est l'élément de G_0 le plus proche de G_1. q_1 est l'élément de G_1 le plus proche de G_0. Le nombre de nœuds de G et de tentatives de connexion entre « sous-graphes » de G sont définis arbitrairement (*respect.* pendant et après la phase *d'apprentissage*)

pour le pré-calcul des mouvements d'un (ou deux) bras articulé(s) à 6 degrés de liberté dans un environnement $3D$.

connSousGraphesPRM(G)
  tant que $\exists(q_a, q_b)$ non joignables
    $G_0 \leftarrow$ plusPetitSousGraphe(G) ;
    $G_1 \leftarrow$ plusGrandSousGraphe(G) ;
    $q_0 \leftarrow$ confLaPlusProche(G_0, G_1) ;
    $q_1 \leftarrow$ confLaPlusProche(G_1, G_0) ;
    $\alpha \leftarrow$ directionAleatoire() ; ①
    tant que q_0 et q_1 non joignables
      $q_{new} \leftarrow$ confAuPointDeCollision(q_0, α) ; ②
      $q_{libre} \leftarrow q_0 + (1 - \epsilon).(q_{new} - q_0)$; ③
      ajouterConfArc(G, q_0, q_{libre}) ;
      $q_0 \leftarrow q_{libre}$;
      $\alpha \leftarrow$ directionDeReflexion(q_0) ; ④
  retourner G ;

ALG. 15: Connexion des « sous-graphes » de G pour une méthode *PRM*.

La réunion des « sous-graphes » de G est initialisée par un déplacement dans une direction aléatoire (ALG. 15 ①). q_{new} est la configuration en collision à partir de q_0 dans la direction α (ALG. 15 ②). Cette collision n'étant pas effectuée dans C, une rétraction de longueur ϵ est appliquée pour écarter la configuration des obstacles et la placer dans C_{libre}. Cette rétraction crée la configuration q_{libre} (ALG. 15 ③). La réflexion sur les obstacles est elle aussi calculée dans W. Le nouvel angle α (ALG. 15 ④) est relatif à l'angle de la normale à l'obstacle en q_{new}. Il est choisi aléatoirement dans l'intervalle $[-85°; +85°]$.

Par flou des obstacles

V. Boor *et al.* [BOS01] proposent, dans le cadre de la méthode *PRM*, d'utiliser un échantillonnage gaussien aux obstacles. La phase *d'apprentissage* classique est entièrement substituée à ce nouvel échantillonnage dans l'espace des configurations. Pour construire un graphe G en adéquation avec la connexité de C, le nombre de nœuds générés situés dans les espaces libres de C est d'autant plus grand que ces espaces sont étroits. Pour assurer ce résultat, une configuration ayant dans son voisinage un nombre important de configurations en collision est

une configuration intéressante. La distribution recherchée est un échantillonnage dont la probabilité de tirer un élément augmente avec le nombre de configurations voisines en collision. Par analogie aux méthodes de traitement d'image, la distribution des configurations q dans C est définie par :

$$\Phi(q,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}^n} e^{\frac{q^2}{2\sigma^2}} \ , \sigma \neq 0$$

avec σ l'écart-type de la distribution gaussienne et n la dimension de C. Le flou des obstacles est défini par :

$$f(q,\sigma) = \int_{p \in C_{obs}} \Phi(q-p,\sigma)dp$$

La distribution gaussienne aux obstacles est définie par :

$$g(q,\sigma) = \begin{cases} f(q,\sigma) & \text{si } q \in C_{libre} \\ 0 & sinon \end{cases}$$

L'échantillonnage gaussien de n configurations est réalisé par l'algorithme 16.

```
echantGaussPRM(n, σ, E)
  pour i ← 1 à n
      q1 ← confAleatoire(E) ;                          ①
      dist ← gauss(σ) ;                                ②
      α ← rand(−π, π) ;
      q2 ← confApartirDe(q1, dist, α) ;                ③
      si q1 ∈ C_libre et q2 ∈ C_obs                    ④
        ajouterConf(q1, G) ;
      sinon si q1 ∈ C_obs et q2 ∈ C_libre              ⑤
        ajouterConf(q2, G) ;
  retourner G ;
```

ALG. 16: Échantillonnage gaussien aux obstacles pour une méthode *PRM*.

La configuration q_1 est générée aléatoirement (ALG. 16 ①). q_1 est une paire de composantes (c_t, c_r) avec c_t les composantes de position (*translational*) et c_r les composantes restantes. La distance $dist$ est générée selon une distribution gaussienne d'écart-type σ (ALG. 16 ②). La configuration q_2 est une paire de composantes (c'_t, c'_r) avec c'_t les composantes de position et c'_r les composantes restantes. Elle est située à une distance $dist$ de q_1 dans la direction α pour ces composantes

de position c'_t (ALG. 16 ③). Ces composantes restantes c'_r sont choisies aléatoirement. Seules les distributions des composantes de position sont gaussiennes aux obstacles. Un tirage gaussien est un couple de configuration (q_1, q_2) dans lequel un élément est dans C_{libre} et un élément est dans C_{obs} (ALG. 16 ④ et ⑤).

Par relation aux obstacles

N.M. Amato *et al.* [ABD+98a] proposent dans le cadre de la méthode *PRM*, de générer des nœuds selon une stratégie optimisant la capture de la connexité de l'espace. Les nœuds sont choisis aléatoirement en contact, à proximité ou autour des C-obstacles (cette variante est nommée *OBPRM* pour *obstacle-based PRM*). La génération des nœuds en contact et à proximité d'un obstacle[2] est définie dans l'algorithme suivant (ALG. 17 avec $d = 0$ pour les configurations en contact et $d > 0$ pour les configurations proches). Les nœuds autour des obstacles sont définis par dilatation des obstacles, en fonction de leur nombre par obstacle et de l'intensité de dilatation recherchée.

La génération d'une configuration en contact ou à proximité des obstacles débute par la sélection de deux points situés respectivement sur le mobile $\mathcal{M}$ et sur l'obstacle $\mathcal{O}$ (ALG. 17 ① et ②). Ces points peuvent être :
- cM le centre de masse ;
- rV un point de contour ;
- eV un point extrême (*i.e.* un extremum sur les axes de l'espace de travail W) ;
- rT un point aléatoire d'un des triangles issu de la tesselation de $\mathcal{O}$; le triangle est lui aussi choisi aléatoirement ;
- wT un point aléatoire d'un des triangles issu de la tesselation de $\mathcal{O}$; le triangle est choisi avec une probabilité proportionnelle à son aire.

Le choix de la nature de ces points se traduit par une stratégie : cM est adapté aux objets sphériques, rV dirige les nœuds vers les portions les plus complexes des objets, eV distribue uniformément les configurations autour des objets, rT dirige les nœuds vers les parties des objets où les formes sont les plus changeantes, wT dirige les nœuds vers les parties les plus larges. En pratique, cM est conseillé pour les objets symétriques et la combinaison $rV + wT$ pour les objets de formes quelconques.

[2] Les tests étant réalisés pour deux passages étroits en $3D$ et pour un *alpha puzzle*, le nombre d'obstacles varie entre 1 et 4. Ceci explique l'absence des considérations de répartition des tirages entre les obstacles. Les tests utilisent un nombre identique de nœuds pour tous les obstacles (quelles que soient leur taille et leur position).

genContEtProxObsPRM$(d, \rho, \mathcal{M}, \mathcal{O})$
- $p_{rob} \leftarrow$ pointAleatoireDunObjet$(\mathcal{M})$; ①
- $p_{obs} \leftarrow$ pointAleatoireDunObjet$(\mathcal{O})$; ②
- $q_{in} \leftarrow$ transMobile$(p_{obs} - p_{rob})$; ③
- tant que $q_{in} \in C_{free}$
  - $\alpha \leftarrow$ directionAleatoire() ; ④
  - $q_{in} \leftarrow$ rotMobile(q_{in}, α) ; ⑤
- $\alpha \leftarrow$ directionAleatoireVersClibre() ; ⑥
- $q_{out} \leftarrow$ confLibreParTrans(q_{in}, α) ; ⑦
- $q \leftarrow$ confContactOuProche$(q_{in}, q_{out}, d, \rho)$; ⑧
- retourner q ;

ALG. 17: Génération d'une configuration en contact ou à proximité des obstacles pour une méthode *PRM*.

La configuration q_{in} est obtenue par translation du mobile dans la direction du vecteur $\overrightarrow{p_{rob}p_{obs}}$ (ALG. 17 ③). La définition d'une unique direction est valable dans un environnement $2D$; elle est exprimée par trois valeurs aléatoires d'angle en $3D$ (ALG. 17 ④). Tant que le mobile dans la configuration q_{in} n'est pas en collision, q_{in} subit une rotation d'angle α (ALG. 17 ⑤).

Après tirage aléatoire (ALG. 17 ⑥) d'une direction partant de q_{in} vers l'extérieur (*i.e.* C_{libre}), q_{out} est la configuration de C_{libre} obtenue par translation de q_{in} dans cette direction (ALG. 17 ⑦). La configuration q en contact ou à proximité de l'obstacle $\mathcal{O}$ est obtenue par dichotomie entre q_{in} et q_{out} à une valeur d près définie par la distance métrique ρ (ALG. 17 ⑧).

3.1.3 Sur les espaces libres

Par rétraction des espaces libres

S. Wilmarth *et al.* [WAS98] proposent dans le cadre de la méthode *PRM*, de diminuer la différence entre n_t et n_a (*respect.* nombre de nœuds générés et nombre de nœuds ajoutés) par rétraction de toutes les configurations (de C_{libre} et de C_{obs}) vers l'axe médian de C_{libre}. L'axe médian n'est pas explicitement calculé. Les configurations sont déviées vers l'axe médian par relation de proximité aux points de collision les plus proches. Cette relation est établie par énumération d'un en-

semble de translations fixées [3].

```
echantAxeMedianPRM(n, C)
  pour i ← 1 à n
      q1 ← confAleatoire(C) ;                                          ①
      q2 ← confAuPointDeCollisionLePlusProche(q1, C) ;                 ②
      si q1 ∈ C_libre
          Δq ← normaliser(q1 − q2) ;                                   ③
          q ← q1 ;
      sinon
          Δq ← normaliser(q2 − q1) ;                                   ④
          q ← q2 ;
      tant que nbConfAuPointDeCollisionLePlusProche(q, C) = 1          ⑤
          q ← transMobile(q, Δq) ;
      ajouterConf(q, G) ;                                              ⑥
  retourner G ;
```

ALG. 18: Échantillonnage dévié vers l'axe médian pour une méthode *PRM*.

Après tirage aléatoire (ALG. 18 ①) d'une configuration q_1 de C, q_2 est définie comme la configuration de C_{obs} la plus proche de q_1 (ALG. 18 ②). Si q_1 est dans C_{libre} (*respect.* dans C_{obs}), le vecteur de déplacement Δq vers l'axe médian est le vecteur $\overrightarrow{q_2q_1}$ et la configuration de départ de ce déplacement est q_1 (*respect.* le vecteur $\overrightarrow{q_1q_2}$ et la configuration q_2). L'axe médian n'étant pas explicitement calculé, le principe de cette méthode est de translater successivement une configuration q d'un déplacement Δq tant que le nombre des configurations en collision les plus proches de q est différent de 2 (ALG. 18 ⑤). Ne disposant d'aucune information sur la distance à l'axe médian, le déplacement Δq doit être unitaire ; le vecteur Δq est normalisé dans C (ALG. 18 ③ et ④). La configuration ainsi déplacée sur l'axe médian (*i.e.* à équidistance de deux obstacles) est ajoutée au graphe G (ALG. 18 ⑥).

[3]Le calcul de la plus petite distance entre deux polygones convexes et de la distance de pénétration en collision est le sujet de nombreuses librairies de détection de collision (*DEEP* pour « Dual-space Expansion for Estimating Penetration Depth » [KLM02], *Enhanced GJK* pour « enhanced version of Gilbert, Johnson and Keerthi » [Cam97], *PQP* pour « Proximity Query Package » [LGLM99], *RAPID* pour « Robust and Accurate Polygon Interference Detection system » [GLM96], *SOLID* pour « SOftware Library for Interference Detection » [vdB01], *SWIFT++* pour « Speedy Walking via Improved Feature Testing (for non-convex objects) » [EL01] et *V-Clip* pour « Voronoi-Clip » [Mir97]).

Par dilatation des espaces libres

D. Hsu *et al.* [HKL$^+$98] proposent dans le cadre de la méthode *PRM*, d'augmenter la capture des espaces étroits par dilatation des espaces libres (ALG. 19). C'_{libre} est le résultat de la dilatation de C_{libre} ; il est calculé par évaluation d'un ensemble de positions susceptibles d'appartenir à son contour [OG96]. La construction de G est composée d'une première étape de construction de G' dans C'_{libre} (C_{libre} dilaté), puis d'une seconde étape de modification de G' par déplacement des nœuds et des arcs de G' situés en dehors de C_{libre}. Le déplacement des éléments de C'_{obs} dans C_{obs} est réalisé par une exploration du voisinage de chacun de ces éléments. Le voisinage d'une configuration q est associé à une distance métrique ρ et est noté $V(q)$. L'échantillonnage étant défini, lors des déplacements de C'_{libre} vers C_{libre} par une méthode locale (§ 2.3.5), l'algorithme définissant l'échantillonnage par dilatation est aussi l'algorithme de construction de G. Les nœuds étant générés par déplacement de C'_{libre} vers C_{libre}, un nœud b généré à partir de a (avec $b \in C_{libre}$ et $a \in C'_{libre}$) est noté $dpl(a) = b$. Si $a \in C_{libre}$, alors $dpl(a) = a$.

Après dilatation de C_{libre} (ALG. 19 ①), les configurations sont choisies aléatoirement dans C (ALG. 19 ②). G' est construit par ajout successif de configurations de C'_{libre} (ALG. 19 ③). Les configurations n'appartenant pas à C'_{libre} mènent à un nouveau tirage aléatoire dans C. Tout chemin local reliant deux configurations voisines de G' correspond à un arc (ALG. 19 ⑤). Cet arc doit être inclus dans l'espace dilaté C'_{libre} (ALG. 19 ④). Le graphe G' construit, chaque nœud q (ALG. 19 ⑥) et chaque arc $\mathcal{L}(q_1, q_2)$ (ALG. 19 ⑩) tente d'être déplacé dans C_{libre}. Les nœuds de G' appartenant à C_{libre} (ALG. 19 ⑦) sont insérés dans G.

Les nœuds q de G' appartenant à C_{obs}, un maximum de m tentatives de substitution vers une nouvelle configuration p choisie aléatoirement (ALG. 19 ⑨) dans le voisinage V_q est appliqué. La boucle de substitution prend fin à la première configuration p dans C_{libre} (ALG. 19 ⑧) ; si cette boucle ne trouve aucune configuration p dans C_{libre} en m itérations, le nœud q (*i.e.* situé dans C'_{libre} et dans C_{obs}) est oublié.

Pour les couples (q_1, q_2) associés à un arc dans G' (*i.e.* associés à un chemin $\mathcal{L}(q_1, q_2)$), si le couple de nœuds associé à leur déplacement dans C_{libre} (*i.e.* le couple $(dpl(q_1), dpl(q_2))$) est un arc sans collision dans C_{libre} (*i.e.* associés à un chemin $\mathcal{L}(dpl(q_1), dpl(q_2))$ dans C_{libre} (ALG. 19 ⑪)), alors cet arc est ajouté dans G (ALG. 19 ⑫). Si le couple de nœuds $(dpl(q_1), dpl(q_2))$ est un arc en collision avec C_{libre}, k configurations sont choisies aléatoirement (ALG. 19 ⑬) dans le voisinage du chemin reliant q_1 à q_2 dans G'. S'il existe une séquence parmi ces k configurations reliant $dpl(q_1)$ à $dpl(q_2)$ dans G, un arc reliant $dpl(q_1)$ à $dpl(q_2)$ est ajouté dans G (ALG. 19 ⑭).

```
consGrapheParDilatClibrePRM(n, m, C)
  | C'_libre ← dilatation(C_libre) ;                                  ①
    i ← 0 ;
    tant que i < n
      | q ← confAleatoire(C) ;                                        ②
      | si q ∈ C'_libre                                               ③
          ajouterConf(q, G') ;
          i ← i + 1 ;
          pour chaque p ∈ V(q)
            | si L(p, q) ⊂ C'_libre                                   ④
                ajouterArc(p, q, G') ;                                ⑤
  | pour chaque q ∈ G'                                                ⑥
    si q ∈ C_libre                                                    ⑦
      | ajouterConf(q, G) ;
    sinon
      | i ← 0 ;
      | p ← q ;
      | tant que i < m et p ∉ C_libre                                 ⑧
          p ← confAleatoire(V(q)) ;                                   ⑨
          si p ∈ C_libre
            | ajouterConf(p, G) ;
          sinon i ← i + 1 ;
  | pour chaque L(q1, q2) ∈ G'                                        ⑩
    si L(dpl(q1), dpl(q2)) ⊂ C_libre                                  ⑪
      | ajouterArc(dpl(q1), dpl(q2), G) ;                             ⑫
    sinon
      | Q ← kConfAleatoires(V_L(dpl(q1), dpl(q2))) ;                  ⑬
      | si existeUneSequenceReliant(dpl(q1), dpl(q2), Q, C_libre)
          ajouterCetteSequence(dpl(q1), dpl(q2), G) ;                 ⑭
  | retourner G ;
```

ALG. 19: Construction de G par dilatation de C_{libre} pour une méthode *PRM*.

La dilatation de taille δ de C_{libre} en C'_{libre} facilite[4] la capture des passages étroits. Cette modification de l'espace de recherche invalide cependant l'adéquation entre

[4]D. Hsu *et al.* [HKL+98] ont établi la propriété suivante : pour un environnement composé de deux espaces libres (de forme carrée et de côté $C1$) reliés par un passage étroit (de forme rectangulaire, de longueur $C1$ et de largeur $C2$ avec $C2 < C1$) et C'_{libre} obtenu par dilatation de taille δ de C_{libre}, si ce passage étroit est capturé dans C_{libre} par un graphe de débattement θ

la connexité du graphe (G' dans ce cas) et celle de l'espace de travail. Cette dilatation peut :
– Créer des passages inexistants ;
– Supprimer des obstacles de W (si la valeur de dilatation est supérieure à la taille de l'obstacle).
Ces deux aspects soulignent l'importance du choix d'une valeur de dilatation δ appropriée.

Par visibilité des espaces libres
C. Nissoux *et al.* [Nis99, SLN00] proposent dans le cadre de la méthode *PRM*, d'assurer simultanément couverture et connexité de G par le biais d'une construction incrémentale reposant sur une relation de visibilité. Pour reprendre les termes de C. Nissoux, « l'idée est de forcer l'algorithme à aller là où il n'est pas allé, à explorer en priorité les régions de l'espace libre qu'il n'a pas encore couvertes, en l'empêchant d'aller là où il est déjà allé ». Les nœuds sont choisis aléatoirement dans C mais doivent satisfaire des propriétés de visibilité [GO97].

Tous les nœuds de G sont des nœuds gardiens, primaires ou secondaires[5] :
– Un gardien sans visibilité avec aucun nœud de G (excepté lui-même) est dit primaire ; par conséquence de leur non-visibilité mutuelle, les gardiens primaires sont sans connexion possible entre eux ;
– Les gardiens secondaires relient les gardiens primaires pour constituer un graphe ; ils sont également appelés nœuds de liaison.

composé de n nœuds, alors ce passage étroit est capturable dans C'_{libre} avec $n/(1+c)^n$ nœuds avec $c = \delta\theta$.

[5]Dans un contexte géométrique, deux objets sont mutuellement « visibles » s'il existe un segment les reliant sans collision avec un autre objet. Dans la galerie d'art des théorèmes de J. O'Rourke [GO97], le gardien est un point, une source de visibilité ou d'illumination.

```
consGrapheParVisibiliteDansClibrePRM(n, C)
    nbEchec ← 0 ;
    tant que nbEchec < n
        q ← confAleatoire(C) ;                                          ①
        Gardiens ← ∅ ;
        R_vis ← ∅ ;
        nb_connect ← 0 ;
        pour chaque Rj ∈ R                                              ②
            noeudVu ← 0 ;
            k ← 0 ;
            tant que noeudVu = 0 et k < nbElement(R_j)                  ③
                r ← KiemeElement(k, R_j) ;                              ④
                si q ∈ Visi_L(r)                                        ⑤
                    noeudVu ← 1 ;
                    Gardiens ← Gardiens ∪ r ;
                    ajouterRegion(R_j, R_vis) ;
                    nb_connect ← nb_connect + 1 ;
                k ← k + 1 ;
        si nb_connect = 0
            ajouterRegion(Visi_L(q), R) ;                               ⑥
            ajouterElementDansRegion(q, Visi_L(q)) ;                    ⑦
            ajouterConf(q, G) ;                                         ⑧
            nbEchec ← 0 ;                                               ⑨
        sinon
            nbEchec ← nbEchec + 1 ;                                     ⑩
            si nb_connect > 1                                           ⑪
                ajouterConf(p, G) ;                                     ⑫
                pour chaque p ∈ Gardiens
                    ajouterArc(p, q, G) ;                               ⑬
                    retirerRegion(R_p, R) ;
                R_fus ← Visi_L(q) ;                                     ⑭
                ajouterElementDansRegion(q, R_fus) ;                    ⑮
                pour chaque R_n ∈ R_vis
                    R_fus ← R_fus ∪ R_n ;                               ⑯
                    deplacerElementsDansRegion(R_n, R_fus) ;            ⑰
                ajouterRegion(R_fus, R) ;
    retourner G ;
```

ALG. 20: Construction de G sous condition de visibilité dans C_{libre} pour une méthode *PRM*.

$Visi(q)$ définit le domaine de visibilité à partir du nœud q. Les nœuds étant reliés dans G par appel d'une méthode locale (produisant un chemin $\mathcal{L}(q,p)$), le domaine de visibilité de $\mathcal{L}$ à partir de q est défini par :

$$Visi_{\mathcal{L}}(q) = \{p \in C_{libre} / \mathcal{L}(q,p) \in C_{libre}\}$$

À chaque itération de la construction du graphe G, une nouvelle configuration q est choisie aléatoirement (ALG. 20 ①). R est la liste des zones de visibilité définie par les nœuds de G. Pour chaque nouvelle configuration q, $Gardiens$ est la liste des nœuds visibles de q. Si un nœud d'une des régions de R (ALG. 20 ② et ④) est inclus dans le domaine de visibilité de q (ALG. 20 ⑤), il est ajouté dans la liste $Gardiens$. Le nombre de nœuds de $Gardiens$, connectés avec q, est comptabilisé dans la variable nb_{connex}. La condition $noeudVu = 0$ implique la sélection d'un seul nœud (ALG. 20 ③) par région R_j. Cette condition sur la liste des nœuds $Gardiens$ permet de réduire[6] le nombre d'arcs générés par la suite (ALG. 20 ⑤). Après le parcours de toutes les régions de R, l'absence de nœud visible (*i.e.* $nb_{connect} = 0$) indique la présence d'un nouveau gardien-primaire q dans C ; le domaine de visibilité du nœud q est ajouté à R (ALG. 20 ⑥) ; q est ajouté dans cette nouvelle région (ALG. 20 ⑦) ; q est ajouté à G (ALG. 20 ⑧). Si q est visible d'au moins deux nœuds (ALG. 20 ⑪), c'est un gardien secondaire ; q est ajouté dans le graphe (ALG. 20 ⑫) et un arc reliant q à chacun des nœuds de $Gardiens$ (ALG. 20 ⑬) est ajouté dans G. Les arcs ainsi créés relient les domaines de visibilité entre eux. Une nouvelle région R_{fus}, résultant d'une fusion du domaine de visibilité de q (ALG. 20 ⑭) et de toutes les régions associées à chaque nœud relié à q (ALG. 20 ⑯), est insérée dans R. Cette fusion s'accompagne du déplacement des nœuds de ces régions dans R_{fus} (ALG. 20 ⑰). Le nœud q est également placé dans R_{fus} (ALG. 20 ⑮).

À chaque ajout d'un nouveau gardien primaire dans G, la variable $nbEchec$ est remise à zéro (ALG. 20 ⑨). À chaque ajout d'un gardien secondaire dans G, la variable $nbEchec$ est incrémentée (ALG. 20 ⑩). La variable $nbEchec$ compte le

[6] La sélection du premier nœud visible comme représentant de sa région peut cependant être source de chemin inutilement long dans G. Ce choix est justifié par la diminution de la densité des arcs de G. La sélection d'un seul nœud réduit le nombre d'arcs insérés dans G. Une nouvelle configuration peut également n'avoir aucun intérêt en termes de couverture et de connexité (si $Visi_{\mathcal{L}}(q) \in R$)). Cette relation relativise l'utilité d'un parcours approfondi de R_j pour la sélection de l'élément visible le plus proche de q [Nis99].

nombre des ajouts consécutifs de nœuds secondaires dans G. L'algorithme recherche des nouveaux nœuds tant que $nbEchec$ est inférieure à une valeur fixée. Ce critère d'arrêt permet de contrôler la génération du graphe en fonction d'une couverture théorique. Au cours de la construction de G, le rapport $1/nbEchec$ est une estimation de la proportion du volume de C non-couvert par le domaine de visibilité de G. Pour assurer une couverture probabiliste des espaces mesurant 1% du volume de C, il faut fixer $n = 100$. Les gardiens secondaires reliés à un seul élément des régions R ne sont pas considérés (ALG. 20 ⑪). Cette condition permet d'éviter de générer des nouveaux nœuds en direction des concavités de C ; elle permet de réduire la taille de G.

3.1.4 Sur des séquences déterministes

S.R. Lindemann *et al.* [LL03] proposent de remplacer les tirages aléatoires par des séquences déterministes[7]. Cette variante est appelée ***RDT*** (abréviation de « Rapidly-Exploring Dense Tree »). Les tirages aléatoires issus d'une séquence déterministe sont appelés q_{dens}. L'association d'une notion de densité suggère l'itération des expansions en direction des configurations q_{dens}. Cette itération transforme la phase d'expansion de durée Δt, en direction de q_{dens}, par une unique expansion de durée n fois Δt en direction de q_{dens}. La valeur de n est définie par la configuration de C_{libre}, la plus éloignée de q_{prox}. La sélection du plus proche voisin d'une configuration q_{dens} n'est plus restreinte aux nœuds de G. q_{prox} devient un nœud de G ou une configuration d'un arc reliant deux configurations de G.

M. Branicky *et al.* [BLOY01] proposent dans le cadre de la méthode ***PRM***, d'utiliser des séquences déterministes pour contrôler la répartition des configurations. Le graphe G est construit avec une garantie de couverture de C, à une densité près, grâce à la discrépance.

[7] Le choix des séquences déterministes (également appelées quasi-aléatoires) s'oppose au choix des séquences aléatoires (également appelées pseudo-aléatoires). Parmi les séquences uniformes : une séquence pseudo-aléatoire est une suite non-prédictible et à répartition uniforme après une période fixée ; une séquence quasi-aléatoire est une suite dont l'uniformité de la répartition est contrôlée. La mesure de la non uniformité de la répartition d'une telle suite est appelée discrépance.

3.1.5 Sur l'objectif

S.M. LaValle *et al.* [LK00] proposent dans le cadre de la méthode ***RRT***, d'utiliser la configuration finale pour dévier la progression de l'arbre dans C. Le principe de cette variante est d'utiliser, à intervalle fixe ou variable, e_{obj} à la place de e_{rand}.

La variante appelée *RRT-GoalBias* [LK00] utilise une valeur d'intervalle fixe v. Une expansion en direction de e_{obj} est intercalée toutes les v expansions en direction de e_{rand}. Une valeur d'intervalle de 20 améliore la convergence vers l'objectif. Une valeur d'intervalle de $1/20$ rapproche la méthode ***RRT*** de la méthode ***RPP***. Dans ce cas, 20 expansions sur 21 sont dirigées vers l'objectif. Une telle utilisation de la déviation vers e_{obj} guide beaucoup plus vite l'arbre vers e_{obj} mais est susceptible d'emprisonner l'arbre à proximité des obstacles (***i.e.*** dans des minima locaux).

La variante appelée *RRT-GoalZoom* [LK00] utilise une valeur d'intervalle variable. *RRT-GoalZoom* modifie la valeur de v en fonction de la distance entre e_{obj} et le plus proche élément de G. La déviation vers e_{obj} est en relation avec la distance au plus proche élément de G. Il résulte de cette relation une accentuation graduelle de la déviation des expansions en direction de e_{obj}. Cette relation ne remédie aucunement aux problèmes des minima locaux introduit par la déviation répétée vers un élément unique[8].

3.1.6 Sur la connexité

L. E. Kavraki [Kav95, KL94b, KL94a] propose dans le cadre de la méthode ***PRM***, d'augmenter les tirages aléatoires dans le voisinage des nœuds de G de faible connexité. Après la phase ***d'apprentissage***, le nombre de connexions établies pour chaque nœud guide le planificateur local au voisinage de chacun des nœuds. Les espaces étroits peuvent générer des nœuds faiblement connectés. Pour rétablir l'adéquation entre connexité de G et connexité des espaces libres, cette méthode propose de réaliser des tirages aléatoires supplémentaires Y (appelés "expansion"

[8] Dans le cadre d'une recherche bidirectionnelle, *RRT-Connect* [KL00] augmente la convergence des deux arbres par déviation respective vers les éléments les plus proches des arbres opposés. *RRT-Connect* est une variante de *RRT-GoalBias* appliquée à une recherche bidirectionnelle. Aucune valeur d'intervalle v n'est fixée. La déviation est effective tant qu'aucune collision n'est détectée. La déviation est réalisée vers les éléments les plus proches dont les positions varient au cours de la progression de l'arbre dans C.

de G) dans le voisinage des nœuds de faible connexité de G. Les nouveaux nœuds de Y sont ensuite reliés à G (sous réserve de planification locale).

Le voisinage d'un nœud est défini par une distance métrique ρ. Sous l'hypothèse que cette distance métrique reflète la connexité des espaces libres : le degré d_q d'un nœud q est le nombre d'arcs associés à q dans ce voisinage ; le poids w_q d'un nœud q, pour un voisinage composé de d_q nœuds et pour un graphe composé de n nœuds (dont les degrés sont d_1 à d_n), est défini par :

$$w_q = \frac{1}{d_q + 1} / \sum_{t=1}^{n} \frac{1}{d_t + 1}$$

Le poids w_q est ici la mesure de connexité[9] du nœud q, normalisée sur l'ensemble des nœuds situé dans le voisinage de q. Plus un nœud est situé dans un espace étroit, plus la valeur de w_q est admise petite. Les nœuds de plus petits w sont donc les nœuds les plus faiblement connectés. Le choix de la distance métrique définissant le voisinage et de leur nombre m dépend du problème considéré. Il est conseillé de choisir m dans l'intervalle $[n/2; n]$. L'algorithme suivant (ALG. 21) [KL94b] réalise une expansion de m nœuds dans le graphe G composé de n nœuds.

L'expansion du graphe G commence par la sélection de m nœuds de poids w minimum (ALG. 21 ①) ; V est l'ensemble de ces m nœuds de G. Cet ensemble V guide la génération de m configurations aléatoires y à partir d'un élément $p(y)$ de V (ALG. 21 ②). Chaque y est générée dans un voisinage de $p(y)$ défini par ρ. $p(y)$ est l'élément de V responsable du nouvel élément y. La probabilité[10] de sélectionner un y dans V est définie par la mesure w. Après génération aléatoire des m configurations, l'algorithme tente de connecter chaque q de Y avec son $p(q)$ (ALG. 21 ③). En cas d'échec, q est abandonné. En cas de succès, la connexion $(q, p(q))$ est ajoutée à G et l'ensemble des k configurations les plus proches de q

[9] Le poids w_q [KSLO94, KSLO96] peut également être :
- l'inverse du nombre de ces nœuds voisins pour une valeur de voisinage fixé ;
- la distance vers l'élément de G non connecté le plus proche ;
- fonction du rapport entre échecs et succès du planificateur local. Cette mesure de w_q est préférée pour les mouvements d'un bras articulé dans le plan [KSLO96]. Un rapport d'échec $r_f(q)$ est associé à chaque nœud, défini par $r_f(q) = f(q)/(n(q) + 1)$, avec $n(q)$ le nombre de tentatives de connexion du nœud q et $f(q)$ le nombre d'échecs parmi $n(q)$. Dans ce cas, le poids w_q devient : $w_q = r_f(q)/(\sum_{a \in N} r_f(a))$.

[10] Le poids w étant normalisé sur n valeurs, un tirage aléatoire dans V est défini dans l'intervalle $[0; a]$ avec $a = \sum_{i=0}^{m} w_{v(i)}$ et $v(i)$ le $i^{ème}$élément de V. La probabilité de tirage de $v(i)$ est $Pr[v(i)] = w_{v(i)}$.

est sélectionné ; cet ensemble est appelé Z. Ces k configurations sont sélectionnées parmi les $n + m - 2$ configurations restantes (ALG. 21 ④). Pour chaque élément p de cet ensemble Z, les connexions possibles avec q sont ajoutées à G (ALG. 21 ⑤).

```
expansGraphePRM(m, ρ, G)
  V ← confDePlusFaiblePoids(m, G) ;                                  ①
  pour i ← 1 à m
      y ← ConfAleatoire(ρ, w, V) ;                                   ②
      ajouterConf(y, Y) ;
  pour chaque q ∈ Y
      si q et p(q) joignables                                        ③
          ajouterArc(q, p(q), G) ;
          Z ← confLesPlusProches(k, q, G ∪ Y − {q; p(q)}) ;          ④
          pour chaque p ∈ Z
              si p et q joignables                                   ⑤
                  ajouterArc(p, q, G) ;
  retourner G ;
```

ALG. 21: Expansion de G pour une méthode *PRM*.

D. Hsu *et al.* [HLM97] proposent une variante de la méthode *PRM* utilisant également la notion de poids w_q pour la sélection des régions dans lesquelles l'échantillonnage doit être plus soutenu. Le graphe G est construit par propagation dans C_{libre} à chaque requête de planification. Chaque nœud de G est associé à un poids w_q, égal à l'inverse de son nombre de nœuds-voisins. La probabilité de sélectionner un nœud est égale à ce poids divisé par la somme des poids des nœuds de G. La sélection d'un nœud implique la génération d'un nouveau nœud dans son voisinage. Ainsi G progresse vers les régions les moins explorées. L'ajout d'un nœud q implique une modification des poids de tous les nœuds de son voisinage $V(q)$.

Cette méthode uniquement utilisée dans l'approche *PRM* pourrait être utilisée pour la gestion des connexions entre arbres des méthodes bi-*RRT* ou de *RRT* en parallèle. La notion de poids en fonction des nœuds-voisins est cependant inadaptée. La notion de densité dans le voisinage nous semble plus conforme à son application à une méthode *RRT*.

L'ensemble des méthodes d'échantillonnage présentées dans ce chapitre s'appuie sur l'adaptation de la localisation de l'échantillonnage sans décomposition cellulaire de C_{libre}. Le principe de réflexion sur les obstacles est appliqué en l'absence de contraintes différentielles sur les arcs de G dans la méthode *PRM*. Dans le cadre de la méthode *RRT*, une réflexion sur les obstacles de C_{libre} satisfaisant les contraintes différentielles de $\mathcal{M}$ implique la définition d'une *amplitude de réflexion* ; À chaque itération, le graphe G progresse du pas défini par le système différentiel associé à $\mathcal{M}$; Pour une configuration q, menant à une succession d'inévitables collisions, il importe de reprendre l'expansion de G à partir d'un nœud q', distant de n pas de q ; nous appelons *amplitude de réflexion* le nombre de pas n nécessaire à une expansion plus probante de G dans le voisinage de q. La définition de n équivaut donc à une évaluation des espaces libres propres au voisinage de q.

3.2 Contributions

Dans le cadre de la méthode *RRT*, l'expansion de G est fonction :
- De la distance séparant q_{rand} et q_{prox} ; en présence d'obstacles, plus la distance est grande et plus la probabilité d'être séparé par un obstacle est grande ;
- Des dimensions et positions des régions inexplorées ; la probabilité d'expansion est plus grande en direction des plus grands triangles du diagramme de Voronoï associé à G ; l'exploration exhaustive des régions de C (notamment les plus grandes) peut cependant être facultative ;
- Des propriétés de l'échantillonnage choisi.

Les variations d'échantillonnage de l'espace de recherche présentées dans ce chapitre dépendent :
- De l'interprétation des collisions avec C_{obs} ; le principe de la variante *OBPRM* souligne le problème des différences inhérentes entre les espaces libres de W et de C ;
- De l'interprétation de relations entre les nœuds de G ; l'association de valeur de connexité aux nœuds de G permet dans le cadre des méthodes *PRM* de dévier les tirages aléatoires à proximité des nœuds de faible connexité ;
- D'une déformation des tirages aléatoires ; le principe de la rétraction permet de déformer les tirages aléatoires vers l'axe médian, améliorant la progression de G dans C. Le flou Gaussien aux obstacles augmente la capture des passages

étroits ;
– De la comparaison entre les propriétés de C et de G ; le principe de visibilité permet de réduire le nombre de nœuds de G. La notion de visibilité associée à la méthode locale $\mathcal{L}$ permet de discriminer les nœuds entre eux ; la relation implicite introduite entre les nœuds de G permet dans le cadre des méthodes *PRM* une définition dynamique du nombre de nœuds nécessaire à une couverture appropriée de G.
– De déformation de C_{libre} ; le principe de la dilatation de C_{libre} permet d'augmenter la taille de l'espace libre, augmentant du même coup la couverture de G dans C. La création de passages inexistants suppose une possibilité de replanification locale, susceptible de créer des nœuds non connectables en Δt ; deux nœuds (q_1, q_2) sont connectables en Δt s'il existe une commande u telle que $q_1 = q_2 + \dot{q}_2$ ou $q_2 = q_1 + \dot{q}_1$ avec $\dot{q}$ la dérivée temporelle de q. Le problème de la connexion de nœuds non connectés en Δt est connexe au problème de la discontinuité[11] de la trajectoire, résultant de la discrétisation de la commande de $\mathcal{M}$.

Nous proposons de définir un échantillonnage progressif de C_{libre} adapté à l'expansion de G avec :
– Une décomposition en tranche de C_{libre} pour assurer la localisation des tirages aléatoires dans C_{libre} ; À chaque itération, un tirage aléatoire guide la progression de G ; Le maintien de q_{rand} dans C_{libre} augmente la probabilité de q_{new} d'être dans C_{libre} ;
– Une évolution de la localisation des tirages aléatoires en fonction de la progression de G dans C_{libre} ; La variante *Goal-zoom* modifie la déviation des tirages aléatoires en fonction de la distance entre q_{obj} et le nœud le plus proche de G. Nous proposons de localiser les tirages aléatoires proches des feuilles de G.

3.3 Sur des contraintes géométriques

Les méthodes de décomposition découpent C_{libre} en un ensemble de régions disjointes appelées cellules. Si la réunion de ces cellules est égale à (*respect.* différente de) C_{libre}, ces méthodes sont exactes (*respect.* approchées[12]). Nous propo-

[11] résoluble par introduction de perturbations

[12] Pour J.C. Latombe [Lat91], les méthodes de décomposition approchées ne permettent pas une représentation exacte de C_{libre}. Pour permettre une décomposition itérative de C_{libre}, les cellules

sons une nouvelle décomposition approchée de C_{libre} en un ensemble de régions délimitées par des obstacles de formes polygonales.

3.3.1 Expressions des espaces libres de C

Les obstacles de $\mathcal{O}$ pouvant être concaves ou convexes, il importe de différencier :
– Les sommets A, situés dans les convexités des obstacles ;
– Les sommets B, situés dans les concavités des obstacles.
Chaque obstacle o_i de W est défini par un ensemble de n points p. Cet ensemble $\{p_i, i = 1, \ldots, n\}$ est une liste circulaire ordonnée telle que les segments $[p_i p_{i+1}]$ constituent les faces du contour de o_i. Un obstacle convexe à n faces est défini par n points A (noté $\{A_i, i = 1, , \ldots, n\}$). Un obstacle concave à n faces est défini par m points A et l points B avec $n = m + l$ (noté $\{A_i|B_i, i = 1, , \ldots, n\}$).

Pour définir une approximation des espaces libres autour des obstacles (par relation aux faces des obstacles), la décomposition de C_{libre} est une succession :
– De trapèzes rectangles (par expression des faces $[A_i\ A_{i+1}]$), notés C_{AA} ;
– De secteurs d'ellipses pleins (par expression des sommets A_i), notés C_A ;
– De polygones à trois ou quatre côtés (par expression des faces $[A_i\ B_{i+1}]$, notés C_{AB} ;
– De polygones à trois ou quatre côtés (par expression des faces $[B_i\ A_{i+1}]$, notés C_{BA} ;
– De polygones à trois ou quatre côtés (par expression des faces $[B_i\ B_{i+1}]$, notés C_{BB}.
Les sommets B_i ne s'expriment pas seuls. C_{libre} ne contient aucune région C_B. Les sommets situés sur des convexités de $\mathcal{O}$ expriment des régions de formes convexes. Les sommets situés sur des concavités de $\mathcal{O}$ expriment des régions de formes quelconques (*i.e.* convexes ou concaves, à trois ou quatre côtés). L'ensemble des régions C_A et C_{AA} est appelé CA. L'ensemble des régions C_{AB}, C_{BA} et C_{BB} est appelé CB. L'ensemble des régions CA et CB est appelé $\mathcal{F}$.

Une région C_{AA} (FIG. 3.1.a) est un trapèze rectangle $(A_i, A_{i+1}, A'_i, A'_{i+1})$ avec :

sont des formes régulières telles que les rectangles pour un espace en $2D$. C_{libre} est approché par décomposition en cellules vides (*i.e.* incluses dans C_{libre}), mixtes (*i.e.* en intersection avec C_{libre} et C_{obs}) et pleines (*i.e.* incluses dans C_{obs}). La décomposition est construite par subdivision successive des cellules mixtes jusqu'à une dimension fixée (définissant également la précision de cette approximation de C_{libre}). Initialement proposées par T. Lozano-Pérez [LP81] et R.A. Brooks [BLP83], les méthodes de décompositions approchées de C_{libre} sont destinées à des espaces de dimensions inférieures à 4.

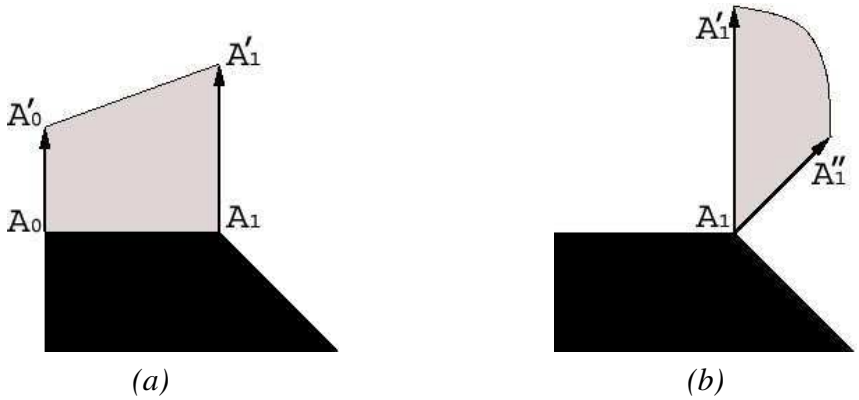


FIG. 3.1 – Régions C_{AA} et C_A de C_{libre}.

Une région C_{AA} en (a) et une région C_A en (b).

– $\overrightarrow{n}$ la normale à la face $[A_i\ A_{i+1}]$;
– $\overrightarrow{u}$ colinéaire à $\overrightarrow{n}$;
– A'_i le point translaté de A_i par $\overrightarrow{u}$;
– A'_{i+1} le point translaté de A_{i+1} par $\overrightarrow{u}$;
– (A_i, A'_i, A'_{i+1}) un triangle inclus dans C_{libre} ;
– (A_i, A'_{i+1}, A_{i+1}) un triangle inclus dans C_{libre}.

Une région C_{AA} est délimitée par quatre côtés :
– $[A_i\ A_{i+1}]$ une face de $\mathcal{O}$;
– $[A_i\ A'_i]$;
– $[A_{i+1}\ A'_{i+1}]$;
– $[A'_i\ A'_{i+1}]$ appelé côté extérieur de C_{AA}.

Une région C_A (FIG. 3.1.b) est un secteur d'ellipse (A_i, A'_i, A''_i) dans le repère $(A_i, \overrightarrow{u}, \overrightarrow{v})$ avec :
– A_i un sommet de $\mathcal{O}$, situé dans l'angle $\widehat{p_{i-1}A_ip_{i+1}}$ convexe ;
– $\overrightarrow{n_1}$ la normale à la face $[p_{i-1}\ A_i]$;
– $\overrightarrow{n_2}$ la normale à la face $[A_i\ p_{i+1}]$;
– $\overrightarrow{u}$ colinéaire à $\overrightarrow{n_1}$;
– $\overrightarrow{v}$ colinéaire à $\overrightarrow{n_2}$;
– A'_i le point translaté de A_i par $\overrightarrow{u}$;
– A''_i le point translaté de A_i par $\overrightarrow{v}$;
– (A_i, A'_i, A''_i) un secteur d'ellipse inclus dans C_{libre}.

Ce secteur d'ellipse (A_i, A'_i, A''_i) est l'image par la fonction affine f du cercle de centre A_i et de rayon $||\overrightarrow{u}||$, avec :

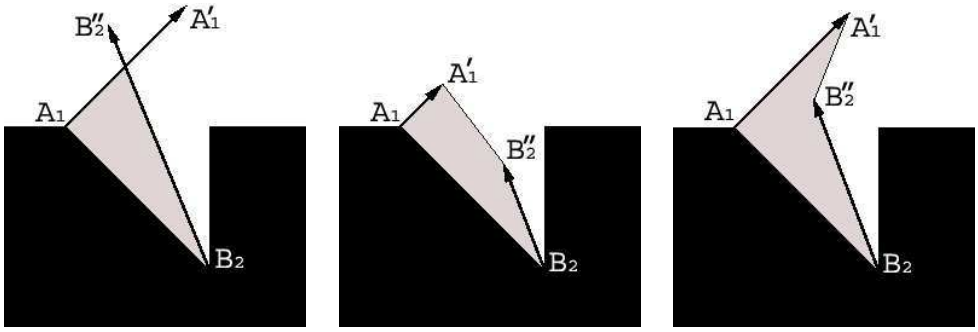


FIG. 3.2 – Régions C_{AB} à 3 et 4 côtés de C_{libre}.

Une intersection entre les segments $[A_1\ A'_1]$ et $[B_2\ B''_2]$ produit une région C_{AB} triangulaire (a). En l'absence d'intersection, une région C_{AB} est un polygone à quatre côtés, convexe (b) ou concave (c). L'éventualité d'une concavité implique la considération des deux triangles (A_i, A'_i, B''_{i+1}) et $(A_i, B''_{i+1}, B_{i+1})$.

$$x \in [0;1]\,, y \in [0;1]\,, \begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{f} \begin{bmatrix} \frac{||\overrightarrow{u}||}{||\overrightarrow{v}||}x \\ y \end{bmatrix} \;;\; ||\overrightarrow{v}|| \neq 0$$

Une région C_A (définie par le sommet A_i) est délimitée par trois côtés :
- $[A_i\ A'_i]$ un côté de secteur d'ellipse ;
- $[A_i\ A''_v]$ un côté de secteur d'ellipse ;
- $\widetilde{[A'_i\ A''_i]}$ la portion d'ellipse définie par f dans le repère $(A_i, \overrightarrow{u}, \overrightarrow{v})$; $\widetilde{[A'_i\ A''_i]}$ est appelé côté extérieur de C_A.

Une région C_A est un secteur de cercle si $||\overrightarrow{u}|| = ||\overrightarrow{v}||$.

Une région C_{AB} (FIG. 3.2) est un polygone $(A_i, B_{i+1}, A'_i, B''_{i+1})$ avec :
- $\overrightarrow{n}$ la normale à la face $[A_i\ B_{i+1}]$;
- $\overrightarrow{u}$ colinéaire à $\overrightarrow{n}$;
- $\overrightarrow{b}$ vecteur normé, colinéaire à la bissectrice de $\widehat{A_i B_{i+1} p_{i+2}}$ et orienté vers l'extérieur de $\mathcal{O}$;
- $\overrightarrow{v}$ colinéaire à $\overrightarrow{b}$;
- A'_i le point translaté de A_i par $\overrightarrow{u}$;
- B''_{i+1} le point translaté de B_{i+1} par $\overrightarrow{v}$;
- (A_i, A'_i, B''_{i+1}) un triangle inclus dans C_{libre} ;
- $(A_i, B''_{i+1}, B_{i+1})$ un triangle inclus dans C_{libre}.

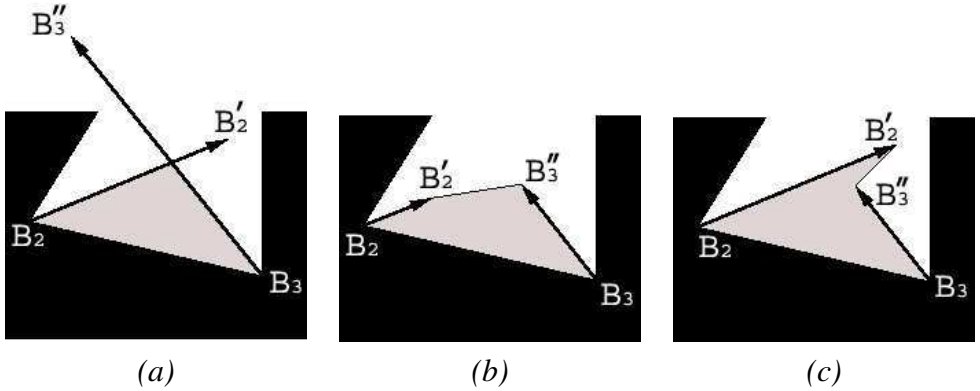
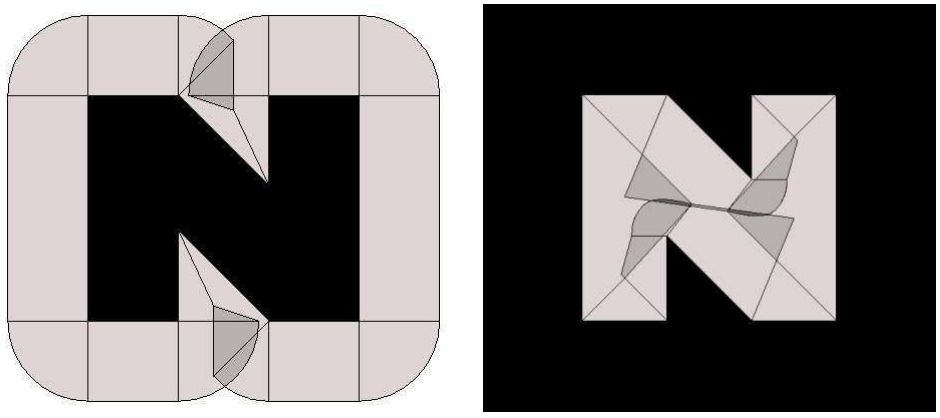


FIG. 3.3 – Régions C_{BB} à 3 et 4 côtés de C_{libre}.

Comme pour les régions C_{AB} et C_{BA}, la présence d'une intersection crée une région C_{BB} triangulaire (a). L'absence d'intersection crée une région C_{AB} convexe (b) ou concave (c).

Une région C_{AB} peut être délimitée par les côtés :
– $[A_i\ B_{i+1}]$ une face de $\mathcal{O}$;
– $[A_i\ A'_i]$;
– $[B_{i+1}\ B''_{i+1}]$;
– $[A'_i\ B''_{i+1}]$ appelé côté extérieur de C_{AB}.
Les vecteurs $\overrightarrow{u}$ et $\overrightarrow{v}$ n'étant pas nécessairement colinéaires (contrairement aux régions C_{AA}), si $[A_i\ A'_i] \cap [B_{i+1}\ B''_{i+1}] \neq \{\emptyset\}$, alors il est nécessaire de déterminer les vecteurs $\overrightarrow{u}$ et $\overrightarrow{v}$ tels que les points A'_i et B''_{i+1} soient confondus. Dans ce cas, C_{AB} est délimitée par les trois côtés $[A_i\ B_{i+1}]$, $[A_i\ A'_i]$ et $[B_{i+1}\ A'_i]$ et le triangle (A_i, A'_i, B''_{i+1}) n'est plus considéré (FIG. 3.2 *(a)*). Si les vecteurs $\overrightarrow{u}$ et $\overrightarrow{v}$ sont disjoints, les longueurs des segments $[A_i\ A'_i]$ et $[B_{i+1}\ B''_{i+1}]$ peuvent définir un polygone à 4 côtés, convexe (FIG. 3.2 *(b)*) ou concave (FIG. 3.2 *(c)*). Les régions C_{BA} sont définies par analogie aux régions C_{AB}.

Une région C_{BB} (FIG. 3.3) est un polygone $(B_i, B_{i+1}, B'_i, B''_{i+1})$ avec :
– $\overrightarrow{b_1}$ vecteur normé, colinéaire à la bissectrice de $\widehat{p_{i-1}B_iB_{i+1}}$ et orienté vers l'extérieur de $\mathcal{O}$;
– $\overrightarrow{u}$ colinéaire à $\overrightarrow{b_1}$;
– $\overrightarrow{b_2}$ vecteur normé, colinéaire à la bissectrice de $\widehat{B_iB_{i+1}p_{i+2}}$ et orienté vers l'extérieur de $\mathcal{O}$;
– $\overrightarrow{v}$ colinéaire à $\overrightarrow{b_2}$;



(a) *(b)*

FIG. 3.4 – Décomposition d'un espace ouvert et d'un espace fermé.

L'expression des espaces libres par relation aux obstacles permet la décomposition d'espaces ouverts (a) ou fermés (b).

– B'_i le point translaté de B_i par $\overrightarrow{u}$;
– B''_{i+1} le point translaté de B_{i+1} par $\overrightarrow{v}$;
– (B_i, B'_i, B''_{i+1}) un triangle inclus dans C_{libre} ;
– $(B_i, B''_{i+1}, B_{i+1})$ un triangle inclus dans C_{libre}.

Une région C_{BB} peut être délimitée par les côtés :
– $[B_i\ B_{i+1}]$ une face de $\mathcal{O}$;
– $[B_i\ B'_i]$;
– $[B_{i+1}\ B''_{i+1}]$;
– $[B'_i\ B''_{i+1}]$ appelé côté extérieur de C_{BB}.

Comme pour les région C_{AB}, les régions C_{BB} peuvent définir des polygones à 3 côtés (FIG. 3.3 *(a)*), à quatre côtés, convexes (FIG. 3.3 *(b)*) ou concaves (FIG. 3.3 *(c)*).

Les longueurs $||\overrightarrow{u}||$ et $||\overrightarrow{v}||$ sont définies par la dimension des espaces libres locaux (*i.e.* au voisinage du sommet p_i. Pour toutes les régions de C_{libre} : d_{max} est la longueur[13] maximale des côtés $[p_i\ p'_i]$ et $[p_{i+1}\ p''_{i+1}]$; $precision$ définit l'intervalle d'approximation de ces côtés.

[13]La valeur d_{max} définit la taille maximale des régions de C_{libre} et permet de décomposer les espaces ouverts (FIG. 3.4, un espace ouvert a et un espace fermé b). Pour une région de C_{libre}, si $[p_i\ p_{i+1}]$ définit « la largeur » de cette région, alors d_{max} définit « la longueur » maximale de cette région.

La forme d'une région de C_{libre} est définie par les longueurs des deux vecteurs $\overrightarrow{u}$ et $\overrightarrow{v}$. Ces longueurs sont recherchées par ajouts successifs d'une longueur d décroissante ; si la région associée est valide, la norme du vecteur considéré est incrémentée de la longueur d. Une valeur d'incrémentation d_0 (*respect.* d_1) est définie pour le vecteur $\overrightarrow{u}$ (*respect.* $\overrightarrow{v}$) ; $\overrightarrow{u_n}$ (*respect.* $\overrightarrow{v_n}$) est le vecteur $\overrightarrow{u}$ (*respect.* $\overrightarrow{v}$) normé. Les points p'_i et p''_{i+1} deviennent les translatés de p_i et p_{i+1} par $d_0\overrightarrow{u_n}$ et $d_1\overrightarrow{v_n}$. Pour équilibrer la recherche des plus grandes distances $||\overrightarrow{u}||$ et $||\overrightarrow{v}||$, la définition d'une région de C_{libre} (ALG. 22 pour les régions CA et ALG. 23 pour les régions CB) est réalisée en deux étapes :

– Une étape d'initialisation ; soit P l'ensemble des points inclus dans la région délimitée par les segments $[p_i\ p'_i]$ et $[p_{i+1}\ p''_{i+1}]$ de longueur d_{max} ; cette étape retourne la distance de l'élément le plus proche de $[p_i\ p_{i+1}]$ dans P ; pour les régions CB, cette étape retourne les distances du point le plus proche de $[p_i\ p_{i+1}]$ sur les axes définis par $\overrightarrow{u}$ et $\overrightarrow{v}$;
– Une étape d'expansion ; à chaque itération de cette étape, la plus grande distance entre d_0 et d_1 est sélectionnée ; les valeurs de d_0 et d_1 sont initialisées à la moitié de la distance ou des distances définies par l'étape d'initialisation ; la phase d'expansion recherche les plus grandes distances de d_0 et d_1 dans C_{libre}. Cette recherche est réalisée à un intervalle de précision fixé.

defCA$(p_i, p_{i+1}, \overrightarrow{u_n}, \overrightarrow{v_n}, d_{max}, precision, C)$
  $d \leftarrow$ initCA$(p_i, p_{i+1}, \overrightarrow{u_n}, \overrightarrow{v_n}, d_{max}, precision, C)$;
  retourner expCA$(p_i, p_{i+1}, \overrightarrow{u_n}, \overrightarrow{v_n}, \frac{d}{2}, d_{max}, precision, C)$;

ALG. 22: Définition des régions C_A et C_{AA} de C_{libre}.

Dans le cas des régions CB, l'absence de condition de colinéarité des vecteurs $\overrightarrow{u_n}$ et $\overrightarrow{v_n}$ implique la détection de l'intersection entre les segments $[p_i\ p'_i]$ et $[p_{i+1}\ p''_{i+1}]$. `initCB` (ALG. 23 ①) retourne un triplet (i, d_0, d_1) : une valeur de i égale à $LIMITE$ limite la définition d'une région à son étape d'initialisation. Cette limitation est la conséquence de l'inclusion dans C_{libre} des régions (p_i, p_{i+1}, p'_i) avec intersection et des régions $(p_i, p_{i+1}, p'_i, p''_{i+1})$ sans intersection.

defCB$(p_i, p_{i+1}, \overrightarrow{u_n}, \overrightarrow{v_n}, d_{max}, precision, C)$
  $(i, d_0, d_1) \leftarrow$ initCB$(p_i, p_{i+1}, \overrightarrow{u_n}, \overrightarrow{v_n}, d_{max}, precision, C)$; ①
  si $i = LIMITE$
    $p'_i \leftarrow p_i + d_0\overrightarrow{u_n}$;
    $p''_{i+1} \leftarrow p_{i+1} + d_1\overrightarrow{v_n}$;
    retourner région$(p_i, p_{i+1}, p'_i, p''_{i+1})$;
  retourner expCB$(p_i, p_{i+1}, \overrightarrow{u_n}, \overrightarrow{v_n}, \frac{d_0}{2}, \frac{d_1}{2}, d_{max}, precision, C)$;

ALG. 23: Définition des régions C_{AB}, C_{BA} et C_{BB} de C_{libre}.

L'étape d'initialisation diffère selon la nature des régions :
– Dans les régions C_A, c'est une recherche de plus proche voisin dans la région $(p_i, p_{i+1}, p_i + d_{max}\overrightarrow{u_n}, p_{i+1} + d_{max}\overrightarrow{v_n})$;
– Dans les régions C_{AA}, c'est l'élément le plus proche du segment $[A_i\ A_{i+1}]$; pour un segment $[A_i\ A_{i+1}]$, la distance au point C est la distance entre ce point et la projection orthogonale de ce point sur $(A_i\ A_{i+1})$;
– Dans les régions CB, cette distance est conditionnée par la différence angulaire des vecteurs $\overrightarrow{u}$ et $\overrightarrow{v}$. Cette étape est détaillée dans l'algorithme 24.

Les distances d_0 et d_1 retournées sont définies sur les axes des vecteurs $\overrightarrow{u}$ et $\overrightarrow{v}$. L'objectif de l'étape d'initialisation est de définir les points p'_i et p''_{i+1} les plus éloignés dans C_{libre} de p_i et p_{i+1} (et à distances respectivement égales).

La détection d'une intersection (ALG. 24 ①) entre les segments $[p_i\ p'_i]$ et $[p_{i+1}\ p''_{i+1}]$ modifie la forme initiale de la région R. Un nouveau point p, résultat de l'intersection (ALG. 24 ②), est utilisé pour définir la région recherchée ; la recherche du plus proche élément est alors limitée à la région définie par les trois points p_i, p_{i+1} et p (ALG. 24 ③). Si cette région ne contient aucun point d'un des obstacles $\mathfrak{o}$, la région recherchée est le triangle défini par les trois points p_i, p_{i+1} et p (ALG. 24 ⑦) ; si cette région contient un ou plusieurs éléments des obstacles $\mathfrak{o}$, une recherche du plus proche voisin est réalisée dans la région R (ALG. 24 ④). La distance prise en compte dans `plusProcheVoisinParProjection` est la distance entre les points de R et leur projection orthogonale sur le segment $[p_i\ p_{i+1}]$. Soit D la droite parallèle à $[p_i\ p_{i+1}]$ passant par le point le plus proche de ce segment ; D définit deux points d'intersection avec $[p_i\ p'_i]$ et avec $[p_{i+1}\ p''_{i+1}]$, dont les distances aux points p_i et p_{i+1} sont d_0 et d_1 (ALG. 24 ⑤ et ⑥). Sans intersection, l'initialisation d'une région CB crée une région R de hauteur d_{max} sur les axes définis par les vecteurs $\overrightarrow{u}$ et $\overrightarrow{v}$ (ALG. 24 ⑧). Si cette région ne contient aucun élément de $\mathcal{O}$ exceptés les éléments du segment $[p_i\ p_{i+1}]$, la région retournée résulte de décalages

de longueur d_{max} à partir des points p_i et p_{i+1} (ALG. 24 ⑨).

```
initCB(p_i, p_{i+1}, u_n, v_n, d_max, precision, C)
  p_0 ← p_i + d_max u_n ;
  p_1 ← p_{i+1} + d_max v_n ;
  si testIntersection(p_i, p_{i+1}, p_0, p_1)                                  ①
    p ← intersection(p_i, p_{i+1}, p_0, p_1) ;                                 ②
    R ← region(p_i, p_{i+1}, p) ;                                              ③
    si contientDesElementsObstacle(R) ;
      p ← plusProcheVoisinParProjectionOrth(p_i, p_{i+1}, R) ;                 ④
      p_0 ← intersection(p_i, p_0, p, p + p_i − p_{i+1}) ;
      p_1 ← intersection(p_{i+1}, p_1, p, p + p_{i+1} − p_i) ;
      d_0 ← distance(p_i, p_0) ;                                               ⑤
      d_1 ← distance(p_{i+1}, p_1) ;                                           ⑥
      retourner (NONLIMITE, d_0, d_1) ;
    d_0 ← distance(p_i, p) ;
    d_1 ← distance(p_{i+1}, p) ;
    retourner (LIMITE, d_0, d_1) ;                                             ⑦
  sinon
    R ← region(p_i, p_{i+1}, p_0, p_1) ;                                       ⑧
    si contientDesElementsObstacle(R) ;
      p ← plusProcheVoisinParProjectionOrth(p_i, p_{i+1}, R) ;
      p_0 ← intersection(p_i, p_0, p, p + p_i − p_{i+1}) ;
      p_1 ← intersection(p_{i+1}, p_1, p, p + p_{i+1} − p_i) ;
      d_0 ← distance(p_i, p_0) ;
      d_1 ← distance(p_{i+1}, p_1) ;
      retourner (NONLIMITE, d_0, d_1) ;
  retourner (LIMITE, d_max, d_max) ;                                           ⑨
```

ALG. 24: Initialisation des régions C_{AB}, C_{BA} et C_{BB} de C_{libre}.

Les points p'_i et p''_{i+1} sont, au cours de l'étape d'expansion des régions CA, initialement décalés de d_{init} (ALG. 25 ① et ②). Les distances de décalage de p'_i et p''_{i+1} sont incrémentées en dissociant les décalages sur $\overrightarrow{u}$ et $\overrightarrow{v}$. À chaque itération, le plus grand des deux décalages (ALG. 25 ④) de longueur d_0 et d_1 est sélectionné. La valeur d'incrémentation d_0 (*respect.* d_1) est divisée par deux à chaque nouvelle itération (ALG. 25 ⑤ et ⑦), les points de p'_i et p''_{i+1} sont recherchés par dichotomie, à une valeur $precision$ près (ALG. 25 ③). `estUneRegionValide` (ALG. 25 ⑥

et ⑧) vérifie la non-intersection de C_{obs} avec les régions $(p_i, p_{i+1}, p, p''_{i+1})$ et (p_i, p_{i+1}, p'_i, p). Une itération est réalisée avec succès si la région générée est valide. Après n itérations avec succès sur $\overrightarrow{u}$ (*respect.* sur $\overrightarrow{v}$), la distance entre p_i et p'_i (*respect.* entre p_{i+1} et p''_{i+1}) est située dans l'intervalle $[d_{max} - precision; d_{max}]$. Après $2n$ itérations avec échecs, la distance retournée est nulle ($p'_i = p_i$ et $p''_{i+1} = p_{i+1}$). L'expansion d'une région CA (ALG. 25) définit ses dimensions en $2\log_2(d_{max} - d_{init} - precision)$ itérations.

```
expCA(p_i, p_{i+1}, u_n, v_n, d_init, d_max, precision, C)
    d_0 ← d_max − d_init ;
    d_1 ← d_max − d_init ;
    p'_i ← p_i + d_init u_n ;                                   ①
    p''_{i+1} ← p_{i+1} + d_init v_n ;                          ②
    tant que d_0 > precision et d_1 > precision                 ③
        si d_0 > d_1                                            ④
            d_0 ← d_0/2 ;                                       ⑤
            p ← p'_i + d_0 u_n ;
                si estUneRegionValide(p_i, p_{i+1}, p, p''_{i+1}, C)   ⑥
                    p'_i ← p ;
        sinon
            d_1 ← d_1/2 ;                                       ⑦
            p ← p''_{i+1} + d_1 v_n ;
                si estUneRegionValide(p_i, p_{i+1}, p'_i, p, C)        ⑧
                    p''_{i+1} ← p ;
    retourner region(p_i, p_{i+1}, p'_i, p''_{i+1}) ;
```

ALG. 25: Expansion des régions C_{AA} et C_A de C_{libre}.

Pour les régions CB (ALG. 26), le nombre d'itérations dépend des intersections entre les segments $[p_i\ p'_i]$ et $[p_{i+1}\ p''_{i+1}]$. Les décalages des points p'_i et p''_{i+1} étant alternés, une intersection entre les segments $[p_i\ p'_i]$ et $[p_{i+1}\ p''_{i+1}]$ crée un point p, redéfinissant la forme de la région R ; le déplacement alterné de vecteurs directeurs $\overrightarrow{u}$ et $\overrightarrow{v}$ implique qu'une intersection crée une forme incluse dans la forme résultant du pas d'itération précédent ; la région, définie par les trois points p_i, p_{i+1} et p, est de ce fait retournée sans test supplémentaire (ALG. 26 ① et ②).

expCB$(p_i, p_{i+1}, \overrightarrow{u_n}, \overrightarrow{v_n}, d_{0init}, d_{1init}, d_{max}, precision, C)$
- $d_0 \leftarrow d_{max} - d_{0init}$;
- $d_1 \leftarrow d_{max} - d_{1init}$;
- $p'_i \leftarrow p_i + d_{0init}\overrightarrow{u_n}$;
- $p''_{i+1} \leftarrow p_{i+1} + d_{1init}\overrightarrow{v_n}$;
- tant que $d_0 > precision$ et $d_1 > precision$
  - si $d_0 > d_1$
    - $d_0 \leftarrow d_0/2$;
    - $p \leftarrow p'_i + d_0\overrightarrow{u_n}$;
    - si testIntersection$(p_i, p_{i+1}, p, p''_{i+1})$
      - $p \leftarrow$ intersection$(p_i, p_{i+1}, p, p''_{i+1})$;
      - retourner region(p_i, p_{i+1}, p, p) ; ①
    - si estUneRegionValide$(p_i, p_{i+1}, p, p''_{i+1}, C)$
      - $p'_i \leftarrow p$;
  - sinon
    - $d_1 \leftarrow d_1/2$;
    - $p \leftarrow p''_{i+1} + d_1\overrightarrow{v_n}$;
    - si testIntersection(p_i, p_{i+1}, p'_i, p)
      - $p \leftarrow$ intersection(p_i, p_{i+1}, p'_i, p) ;
      - retourner region(p_i, p_{i+1}, p, p, C) ; ②
    - si estUneRegionValide$(p_i, p_{i+1}, p'_i, p, C)$
      - $p''_{i+1} \leftarrow p$;
- retourner region$(p_i, p_{i+1}, p'_i, p''_{i+1})$;

ALG. 26: Expansion des régions C_{AB}, C_{BA} et C_{BB} de C_{libre}.

La valeur d_{max} fixe les valeurs maximales de chaque décalage à partir d'un point p_i d'un obstacle. Nous étudions la couverture de l'espace C_{libre}, réalisée par un ensemble de régions $\mathcal{F}$, sous deux aspects :

– La couverture par union, notée $\mathcal{CV}_U$, exprime la proportion de C_{libre} occupée par l'union des régions $\mathcal{F}$; dans le cas d'un espace $2D$, elle exprime l'union des surfaces définies par les régions $\mathcal{F}$; elle vérifie $0 \leq \mathcal{CV}_U \leq 1$;
– La couverture par addition, notée $\mathcal{CV}_A$, exprime la proportion de C_{libre} occupée par l'addition des régions $\mathcal{F}$; dans le cas d'un espace $2D$, elle exprime la somme des surfaces définies par les régions $\mathcal{F}$; l'addition de régions $\mathcal{F}$ non disjointes peut créer une valeur de $\mathcal{CV}_A$ supérieure à la taille de C_{libre}.

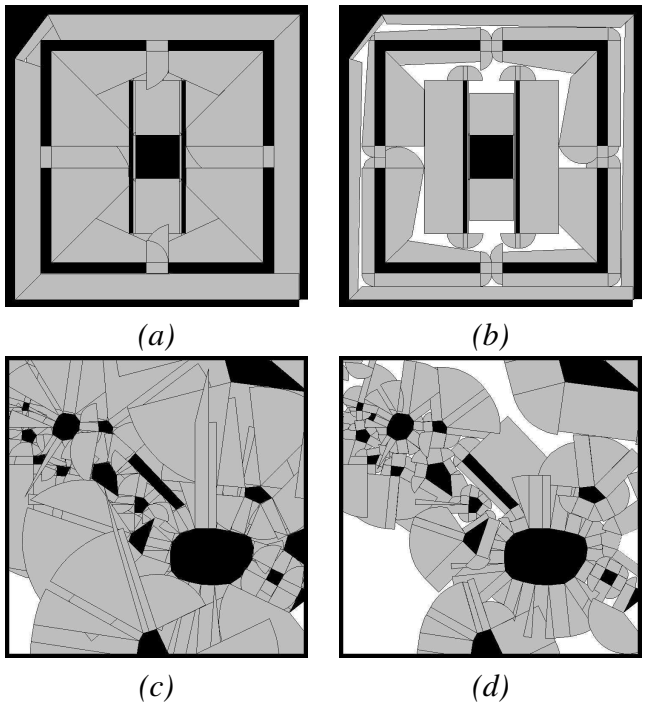


FIG. 3.5 – Deux stratégies de recouvrement de C_{libre}.

Minimiser les vides en (a) et (c) ; minimiser les intersections en (b) et (d).

Un vide est une région de C_{libre} exclue de toutes les régions de $\mathcal{F}$. Pour construire un recouvrement de C_{libre}, nous proposons deux stratégies influant sur la taille des régions de $\mathcal{F}$:

– Pour minimiser les vides (FIG. 3.5 *(a)* et *(c)*), les côtés $[p'_i\ p''_{i+1}]$ de ces régions sont placés à proximité des faces des obstacles opposés. Cette décomposition est obtenue en fixant la valeur d_{max} à la plus grande distance entre deux obstacles de C. Cette décomposition peut produire des valeurs de $\mathcal{CV}_A$ supérieures à 1 ;
– Pour minimiser les intersections entre les régions de $\mathcal{F}$ (FIG. 3.5 *(b)* et *(d)*), les côtés $[p'_i\ p''_{i+1}]$ de ces régions sont placés à proximité de l'axe médian de C_{libre} ; soit m_i le milieu du segment $[p_i\ p'_i]$ et m_{i+1} le milieu du segment $[p_{i+1}\ p''_{i+1}]$; la position de l'axe médian est estimée à proximité du segment $[m_i\ m_{i+1}]$; les quadruplets $(p_i, p_{i+1}, p'_i, p''_{i+1})$, définissant précédemment les régions de $\mathcal{F}$, sont substitués par $(p_i, p_{i+1}, m_i, m_{i+1})$; Cette décomposition diminue les chances d'obtenir des valeurs de $\mathcal{CV}_U$ supérieures à 1.

3.3.2 Échantillonnage approché dans les régions de C_{libre}

Échantillonnage uniforme

La génération de tirage aléatoire uniforme dans les régions de $\mathcal{F}$ est réalisée par relation au barycentre des triangles [Tur90]. Les régions de $\mathcal{F}$ sont discrétisées en triangles :

- Une région C_{AA} est définie par les deux triangles (A_i, A'_i, A'_{i+1}) et (A_i, A'_{i+1}, A_{i+1}) ;
- Une région C_A est discrétisée en triangles à un intervalle angulaire régulier ; un secteur d'ellipse d'angle θ est approximé par une succession de n triangles d'angle α, complétée par un triangle d'angle $(\theta - n\alpha)$;
- Une région CB est définie par un ou deux triangles ; avec une intersection dans C_{libre} des segments $[p_i\ p'_i]$ et $[p_{i+1}\ p''_{i+1}]$, une région CB est délimitée par le triangle (p_i, p'_i, p_{i+1}) ; sans intersection, elle peut être concave ou convexe ; la définition d'une région CB concave dépend de la position de la concavité (*i.e.* de la position du sommet B) ; une région C_{AB} concave est définie par les deux triangles (A_i, A'_i, B''_{i+1}) et (A_i, B''_{i+1}, B_i) ; une région C_{BA} concave est définie par les deux triangles (B_i, B'_i, A_{i+1}) et $(A_{i+1}, B'_i, A''_{i+1})$; une région CB convexe est définie par les deux triangles (p_i, p'_i, p_{i+1}) et (p_i, p'_i, p''_{i+1}).

Pour l'ensemble des régions de $\mathcal{F}$, l'espace de recherche est échantillonné en deux étapes :

- Une première distribution aléatoire sélectionne une région ;
- Une seconde distribution aléatoire sélectionne une position de cette région.

Échantillonnage gaussien

Pour diminuer la distance à l'axe médian, une déformation gaussienne est appliquée aux tirages aléatoires. Cette déformation est définie selon un axe de translation, par la loi normale gaussienne centrée réduite, d'espérance 0 et d'écart-type 1, avec :

$$f(x, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

La loi f est utilisée pour définir une fonction de répartition[14] dans les régions de $\mathcal{F}$, avec un décalage initial f_i, une échelle d'application f_e et une densité initiale f_d : le décalage initial f_i repousse la fonction f vers l'axe médian[15] ; l'échelle f_e

[14] En termes de statistiques, 68% des tirages sont dans l'intervalle $[-1; 1]$, 95% des tirages sont dans l'intervalle $[-2; 2]$ et 99% des tirages sont dans l'intervalle $[-3; 3]$.

[15] Une région de $\mathcal{F}$ résulte de l'expression d'un sommet ou d'une face de $\mathcal{O}$. La configuration q obtenue par tirage aléatoire uniforme dans cette région est l'expression d'un point p de $\mathcal{O}$ dont

modifie la largeur[16] de la fonction f ; la densité f_i ajoute une densité minimale[17] dans toute la région de $\mathcal{F}$.

Les figures 3.6 *(a-c)*, 3.7 *(a-c)* et 3.8 *(a-l)* présentent les régions de $\mathcal{R}$ dans un espace ouvert (FIG. 3.6 *(a)*) et dans un espace fermé (FIG. 3.7 *(a)*) pour un mobile ponctuel. Ces figures détaillent les régions C_{AA} en 3.8 *(a)*, les régions C_A en *(d)*, les régions CB en *(g)* et la superposition de toutes ces régions en *(j)*, dont les échantillonnages uniformes puis gaussiens sont présentés en *(b)* et *(c)*, *(e)* et *(f)*, *(h)* et *(i)*, *(k)* et *(l)*. La décomposition des régions CB est dépourvue de région C_{BB}. Le nombre de tirages aléatoires dans les régions de $\mathcal{R}$ varie pour des raisons de clarté. Les valeurs de d_{max} et $precision$ définissent le nombre d'itérations nécessaires à la définition de chaque région de $\mathcal{R}$; les résultats sont obtenus à partir de valeurs extrêmes de d_{max} et $precision$: d_{max} est égale à la plus grande variation possible des positions sur les axes dans l'espace de travail (*i.e.* 2800 pour les figures 3.8 et 1400 pour les figures 3.9) et $precision = 1$. Les données relatives à l'analyse de ces décompositions sont présentées dans le tableau 3.1. Le temps d'échantillonnage de ces régions est comparé à un échantillonnage uniforme de C avec détection de collision (FIG. 3.6 *(b)*) et sans détection de collision (FIG. 3.6 *(c)*). La décomposition des régions de $\mathcal{R}$ choisie minimise les vides (§ 3.3.1). Les colonnes $\mathcal{CV}_U$ et $\mathcal{CV}_A$ donnent la couverture de C_{libre} ; les colonnes const., uniforme et gaussien sont en secondes[18] : const. est le temps de construction des régions, uniforme est le temps nécessaire pour obtenir des tirages aléatoires uniformes dans ces régions, gaussien est le temps nécessaire pour obtenir des tirages aléatoires gaussiens dans ces régions. Les temps de construction des régions de $\mathcal{R}$ sont inclus dans les temps d'obtention des tirages aléatoires.

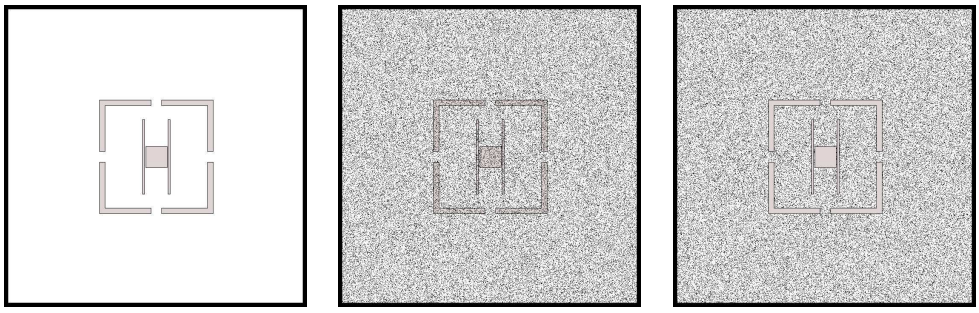
---

la normale passe par q. f_i est la distance entre p et q. Une valeur de f_i supérieure à la hauteur du segment libre au point p définie par la normale en 0 annihile l'expression de cette région de $\mathcal{F}$. La variation f_i correspond à l'espérance de la distribution. Elle est retirée de l'expression de f pour permettre le pré-calcul des variations de f. Le pas de discrétisation de ce pré-calcul est supérieur à la valeur d_{max}. Cette déformation des tirages aléatoires uniformes sur l'axe des normales des obstacles suggère une subdivision des régions selon ces normales, divisant les régions C_{AB} en trois triangles et les régions C_{BB} en quatre triangles. Cette décomposition optimise les calculs de déformation dûs à la fonction f.

[16] f_e est l'ensemble de définition des valeurs de x. f_e est l'écart-type de la distribution. À l'instar de l'espérance, son expression en dehors de f est justifiée par le pré-calcul de f. En fonction du mode de décomposition choisi (*i.e.* minimiser les vides ou les intersections), f est pré-calculée sur les intervalles $[-4; 4]$ ou $[-4; 0]$.

[17] La densité minimale f_i permet d'utiliser une gaussienne d'écart-type faible. Un écart-type faible implique une forte contrainte de localisation des tirages aléatoires autour de l'espérance. Cette contrainte est relâchée par la densité f_i.

[18] Ces temps de calcul sont réalisés avec un processeur Intel Pentium4 2.66Ghz avec 512MO DDR SDRAM sous Linux (5295 bogomips).



(a) *(b)* *(c)*

FIG. 3.6 – Échantillonnage aléatoire uniforme de C dans un espace ouvert.

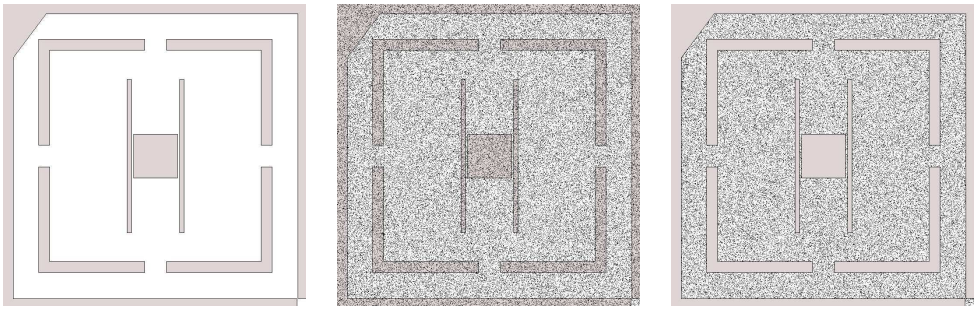
Les obstacles en (a), tirages aléatoires sans détection de collision en (b), tirages aléatoires avec détection de collision en (c). La définition de l'intervalle des tirages aléatoires présuppose une solution.

L'échantillonnage aléatoire des figures 3.6 *(b)* et *(c)* est réalisé dans l'intervalle défini par la fenêtre de visualisation, soit dans l'intervalle $[-2800; 2800]$ sur deux axes. L'espace occupé par les obstacles, dans les figures 3.6 *(a-c)* et 3.8 *(a-l)*, représente 3.6% de la surface définie par ces deux intervalles.

L'échantillonnage aléatoire des figures 3.7 *(b)* et *(c)* est réalisé dans l'intervalle défini par la fenêtre de visualisation, soit dans l'intervalle $[-1400; 1400]$ sur deux axes. L'espace occupé par les obstacles, dans les figures 3.7 *(a-c)* et 3.9 *(a-l)*, représente 27.7% de la surface définie par ces deux intervalles.

La comparaison des échantillonnages relatifs aux espaces définis dans les figures 3.6 *(a)* et 3.7 *(a)* montre :

– L'influence supra-linéaire de l'encombrement sur un échantillonnage uniforme classique de C. L'échantillonnage uniforme de C génère des tirages inclus dans C_{obs}. La liaison de cet échantillonnage à une détection de collision permet de restreindre les configurations générées dans C_{libre} ; cette solution d'échantillonnage de C_{libre} est fortement dépendante à la proportion de C occupée par les obstacles ; le temps d'échantillonnage de C_{libre} passe de $0.40s$. (FIG. 3.6 *(c)*, espace occupé à 3.6%) à $1.22s$. (FIG. 3.7 *(c)*, espace occupé à 27.7%). Les obstacles sont englobés dans une hiérarchie de formes *A*ABB pour la détection de



(a) *(b)* *(c)*

FIG. 3.7 – Échantillonnage aléatoire uniforme de C dans un espace fermé.

Les obstacles en (a), tirages aléatoires sans détection de collision en (b), tirages aléatoires avec détection de collision en (c). La définition de l'intervalle des tirages aléatoires est défini par les limites de l'espace de travail.

TAB. 3.1 – échantillonnage dans un espace ouvert et dans un espace fermé.

FIG.	régions	$\mathcal{CV}_U$	$\mathcal{CV}_A$	nb tirages	const.	uniforme	gaussien
3.6 *(b)*				100k		0.03	
3.6 *(c)*				100k		0.40	
3.8 *(a)*	C_{AA}	0.34	0.34	50k	0.05	0.18	0.33
3.8 *(d)*	C_A	0.18	0.19	50k	0.11	0.29	0.42
3.8 *(g)*	CB	0.07	0.07	50k	0.04	0.09	0.27
3.8 *(j)*	$\mathcal{R}$	0.51	0.60	100k	0.15	0.65	0.92
3.7 *(b)*				100k		0.03	
3.7 *(c)*				100k		1.22	
3.9 *(a)*	C_{AA}	0.58	0.85	50k	0.05	0.17	0.33
3.9 *(d)*	C_A	0.22	0.27	50k	0.14	0.32	0.45
3.9 *(g)*	CB	0.79	0.80	50k	0.05	0.10	0.30
3.9 *(j)*	$\mathcal{R}$	1.00	1.87	100k	0.19	0.82	1.14

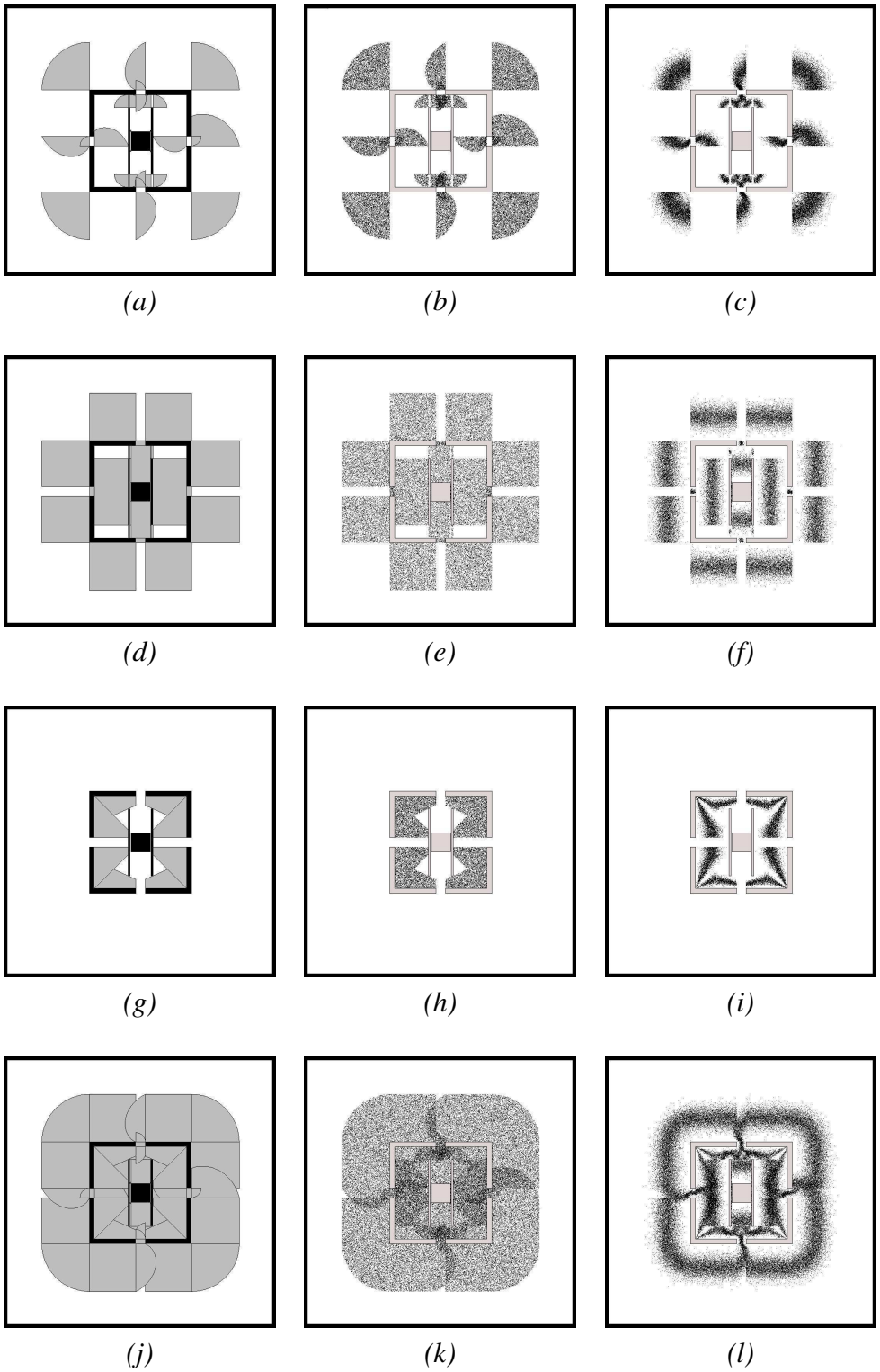


FIG. 3.8 – Régions de $\mathcal{R}$ et échantillonnage aléatoire dans un espace ouvert.

Les régions C_A en (a), C_{AA} en (d), CB en (g) et leur superposition en (j).

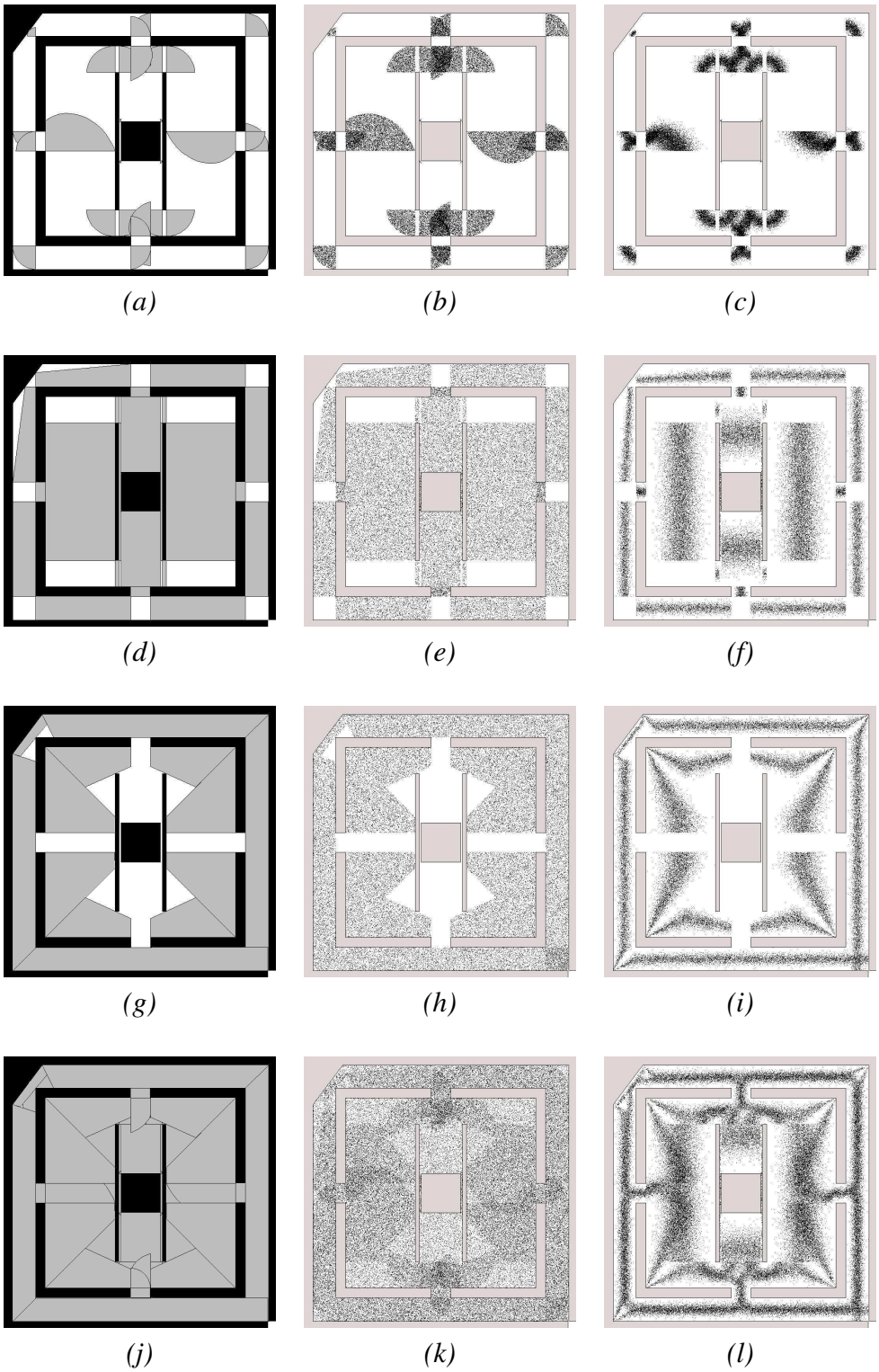


FIG. 3.9 – Régions de $\mathcal{R}$ et échantillonnage aléatoire dans un espace fermé.

Les régions C_A en (a), C_{AA} en (d), CB en (g) et leur superposition en (j).

collision. Si un tirage aléatoire est dans une forme *AABB*, la détection de collision nécessite un approfondissement. La relation entre la surface occupée par les obstacles et le nombre de tirages aléatoires implique une augmentation plus que linéaire du nombre de recherches approfondies par rapport à l'augmentation de l'occupation de l'espace ;
– L'indépendance relative de l'encombrement de l'espace et de la construction des régions de $\mathcal{R}$; pour une augmentation de l'encombrement de l'espace de 3.6% à 27.7%, le temps de construction de toutes les régions de $\mathcal{R}$ passe de $0.15s.$ à $0.19s.$. La décomposition des régions de $\mathcal{R}$ est fonction du nombre de sommets définissant les obstacles et des interférences entre les domaines de visibilité associés à ces sommets. La complexité du problème étant exponentielle, la proposition d'heuristiques est impérative. L'augmentation de l'occupation de l'espace de 3.6% à 27.7% est en fait une augmentation du nombre de sommets décrivant les obstacles, de 36 à 42 ; soit une augmentation de 116% du nombre de sommets pour une augmentation du 126% du temps de calcul des régions de $\mathcal{R}$. La phase d'expansion des régions CB étant conditionnée par la dimension des espaces libres, la diminution des espaces libres a donc une conséquence sur le temps de définition des régions de $\mathcal{R}$.

Dans un espace fermé (FIG. 3.7 *(a)*), la couverture par addition de l'ensemble des régions de $\mathcal{R}$ est quasiment le double de la couverture par union. Dans un espace ouvert (FIG. 3.6 *(a)*), les régions de $\mathcal{R}$ permettent de restreindre l'espace associé à la solution. Cette restriction permet de diminuer la taille de l'espace de recherche.

3.4 Sur la visibilité et la progression dans C

Pour réduire l'expression des espaces libres de C, l'ensemble $\mathcal{F}$ est construit incrémentalement par union des domaines de visibilité des nœuds de G. La notion de visibilité est définie par l'absence d'intersection le long de la trajectoire associée à une méthode locale $\mathcal{L}$. Pour une trajectoire définie par une suite de configurations, les deux configurations situées aux extrémités de la trajectoire sont dites visibles si toutes les configurations intermédiaires sont incluses dans C_{libre}. Le domaine de visibilité d'une configuration est défini par l'ensemble des configurations visibles par $\mathcal{L}$. La méthode locale choisie est la ligne droite dans C. Nous définissons les modalités d'ajout et de suppression des régions de $\mathcal{F}$.

3.4.1 Ajout sous contrainte de visibilité

L'algorithme de construction du graphe G génère itérativement des nouvelles régions de $\mathcal{F}$. Nous définissons le domaine de visibilité (également appelé domaine d'accessibilité) d'une configuration $q(x, y, \theta)$, pour une méthode locale $\mathcal{L}$, par l'ensemble des configurations accessibles dans un intervalle de tranches $[\theta' - v\Delta\theta; \theta' + v\Delta\theta]$; $\Delta\theta = 2\pi/m$ définissant la variation angulaire entre deux configurations, l'espace des configurations C est approximé à $\Delta\theta$ près en une succession de m tranches ; cette discrétisation de C implique la sélection de θ', la valeur angulaire la plus proche de θ avec $\theta' = k\Delta\theta$ tel que $k\Delta t \leq \theta < (k+1)\Delta\theta$, $k \in$ IN*. Pour $v = 0$, les régions visibles d'une configuration q d'angle θ sont les configurations visibles de q dans la tranche d'angle θ'. Pour $v = 1$, les régions visibles de q sont les configurations visibles de q dans les trois tranches d'angles $\theta' - \Delta\theta$, θ' et $\theta' + \Delta\theta$. Les valeurs possibles de v sont comprises entre 0 et $m/2$. L'ensemble des tranches de C définies par $\Delta\theta$ n'étant pas pré-calculé, chacun des sommets et des côtés de $\mathcal{O}$ est associé au moins à un vecteur à m composantes booléennes ; pour un élément e de ce vecteur, $\mathcal{R}_e$ est la région de $\mathcal{R}$ associée ; e est $VRAI$ si $\mathcal{R}_e$ est déjà calculée et est $FAUX$ dans le cas contraire. L'ensemble des obstacles $\mathcal{O}$, composé de n sommets, est alors associé à une matrice EO de dimension minimale $2n \times m$. $\mathcal{V}$ est l'ensemble des régions de C_{libre} appartenant au domaine de visibilité de G. EO maintient la liste des régions de C_{libre} exprimées dans $\mathcal{V}$.

```
consVisiObs(q_init, q_obj, k, v, Δt, Δθ, M, O)
  | EO ← expressionDesObstacles(Δθ, O) ;                                  ①
  | (V, EO) ← nouvellesRegionsVisibles(q_init, v, ∅, EO, M, O) ;          ②
  | pour i ← 1 à k
      q_rand ← confAleatoire(V) ;                                         ③
      q_prox ← confLaPlusProche(q_rand, G) ;
      q_new ← nouvelleConf(q_prox, q_rand, Δt) ;
      si ConfArcSansCollision(q_prox, q_new, M, O)
        | ajouterConfArc(q_prox, q_new, G) ;
        | (V, EO) ← nouvellesRegionsVisibles(q_new, v, V, EO, M, O) ;     ④
  | retourner G ;
```

ALG. 27: Construction du graphe sous condition de visibilité.

La construction des régions de $\mathcal{R}$ est initialisée par la définition de la taille de

matrice EO (ALG. 27 ①). La construction du graphe G est guidée par $\mathcal{V}$. Partant de la configuration initiale q_{init}, `nouvellesRegionsVisibles` modifie EO et ajoute les régions visibles à partir de q_{init} dans $\mathcal{V}$ (ALG. 27 ②). À chaque itération, une nouvelle configuration est choisie aléatoirement dans $\mathcal{V}$ (ALG. 27 ③). Chaque nouvelle configuration reliée avec succès à une configuration q_{prox} de G (ALG. 27 ④) suscite la mise à jour du couple $(\mathcal{V}, EO)$.

Cette variation implique la définition de v. Assimilable à une profondeur de visibilité dans les tranches[19] de C, cette valeur permet de diminuer le temps de calcul des régions visibles d'une configuration. La distance de la ligne droite dans C associée à la méthode locale $\mathcal{L}$ est également un critère limitant le calcul des régions de $\mathcal{R}$.

3.4.2 Déviation des expansions vers l'objectif

Pour augmenter la convergence du graphe vers l'objectif, il est possible de dévier les expansions vers la configuration objectif q_{obj}. Cette déviation des expansions en direction de q_{obj} est réalisée par substitution des configurations aléatoires q_{rand} des régions de $\mathcal{R}$. Nous proposons deux[20] stratégies de substitution des configurations aléatoires q_{rand} :

– La déviation est activée et n'est maintenue que sous réserve de visibilité (ALG. 29 `echantConditionne_01`);
– La déviation est activée dès la première visibilité (ALG. 29 `echantConditionne_02`).

L'ajout de stratégies substitue la sélection d'une configuration aléatoire q_{rand} de $\mathcal{V}$ par `echantConditionne_xx` (ALG. 29 ①). Les paramètres utilisés pour définir ces stratégies sont : la dernière configuration q_{new} ajoutée dans le graphe G, la configuration objectif q_{obj}, l'indice i de l'itération en cours, la fréquence f de déviation des expansions, l'ensemble des régions $\mathcal{V}$ visibles de $\mathcal{R}$.

[19] Dans le cadre d'une décomposition géométrique en tranches de C, l'existence d'une solution est fonction de la valeur de $\Delta\theta$ (*i.e.* du nombre de tranches de C) ; Dans le cadre des régions de $\mathcal{R}$, le nombre de tranches définit l'expression des régions de $\mathcal{R}$ indépendamment de l'existence d'une solution. Le nombre de tranches définit également le temps de calcul nécessaire à la définition des régions de $\mathcal{R}$ dans toutes les tranches de C. Pour conserver un temps de résolution similaire pour toutes les résolutions (*i.e.* avec ou sans calcul des régions de $\mathcal{R}$), le nombre de tranches de C utilisé pour les régions de $\mathcal{R}$ est fixé à 10.

[20] Il est possible de réaliser de nombreuses variations des expansions en fonction des résultats des expansions, de la localisation des tirages aléatoires, des relations entre tirages aléatoires et plus proches voisins dans G.

```
consGDomVisi(q_init, q_obj, k, v, f, Δt, Δθ, M, O)
  EO ← expressionDesObstacles(Δθ, O) ;
  (V, EO) ← nouvellesRegionsVisibles(q_init, v, ∅, EO, M, O) ;
  pour i ← 1 à k
    q_rand ← echantConditionne_xx(q_new, q_obj, i, f, V) ;                ①
    q_prox ← confLaPlusProche(q_rand, G) ;
    q_new ← nouvelleConf(q_prox, q_rand, Δt) ;
    si ConfArcSansCollision(q_prox, q_new, M, O)
      ajouterConfArc(q_prox, q_new, G) ;
      (V, EO) ← nouvellesRegionsVisibles(q_new, v, V, EO, M, O) ;
  retourner G ;
```

ALG. 28: Construction du graphe G associée à une stratégie d'échantillonnage de C.

```
echantConditionne_01(q_new, q_obj, i, f, V)
  si sontVisibles(q_new, q_obj)                                           ①
    si i ≡ O[f]                                                           ②
      q_rand ← q_obj ;                                                    ③
  sinon
    q_rand ← confAleatoire(V) ;                                           ④
  retourner q_rand ;
```

```
echantConditionne_02(q_new, q_obj, i, f, V)
  si dev = PASVU                                                          ⑤
    si sontVisibles(q_new, q_obj)
      dev ← VU                                                            ⑥
  si dev = VU                                                             ⑦
    si i ≡ O[f]
      q_rand ← q_obj ;                                                    ⑧
  sinon
    q_rand ← confAleatoire(V) ;                                           ⑨
  retourner q_rand ;
```

ALG. 29: Deux stratégies de déviation des expansions.

À chaque appel de `echantConditionne_01`, la visibilité entre les deux configurations q_{new} et q_{obj} est testée (ALG. 29 ①). Si les deux configurations sont visibles (*i.e.* si q_{new} est dans le domaine de visibilité[21] de q_{obj}), q_{rand} est remplacée par q_{obj} (ALG. 29 ③) toutes les f itérations (ALG. 29 ②). Cette prise en compte de la fréquence de déviation implique une déviation effective inférieure à la valeur f. Dans le cas contraire, q_{rand} est une configuration aléatoire d'une distribution aléatoire (uniforme ou gaussien) dans $\mathcal{V}$ (ALG. 29 ④).

À chaque appel de `echantConditionne_02`, la variable booléenne dev est consultée (ALG. 29 ⑤) ; la variable dev est affectée à la valeur $PASVU$ à l'initialisation d'une résolution d'un problème ; pour une valeur $PASVU$, la déviation vers l'objectif n'est pas active et n'a jamais été activée. Dans ce cas, la visibilité des configurations q_{new} q_{obj} change sa valeur à VU (ALG. 29 ⑥), et la visibilité des configurations q_{new} q_{obj} ne sera plus testée. Dans le cas d'une déviation active (ALG. 29 ⑦), q_{obj} se substitue à q_{rand} toutes les f itérations(ALG. 29 ⑧). Dans le cas contraire, q_{rand} reste une configuration aléatoire de $\mathcal{V}$ (ALG. 29 ⑨).

3.5 Résultats

Pour évaluer les algorithmes décrits ci-dessus, nous présentons les chemins calculés à l'aide d'une décomposition non uniforme de C_{libre}. Les résultats suivants sont présentés dans deux types d'environnement $2D$:
– Un passage étroit ;
– Un champ d'obstacles.

Les déplacements du mobile $\mathcal{M}$ utilisé sont définis par un modèle kinodynamique $5D$ [CL01] muni d'un ensemble U composé de 6 commandes, associé à la méthode de résolution numérique *Runge-Kutta* d'ordre 4. L'ensemble des commandes U utilisé déplace le mobile $\mathcal{M}$ en marche avant et en marche arrière : l'angle de braquage est défini par 3 commandes du type U_1 (§ 2.6.3), à vitesse constante en marche avant ou arrière.

[21] Le calcul du domaine de visibilité de q_{obj} est ainsi effectué une seule fois, voire pré-calculable. Dans notre implémentation, il est calculé au premier appel et réutilisé par la suite. Le domaine de visibilité de q_{obj} reste valide tant que la méthode locale considérée $\mathcal{L}$ reste inchangée.

Les obstacles sont définis par des polygones concaves ou convexes. La détection de collision utilise un partitionnement de l'espace en *quadtree*, associée à une détection de collision approximative en volume *AABB*. La détection de collision exacte est basée sur une somme de Minkowsky suivie d'une décomposition en polygones convexes. Les chemins de l'espace des configurations sont définis par le calcul approximatif du volume balayé par $\mathcal{M}$ sous forme de segments de droite dans l'espace des configurations C ; À chaque déplacement, la liste des obstacles, susceptibles d'être en collision avec le mobile de configuration q, est utilisée pour calculer les C-obstacles desquels q doit être exclue. L'espace des configurations n'est pas maintenu en mémoire. La représentation des régions de $\mathcal{F}$ est associée à une décomposition de C_{libre} en tranches. Lors de l'expansion d'une configuration q_{prox} d'angle θ_1, la somme de Minkowsky de $\mathcal{M}$ avec les obstacles visibles de q_{prox} est utilisée pour définir les régions de $\mathcal{F}$. Cette méthode permet d'associer une détection de collision exacte avec un calcul approximatif des régions de $\mathcal{F}$.

La distance métrique, utilisée pour l'évaluation des configurations $q_1(x_1, y_1, \theta_1)$ et $q_2(x_2, y_2, \theta_2)$, est définie par :

$$d(q_1, q_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + rm^2(\theta_1 - \theta_2)^2}$$

avec rm la variation maximale de x et y dans W. La stratégie de planification locale utilisée est la ligne droite dans C. L'échantillonnage uniforme de C est abrégé ***CR*** ; ***GB*** utilise une déviation permanente vers q_{obj} (la fréquence de déviation choisie est de 1 tirage sur 10) ; ***VGB*** utilise une déviation activée et maintenue sous réserve de visibilité (ALG. 29 `echantConditionne_01`). ***BG*** est un échantillonnage gaussien aux obstacles [BOS01] (§ 3.1.2) ; ***LR*** est l'échantillonnage défini par les régions de $\mathcal{R}$ (***Uni*** pour un échantillonnage uniforme de C_{libre} et ***Gauss*** pour un échantillonnage Gaussien de C_{libre} sur l'intervalle $[-4; 4]$).

Les résultats sont donnés en moyenne pour une série de 400 instances de problème (*i.e.* requêtes de planification entre deux positions de C). Chacun des problèmes est résolu à partir d'une initialisation identique des générateurs aléatoires ; un problème est défini par un environnement, une valeur d'initialisation des générateurs aléatoires, un couple de configurations (q_{init}, q_{obj}). Les conditions sur les positions de q_{init} et q_{obj} diffèrent dans le cas du passage étroit et dans le cas du champ d'obstacles.

3.5.1 Dans un passage étroit

Les configurations initiale et finale sont choisies aléatoirement de part et d'autre du passage étroit ; les configurations q_{init} et q_{obj} sont placées à une distance d, à Δd près, dans C_{libre}. Par décomposition d'un chemin de C_{libre} reliant q_{init} et q_{obj} avec une méthode locale $\mathcal{L}$ (la ligne droite dans C_{libre}), la distance entre ces deux configurations est définie par :

$$d - \Delta d \leq d_{\mathcal{L}}(q_{init}, q_{obj}) \leq d + \Delta d$$

Partant de la configuration q_{init} choisie aléatoirement dans C_{libre}, la configuration q_{obj} est placée par construction aléatoire d'un chemin holonome dans C_{libre}. Cette connaissance approximative (à Δd près) de la distance séparant les deux configurations q_{init} et q_{obj} nous permet de supposer qu'il existe une solution en $2d\pi$ itérations[22] . Les temps de résolution sont inférieurs à la seconde pour toutes les variantes du tableau 3.2 ; La recherche de la solution est bornée à 2500 itérations.

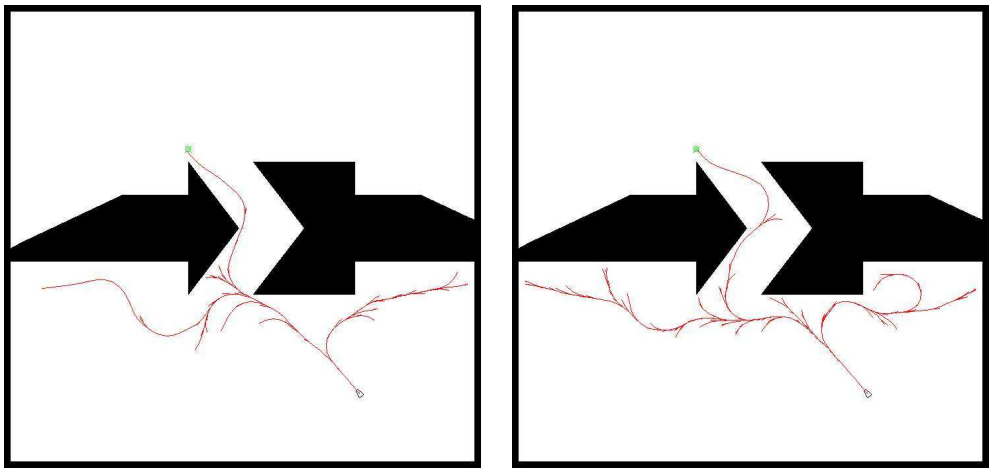
TAB. 3.2 – Résultats pour la traversée d'un passage étroit.

	succès (cfgs de G)	échec(cfgs de G)	solution
$CR\,GB_{10}$	0.04 (1163)	0.96 (1585)	79
$CR\,VGB_{10}$	0.21 (1713)	0.79 (2422)	65
$BG\,GB_{10}$	0.02 (600)	0.98 (485)	64
$BG\,VGB_{10}$	0.01 (628)	0.99 (332)	62
$LR\,Uni\,GB_{10}$	0.21 (1411)	0.79 (1650)	83
$LR\,Uni\,VGB_{10}$	0.47 (1792)	0.53 (2500)	69
$LR\,Gauss\,GB_{10}$	0.29 (1141)	0.71 (1654)	93
$LR\,Gauss\,VGB_{10}$	0.61 (1749)	0.39 (2514)	67

Les résultats, présentés dans le tableau 3.2, mettent en évidence l'influence de la notion de visibilité, appliquée au mobile et aux espaces libres de C. Ce tableau compare l'expansion de G associée aux différentes variantes ; le nombre de configurations de G est présenté en cas d'échec et en cas de succès ; la moyenne du nombre de nœuds séparant q_{init} de q_{obj} est calculée sur les succès.

La déviation des tirages associée à la visibilité de q_{obj} augmente le pourcentage de succès de la méthode ***RRT*** classique (noté CR) et de notre méthode associée à

[22]Le chemin solution estimé est ici approximé par une succession de segments de droite et d'arcs de cercle. Dans un espace sans obstacle, deux configurations distantes de d sont reliées par un chemin de longueur d par une ligne droite (annexe B) et de longueur inférieure à $2d\pi$ par une succession d'arcs de cercle.



(a) *(b)*

FIG. 3.10 – Résultats dans un problème de passage étroit.

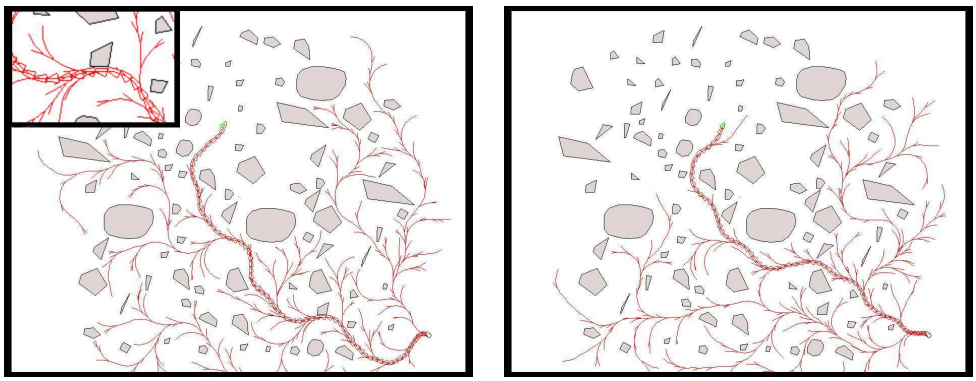
L'échantillonnage des régions de $\mathcal{R}$ implique une meilleure expansion de G dans C ; L'échantillonnage est uniforme en (a) et gaussien en (b). L'échantillonage gaussien diminue l'influence des obstacles sur l'expansion de G dans C.

la restriction des tirages dans les régions de $\mathcal{R}$; Le mobile tend à se diriger vers l'objectif si q est dans son domaine de visibilité. La visibilité augmente également le nombre de configurations ajoutées avec succès dans G. La variante BR initialement proposée pour les méthodes *PRM* construit un graphe contenant moins de configurations comparé aux autres variantes ; le parcours de l'environnement à proximité des obstacles permet de trouver des trajectoires plus courtes. Le calcul des régions de $\mathcal{R}$ (*i.e.* $LR\ Uni$) est sans conséquence sur le temps total de résolution ; les régions de $\mathcal{R}$ impliquent un plus grand nombre de configurations générées dans C_{libre}. Les régions de $\mathcal{R}$ permettent de restreindre les expansions en direction de configurations libres. L'ajout d'une déformation gaussienne augmente le nombre des configurations dans C_{libre}, augmentant également le nombre de configurations dans G. Cette déformation favorise cependant les expansions au centre des espaces libres. La déformation gaussienne diminue l'influence des obstacles sur la progression de G dans C ; la visibilité permet d'adapter la déviation des tirages aléatoires vers q_{obj}. Des exemples de solutions sont présentés en annexe B.

3.5.2 Dans un champ d'obstacles

La configuration initiale et la configuration finale sont placées aléatoirement dans C_{libre} pour éviter toute influence de la répartition des tirages aléatoires sur le succès des différentes résolutions. À chaque expansion, la distance métrique oriente les nouvelles configurations de G en direction de q_{obj} ; il existe de ce fait une déviation en direction de q_{obj} inhérente à la construction de G. Pour évaluer l'influence des déviations vers l'objectif, l'algorithme classique (CR, *i.e.* sans déviation des tirages vers l'objectif) est ajouté dans le tableau 3.3. Le champ d'obstacles est composé uniquement d'obstacles convexes (FIG. 3.11). C est un espace ouvert ; les régions de $\mathcal{R}$ délimitent l'espace des tirages aléatoires ; l'échantillonnage BR est défini par les obstacles ; les variantes CR réalisent un échantillonnage dans la fenêtre de visualisation. La recherche est bornée à 5000 itérations.

Les résultats du tableau 3.3 mettent en évidence l'influence de la visibilité dans un environnement de grande taille et composé d'obstacles épars. Ce tableau compare les pourcentages de succès des différentes variantes et les positions des expansions : la colonne *cfgs de* C_{libre} dénombre les expansions réalisées en direction d'une configuration de C_{libre} ; les expansions impliquant une inévitable collision sont dénombrées dans la colonne *cfgs de* C_{obs}. La colonne *cfgs de* G dénombre le total des configurations ; ce total est le nombre moyen de configurations associées à une résolution couronnée d'un succès.



(a) *(b)*

FIG. 3.11 – Résultats dans un champ d'obstacles convexes.

Pour un couple de configurations $(q_{init}, q_{obj}$*, l'échantillonnage gaussien en (b) des régions de* $\mathcal{R}$ *écarte* $\mathcal{M}$ *des obstacles. Les déplacements à proximité des obstacles, en encart en (a), générés par échantillonnage uniforme des régions de* $\mathcal{R}$ *sont supprimés. En (a),* G *est composé de* 1783 *configurations pour une solution en* 169 *configurations ; En (b),* G *est composé de* 1592 *configurations pour une solution en* 106 *configurations.*

TAB. 3.3 – Résultats pour la traversée d'un champ d'obstacles.

	succès	cfgs de G	cfgs de C_{libre}	cfgs de C_{obs}
CR	0.44	4250	1861	2388
$CR\,GB_{10}$	0.36	3195	377	2817
$CR\,VGB_{10}$	0.66	2594	787	1807
$BG\,GB_{10}$	0.31	2906	560	2346
$BG\,VGB_{10}$	0.33	2890	631	2259
$LR\,Uni\,GB_{10}$	0.76	2794	1813	981
$LR\,Uni\,VGB_{10}$	0.77	2735	1766	968
$LR\,Gauss\,GB_{10}$	0.62	2616	1682	934
$LR\,Gauss\,VGB_{10}$	0.64	2635	1763	912

La déviation des tirages associée à la visibilité de q_{obj} augmente le pourcentage de succès de toutes les variantes présentées ; Associée à une fréquence de déviation fixée, le pourcentage de succès de la variante CR est équivalent ou amélioré ; associée à une fréquence de déviation dynamique, le pourcentage de succès de la variante CR est proche des variantes LR.

La déviation des tirages vers q_{obj} diminue le nombre de configurations de G nécessaire à l'obtention d'une solution ; toutes les variantes GB et VGB possèdent un nombre de configurations de G inférieur à la variante CR ; la restriction des tirages aléatoires dans les régions de $\mathcal{R}$ diminue le nombre de configurations menant à des collisions et augmente le nombre de configurations étendues avec succès. Des exemples de solutions sont présentés en annexe C.

3.6 Conclusion

La décomposition de C en région de $\mathcal{R}$ augmente le nombre d'expansions réalisées avec succès ; elle augmente l'expansion de G dans C_{libre}. La déformation Gaussienne positionne le graphe G à proximité de l'axe médian ; elle augmente la progression de G dans les espaces étroits. Les régions de $\mathcal{R}$ permettent de définir un échantillonnage progressif, contrôlé par la progression de G dans C ; À chaque itération, une nouvelle configuration est sélectionnée dans le domaine de visibilité de G pour maximiser la probabilité d'expansion. Les régions de $\mathcal{R}$ permettent de contrôler la progression de G dans C, par restriction des tirages aléatoires à C_{libre}.

L'ensemble des régions de $\mathcal{R}$ définit les configurations de C_{libre} à un pas de discrétisation $\Delta\theta$ près. L'absence d'une partition de C_{libre} en tranches implique une complétude liée au pas de discrétisation de U (et à son intégration numérique) ; l'existence d'une trajectoire sans collision dans l'espace des configurations est caractérisée par l'existence d'une trajectoire associée à l'ensemble des commandes U du mobile $\mathcal{M}$.

Chapitre 4

Conclusion

Les travaux résumés dans ce mémoire présentent un algorithme de planification de mouvement probabiliste incrémentale et une nouvelle décomposition en cellules irrégulières de l'espace des configurations basée sur la propriété de visibilité pour un robot mobile.

L'étude des algorithmes de planification probabiliste incrémentale nous a permis de proposer un algorithme de construction d'un arbre d'exploration de l'espace libre ; cet algorithme accélère la résolution des expansions du graphe dans l'espace de recherche. La complétude des méthodes probabilistes étant garantie pour un temps d'exécution infini, la synthèse d'algorithme optimisant la phase d'expansion est une de leurs composantes principales.

L'étude des techniques d'échantillonnage de l'espace des configurations nous a permis de proposer une décomposition des espaces libres, adaptée à la notion de visibilité. La décomposition des régions de $\mathcal{R}$ permet de représenter les espaces libres pour des environnements composés d'obstacles concaves et convexes ; Pour un nombre d'itérations fixé, elle augmente l'expansion du graphe dans l'espace de recherche. Cette augmentation du nombre de nœuds dans le graphe permet de compenser le temps de calcul nécessaire à leur définition. Associées à une déformation gaussienne, les régions de $\mathcal{R}$ permettent d'éviter les déplacements à proximité des obstacles (en dehors des cas nécessaires). Dans les passages étroits, les régions de $\mathcal{R}$ permettent une expansion du graphe intégrant les contraintes du mobile et les contraintes géométriques des obstacles.

Ces résultats doivent être validés dans des environnements tridimensionnels. La

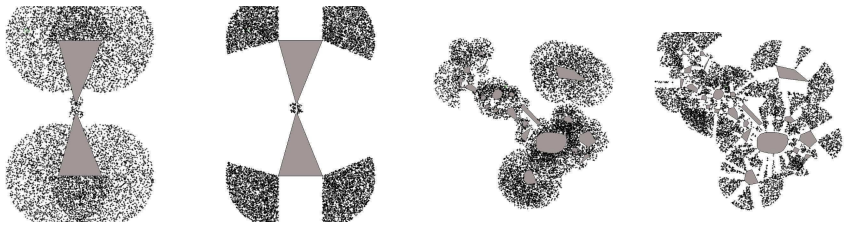
détection de collision étant une des sources de ralentissement des calculs, notre algorithme devrait conserver une vitesse d'exécution plus rapide, voire augmenter sa rapidité vis-à-vis des algorithmes existants ; l'augmentation de la dimension du système définissant les mouvements du mobile devrait également augmenter sa rapidité ; ces deux caractéristiques semblent favorables à l'extension de notre algorithme dans des environnements de grandes dimensions. Cette extension implique cependant l'étude d'une méthode de détection de collision appropriée aux déplacements et à l'espace des configurations considérés.

Dans le cadre de l'étude de dépendance entre les régions de $\mathcal{R}$, nous avons proposé une nouvelle décomposition [JC04b] pour les environnements convexes ; la définition des espaces libres est limitée aux libertés inhérentes aux sommets des obstacles, traduites en un ensemble d'arcs de cercles dans W. Un obstacle composé de n sommets est défini par n arcs de cercles, caractérisant les espaces libres autour de cet obstacle. Les valeurs d_{max} et $precision$ des régions de $\mathcal{R}$ sont conservées pour définir une taille limite des régions de C_{libre} ; Les arcs de cercles sont délimités soit par les côtés des obstacles soit par les normales aux obstacles (FIG. 4.1). Associée à un échantillonnage uniforme , cette décomposition maintient la progression de G dans les espaces libres pour les espaces convexes ; cet échantillonnage de C_{libre}, comparé à un échantillonnage uniforme de C, diminue le nombre de configurations nécessaire à l'obtention d'une solution.

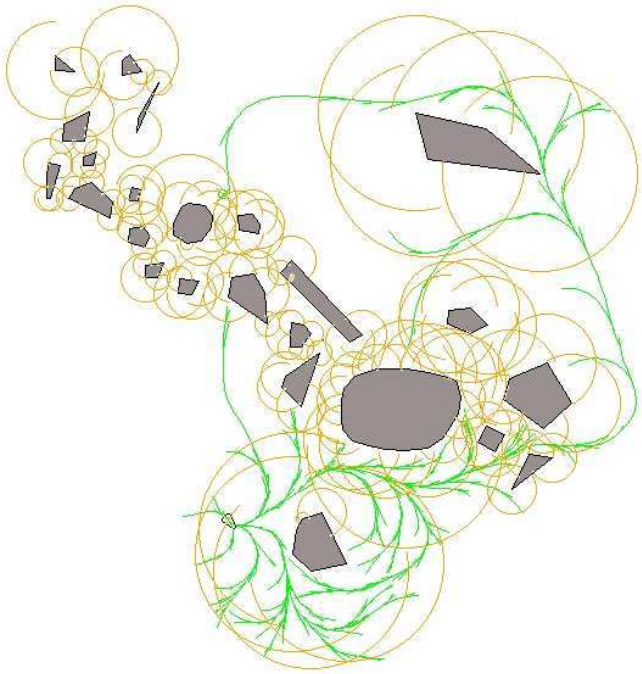
Les résultats relatifs à nos travaux nous ont permis de commencer des extensions sur la planification en environnement dynamique et sur la mise en œuvre de stratégies de planification adaptées aux espaces libres. La planification en environnement dynamique implique cependant une contrainte supplémentaire de connaissance de l'environnement ; Elle implique la modélisation de capteurs, permettant d'évaluer à chaque instant la vitesse linéaire des obstacles. Il en résulte un environnement partiellement connu.

L'étude des régions de $\mathcal{R}$ met en évidence la possibilité d'associer une stratégie de planification à la décomposition des espaces libres proposée. L. Yang *et al.* [YL00] proposent d'évaluer les espaces de C_{libre} par un ensemble de cercles reliés dans un graphe appelé *RNG*[1]. L'ensemble des cercles exprime une fonction globale de navigation dans W, par propagation d'une fonction de potentiel à partir de la configuration initiale. Un front d'ondes est émis à partir de la configuration initiale. Il en découle une stratégie de navigation dans W. Les intersections du front d'ondes de la fonction de potentiel avec les cercles définissent une liste de points de passage. Pour un espace de travail $2D$, le *RNG* se construit par ajouts successifs de

[1] *RNG*, Random Neighborhood Graph.



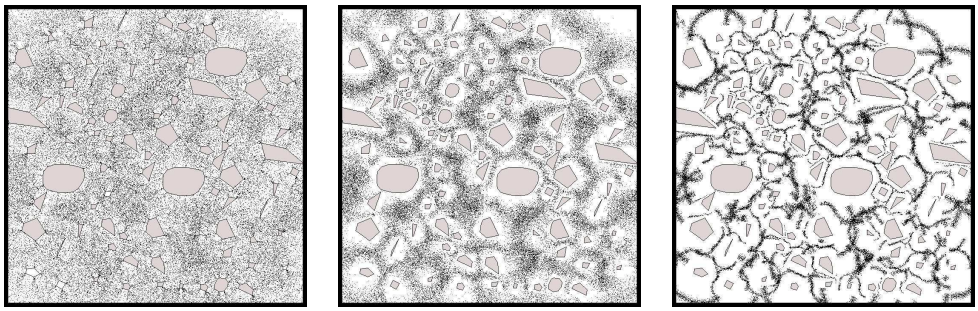
(a) *(b)* *(c)* *(d)*



(e)

FIG. 4.1 – Expression des espaces libres à partir des sommets des obstacles.

L'échantillonnage des régions définies est uniforme, à partir des sommets des obstacles dans W. Les arcs sont délimités par les côtés des obstacles en (a) et (c) ou par les normales des obstacles en (b) et (d). Les déplacements résultants en (e) contournent les obstacles.



(a) *(b)* *(c)*

FIG. 4.2 – Variation des déformations des échantillonnages des régions de $\mathcal{R}$ pour la définition d'une stratégie de planification.

La variation des déformations devrait nous permettre de modifier le comportement du mobile en fonction de sa position dans l'espace de recherche.

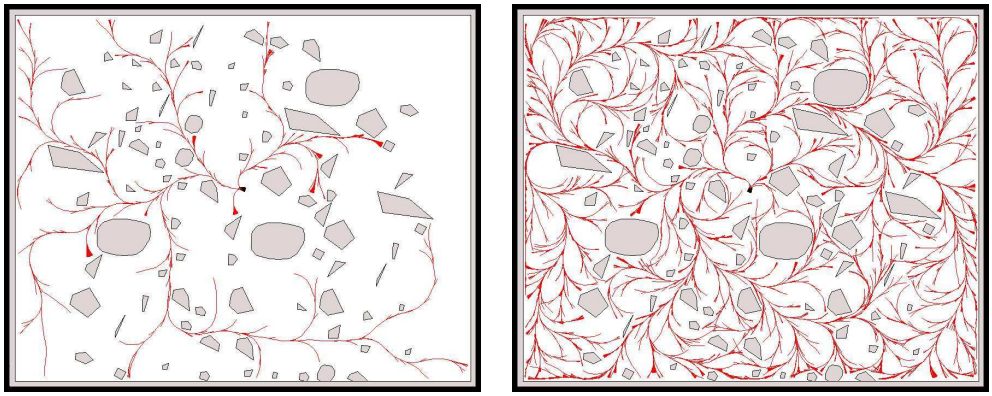
cercles centrés sur des tirages aléatoires dans C_{libre}. S.R. Lindemann *et al.* [LL04] proposent de remplacer les tirages aléatoires de la méthode *RRT* par la sélection de k points du diagramme de *Voronoï* des espaces libres. L'expansion de G étant naturellement dirigée vers les régions inexplorées, cette variante propose de focaliser la croissance de G sur des tirages plus sûrs. Ces tirages aléatoires, dits plus sûrs, sont des points du diagramme de *Voronoï*.

Nous pensons définir une stratégie de planification basée sur :
– Une évaluation des configurations par rapport aux régions de $\mathcal{R}$;
– Une évolution dynamique de la déformation des tirages aléatoires par relation à l'accessibilité du mobile. Les figures 4.2 *(a-c)* présentent trois variations de déformation.

La variation des déformations devrait nous permettre de modifier le comportement du mobile en fonction de sa position dans l'espace de recherche et en adéquation avec la définition d'une commande d'accélération par évaluation des espaces libres voisins basée sur les régions de $\mathcal{R}$.

Annexe A

Exemples de graphes



(a) *(b)*

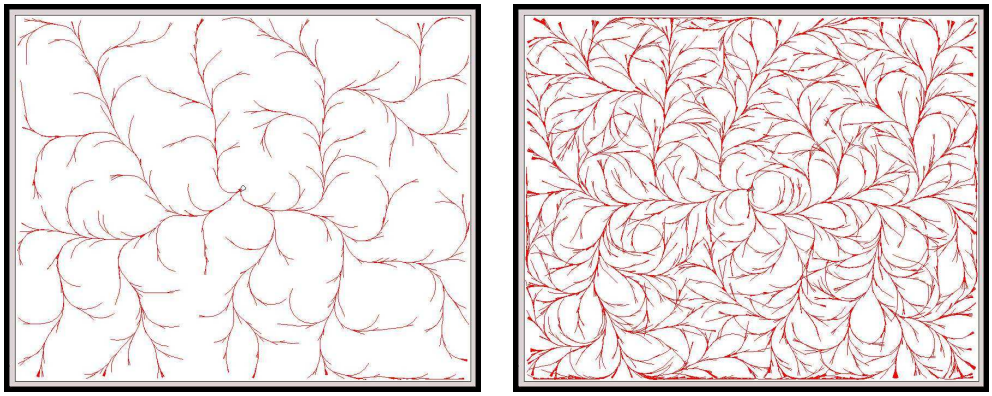
FIG. A.1 – Exemples de graphes dans l'environnement 2.9 *(a)*.

Le graphe G est construit à partir du centre, dans une position angulaire aléatoire ; il résulte de 2000 *en (a) et de* 12000 *itérations en (b).*

(a) *(b)*

FIG. A.2 – Exemples de graphes dans l'environnement 2.9 *(b)*.

Le graphe G est construit à partir du centre, dans une position angulaire aléatoire ; il résulte de 2000 *en (a) et de* 12000 *itérations en (b) ; la position centrale restreint les possibilités de progresssion et augmente les collisions.*



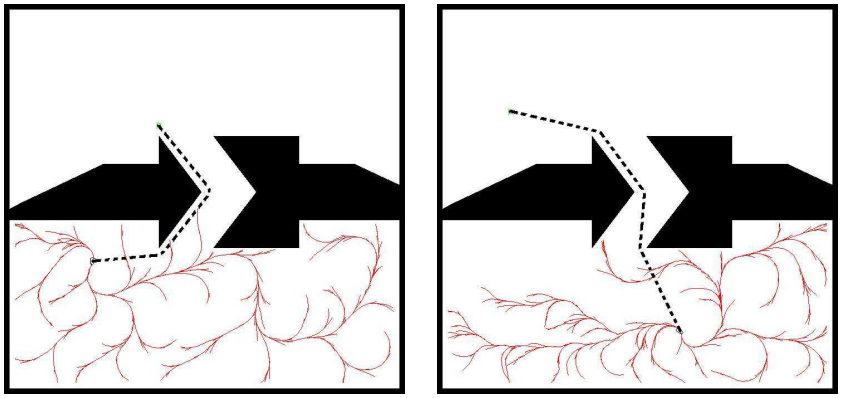
(a) *(b)*

FIG. A.3 – Exemples de graphes dans un espace sans obstacle.

Le graphe G est construit à partir du centre, dans une position angulaire aléatoire ; il résulte de 2000 *en (a) et de* 12000 *itérations en (b) ; l'absence d'obstacle (en dehors des contours) diminue le nombre de collisions pour les premières itérations.*

Annexe B

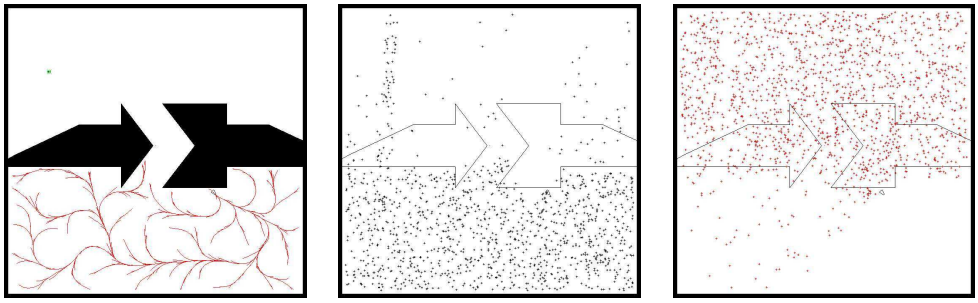
Exemples de solutions dans un passage étroit



(a) *(b)*

FIG. B.1 – Exemple de problème de passage étroit.

Les positions initiales et finales sont choisies aléatoirement en vérifiant $d - \Delta d \leq d_{\mathcal{L}}(q_{init}, q_{obj}) \leq d + \Delta d$. *Le chemin, défini par* $\mathcal{L}$ *(en pointillé), est associé ici à une résolution* $CRGB_{10}$.



(a) *(b)* *(c)*

FIG. B.2 – Distinction des tirages aléatoires dans un problème de passage étroit.

Le graphe G *représenté en (a) est construit par expansion en direction des tirages aléatoires (a) et (b). Chaque expansion peut se solder par un succès ; chacune des configurations* q_{rand}*, représentées en (b), implique l'insertion d'un nouveau noeud* q_{new} *dans* G *; en cas d'échec, Les configurations* q_{rand} *associées à une expansion se soldant par un échec (*i.e. *une collision) sont représentées en (c).*

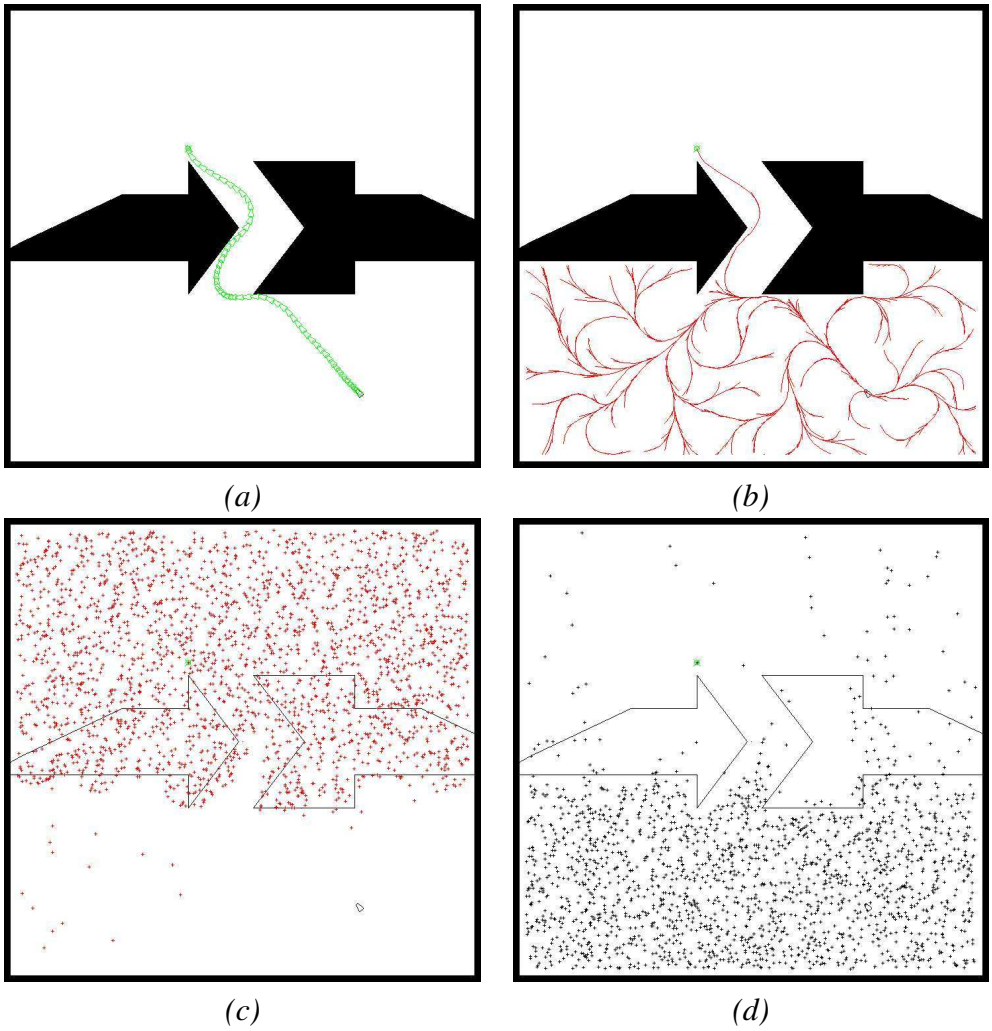


FIG. B.3 – Exemple de $CRVGB_{10}$ dans un problème de passage étroit.

L'échantillonnage uniforme de C produit indifféremment des configurations dans C_{libre} et dans C_{obs} ; le chemin résultat en (a) ; le graphe G en (b) ; les tirages aléatoires associés à une expansion couronnée par un succès en (c) ; les tirages aléatoires associés à une expansion sans succès en (d).

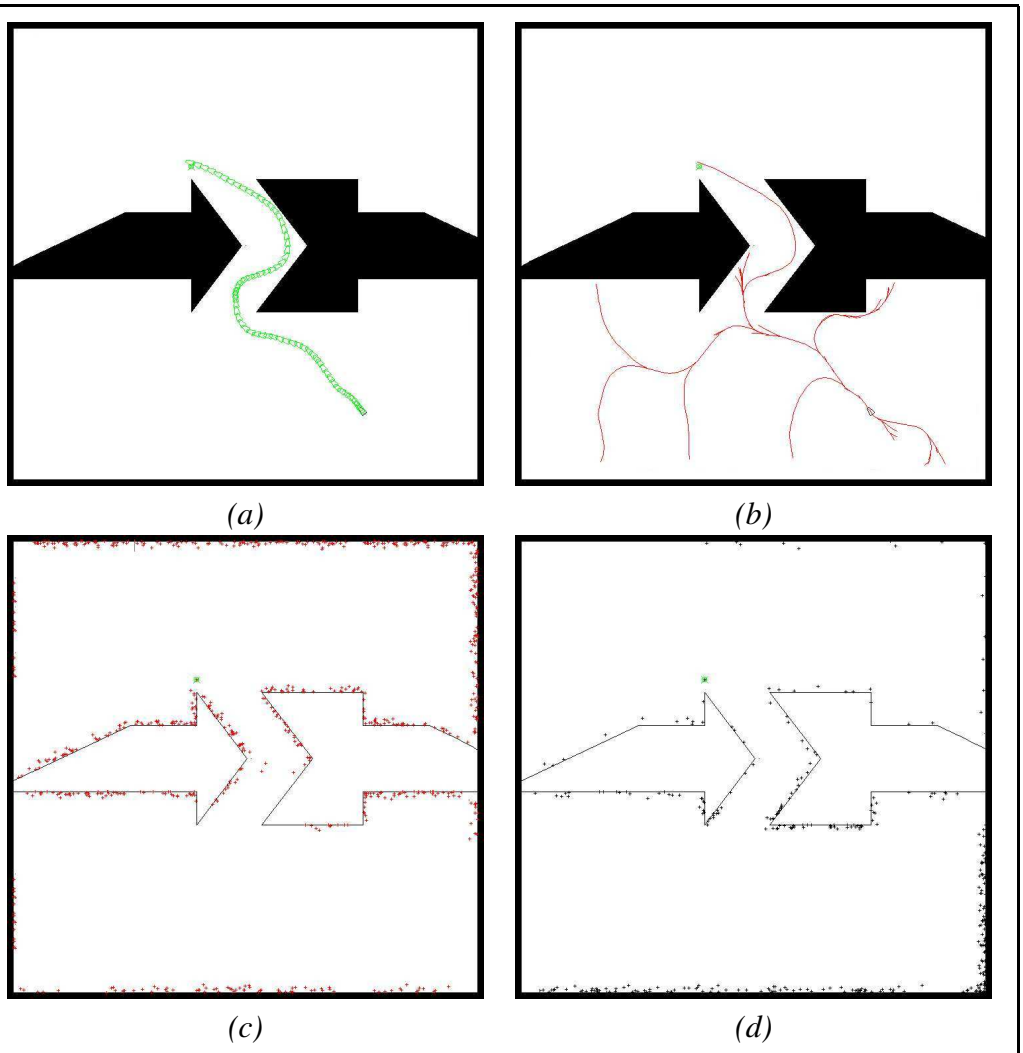


FIG. B.4 – Exemple de $BRVGB_{10}$ dans un problème de passage étroit.

L'échantillonnage gaussien aux obstacles [BOS01] ne produit que des tirages à proximité des obstacles ; le chemin résultat en (a) ; le graphe G en (b) ; les tirages aléatoires associés à une expansion couronnée par un succès en (c) ; les tirages aléatoires associés à une expansion sans succès en (d). La position du graphe oscille entre des positions proches de l'axe médian et des positions proches des obstacles.

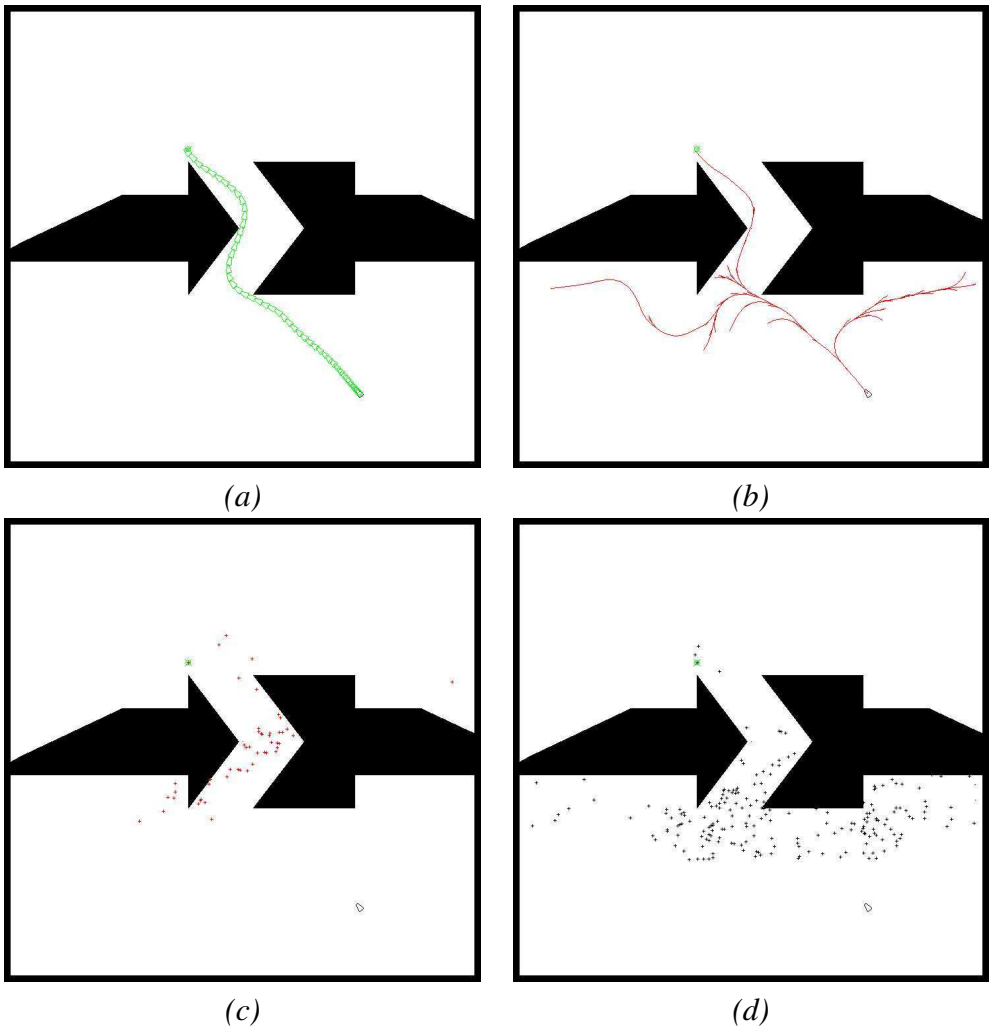


FIG. B.5 – Exemple de $LRUniVGB_{10}$ dans un problème de passage étroit.

L'échantillonnage uniforme des régions de $\mathcal{R}$ produit des tirages dans les zones visibles de G ; le chemin résultat en (a) ; le graphe G en (b) ; les tirages aléatoires de C_{libre} peuvent cependant étendre G vers des configurations en collision en (c) ; une majorité de ces configurations permettent une extension de G dans C_{libre} en (d).

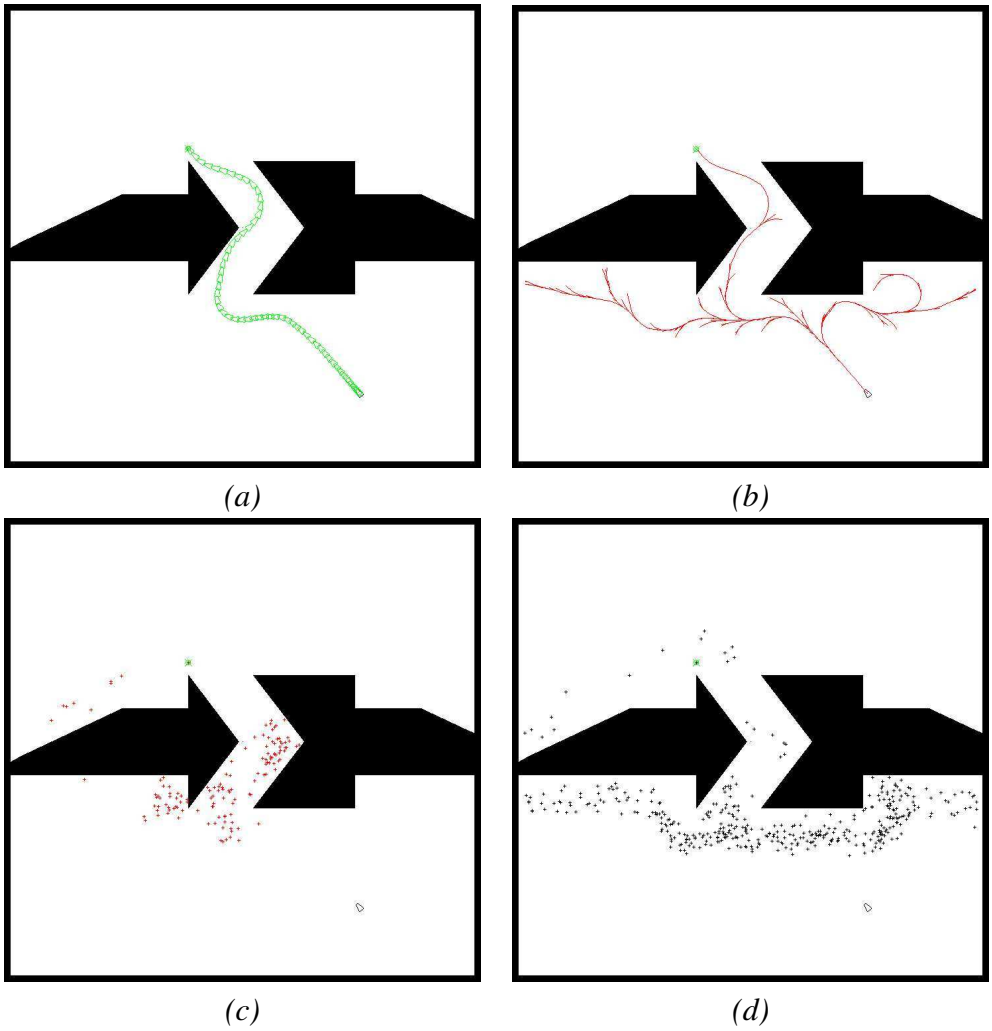


FIG. B.6 – Exemple de $LRGaussVGB_{10}$ dans un problème de passage étroit.

L'échantillonnage Gaussien des régions de $\mathcal{R}$ écarte le mobile des obstacles ; le chemin résultat en (a) ; le graphe G en (b) ; les expansions sans succès sont représentées en (c) ; cet échantillonnage favorise les expansions de G, augmentant le nombre des expansions couronnées de succès en (d).

Annexe C

Exemples de solution dans un champ d'obstacles

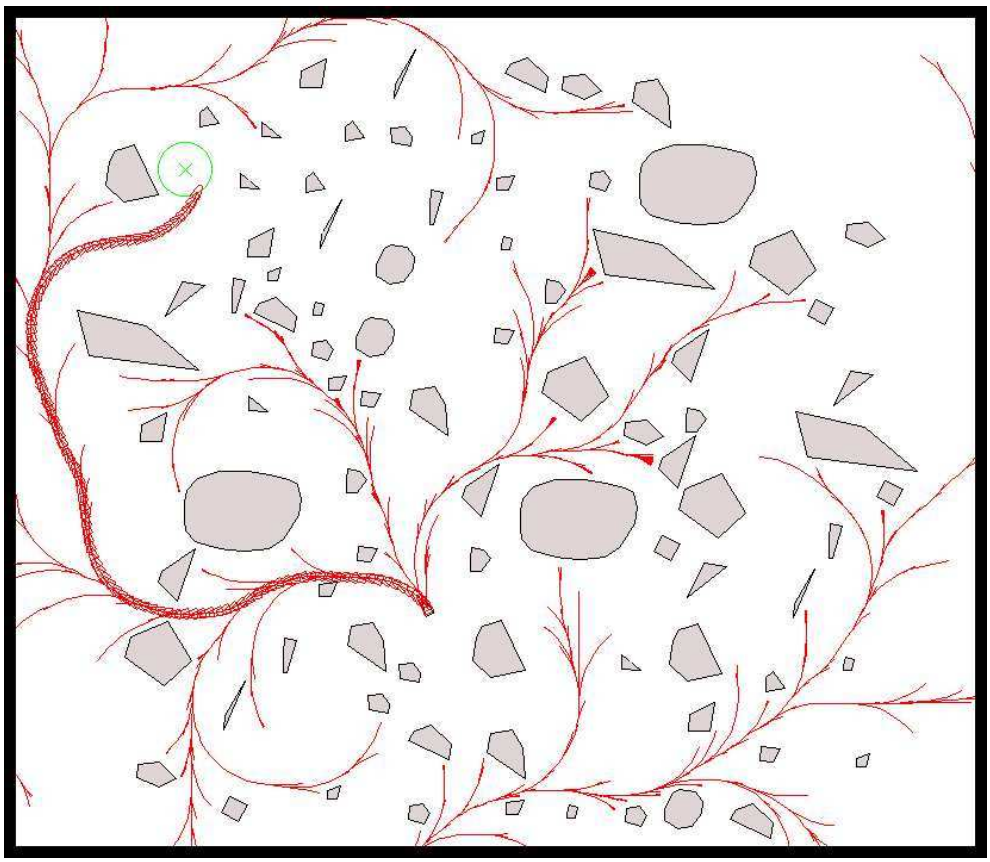


FIG. C.1 – Exemple de $CRVGB_{10}$ dans un champ d'obstacles.

Échantillonnage uniforme de C dévié vers q_{obj} ; le maintien de la déviation est conditionné par la notion de visibilité ; g peut explorer une grande partie de l'espace libre pour converger vers l'objectif.

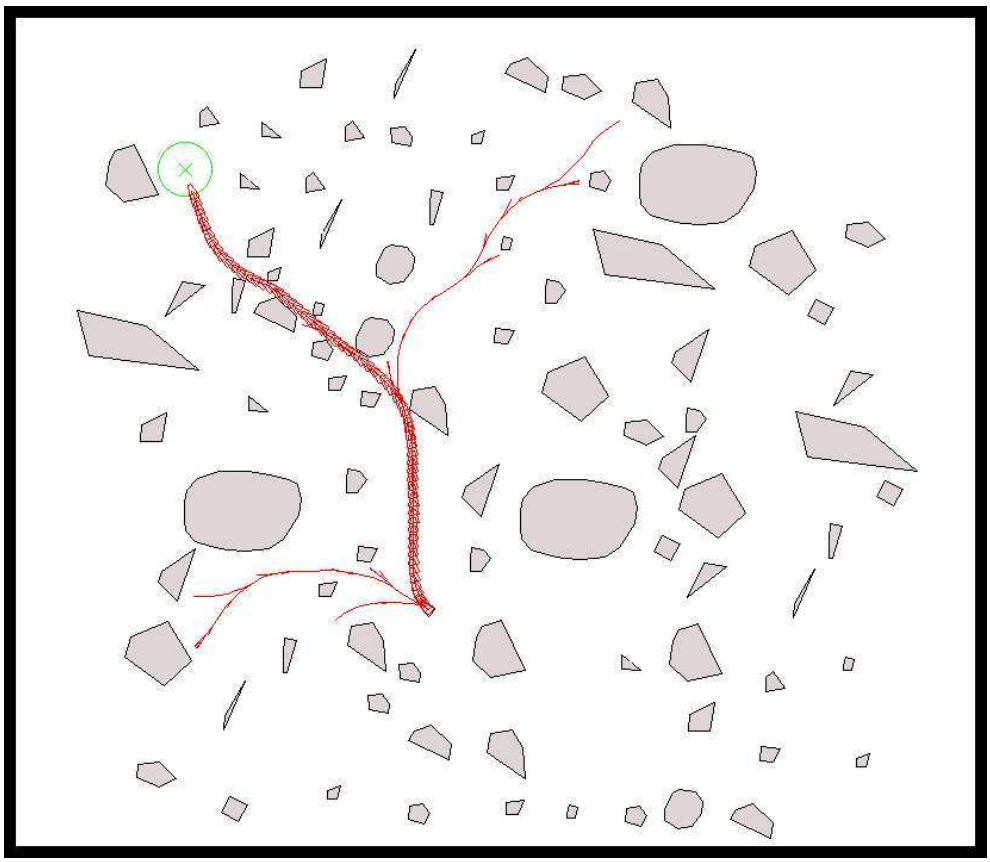


FIG. C.2 – Exemple de $BRVGB_{10}$ dans un champ d'obstacles.

Échantillonnage gaussien aux obstacles de C dévié vers q_{obj} ; le maintien de la déviation est conditionné par la notion de visibilité ; les déplacements ainsi définis passent à proximité des obstacles.

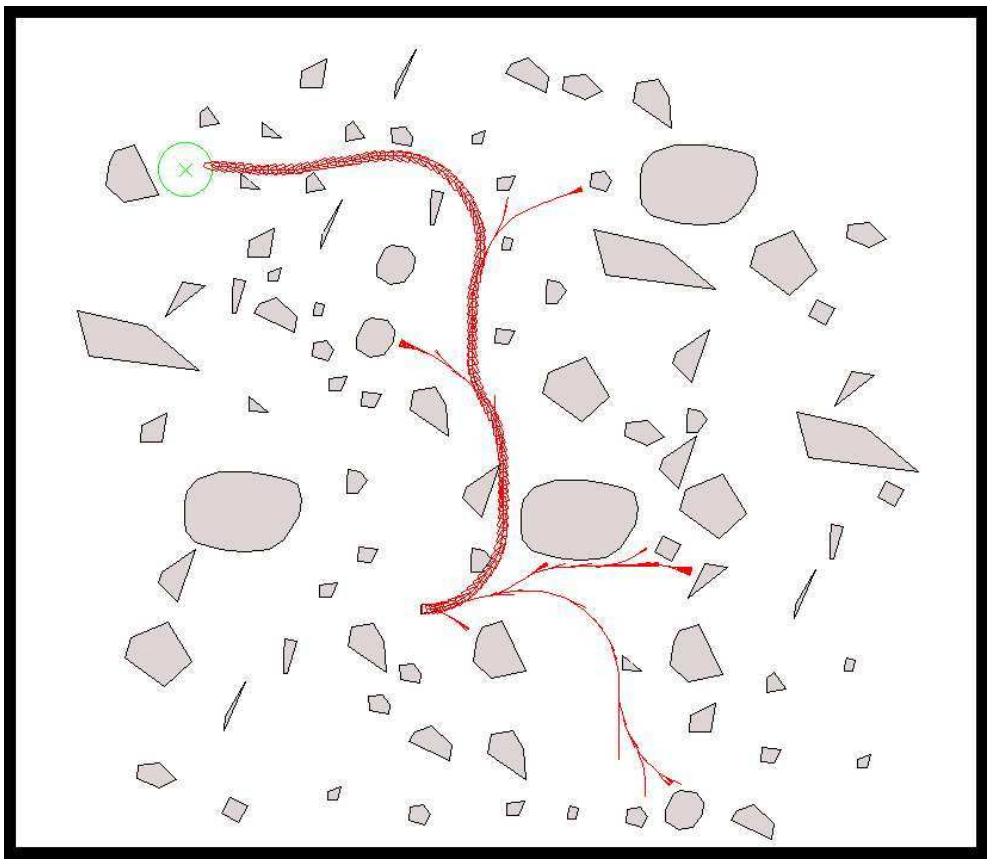


FIG. C.3 – Exemple de $LR\,Uni\,VGB_{10}$ dans un champ d'obstacles.

Échantillonnage uniforme des régions de $\mathcal{R}$ dévié vers q_{obj} ; le maintien de la déviation est conditionné par la notion de visibilité ; l'échantillonnage de C_{libre} augmente le nombre d'expansions couronnées par un succès.

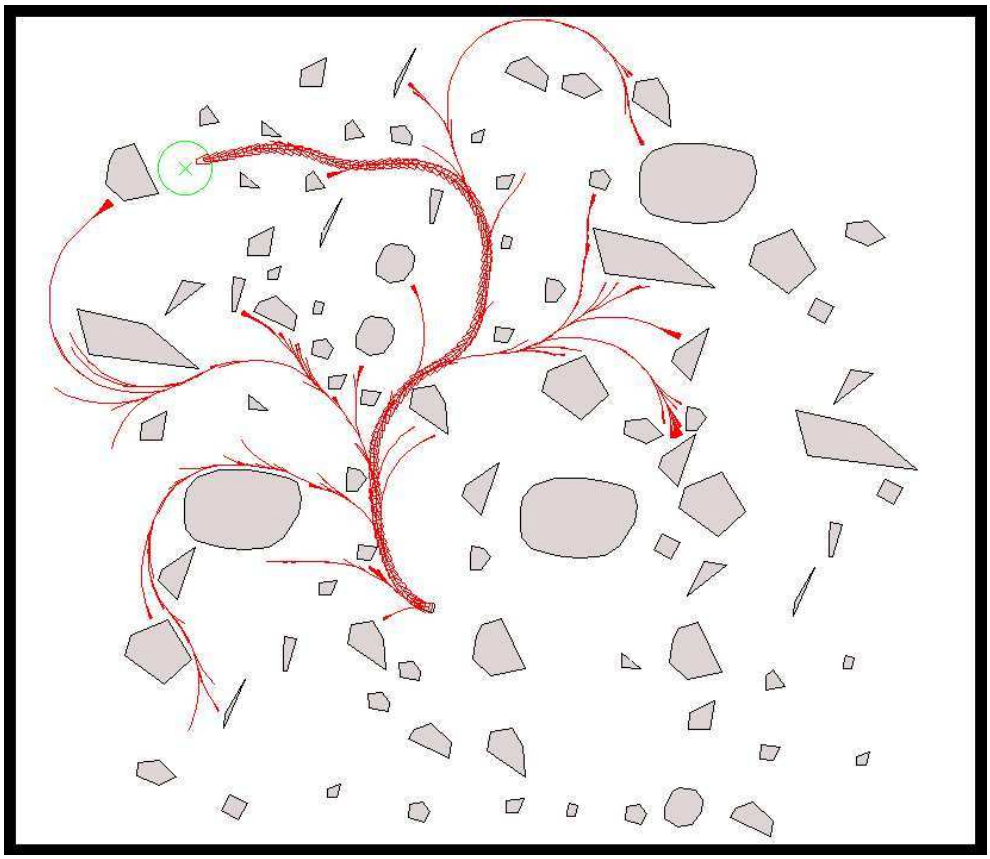


FIG. C.4 – Exemple de $LR\ Gauss\ VGB_{10}$ dans un champ d'obstacles.

Échantillonnage gaussien aux espaces libres des régions de $\mathcal{R}$ dévié vers q_{obj} ; le maintien de la déviation est conditionné par la notion de visibilité ; l'échantillonnage gaussien de C_{libre} augmente la progression de G dans les régions proches de l'objectif.

Bibliographie

[ABD+98a] N.M. AMATO, O.B. BAYAZIT, L.K. DALE, C. JONES et D. VALLEJO : OBPRM : An Obstacle-Based PRM for 3D Workspaces. *In Workshop on the Algorithmic Foundations of Robotics (WAFR'98)*, 1998.

[ABD+98b] N.M. AMATO, O. BAYAZITA, L. DALE, C. JONES et D. VALLEJO : Choosing good distance metrics and local planners for probabilisitc roadmap methods. *In Int. Conf. Robotics and Automation (ICRA'98)*, 1998.

[AD99] N.M. AMATO et L.K. DALE : Probabilistic Roadmap Methods are Embarrassingly Parallel. *In Int. Conf. Robotics and Automation (ICRA'99)*, 1999.

[AGSS89] A. AGGARWAL, L. J. GUIBAS, J. SAXE et P. W. SHOR : A linear-time algorithm for computing the Voronoï diagram of a convex polygon. *In Discrete Comput. Geom.*, 1989.

[AL94] J.M. AHUACTZIN-LARIOS : *Le Fil d'Ariane : Une Méthode de Planification Générale. Application à la Planification Automatique de Trajectoires*. Thèse de doctorat, Inst. Nat. Polytechnique de Grenoble, 1994.

[AL97a] A. AUTERE et J. LEHTINEN : A multiresolution A* method for robot path planning. *In 12th Int. Conf. on Application of Artificial Intelligence in Engineering (AIENG'97)*, 1997.

[AL97b] A. AUTERE et J. LEHTINEN : Robot motion planning by a hierarchical search on a modified discretized configuration space. *In Int. Conf. on Intelligent Robots and Systems (IROS'97)*, 1997.

[AL02] A. ATRAMENTOV et S.M. LAVALLE : Efficient Nearest Neighbor Searching for Motion Planning. *In Int. Conf. on Robotics and Automation (ICRA'02)*, 2002.

[AW96] N.M. AMATO et Y. WU : A randomized roadmap method for path and manipulation planning. *In Int. Conf. Robotics and Automation (ICRA'96)*, 1996.

[Bar91] J. BARRAQUAND : Automatic Motion Planning for Complex Articulated Bodies. Rapport technique, Laboratoire de Robotique de Paris (LRP), 1991.

[BDS+92] J.D. BOISSONNAT, O. DEVILLERS, R. SCHOTT, M. TEILLAUD et M. YVINEC : Applications of random sampling to on-line algorithms in computationnal geometry. *In Discrete Comput. Geom.*, 1992.

[BL90] J. BARRAQUAND et J.C. LATOMBE : A Monte-Carlo Algorithm for Path Planning with many degrees of Freedom. *In Int. Conf. on Robotics and Automation (ICRA'90)*, 1990.

[BL91] J. BARRAQUAND et J.C. LATOMBE : Robot motion planning : A distributed representation approach. *Int. Journal of Robotics Research (IJRR'91)*, 1991.

[BL93] J. BARRAQUAND et J.C. LATOMBE : Nonholonomic multibody mobile robots : Controllability and motion planning in the presence of obstacles. *In Algorithmica*, 1993.

[BL96] J.D. BOISSONNAT et S. LAZARD : A Polynomial-Time Algorithm for Computing a Shortest Path of Bounded Curvature Amidst Moderate Obstacles. *In Symp. on Computational Geometry*, 1996.

[BLOY01] M. BRANICKY, S.M. LAVALLE, K. OLSON et L. YANG : Quasirandomized path planning. *In Int. Conf. on Robotics and Automation (ICRA'01)*, 2001.

[BLP83] R.A. BROOKS et T. LOZANO-PÉREZ : A Subdivision Algorithm in Configuration Space for Findpath with Rotation. *In Int. Conf. on Artificial Intelligence (ICAI'83)*, 1983.

[BOS01] V. BOOR, M.H. OVERMARS et A.F. Van Der STAPPEN : Gaussian Sampling for Probabilistic Roadmap Planners. *In Int. Conf. on Robotics and Automation (ICRA'01)*, 2001.

[Cam97] S. CAMERON : Enhancing GJK : Computing Minimum and Penetration Distances between Convex Polyhedra. *In Int. Conf. on Robotics and Automation (ICRA'97)*, 1997.

[Can87] J.F. CANNY : *The complexity of robot motion planning*. Thèse de doctorat, Massachusetts Institute of Technology. Artificial Intelligence Laboratory., 1987.

[CFL04] P. CHENG, E. FRAZZOLI et S.M. LAVALLE : Improving the Performance of Sampling-Based Planners by Using a Symmetry-Exploiting Gap Reduction Algorithm. *In Int. Conf. on Robotics and Automation (ICRA'04)*, 2004.

[Che01] P. CHENG : *Reducing RRT metric sensitivity for motion planning with differential constraints*. Thèse de doctorat, Iowa State University, 2001.

[CL01] P. CHENG et S.M. LAVALLE : Reducing Metric Sensitivity in Randomized Trajectory Design. *In Int. Conf. on Intelligent Robots and Systems (IROS'01)*, 2001.

[CL02] P. CHENG et S.M. LAVALLE : Resolution Complete Rapidly-Exploring Random Trees. *In Int. Conf. on Robotics and Automation (ICRA'02)*, 2002.

[CP02] S. CARPIN et E. PAGELLO : On Parallel RRTs for Multi-robot Systems. *In 8th Conf. of the Italian Association for Artificial Intelligence (AI*IA'02)*, 2002.

[CSL01] P. CHENG, Z. SHEN et S.M. LAVALLE : RRT-Based Trajectory Design. *In Archives of Control Sciences*, volume 11, 2001.

[dBSvKO00] M. de BERG, O. SCHWARZKOPF, M. van KREVELD et M. OVERMARS : *Computational Geometry*. Springer-Verlag, 2000. 2nd rev.

[DXCR93] B.R. DONALD, P.G. XAVIER, J.F. CANNY et J.H. REIF : Kinodynamic Motion Planning. *Journal of the ACM*, 1993.

[EL01] S.A. EHMANN et M.C. LIN : Accurate and Fast Proximity Queries between Polyhedra Using Surface Decomposition. *In Eurographics*, 2001.

[eMY95] J.D. BOISSONNAT et M. YVINEC : *Géométrie Algorithmique*. Ediscience international, Paris, 1995.

[Fle96] S. FLEURY : *Architecture de contrôle distribuée pour robot mobile autonome : principes, conception et applications*. Thèse de doctorat, LAAS, 1996.

[Fra93] T. FRAICHARD : Dynamic trajectory planning with dynamic constraints : a "state-time space" approach. *In Int. Conf. Robotics and Automation (ICRA'93)*, 1993.

[GLM96] S. GOTTSCHALK, M.C. LIN et D. MANOCHA : OBB-Tree : A Hierarchical Structure for Rapid Interference Detection. Rapport technique 96-013, Department of Computer Science, University of N. Carolina, 1996.

[GO97] J. E. GOODMAN et J. O'ROURKE : *Handbook of Discrete and Computational Geometry*. CRC Press LLC, 1997.

[HC95] Y.K. HWANG et P.C. CHEN : A heuristic and complete planner for the classical mover's problem. *In Int. Conf. Robotics and Automation (ICRA'95)*, 1995.

[Hen96] D. HENRICH : A review of parallel processing approaches to motion planning. *In Int. Conf. Robotics and Automation (ICRA'96)*, 1996.

[Hen97] D. HENRICH : Fast motion planning by parallel processing - A review. *In Journal of Intelligent and Robotic Systems*, 1997.

[HKL+98] D. HSU, L.E. KAVRAKI, J.C. LATOMBE, R. MOTWANI et S. SORKIN : On finding narrow passages with probabilistic roadmap planners. *In Workshop on the Algorithmic Foundations of Robotics (WAFR'98)*, 1998.

[HLM97] D. HSU, J.C. LATOMBE et R. MOTWANI : Path planning in expansive configuration spaces. *In Trans. on Robotics and Automation*, 1997.

[HST94] Th. HORSCH, F. SCHWARZ et H. TOLLE : Motion planning for many degrees of freedom — random reflections at C-Space obstacles. *In Int. Conf. Robotics and Automation (ICRA'94)*, 1994.

[Hsu00] D. HSU : *Randomized Single-Query Motion Planning in Expansive Spaces*. Thèse de doctorat, Dept. of Computer Science, Stanford University, 2000.

[ICK+99] K.E. Hoff III, T. CULVER, J. KEYSER, M. LIN et D. MANOCHA : Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. *In Int. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH'99)*, 1999.

[Jao97] H. JAOUNI : *Etude d'une métrique non-holonome et applications aux robots mobiles*. Thèse de doctorat, LAAS, 1997.

[JC02] N. JOUANDEAU et A. Ali CHÉRIF : Environmental Motion Planning. *In Int. Symp. on Robotics (ISR'02)*, 2002.

[JC03] N. JOUANDEAU et A. Ali CHÉRIF : Primitives d'échantillonnage gaussien pour la planification de mouvement. *In MAnifestation des JEunes Chercheurs STIC (MAJECSTIC'04)*, 2003.

[JC04a] N. JOUANDEAU et A. Ali CHÉRIF : Fast Approximation to gaussian random sampling for randomized motion planning. *In Int. Symp. on Intelligent Autonomous Vehicules (IAV'04)*, 2004.

[JC04b] N. JOUANDEAU et A. Ali CHÉRIF : Planification de mouvements randomisée localisée. *In Journées Jeunes Chercheurs en Robotique (JJCR'04)*, 2004.

[Jou03] N. JOUANDEAU : Planification de mouvement. *In Journées Nationales de la Recherche en Robotique - session poster - (JNRR'03)*, 2003.

[Kav95] L.E. KAVRAKI : *Random networks in configuration space for fast path planning*. Thèse de doctorat, Stanford University, 1995.

[KGW96] E. KRUSE, R. GUTSCHE et F. WAHL : Efficient, iterative, sensor based 3-d map building using rating functions in configuration space. *In Int. Conf. Robotics and Automation (ICRA'96)*, 1996.

[Kha85] O. KHATIB : Real-time obstacle avoidance for manipulators and mobile robots. *In Int. Conf. on Robotics and Automation (ICRA'85)*, 1985.

[Kha96] M. KHATIB : *Contrôle du mouvement d'un robot mobile par retour sensoriel*. Thèse de doctorat, LAAS, 1996.

[KKL96] L.E. KAVRAKI, M. KOLOUNTZAKIS et J.C. LATOMBE : Analysis of probabilistic roadmaps for path planning. *In Int. Conf. Robotics and Automation (ICRA'96)*, 1996.

[KL94a] L.E. KAVRAKI et J.C. LATOMBE : Randomized Preprocessing of Configuration Space for Fast Path Planning. *In Int. Conf. Robotics and Automation (ICRA'94)*, 1994.

[KL94b] L.E. KAVRAKI et J.C. LATOMBE : Randomized Preprocessing of Configuration Space for Path Planning : Articulated Robots. *In Int. Conf. on Intelligent Robots and Systems (IROS'94)*, 1994.

[KL00] J. KUFFNER et S.M. LAVALLE : RRT-Connect : An efficient approach to single-query path planning. *In Int. Conf. on Robotics and Automation (ICRA'00)*, 2000.

[KLM02] Y.J. KIM, M.C. LIN et D. MANOCHA : DEEP : Dual-space Expansion for Estimating Penetration Depth between convex polytopes. *In Int. Conf. Robotics and Automation (ICRA'02)*, 2002.

[KLMR95] L.E. KAVRAKI, J.C. LATOMBE, R. MOTWANI et P. RAGHAVAN : Randomized query processing in robot path planning. *In Symp. on Theory of Computing*, 1995.

[KSLO94] L.E. KAVRAKI, P. SVESTKA, J.C. LATOMBE et M. OVERMARS : Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. Rapport technique 94-1519, 1994.

[KSLO96] L.E. KAVRAKI, P. SVESTKA, J.C. LATOMBE et M. OVERMARS : Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *In Trans. on Robotics and Automation*, 1996.

[Lat91] J.C. LATOMBE : *Robot Motion Planning (4ème édition)*. Kluwer Academic, 1991.

[Lat99] J.C. LATOMBE : Motion Planning : A Journey of Robots, Molecules, Digital Actors, and Other Artifacts. *Int. Journal of Robotics Research (IJRR'99), Special Issue on Robotics at the Millennium*, Novembre 1999.

[Lau86] J.P. LAUMOND : Feasible trajectories for mobile robots with kinematic and environnment constraints. *In Int. Conf. on Intelligent Autonomous Systems (IAS'86)*, 1986.

[LaV98] S.M. LAVALLE : Rapidly-exploring random trees : A new tool for path planning. Rapport technique 98-11, Dept. of Computer Science, Iowa State University, 1998.

[LaV04] S.M. LAVALLE : *Planning Algorithms*. [livre en-ligne], 2004. disponible sur http ://msl.cs.uiuc.edu/planning/.

[LDE+01] J.P. LAUMOND, M. DEVY, B. ESPIAU, F. GÉNOT, M. GHALLAB, F. LAMIRAUX, P. MORIN, P. RIVES, C. SAMSON et S. SEKHAVAT : *La robotique mobile*. Hermès, 2001.

[Ler98] S. LEROY : *Outils géométriques pour la planification de chemins de robots mobiles non holonomes*. Thèse de doctorat, LAAS, 1998.

[LGLM99] E. LARSEN, S. GOTTSCHALK, M.C. LINAND et D. MANOCHA : Fast Proximity Queries with Swept Sphere Volumes. Rapport technique 99-018, Department of Computer Science UNC Chapel Hill, 1999.

[LJTM94] J.P. LAUMOND, P. JACOBS, M. TAÏX et R. MURRAY : A Motion Planner for Non Holonomic Mobile Robots. *In Trans. on Robotics and Automation*, 1994.

[LK99] S.M. LAVALLE et J.J. KUFFNER : Randomized kinodynamic planning. *In Int. Conf. on Robotics and Automation (ICRA'99)*, 1999.

[LK00] S.M. LAVALLE et J.J. KUFFNER : Rapidly-exploring random trees : Progress and prospects. *In Workshop on the Algorithmic Foundations of Robotics (WAFR'00)*, 2000.

[LK01] S.M. LAVALLE et J.J. KUFFNER : Randomized kinodynamic planning. *In Int. Journal of Robotics Research (IJRR'01)*, 2001.

[LL96] F. LAMIRAUX et J.P. LAUMOND : On the expected complexity of random path planning. *In Int. Conf. on Robotics and Automation (ICRA'96)*, 1996.

[LL03] S. R. LINDEMANN et S. M. LAVALLE : Current issues in sampling-based motion planning. *In Int. Symp. on Robotics Research (ISRR'03)*, 2003.

[LL04] S.R. LINDEMANN et S.M. LAVALLE : Incrementally reducing dispersion by increasing Voronoi bias in RRTs. *In Int. Conf. on Robotics and Automation (ICRA'04)*, 2004.

[LP81] T. LOZANO-PÉREZ : Automatic Planning of Manipulator Transfer Movements. *In Trans. on Systems, Man and Cybernetics*, 1981.

[LP83] T. LOZANO-PÉREZ : Spatial Planning : A Configuration Space Approach. *In Trans. on Computers*, 1983.

[LS91] G. LAFFERIERE et H.J. SUSSMAN : Motion Planning for controllable systems without drift. *In Int. Conf. on Robotics and Automation (ICRA'91)*, 1991.

[LSL+98] J.P. LAUMOND, S. SEKHAVAT, F. LAMIRAUX, A. BELLAICHE, F. JEAN, J.J. RISLER, P. SOUÈRES, J.D. BOISSONNAT, A. De LUCA, G. ORIOLO, C. SAMSON, P. SVESTKA, M.H. OVERMARS, P. JIMÉNEZ, F. THOMAS et C. TORRAS : *Robot Motion Planning and Control, Lectures Notes in Control and Information Sciences*. Springer, 1998.

[Mir97] B. MIRTICH : V-clip : Fast and Robust Polyhedral Collision Detection. Rapport technique 97-05, Mitsubishi Electric Research Laboratory, 1997.

[MRS95] R.M. MURRAY, M. RATHINAM et W.M. SLUIS : Differential flatness of mechanical control systems. *In ASME Int. Cong. and Exp.*, 1995.

[MS93] R.M. MURRAY et S. SASTRY : Nonholonomic motion planning : Steering using sinusoids. *In Trans. on Automatic Control*, 1993.

[Nis99] C. NISSOUX : *Visibilité et méthodes probabilistes pour la planification de mouvement en robotique*. Thèse de doctorat, Université Paul Sabatier, 1999.

[OG96] C.J. ONG et E.G. GILBERT : Growth Distances : New Mesures for Object Separation and Penetration. *In Trans. on Robotics and Automation*, 1996.

[O'R94] J. O'ROURKE : *Computational Geometry in C*. Cambridge University Press, 1994.

[Pru96] A. PRUSKI : *Robotique mobile, la planification de trajectoire*. Hermès, 1996.

[RN03] S. RUSSELL et P. NORVIG : *Artificial Intelligence, A Modern Approach (2ème édition)*. Prentice Hall, 2003.

[RS90] J.A. REEDS et R.A. SHEPP : Optimal paths for a car that goes both forward and backwards. *In Pacific Journal of Mathematics*, 1990.

[Rus02] S. RUSSELL : Rationality and Intelligence. *In* Oxford University PRESS, éditeur : *Common sense, reasoning, and rationality*, 2002.

[SA95] M. SHARIR et P. AGARWAL : *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, 1995.

[SLL98] T. SIMÉON, S. LEROY et J.P. LAUMOND : A collision checker for car-like robots coordination. *In Int. Conf. on Robotics and Automation (ICRA'98)*, 1998.

[SLN00] T. SIMÉON, J.P. LAUMOND. et C. NISSOUX. : Visibility based probabilistic roadmaps for motion planning. *In Advanced Robotics Journal*, 2000.

[SLS02] B. SIMOV, S.M. LAVALLE et G. SLUTZKI : A complete pursuit-evasion algorithm for two pursuers using beam detection. *In Int. Conf. on Robotics and Automation (ICRA'02)*, 2002.

[SO98] P. SVESTKA et M. OVERMARS : Coordinated path planning for multiple robots. *In Journal of Robotics and Autonomous Systems*, 1998.

[SS83] J.T. SCHWARTZ et M. SHARIR : On the piano movers problem :I, II, III, IV, V. Rapport technique, New York University, Courant Institute, Department of Computer Sciences, 1983.

[SSLO96] S. SEKHAVAT, P. SVESTKA, J.P. LAUMOND et M.H. OVERMARS : Probabilistic path planning for tractor trailer. Rapport technique 96007, LAAS-CNRS, 1996.

[Sve97] P. SVESTKA : *Robot Motion Planning using Probabilistic Roadmaps*. Thèse de doctorat, Utrecht University, 1997.

[TLM03] B. TOVAR, S.M. LAVALLE et R. MURRIETA : Optimal navigation and object finding without geometric maps or localization. *In Int. Conf. on Robotics and Automation (ICRA'03)*, 2003.

[Tur90] G. TURK : *Graphics Gems*, chapitre Generating random points in triangles. Morgan Kauffmann, 1990.

[vdB01] G. van den BERGEN : Proximity Queries and Penetration Depth Computation on 3D Game Objects. *In Game Developers Conference*, 2001.

[Ven96] Visible positions for a car-like robot amidst obstacles. *In Workshop on Algorithmic Foundations of Robotics (WAFR'96)*, 1996.

[WAS98] S. WILMARTH, N. AMATO et P. STILLER : MAPRM : A probabilistic roadmap planner with sampling on the medial axis of the free space. Rapport technique 98-022, Department of Computer Science, Texas A&M University, 1998.

[WB00] K.D. WISE et A. BOWYER : A Survey of Global Configuration-Space Mapping Techniques for a Single Robot in a Static Environment. *Int. Journal of Robotics Research (IJRR'00)*, 19(8), August 2000.

[WBK97] A. WITKIN, D. BARAFF et M. KASS : Physically Based Modeling :Principles and Practice. *In Int. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH'97)*, 1997.

[Xav97] P.G. XAVIER : Fast swept-volume distance for robust collision detection. *In Int. Conf. on Robotics and Automation (ICRA'97)*, 1997.

[YL00] L. YANG et S.M. LAVALLE : A framework for planning feedback motion strategies based on a random neighborhood graph. *In Int. Conf. on Robotics and Automation (ICRA'00)*, 2000.

Résumé

Mots-clé : Robot mobile, Planification de mouvement.

Nous étudions dans ce mémoire la planification de mouvement probabiliste incrémentale. Nos travaux se concrétisent par un nouvel algorithme de construction des arbres aléatoires d'exploration rapide et une nouvelle décomposition de l'espace des configurations en cellules irrégulières. Partant d'une synthèse des méthodes de planification incrémentale probabiliste, nos travaux présentent un algorithme de construction accélérant l'exploration de l'espace de recherche. Partant des principales approches d'échantillonnage de l'espace de recherche utilisées dans les méthodes probabilistes, l'analyse des propriétés associées à ces échantillonnages nous conduit à proposer une décomposition de l'espace en cellules irrégulières adaptée à la notion d'accessibilité. Après définition de cette décomposition, ces algorithmes de construction sont évalués comparativement à un échantillonnage uniforme de l'espace de recherche.

Abstract

Keywords : Mobile robots, Motion planning.

In this thesis, we are studying incremental probabilistic motion planning. Our studies present a new fast algorithm to expand Rapidly exploring Random Tree (RRT) and a new irregular cell partition based on visibility. Our algorithm improves the existing successful probabilistic path planner called RRT by restricting each expansion step to the first collision free configuration. The analysis of the principal sampling's properties used in probabilistic motion planning leads us to propose a new irregular cell partition based on visibility. This new decomposition is tested in narrow environments and in cluthered ones. Results show that this new algorithm and this new decomposition are two significant componants of RRT methods. The motion planner we developped is implemented for mobile robot, evolving in a static well-known environment.