Planification de Mouvement Pour la Manipulation Dextre d'Objets Rigides

Jean-Philippe Saut

UNIVERSITE PIERRE & MARIE CURIE
LA SCIENCE A PARIS

THÈSE DE DOCTORAT DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

ROBOTIQUE - INFORMATIQUE

Présentée par

M. Jean-Philippe SAUT

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

PLANIFICATION DE MOUVEMENT POUR LA MANIPULATION DEXTRE D'OBJETS RIGIDES

soutenue le 20 Novembre 2007 devant le jury composé de:

Rapporteurs :	Rachid	ALAMI	Directeur de recherche, LAAS-CNRS
	Alain	MICAELLI	Directeur scientifique, CEA/Fontenay-aux-Roses
Examinateurs :	Philippe	BIDAUD	Professeur, Université Paris 6
	Richard	GREENHILL	Managing Director, Shadow Robot Company
	Bruno	SICILIANO	Professeur, Università di Napoli Federico II
Directrice de thèse :	Véronique	PERDEREAU	Professeur, Université Paris 6
Co-directeur de thèse :	Anis	SAHBANI	Maître de conférences, Université Paris 6

Table des matières

Introduction

PARMI les actions les plus utiles qu'un robot est susceptible d'exercer sur son environnement, figurent certainement toutes celles qui sont liées à la manipulation d'objets. C'est pourquoi, la plupart des robots industriels, des robots domestiques d'aide aux personnes et des robots d'exploration sont équipés d'organes appelés préhenseurs, constitués d'un bras muni d'une pince ou d'une main mécanique. Un préhenseur permet à un robot de ramasser ou d'attraper un objet et de lui transmettre un mouvement donné, adapté à une tâche précise : montrer ou donner l'objet à quelqu'un, l'utiliser comme un outil, le ranger, etc. Des différents types de préhenseurs existants, les plus perfectionnés sont ceux munis d'une main mécanique ou main robotisée. Les mains robotisées sont des mécanismes complexes, composés de doigts à plusieurs degrés de liberté et dotés d'une instrumentation élaborée.

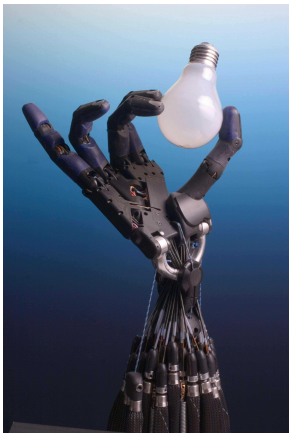


FIGURE 1 – Un exemple de main robot : la main Shadow (©Shadow Robot Company).

La manipulation pratiquée à l'aide de mouvements des doigts d'un tel dispositif est appelée manipulation dextre. Contrairement à la manipulation réalisée par un bras robot, elle se fait à l'aide de mouvements de faible amplitude, ce qui permet de travailler dans des environnements plus exigus, et elle autorise le changement de prise en repositionnant successivement les doigts sur la surface de l'objet saisi. Elle induit, par contre, une bien plus grande complexité, qui concerne des sujets variés tels que la mécanique (conception de la main, étude de la cinématique et de la dynamique de la manipulation), l'instrumentation (capteurs de force et/ou de couple, capteurs tactiles), l'automatique (contrôle des doigts en force et/ou position) et, à un niveau supérieur, la planification. Ce dernier sujet est celui du présent mémoire de thèse.

Le but de la planification pour la manipulation dextre est de concevoir une méthode calculant automatiquement une séquence de mouvements des doigts de la main, qui permettra d'amener l'objet saisi d'une pose de départ à une pose d'arrivée, tout en assurant une prise stable. Ce sujet s'inscrit dans le cadre plus large de la planification de mouvement. La manipulation dextre présente cependant certaines caractéristiques qui rendent nécessaire le développement d'une méthode spécifique au problème. Ces caractéristiques sont de plusieurs natures :

- présence de plusieurs chaînes cinématiques fermées
- contrainte cinématique de non glissement des contacts
- contrainte de stabilité de la prise
- existence de plusieurs "états" correspondant aux différents types de prises possibles, en fonction du nombre de doigts participant à la prise
- géométrie pouvant être complexe, selon la forme de l'objet manipulé

Parmi les méthodes développées jusqu'à aujourd'hui pour traiter du problème de planification de tâches de manipulation dextre, il n'en existe aucune traitant tous les aspects du problème pour des objets et manipulateurs non planaires. En effet, certaines méthodes ne s'intéressent pas au repositionnement des doigts alors que celles qui s'y intéressent ne considèrent que des doigts simplifiés, dont on supprime les contraintes cinématiques et géométriques en les considérant comme des solides libres à six degrés de liberté. Le souhait de disposer d'une méthode plus générale a été une des motivations de ce travail de thèse. Par ailleurs, l'utilisation récente des méthodes de planification probabilistes pour la manipulation à l'aide d'un bras robot a fourni des résultats intéressants. Ces résultats nous ont suggéré d'étendre leur emploi au problème de la manipulation dextre, ce qui a été fait par l'intermédiaire de la méthode proposée dans cette thèse.

Contribution et organisation de la thèse

La contribution de cette thèse est une méthode de planification de tâches de manipulation dextre. Cette méthode permet de traiter la manipulation d'objets de toutes les formes, sans restriction sur les positions des contacts entre le bout des doigts et l'objet. Des résultats de simulation sont également présentés pour démontrer l'efficacité de la méthode. Le présent mémoire se compose de cinq chapitres. Le premier chapitre présente la problématique de la planification de tâches de manipulation dextre et justifie son utilité en robotique. Il propose également une synthèse des différents travaux existants concernant ce sujet de recherche. Parmi ces travaux, certains s'appuient sur des techniques issues du domaine de la planification de mouvement. Ce sera aussi le cas de la méthode de planification proposée dans cette thèse. C'est pourquoi, le premier chapitre présente aussi les méthodes de planification de mouvement, en s'attardant sur celles que nous avons utilisées, à savoir les méthodes probabilistes et, à un niveau supérieur, les méthodes de planification de mouvement pour la manipulation à l'aide d'un bras robot. Le deuxième chapitre concerne une formulation théorique du problème de manipulation dextre, dans le cas où on néglige les effets du roulement des points de contact. C'est sur cette formulation que repose la méthode originale proposée dans le troisième chapitre. Le troisième chapitre contient la présentation et la description détaillée de cette méthode, qui constitue l'apport principal de cette thèse. Le quatrième chapitre présente les résultats obtenus, en simulation, pour différents problèmes de manipulation dextre. Il contient également une analyse des performances de la méthode de planification proposée. Le cinquième chapitre décrit une extension de la méthode proposée, qui prend en compte le roulement des points de contact, et montre des résultats obtenus avec la méthode de planification étendue. Enfin, le mémoire se conclut par un résumé des apports de ce travail de thèse et des améliorations ou développements possibles.

Chapitre 1

La Planification de Tâches de Manipulation Dextre

Ce chapitre introduit la problématique de la planification pour la manipulation à l'aide d'une main robotisée. L'essentiel des travaux les plus significatifs sur le sujet sont passés en revue et analysés. Certaines des méthodes de planification de tâches de manipulation dextre utilisent des techniques de planification de mouvement, ce qui est également le cas de la méthode proposée dans cette thèse. C'est pourquoi, ce chapitre présente aussi les différentes techniques de planification de mouvement, en s'attardant sur les mieux adaptées à la manipulation dextre.

1.1 Problématique

La manipulation dextre en robotique consiste à amener un objet tenu par une main robotisée d'une configuration initiale à une configuration finale, caractérisées chacune par une position et une orientation, à l'aide d'une suite de mouvements des doigts. Un doigt est un mécanisme articulé formant une chaîne cinématique. Cette chaîne est formée d'une succession de différents corps, considérés comme des solides rigides et reliés par des liaisons motorisées. La commande des moteurs des liaisons permet de contrôler les articulations des doigts. Une articulation peut être contrôlée géométriquement (contrôle de l'angle formé par les deux corps au niveau de leur articulation), cinématiquement (contrôle de la vitesse angulaire de la liaison) ou dynamiquement (contrôle du couple exercé au niveau de l'articulation). On peut ainsi calculer des consignes adaptées en position, vitesse ou couple pour faire suivre aux doigts un mouvement désiré ou exercer une force donnée sur l'objet. L'évolution temporelle de ces consignes constituera des trajectoires. L'objectif de la planification de tâches de manipulation dextre est de déterminer des trajectoires pour les doigts qui, correctement suivies, amèneront la main à déplacer l'objet saisi de la configuration de départ à la configuration d'arrivée. Le calcul automatique des trajectoires est le sujet principal de cette thèse, alors que nous ne traiterons pas du suivi de ces trajectoires qui ne concerne plus la planification mais la commande. Le calcul des trajectoires doit assurer à tout moment la stabilité de la prise et prendre en compte les diverses contraintes du problème telles que les contraintes cinématiques et dynamiques des contacts. Différents types de mouvement interviennent au cours de la manipulation comme le déplacement de l'objet induit par celui des doigts, la rupture d'un contact de la prise par lever d'un doigt, le déplacement d'un doigt sans contact ou l'établissement d'un contact par un des doigts.

Avant de présenter les travaux de recherche existant concernant la planification de tâches de manipulation dextre, il est nécessaire de rappeler deux notions de base qui seront fréquemment utilisées dans la suite du document. Il s'agit de la notion de configuration et de celle d'espace des configurations. Le terme de configuration a déjà été employé plus haut, sans être rigoureusement défini. La configuration d'un système est un ensemble de paramètres indépendants caractérisant de façon unique l'état de ce système. Elle est généralement représentée sous la forme d'un vecteur, appelé vecteur de configuration. Les trajectoires solutions d'un problème de planification seront donc des suites continues de configurations. L'ensemble des configurations que peut prendre le système est l'espace des configurations de ce système, noté $\mathcal{CS}$ (*Configuration Space*) et a été introduit par Lozano-Pérez [LP83][1]. La dimension de cet espace est la dimension du vecteur de configuration et est équivalente au nombre de degrés de liberté du système considéré. Pour un système donné, plusieurs choix sont souvent pos-

[1] T. Lozano-Pérez. Spatial planning : a configuration space approach. *IEEE Transactions on Computers*, 32(2) :108–120, 1983.

sibles pour l'espace des configurations, selon la façon dont on considère le problème de planification associé et notamment selon les hypothèses et simplifications émises. Le choix de cet espace des configurations fait partie de la formulation du problème et une méthode de recherche d'une trajectoire faisable reliant les configurations initiale et finale du problème de planification est une méthode de planification.

Beaucoup des premiers travaux publiés concernant la planification pour la manipulation dextre se limitaient principalement à la formulation du problème, ce qui s'explique par la complexité du sujet. En effet, il met en jeu de nombreuses difficultés : présence de chaînes cinématiques fermées, contraintes cinématiques (contacts) et dynamiques (forces de contact), grand nombre de degrés de liberté. Il a fallu attendre quelques années pour que les premières méthodes de résolution ne soient mises au point.

1.1.1 Formulation et description du problème

Parmi les travaux concernant la formulation du problème, un des tout premiers est celui de [LCS89][2], qui propose notamment un choix pour l'espace des configurations du système. Le système considéré est constitué de l'objet et des doigts qui sont représentés simplement par leur dernière phalange mais subissent néanmoins des contraintes sur leur vitesse puisqu'ils font partie d'une chaîne cinématique. L'espace des configurations choisi est alors le produit cartésien des espaces des configurations de chaque doigt, qui sont eux-mêmes assimilés à $SE(3)$ (le groupe spécial euclidien, utilisé pour représenter la position et l'orientation d'un solide dans l'espace). Li *et al.* décrivent ensuite les différentes contraintes du problème : contrainte de non collision entre les doigts, qui impose que la distance entre les doigts reste toujours positive, contrainte sur les vitesses relatives des doigts et de l'objet pour que les contacts de la prise soient maintenus et contrainte cinématique, due au fait que les doigts ont des domaines d'accessibilité limités et appartiennent chacun à une chaîne cinématique.

Montana [Mon95][3] a aussi proposé une formulation de toutes les contraintes liées à la cinématique de la manipulation par une main robotisée. Il introduit notamment l'idée de considérer les doigts saisissant l'objet comme des chaînes cinématiques fermées virtuelles. La fermeture des chaînes se fait en considérant les contacts comme des liaisons virtuelles entre les doigts et l'objet, qui sont définies à un instant donné puisque les positions des contacts changent au cours de la manipulation, comme lorsque les doigts glissent ou roulent sur la surface de l'objet par exemple. Montana ajoute à sa description de la manipulation le principe du repositionnement des points de contact.

[2] Z. Li, J. Canny, and S. Sastry. On motion planning for dexterous manipulation, part I : The problem formulation. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA'89)*, pages 775–780, 1989.

[3] D.J. Montana. The kinematics of multi-fingered manipulation. *IEEE Transactions on Robotics and Automation*, 11(4) :491–503, août 1995.

En effet, manipuler un objet en maintenant les contacts de chaque doigt ne permettant que des déplacements limités de l'objet, en raison des butées articulaires des doigts, il est souvent nécessaire de repositionner les contacts en levant les doigts. Un lever de doigt peut être considéré comme le passage à un espace des configurations de dimension supérieure car un doigt qui n'est pas en contact avec l'objet n'est plus soumis aux contraintes cinématiques des points de contact. La figure 1.1 montre ainsi comment il faut, pour tourner un bâton, alterner entre des mouvements de l'objet obtenus en contrôlant la vitesse des points de contact et des mouvements de saut de doigt.

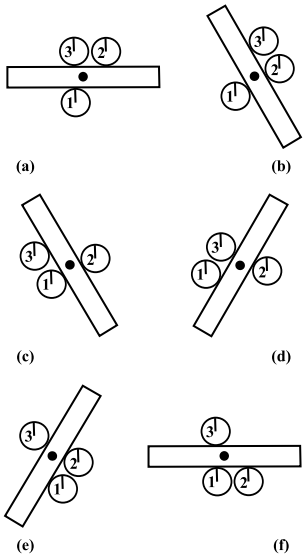


FIGURE 1.1 – Pour tourner l'objet, il est nécessaire d'alterner entre des phases de mouvement de l'objet obtenus par contrôle de la vitesse des points de contact et des phases de reconfiguration de la prise (tiré de [Mon95]).

Hong *et al.* [HLMT90][1] se sont également intéressés à l'existence de séquences de prise permettant de manipuler un objet. Leurs travaux supposent que la surface de l'objet est lisse, c'est-à-dire ne comporte aucune arête. Pour une main planaire à quatre doigts supposés ponctuels, ils montrent qu'il existe deux types de prises stables élémentaires, le critère de stabilité étant ici la fermeture de force (voir partie 2.1.4.2). Le premier type de prise se fait avec deux doigts antipodaux. Deux points P_1 et P_2 sont

[1] J. Hong, G. Lafferriere, B. Mishra, and X. Tan. Fine manipulation with multifinger hands. *Proceedings of the International Conference on Robotics and Automation (ICRA'90)*, pages 1568–1573, 1990.

antipodaux si

$$(P_1 - P_2)^T \mathbf{t_1} = 0 \tag{1.1}$$
$$(P_2 - P_1)^T \mathbf{t_2} = 0 \tag{1.2}$$
$$\mathbf{n_1} + \mathbf{n_2} = 0 \tag{1.3}$$

où $\mathbf{t_1}$ et $\mathbf{t_2}$ sont les tangentes à la surface de l'objet aux points P_1 et P_2 et $\mathbf{n_1}$ et $\mathbf{n_2}$ sont les normales à la surface de l'objet en ces mêmes points. Le deuxième type de prise stable élémentaire correspond à des prises stables à trois doigts $P_1P_2P_3$ telles que les points de contact pris deux à deux ne soient pas antipodaux. Ainsi on peut imaginer deux types d'alternance de prise pour manipuler l'objet : soit on utilise des prises à deux contacts antipodaux pour alterner les prises à deux et quatre doigts, soit on utilie des prises à trois doigts sans contacts antipodaux pour alterner les prises à trois et quatre doigts. Ces deux types de mouvements peuvent alors servir de primitives pour l'exécution d'une tâche de manipulation complexe. Han et Trinkle ont proposé une description du problème assez similaire [HT98][2]. Ils se sont intéressés à la manipulation d'une sphère avec trois doigts hémisphériques et ont pour cela pris en compte la possibilité de faire rouler les doigts sur la surface de l'objet pour modifier sa configuration ainsi que la possibilité de lever un doigt pour le repositionner sur la surface de l'objet.

Plus récemment, Han *et al.* ont présenté une définition du problème de la planification pour la manipulation dextre [HLT+00][3] mais sans proposer de technique pour le résoudre. Ils choisissent de représenter un état du système, constitué par la main tenant l'objet, par le vecteur :

$$\mathbf{c} = (\mathbf{g_{po}}, \eta, \theta, F) \in SE(3) \times \mathbb{R}^{5k} \times \mathbb{R}^n \times \mathbb{R}^6 \tag{1.4}$$

où $\mathbf{g_{po}}$ est la configuration de l'objet, η la position des points de contact (les coordonnées du point de contact sur la surface de l'objet, sur celle du doigt et l'angle entre les deux surfaces autour de la normale au point de contact), θ l'ensemble des coordonnées articulaires de tous les doigts et F l'effort qui doit être exercé sur l'objet pour le maintenir dans sa configuration courante. Les auteurs définissent alors le problème de planification de tâches de manipulation dextre comme la recherche d'un chemin faisable entre deux configurations $\mathbf{c_0}$ et $\mathbf{c_1}$ données. Un chemin faisable est un ensemble continu de configurations respectant l'absence de collision entre les différents corps du système ailleurs qu'aux points de contact en bouts de doigts, les contraintes

[2] L. Han and J.C Trinkle. Dextrous manipulation by rolling and finger gaiting. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'98)*, pages 730–735, mai 1998.

[3] L. Han, Z. Li, J.C. Trinkle, Z. Qin, and S. Jiang. The planning and control of robot dextrous manipulation. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2000)*, pages 263–269, 2000.

de fermeture des chaînes cinématiques formées par les doigts et l'objet, les contraintes imposées par les butées articulaires des doigts, la contrainte d'équilibre dynamique de la prise et les contraintes sur les couples maximaux que peuvent exercer les moteurs des doigts.

Les auteurs des différentes formulations et descriptions du problème de la planification de tâches de manipulation dextre présentées plus haut n'ont généralement pas accompagné leurs travaux de méthodes de résolution. Cependant, ces formulations ont servi de base à des travaux plus récents, qui développent des solutions pour résoudre le problème de planification.

1.1.2 État de l'art des méthodes de planification de tâches de manipulation dextre

Parmi les approches proposées, on distingue des approches partielles ou locales, qui s'intéressent à un aspect particulier de la planification de tâches de manipulation dextre, et des approches globales, qui cherchent à prendre en compte tous ses aspects. Les premières vont s'intéresser à des mouvements assez limités de l'objet, sans se préoccuper du besoin de changement de prise qui intervient pour des mouvements plus amples, ou au contraire ne s'intéresser qu'au repositionnement des doigts sans se soucier de la cinématique des doigts et de l'objet. Les secondes essaient de prendre en compte toutes les caractéristiques de la manipulation, en utilisant, au niveau local, les approches précédentes ou des techniques plus simples. L'existence de ces deux niveaux d'étude s'explique par la grande difficulté de la planification des tâches de manipulation dextre, qui a incité les chercheurs à se restreindre à l'étude d'un aspect précis de ce problème ou à limiter l'étude du problème général à des situations particulières ou en émettant des hypothèses simplificatrices.

1.1.2.1 Les approches locales

Les approches locales concernent seulement un des sous-problèmes associés à la planification de tâches de manipulation dextre. Ainsi, Rus [Rus97, Rus98][1,2] s'est intéressée aux mouvements des doigts qui permettent la rotation d'un objet, sans prendre en compte les restrictions des mouvements des doigts dues aux butées articulaires ni le problème d'éventuelles collisions. Elle a proposé une approche complète [1] qui prend en compte la dynamique du mouvement de réorientation. Elle a développé la technique appelée *tracking finger algorithm* pour faire tourner l'objet. Cette technique propose

1. dans le sens algorithmique du terme

[1] D. Rus. Coordinated manipulation of objects in a plane. *Algorithmica*, 19(1/2) :129–147, 1997.

[2] D. Rus. In-hand dexterous manipulation of 3D piecewise-smooth objects. *International Journal of Robotics Research*, 1998.

de choisir un ensemble de doigts fixes, c'est-à-dire dont la position est fixe par rapport au repère monde, et un doigt mobile. Le doigt mobile se déplace le long de la surface de l'objet tout en continuant à exercer une force selon la normale à sa surface. Il en résulte l'application d'un couple, qui fait tourner l'objet. Dans le cas d'un objet plan, il y a deux doigts fixes et un mobile et dans le cas d'un objet tridimensionnel, il y a trois doigts fixes et un doigt mobile. La figure 1.2 illustre ce principe dans le cas d'un objet plan, pour des contacts sans frottement. Cette méthode utilise une définition al-

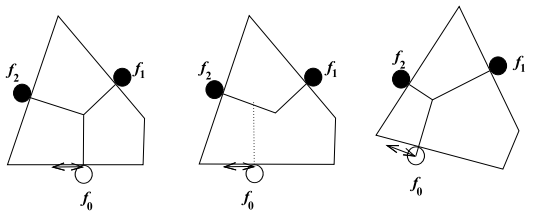


FIGURE 1.2 – Tant que les forces de contact ne sont pas concourantes, la prise n'est pas stable et l'objet tourne. Quand elles le redeviennent un nouvel équilibre est atteint, qui correspond à une nouvelle orientation de l'objet (tiré de [Rus97]).

gébrique de l'espace des configurations du système composé par l'objet à réorienter et les doigts, respectant le principe du *tracking finger*. Les contraintes que doit respecter le système concernent les positions relatives des doigts par rapport à l'objet. Rus montre que, dans le cas d'un objet tridimensionnel, l'espace des configurations est difféomorphique à $S0(3)$ (le groupe spécial orthogonal qui correspond à l'ensemble des rotations dans l'espace). Autrement dit, chaque orientation de l'objet est déterminée par la position du doigt mobile et cette relation est différentiable. On peut donc obtenir un système d'équations différentielles permettant de déterminer les forces que doivent exercer les doigts en fonction de l'accélération angulaire désirée de l'objet. Toutefois les forces obtenues ne sont pas forcément applicables car elles peuvent être négatives (dirigées vers l'extérieur de l'objet). Dans ce cas, le mouvement de rotation doit être stoppé et un doigt doit rompre son contact avec l'objet et se repositionner pour modifier la prise. Rus ne propose pas de solution au problème de la reconfiguration de la prise dans le cas général d'un objet tridimensionnel.

A l'opposé de ces travaux, on trouve ceux de Sudsang et Phoka [SP03][3], qui concernent uniquement la détermination d'une séquence de prises stables permettant de passer d'une prise stable quelconque à une autre, sans considérer le mouvement de l'objet. Le critère de stabilité retenu est la fermeture de force, c'est-à-dire qu'une prise est considérée stable si elle peut résister à n'importe quelle perturbation, en force et

[3] A. Sudsang and T. Phoka. Regrasp planning for a 4-fingered hand manipulating a polygon. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'03)*, 2 :2671–2676, septembre 2003.

moment, exercée sur l'objet, en appliquant les forces de contact appropriées tout en respectant les contraintes de non glissement des points de contact (voir partie 2.1.4.2). La méthode s'applique à un système plan composé d'un objet polygonal fixe tenu par quatre doigts ponctuels. Une prise est caractérisée par un ensemble de points de contact, c'est-à-dire que les doigts sont considérés comme des points aux mouvements non contraints. Sudsang et Phoka utilisent une représentation d'ensembles de prises pour la construction d'un graphe, qui permet de trouver une séquence de prises stables reliant les prises initiale et finale. Les noeuds du graphe correspondent alors à des prises appelées prises parallèles ou à des prises appelées prises concourantes. Les prises parallèles (figure 1.3) sont les prises à trois doigts telles qu'il existe trois vecteurs parallèles, chacun inclus dans un des trois cônes de frottement des contacts, et que la direction du vecteur du milieu soit opposée à celles des deux autres. Les prises concourantes (figure 1.3) sont les prises à trois contacts telles qu'il existe trois demi-droites concourantes, appartenant chacune à un des cônes de frottement, et telles que les trois cônes de frottement couvrent positivement $\mathbb{R}^2$ (c'est-à-dire que n'importe quel vecteur de $\mathbb{R}^2$ peut être écrit comme une combinaison linéaire à coefficients positifs de n'importe quel triplet de vecteurs appartenant chacun à un des cônes de frottement). Une condition suffisante pour qu'une prise à trois doigts forme une prise concourante

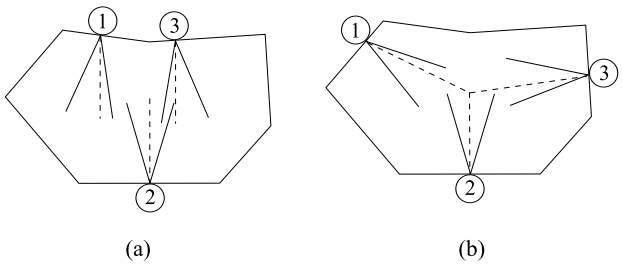


FIGURE 1.3 – Prises parallèle (a) et concourante (b).

est que les cônes de frottement couvrent positivement $\mathbb{R}^2$ et qu'il existe un point x_0 tel que les cônes de frottement inversés partant de x_0 intersectent les trois arêtes de l'objet sur lesquelles se font les contacts. Cette condition sert aux auteurs à montrer que l'ensemble des prises concourantes, pour des contacts sur trois arêtes différentes données, peut être représenté par un ensemble de points du plan inclus dans un polygone appelé *focus cell*. Le principe de construction d'un *focus cell* est illustré par la figure 1.4. Sur la figure 1.4-b, on voit qu'au point x_0 sont associées trois segments E'_a, E'_b et E'_c telles qu'on peut choisir indépendamment les contacts sur chacun de ces trois segments et avoir toujours une prise concourante. Les doigts pourront ainsi glisser librement dans ces zones tout en maintenant la stabilité de la prise. L'ensemble des points qui partagent les mêmes zones de contact indépendantes E'_a, E'_b et E'_c est représenté en gris sur la figure 1.4-b. L'ensemble des points x_0 auxquels on peut associer des zones de

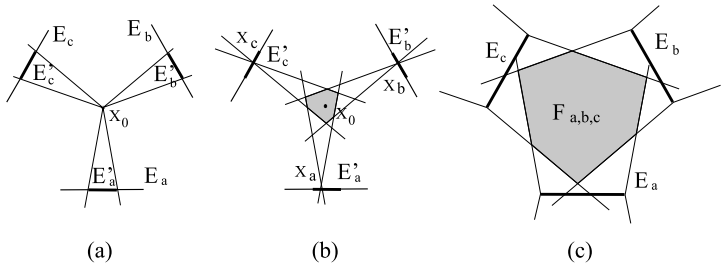


FIGURE 1.4 – Principe de construction d'un *focus cell* : (a) les cônes de frottement inversés, (b) les zones de contact indépendantes, (c) *focus cell* obtenu à partir de l'intersection des unions des cônes de chaque zone de contact indépendante (tiré de [SP03]).

contact indépendantes, appartenant à trois arêtes E_a, E_b, E_c données, est le *focus cell* $F_{a,b,c}$ de ces trois arêtes (figure 1.4-c). Sudsang et Phoka proposent alors de construire un graphe dont les noeuds sont des *focus cells*. Deux noeuds peuvent être reliés si les *focus cells* associés ont une intersection non nulle et si ils ont deux arêtes sur lesquelles se font les contacts en commun. Si l'intersection de leurs *focus cells* est nulle, on peut essayer de glisser un doigt le long d'une des arêtes jusqu'à ce que l'intersection soit non nulle. Cette méthode de construction peut être étendue en prenant en compte les prises parallèles et les prises stables à deux doigts. La figure 1.5 montre une suite de reconfiguration de prises obtenue par la méthode de Sudsang et Phoka.

Les méthodes présentées jusqu'ici ne concernent que certains des aspects de la manipulation (mouvement de l'objet avec une prise constante ou changement de la prise sans bouger l'objet et sans considérer les chaînes cinématiques des doigts). Des approches "globales" intègrent ces différents aspects et sont décrites dans la partie suivante.

1.1.2.2 Les approches globales

Les approches par construction de graphes et utilisation de primitives

Un des premiers travaux à s'intéresser à la planification complète (mouvements des doigts et de l'objet, changement des points de contact) d'une tâche de manipulation dextre est celui de Trinkle et Hunter [TH91][1], qui s'intéresse au cas particulier d'un système plan constitué d'un robot manipulant un objet, les deux ayant des surfaces polyédriques. Leur méthode autorise des contacts n'importe où sur la main. Le principe de cette méthode est d'utiliser une description du système appelée *Contact Formation*

[1] J.C. Trinkle and J. Hunter. A framework for planning dexterous manipulation. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA'91)*, pages 775–780, avril 1991.

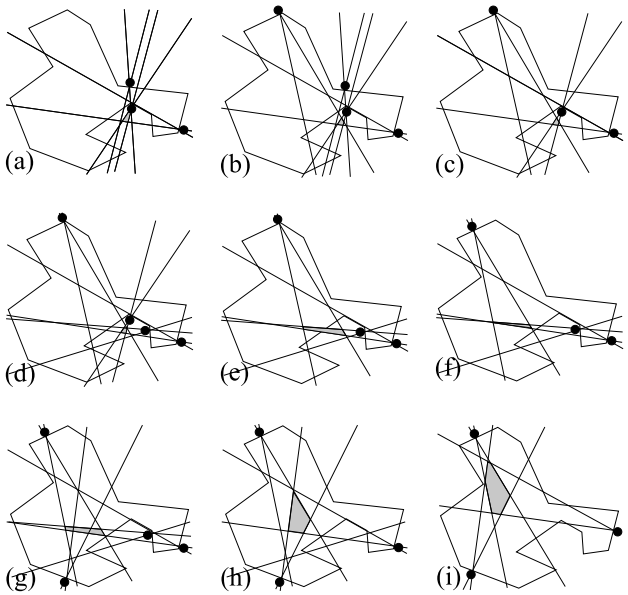


FIGURE 1.5 – Une suite de reconfiguration de prise obtenue par la méthode de Sudsang et Phoka ([SP03]). Les cônes de frottement sont représentés ainsi que les *focus cells* associés aux prises.

(notée CF) introduite dans [DV89][1]. Un CF décrit un ensemble de prises sous la forme d'une liste de contacts entre les éléments de la main et de l'objet, qui peuvent être des sommets ou des arêtes. Les contacts sont de type sommet-arête ou arête-arête. Par exemple, un CF peut indiquer, entre autres, que le sommet i de l'objet est en contact avec l'arête j de la main. Ainsi, un CF peut être vu comme une surface unidimensionnelle, puisque les positions des contacts ne sont pas fixées le long des arêtes. La surface correspondant à un CF sera limitée par le changement d'arête des contacts. En effet, si le sommet i de l'objet entre en contact avec une arête de l'objet autre que j, on change de CF. Les configurations initiale et finale du problème de manipulation sont considérées comme des CF. Elles servent chacune à la construction d'un arbre dont les noeuds sont des CF. Les deux arbres sont développés indépendamment en cherchant pour chaque noeud quels sont les CF qu'il permet d'atteindre. Une fois que les deux arbres contiennent un CF identique, on connaît la suite de CF reliant les configurations initiale et finale. Le modèle physique du système est ensuite utilisé pour déterminer les trajectoires continues permettant de passer d'un CF à un autre. Trinkle et Hunter ont testé leur méthode pour un système planaire à cinq degrés de liberté. Ce système se

[1] R.S. Desai and R.A. Voltz. Identification and versification of termination conditions in fine motion in presence of sensor errors and geometric uncertainties. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'89)*, pages 800–807, 1989.

compose d'une main munie de deux doigts à un degré de liberté chacun et d'un objet à trois degrés de liberté (figure 1.6). Cependant, les dimensions du système peuvent être réduites à deux car sa géométrie permet de calculer la configuration de l'objet connaissant les angles des articulations des doigts. La figure 1.7 montre quelques étapes de

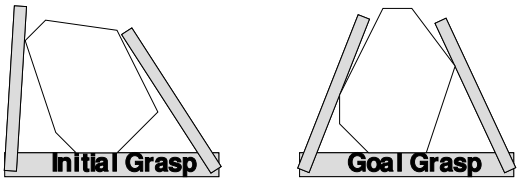


FIGURE 1.6 – Configurations initiale et finale désirées (tiré de [TH91]).

la manipulation d'un objet à sept côtés, pour relier les deux configurations de la figure 1.6.

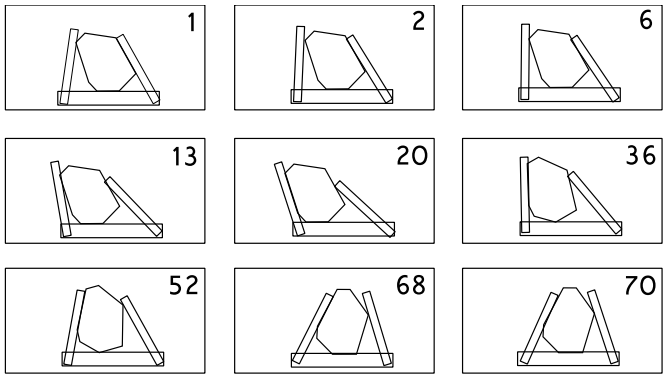


FIGURE 1.7 – Quelques étapes de la manipulation d'un polyèdre (tiré de [TH91]).

Une méthode similaire a été proposée par Zhang *et al.* [ZTM96][2]. Ils l'ont appliqué à un exemple tridimensionnel, avec une main à trois doigts et six degrés de liberté au total. La méthode se base sur une extension de la notion de CF, qui utilise des primitives topologiques plus complexes pour décrire la prise. Elles peuvent être de type arête, sommet, arête courbe, surface plane, surface courbe, etc. Les auteurs se servent de cette représentation pour construire un ensemble de prises appelées *Canonical Grasps* (notées CG). Les CG sont alors utilisés pour relier deux configurations données à l'aide d'une technique de construction d'arbre semblable à celle de Trinkle

[2] H. Zhang, K. Tanie, and H. Maekawa. Dextrous manipulation planning by grasp transformation. *Proceedings of the IEEE Conference on Robotics Automation (ICRA'96)*, avril 1996.

et Hunter. Comme la méthode de Trinkle et Hunter, la méthode de Zhang *et al.*, s'applique plutôt à des problèmes simples, pour lesquels les configurations initiale et finale sont proches ou les objets comportent peu de faces ou de surfaces complexes et la main peu de degrés de liberté. En effet, plus la forme de l'objet est complexe et plus la main comporte d'éléments, plus il y a de CF (ou de CG) et plus le graphe sera grand et les configurations initiale et finale éloignées dans le graphe.

La complexité du problème de la planification de tâches de manipulation dextre a incité certains auteurs à trouver des moyens de le simplifier. Omata et Farooqi [ON94, OF96][1,2] suggèrent ainsi d'utiliser des primitives pour réduire l'étendue des mouvements possibles du système main-objet. Ces primitives concernent les mouvements des doigts permettant de faire tourner l'objet. Les auteurs en définissent trois, qui sont représentées sur la figure 1.8. Telles qu'elles ont été formulées et utilisées, elles ne s'appliquent qu'à des objets de forme prismatique. La primitive R est une rotation effectuée en déplaçant plus de deux doigts. Si un doigt atteint la limite de son domaine de travail, il est ramené en arrière dans la direction opposée à la rotation. La primitive P est la primitive de pivotement de l'objet : les points de contact de deux doigts forment un axe de pivotement et un troisième fait tourner l'objet autour de l'axe de pivotement en exerçant une force sur une des faces latérales de l'objet. La primitive C est une variante de la primitive P pour laquelle deux doigts en contact sur une même face de l'objet exercent un couple pour le faire tourner.

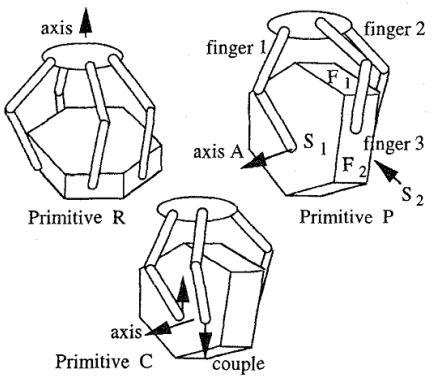


FIGURE 1.8 – Les primitives de réorientation d'un objet polyédrique (tiré de [OF96]).

Un arbre, dont les noeuds sont des orientations de l'objet, est développé à partir de l'orientation initiale de l'objet. La recherche se fait en développant l'arbre par niveaux,

[1] T. Omata and K. Nagata. Planning reorientation of an object with a multifingered hand. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA'94)*, 4 :3104–3110, mai 1994.

[2] T. Omata and M.A. Farooqi. Regrasps by a multifingered hand based on primitives. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA'96)*, 3 :2774–2780, avril 1996.

à partir du premier niveau qui est l'orientation initiale. A la construction d'un nouveau niveau, l'axe r_0 et l'angle γ_0 de rotation entre chaque noeud du plus haut niveau de l'arbre et l'orientation finale désirée sont calculés. Si une des primitives permet d'effectuer directement cette rotation, une solution du problème de planification est trouvée. Sinon pour chaque noeud, on choisit, parmi les axes de rotation autour desquels il est possible de faire tourner l'objet en utilisant une des primitives de ressaisie, celui qui est le plus proche de r_0. On calcule alors l'angle de rotation autour de l'axe choisi qui rapprochera le plus l'objet de son orientation finale. En appliquant cette rotation à l'objet, on obtient une nouvelle orientation qui sera un noeud de niveau supérieur de l'arbre. Le développement de l'arbre se poursuit en suivant les règles énoncées plus haut.

Une autre méthode travaillant en construisant un graphe est celle de Cherif et Gupta [CG98, CG99][3,4]. Leur planificateur travaille sur plusieurs niveaux et son but est de construire un chemin entre une orientation initiale et une orientation finale de l'objet. Le niveau supérieur (ou niveau global) génère une trajectoire nominale pour l'objet, qui le relie à l'orientation finale. Cette trajectoire est choisie pour être la plus courte possible. Itérativement, le niveau global calcule une orientation de l'objet, sur la trajectoire nominale, proche de l'orientation précédente sur le chemin. Cette nouvelle orientation de l'objet servira de sous-but. Le niveau immédiatement inférieur (le niveau local) essaye chaque mode de manipulation parmi trois possibles, tant qu'aucun ne permet d'atteindre le sous-but. Ces trois modes sont les suivants : mouvement de l'objet avec déplacement de tous les doigts ou en gardant fixe la position de certains doigts ou déplacement des doigts sans déplacer l'objet. Dans ces différents modes, le déplacement des doigts se fait soit par roulement soit par glissement sur la surface de l'objet. Le niveau inférieur au niveau local (le niveau instantané) utilise ensuite la cinématique des contacts et les modèles inverses des doigts pour atteindre la nouvelle orientation de l'objet. Si le niveau instantané trouve une solution, on a alors un nouvel état du système (nouvelle orientation de l'objet et/ou nouvelle prise) et on l'ajoute au chemin. Sinon un autre sous-but est généré sur la trajectoire nominale. Cette méthode marche avec des objets convexes polyédriques ou lisses tels que des ellipsoïdes.

La figure 1.9 montre un résultat obtenu par les auteurs.

---

[3] M. Cherif and K. Gupta. 3D in-hand manipulation planning. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)*, pages 146–151, octobre 1998.

[4] M. Cherif and K. Gupta. Global planning for dextrous re-orientation of rigid objects : Finger tracking with rolling and sliding. *Rapport de recherche RR-3770, INRIA Rhône-Alpes*, septembre 1999.

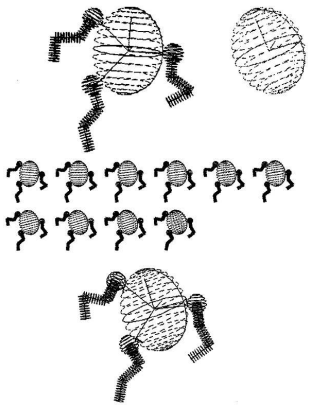


FIGURE 1.9 – Un exemple de réorientation d'un ellipsoïde avec trois doigts à quatre degrés de liberté (tiré de [CG98]).

Les approches issues de la théorie du contrôle Différents travaux [Goo98, GB02, HLP02, Zef02][1,2,3,4] traitent du problème de la planification de tâches de manipulation dextre avec des techniques qui sont celles de la théorie du contrôle des systèmes non linéaires et plus précisément des systèmes hybrides. Un système hybride est un système faisant intervenir simultanément des phénomènes ou des modèles continus et événementiels. Ces systèmes sont constitués de processus continus interagissant avec ou supervisés par des processus discrets. Le système constitué par une main robot manipulant un objet peut être vu comme un système hybride dont les processus continus correspondent aux mouvements possibles pour une prise donnée et les processus discrets sont les ruptures de contact (changement de prise). Les méthodes employées sont alors celles des systèmes stratifiés. Les systèmes stratifiés sont des systèmes dont l'espace des configurations est constitué de différentes surfaces appelées strates, qui s'intersectent pour former d'autres strates de dimension inférieure. Si le système est composé de deux doigts et d'un objet, par exemple, il y aura quatre strates correspondant aux situations suivantes : un des deux doigts est en contact avec l'objet (deux strates), aucun doigt n'est en contact avec l'objet (une strate) et les deux doigts sont

[1] B. Goodwine. Stratified motion planning with application to robotic finger gaiting. *Proceedings of the IFAC World Congress*, 1998.

[2] B. Goodwine and J. Burdick. Motion planning for kinematic stratified systems with application to quasi-static legged locomotion and finger gaiting. *IEEE Transactions on Automatic Control*, 18(2) :209–222, 2002.

[3] I. Harmati, B. Lantos, and S. Payandeh. On fitted stratified and semi-stratified geometric manipulation planning with fingertip relocations. *The International Journal of Robotics Research*, 21(5-6) :489–510, juin 2002.

[4] M. Zefran. A feedback strategy for dextrous manipulation. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2002)*, 3 :2479–2484, 2002.

en contact avec l'objet (une strate). Les strates sont ordonnées de la plus basse (la plus contrainte) à la plus haute (la moins contrainte). Les contraintes correspondant à chaque strate sont différentes et, par conséquent, les équations du mouvement associées à chaque strate sont aussi différentes.

Le principe du contrôle des systèmes stratifiés est de définir un espace d'état commun où tous les champs de vecteurs, qui sont définis pour chaque strate, peuvent être considérés. Cet espace d'état utilise des entrées fictives. Ces entrées peuvent être la configuration de l'objet (six paramètres) et les positions des bouts des doigts ([Goo98, GB02]) ou la configuration de l'objet et les positions des bouts des doigts relatives à la surface de l'objet (deux coordonnées sur la surface plus la distance par rapport à la projection du point sur la surface) ([HLP02]). Ces entrées sont fictives car les vraies entrées sont les paramètres articulaires des doigts. Une fois la trajectoire trouvée, il faut donc calculer ces paramètres à partir des entrées fictives, ce qui se fait à l'aide des modèles cinématiques inverses.

Pour amener l'objet d'une configuration à une autre, ces méthodes de planification de mouvement stratifié calculent une vitesse qui permettra le changement de configuration en un temps donné. Les positions des bouts des doigts constituent une autre entrée du système. Elles doivent être choisies de sorte qu'elles resteront dans les domaines d'accessibilité des doigts le long du mouvement et permettront d'assurer la propriété de fermeture de force de la prise. La figure 1.10 montre un résultat obtenu par Goodwine [Goo98]. Il s'agit de faire subir à un objet en forme d'oeuf une rotation de 2π autour de l'axe $(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$.

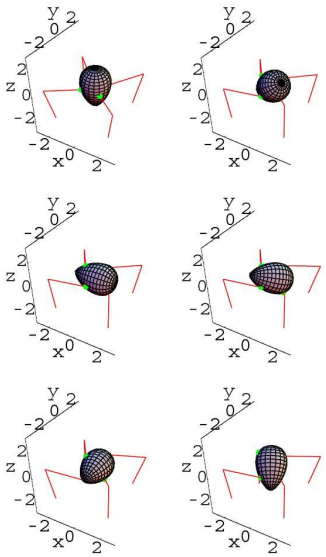


FIGURE 1.10 – Un exemple de réorientation d'un objet en forme d'oeuf avec quatre doigts à trois degrés de liberté (tiré de [Goo98]).

Ces différentes méthodes déterminent quand reconfigurer la prise et calculent les mouvements de repositionnement des doigts. Par contre, elles ont besoin de recevoir en entrée la trajectoire de l'objet. C'est pourquoi elles doivent être utilisées en complément d'une autre méthode de planification.

Les méthodes par construction de graphes probabilistes Des travaux de planification de mouvement de manipulation dextre ont employé des méthodes probabilistes [KSLO96, Sve96, LaV98][1,2,3]. Il s'agit des travaux de Yashima *et al.* [YY02, YSY03][4,5] qui présentent des points communs avec la méthode de Chérif et Gupta mais s'appuient sur des techniques de planification plus récentes. L'espace des configurations choisi pour leur méthode est celui de l'objet, c'est-à-dire $SE(3)$. Leur planificateur opérant sur deux niveaux. Le niveau supérieur planifie la trajectoire de l'objet et utilise une méthode fonctionnant selon le principe de la méthode RRT monodirectionnelle [LaV98][3] (voir partie 1.5.2). La méthode concerne la construction d'un arbre partant d'une configuration initiale X_{init} et dont les noeuds sont des configurations de l'objet. L'arbre est développé itérativement en partant de la configuration X_{init} et il a pour but d'atteindre la configuration X_{goal}. A chaque itération de l'algorithme, une nouvelle configuration de l'objet, X_{rand}, est générée aléatoirement. L'algorithme cherche alors à connecter le noeud de l'arbre le plus proche de X_{rand} (figure 1.11). Ce noeud le plus proche est noté X_{near}. La fonction distance utilisée pour déterminer le noeud le plus proche est une distance entre deux configurations de l'objet. Cette dernière est calculée en sommant les distances entre les positions et entre les orientations correspondantes de l'objet. C'est le planificateur local qui se charge d'essayer de relier les noeuds. Il commence par choisir aléatoirement une combinaison de modes de contact pour les doigts, parmi les différents modes possibles qui sont au nombre de huit. Ces modes sont obtenus en imposant la nullité de certaines composantes de la vitesse relative de l'objet et du doigt au point de contact. On a ainsi des contacts avec glissement pur, si on impose seulement la nullité de la composante normale de la vitesse au point de contact, des contacts avec roulement, si on impose également la

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

[2] P. Svestka. On probabilistic completeness and expected complexity of probabilistic path planning. *Technical Report*, 1996.

[3] S.M. LaValle. Rapidly-exploring random trees : a new tool for path planning. Technical Report 98-11, Computer Science Dept., Iowa State University, octobre 1998.

[4] M. Yashima and H. Yamaguchi. Dynamic motion planning whole arm grasp systems based on switching contact modes. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA'2002)*, mai 2002.

[5] M. Yashima, Y. Shiina, and H. Yamaguchi. Randomized manipulation planning for a multi-fingered hand by switching contact modes. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA'2003)*, septembre 2003.

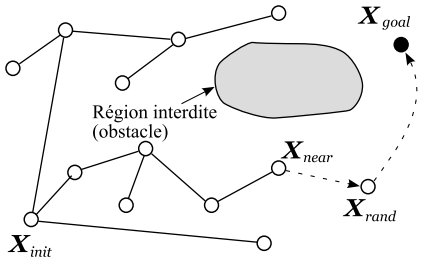


FIGURE 1.11 – Utilisation de l'algorithme RRT dans la méthode de Yashima *et al.* (tiré de [YSY03])

nullité des composantes tangentielles de la vitesse, etc. Un petit intervalle de temps est également tiré aléatoirement. La trajectoire locale de l'objet est choisie comme étant un polynôme cubique reliant les deux configurations de l'objet tel que les vitesses au début et à la fin de l'intervalle de temps soient nulles. Ensuite, on calcule à l'aide du modèle dynamique inverse les valeurs des couples à exercer au niveau des liaisons des doigts, pour faire suivre à l'objet la trajectoire locale souhaitée, pour la combinaison de modes de contact précédemment choisie. S'il n'y a pas de solution à ce problème inverse qui respecte les contraintes cinématiques et dynamiques du système, les deux configurations ne sont pas reliées dans le graphe et le planificateur global tire au hasard une nouvelle configuration de l'objet. Choisir régulièrement entre les différents modes de contact permet de changer les contraintes cinématiques. En effet, certains obstacles de l'espace des configurations, dûs à des contraintes cinématiques disparaîtront si on change de mode de contact comme l'illustre la figure 1.12.

Les repositionnements des points de contact par lever de doigt ne sont pas gérés. Ils

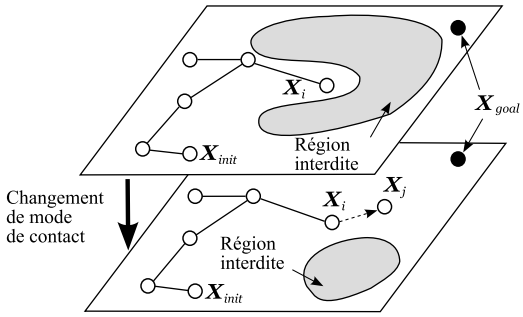


FIGURE 1.12 – Changer de modes de contact permet de changer la forme des obstacles de l'espace des configurations et par conséquent de les contourner éventuellement (tiré de [YSY03]).

sont seulement possibles si on fait glisser les doigts sur la surface de l'objet. De même les contraintes de non-collision ne sont pas prises en compte par la méthode. Ces dernières pourraient néanmoins être introduites dans le planificateur local.

La figure 1.13 montre un exemple de résultats obtenu par le planificateur de Yashima. Il s'agit de translater selon un vecteur $(0.02, 0.02, 0.02)$ et de faire tourner d'un angle de 30^o autour de l'axe $(-0.5, 0.5, 0.707)$ un objet ellipsoïdal de rayons $(0.2, 0.2, 0.16)$. La main a trois doigts pourvus de phalanges terminales plates. Chaque doigt a deux degrés de liberté.

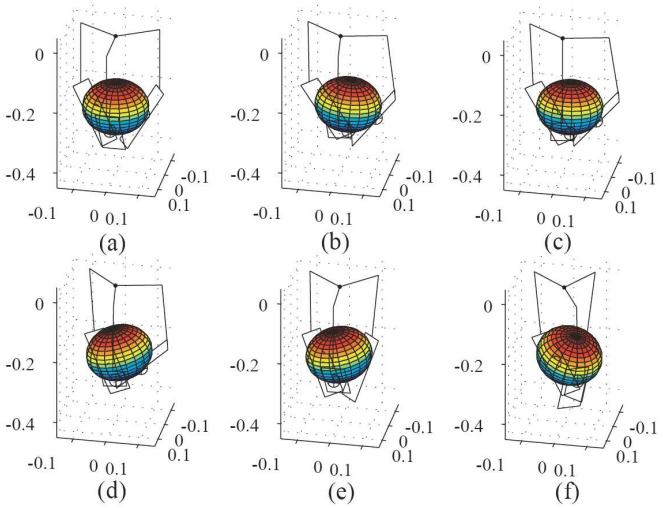


FIGURE 1.13 – Un exemple de réorientation d'un objet ellipsoïdal par la méthode de Yashima (tiré de [YSY03]).

Le défaut majeur de la méthode de Yashima est qu'elle conduit à des temps de calcul très importants. A chaque tentative de connexions de deux noeuds du graphe, le problème dynamique inverse est résolu, or c'est un problème très coûteux en terme de calculs. Ainsi l'exemple de la figure 1.13 —pourtant très simple— nécessite en moyenne 154 minutes pour un programme écrit avec Matlab, exécuté sur un PC équipé d'un processeur cadencé à 1,4GHz et de 256Mo de RAM.

1.2 Analyse

La partie précédente a présenté l'essentiel des techniques proposées jusqu'à aujourd'hui pour le problème de la planification de tâches de manipulation dextre. Elles font toutes appel à des idées différentes ; l'utilisation de graphes, bien que de significations variées, est néanmoins quasiment toujours présente –sauf dans les techniques venant de la théorie du contrôle–, comme dans la plupart des méthodes de planification de mouvement en général. Un défaut commun aux méthodes présentées est qu'aucune ne considère tous les aspects du problème, même d'un point de vue purement cinématique. Cela s'explique par le besoin de simplifier le problème. On retrouve ainsi une

simplification particulièrement gênante dans la plupart des méthodes analysées plus haut, qui est de poser comme préalable le calcul de la trajectoire de l'objet. On sait en effet qu'il n'y a aucune garantie sur la convergence d'un algorithme de planification s'il considère séparément les différentes parties d'un même système. Dans le cas de la manipulation dextre, la trajectoire que doit suivre l'objet dépend fortement de la forme de celui-ci et de la géométrie de la main. Un algorithme calculant d'abord la trajectoire de l'objet risque fort de calculer une trajectoire qui rende l'objet inaccessible à la main.

Une autre simplification fréquente est de considérer les doigts comme des points ou des sphères ([HLMT90, Rus97, Rus98, CG99, OF96, Goo98, SP03, HT98]), ce qui empêche de prendre en compte les limites des domaines accessibles des doigts et les collisions entre les doigts ou entre les doigts et l'objet. On met ainsi de côté l'une des raisons principales de la planification de mouvement de tâches de manipulation dextre, qui est d'assurer que les trajectoires des doigts n'entraîneront pas de collision indésirables.

La forme de l'objet manipulé est aussi régulièrement soumise à des contraintes fortes. Les méthodes de la littérature font généralement l'hypothèse que l'objet a une surface paramétrée ou simplement polyédrique. Dans le premier cas, il faut pouvoir calculer facilement, sur la surface de l'objet, des points ou des trajectoires (glissement des doigts). Les exemples proposés concernent alors des sphères ([HT98, Zef02, HLT+00, XL05]), des ellipsoïdes ([CG98, CG99, YY02, YSY03]) ou des objets en forme d'oeuf ([Goo98, GB02, HLP02]). Beaucoup d'objets du quotidien présentent des arêtes ou des surfaces non paramétrables et ne pourraient donc pas être manipulés avec de telles méthodes. Dans le second cas ([TH91, OF96, Rus97, SP03]), la complexité des calculs des algorithmes de planification dépend directement du nombre de faces (ou d'arêtes dans le cas plan) de l'objet. Ainsi les exemples présentés montrent des formes polyédriques très simples, souvent pour des problèmes plans.

Enfin, la complexité de certaines méthodes les rend tout simplement inapplicables dans la plupart des cas. C'est le cas des méthodes qui nécessitent une représentation explicite de l'espace des configurations ([Rus97, Rus98]) ou des calculs symboliques qui peuvent être très complexes ([Goo98, GB02, HLP02, Zef02]).

Le développement de la méthode proposée dans cette thèse aura donc été motivé par la volonté de remédier aux inconvénients des méthodes existantes présentés ci-dessus. Les caractéristiques dont nous avons souhaité la voir pourvue peuvent se résumer comme suit :

- le calcul de la trajectoire de l'objet doit tenir compte de la géométrie de la main pour garantir qu'une solution au problème de planification sera trouvée,
- les collisions entre tous les corps (paume de la main, doigts , objet manipulé, obstacles éventuels de l'environnement) doivent être évitées,
- les domaines d'accessibilité des doigts doivent être pris en compte,
- les hypothèses sur la forme de l'objet doivent être le moins contraignantes possible (idéalement il ne doit pas y en avoir),

– la méthode de planification doit être suffisamment simple pour pouvoir être implémentée et testée sur des exemples “réalistes”.

Devant la complexité du problème, nous serons cependant amener à émettre, par ailleurs, des hypothèses simplificatrices concernant des points ne faisant pas partie de ceux énumérés ci-dessus comme l'expliquerons les chapitres 2 et 3. Convaincus que tous ces problèmes ne pouvaient être résolus sans revenir au problème de base qui est la planification de mouvement, nous nous sommes intéressés à ce sujet de façon générale mais aussi à ces aspects les plus susceptibles de nous être utiles pour la manipulation dextre. La méthode proposée dans le chapitre 3 utilise des techniques issues du domaine de la planification de mouvement. C'est pourquoi, la partie suivante présente une synthèse des techniques de planification de mouvement, s'attardant plus particulièrement sur celles qui ont été utilisées par notre méthode ou qui l'ont inspirée, à savoir les méthodes probabilistes et les méthodes de planification pour la manipulation à l'aide d'un bras robotisé.

1.3 La planification de mouvement

La planification de tâches de manipulation dextre est un cas particulier de planification de mouvement. La planification de mouvement peut concerner tout système mécanique, ou robot, composé d'un ou de plusieurs corps, devant trouver son chemin dans un environnement pouvant être encombré d'obstacles fixes ou mobiles. A la notion déjà introduite d'espace des configurations $\mathcal{CS}$, il est nécessaire de rappeler celle d'espace des configurations libres, appelé $\mathcal{CS}_{free}$. Cet espace est le sous-espace de $\mathcal{CS}$ formé par toutes les configurations sans collision aussi appelées configurations libres. Une configuration libre est une configuration qui n'est pas rendue invalide par la présence d'un obstacle. Un obstacle peut être un objet, qui va bloquer le mouvement du système, une limitation telle qu'une butée articulaire ou, suivant le problème, la violation d'une contrainte. La planification de mouvement est la recherche d'un chemin dans $\mathcal{CS}_{free}$ reliant deux points donnés de cet espace. Ce chemin devra éventuellement être soumis à certaines contraintes, telles que des contraintes sur la vitesse ou la dynamique du système. Nous reprenons la classification de LaValle [LaV06][1] pour présenter les méthodes de planification de mouvement selon trois catégories principales :

– les méthodes combinatoires ou méthodes exactes
– les méthodes heuristiques
– les méthodes par échantillonnage de l'espace des configurations

[1] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Consultable en ligne : http ://planning.cs.uiuc.edu/.

1.3.1 Les méthodes exactes

Ces méthodes sont complètes, c'est-à-dire qu'elles déterminent en un temps fini s'il existe une solution au problème de planification et le cas échéant trouve cette solution en un temps fini. Elles s'appuient sur une représentation de $\mathcal{CS}_{free}$ sans faire aucune approximation. La nécessité de construire une telle représentation rend les méthodes exactes uniquement applicables à des systèmes simples. Par ailleurs, les temps de calcul associés à l'usage de ces méthodes croissent exponentiellement avec le nombre de degrés de liberté du système étudié. Elles sont inutilisables, en pratique, pour des systèmes à plus de quatre degrés de liberté. Elles sont par conséquent inadaptées au problème de la manipulation dextre mais ont été employées avec succès pour des problèmes tels que la navigation d'un robot mobile dans le plan. On compte parmi ces techniques des méthodes comme la décomposition cellulaire ou les graphes de visibilité [LaV06][1].

1.3.2 Les méthodes heuristiques

Ces méthodes sont basées sur l'optimisation d'une fonctionnelle, dont le but est d'amener le robot vers sa configuration finale désirée. Le choix de la fonctionnelle dépend du système considéré. Le problème de planification se ramène à un problème d'optimisation sous contrainte. Dans le cas de la méthode des potentiels, par exemple [Kha86][2], le but est de minimiser la distance du robot à sa position finale. On simule alors l'application d'une force fictive sur le robot qui le pousse vers sa destination. Des forces répulsives sont générées pour le tenir éloigné des obstacles.

Ces méthodes présentent des problèmes de convergence quand la fonctionnelle possède des minima locaux. C'est pourquoi, elles sont plutôt utilisées comme méthodes locales, c'est-à-dire pour relier deux configurations proches.

1.3.3 Les méthodes par échantillonnage

Les méthodes par échantillonnage réalisent une simplification de l'espace des configurations en choisissant un nombre fini de configurations pour le représenter. L'espace des configurations approché est alors un ensemble de points de $\mathcal{CS}$. L'algorithme de planification va déterminer comment ces points peuvent être reliés par des chemins. Cela mènera à l'obtention d'un graphe dont les noeuds seront des configurations de $\mathcal{CS}_{free}$ et les arêtes des chemins dans ce sous-espace. La façon dont on échantillonne $\mathcal{CS}$ influe directement sur la façon dont se développe le graphe. C'est pourquoi des critères de qualité de l'échantillonnage ont été introduits. De manière générale, si on souhaite prélever un ensemble P de N échantillons d'un espace $X = [0,1]^d \subset \mathbb{R}^d$,

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

trois critères peuvent être pris en considération pour caractériser la qualité de l'échantillonnage P [LBL04][1] :

- l'uniformité : l'échantillonnage couvre l'espace des configurations sans laisser de zones non ou trop couvertes. Il optimise alors une mesure d'uniformité comme la dispersion [2] ou la disparité [3].
- la structure en treillis : la répartition des échantillons suit une structure particulière qui permet de trouver facilement les voisins d'un échantillon donné.
- la qualité incrémentale : la qualité de la couverture de l'échantillonnage est indépendante du nombre d'échantillons tirés. Autrement dit, si on interrompt l'échantillonnage on aura une répartition uniforme des points pour le nombre d'échantillons tirés.

Il existe deux familles de techniques d'échantillonnage : les échantillonnages déterministes et les échantillonnages probabilistes. Elles offrent des performances différentes au regard des trois critères énoncés. Le lecteur trouvera en annexe de ce document plus de détails sur ces types d'échantillonnage et leurs qualités et défauts. Pour résumer, l'échantillonnage déterministe offre de meilleures performances pour les trois critères énoncés plus haut. Cependant il est souvent moins simple à mettre en oeuvre et, surtout, si les différentes techniques d'échantillonnage déterministe existantes permettent de calculer des suites de points uniformes dans $\mathbb{R}^d$, leur application à d'autres espaces topologiques n'est pas toujours possible. En robotique, les espaces auxquels on s'intéresse sont généralement des produits cartésiens de nombreux espaces donc de la forme $E = E_1 \times E_2 \times \ldots \times E_n$. Même si on sait échantillonner uniformément chacun des E_i à l'aide de méthodes déterministes, on ne saura pas forcément échantillonner uniformément E. L'échantillonnage aléatoire ne présente pas ce défaut. Si on peut générer des suites aléatoires uniformes de points $x_1, x_2, \ldots, x_n$ dans les espaces $E_1, E_2, \ldots, E_n$ on peut alors générer une suite aléatoire uniforme dans $E = E_1 \times E_2 \times \ldots \times E_n$ en prenant simplement les points $(x_1, x_2, \ldots, x_n)$. C'est ce qui rend les échantillonnages aléatoires particulièrement pratiques et a assuré leur grand succès, ainsi que le fait que les fonctions de génération de nombres aléatoires soient très répandues et faciles d'utilisation. Autour du principe de l'échantillonnage

2. La dispersion δ correspond au rayon de la plus grande boule ne contenant aucun échantillon et dépend de la métrique ρ choisie, une métrique étant une fonction définissant une distance entre les éléments d'un ensemble : $\delta(P, \rho) = \sup_{q \in X} \{\min_{p \in P} \rho(q, p)\}$

3. Soit $\mathcal{R}$ une famille de sous-ensemble de X (par exemple l'ensemble des boules) et μ une mesure (généralisation de la notion de volume à un espace métrique quelconque), la disparité est alors $D(P, \mathcal{R}) = \sup_{R \in \mathcal{R}} \left| \frac{|P \bigcap R|}{N} - \frac{\mu(R)}{\mu(X)} \right|$ où $|\cdot|$ s'applique à un ensemble dénombrable et vaut son nombre d'éléments.

[1] S.M. LaValle, M.S. Branicky, and S. R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics Research*, 23(7/8) :673–692, juillet/août 2004.

aléatoire de l'espace des configurations, se sont construites différentes techniques, appelées de façon générale méthodes probabilistes, qui ont connu un essor important dans la communauté robotique.

Les méthodes probabilistes ont été introduites, au milieu des années 90, par Kavraki et Latombe [KSLO96] sous le nom PRM (Probabilistic Roadmap Method) et par Švestka et Overmars [Or94] sous le nom PPP (Probabilistic Path Planner). Elles échantillonnent l'espace des configurations à l'aide de tirages aléatoires. Les méthodes probabilistes sont complètes en probabilité, ce qui signifie que la probabilité qu'elles trouvent une solution, s'il en existe, tend vers 1 avec le nombre de points tirés. Comme le nombre de points croît strictement avec le temps d'exécution de ces méthodes, elles garantissent de trouver une solution, s'il en existe, en un temps fini. Leur grand avantage est leur adaptabilité et leur simplicité d'utilisation et surtout l'efficacité qu'elles ont montré dans la résolution de problèmes difficiles (systèmes comportant un grand nombre de degrés de liberté).
Les algorithmes de planification probabilistes travaillent en trois étapes successives : construction d'un graphe, recherche d'un chemin dans le graphe et optimisation du chemin. Selon que ce graphe, obtenu pour un environnement donné, peut être utilisé pour résoudre un seul ou plusieurs problèmes de planification, on parlera de méthode mono-requête ou multi-requête. Comme le planificateur du chapitre 3 utilise des méthodes probabilistes, nous nous attardons sur ces méthodes de planification de mouvement en les présentant en détail dans la partie suivante.

1.4 Les méthodes probabilistes

1.4.1 Construction du graphe ou étape d'apprentissage

Cette étape concerne la construction d'un graphe [4] dont les noeuds sont des configurations de $\mathcal{CS}_{free}$ et les arêtes des chemins réalisables dans $\mathcal{CS}_{free}$. Ce graphe doit capturer le mieux possible la connexité de $\mathcal{CS}_{free}$ c'est-à-dire qu'idéalement à chaque composante connexe [5] de $\mathcal{CS}_{free}$ doit être associée une composante connexe du graphe [6]. Par ailleurs, on souhaite que toute configuration de $\mathcal{CS}_{free}$ puisse être "facilement" connectée à un des noeuds du graphe. Une fois tirée au hasard une configuration dans

4. De manière générale, un graphe est un ensemble de noeuds dont certains sont connectés entre eux par des arcs ou arêtes. Les arêtes traduisant l'existence d'une relation entre les noeuds qu'elles connectent.

5. Nous utilisons le terme connexe dans le sens connexe par arcs. Un espace connexe par arcs est un espace tel qu'on peut relier deux quelconques de ses points par un chemin. Un chemin γ d'origine x et d'extrémité y, dans un espace E, est une application continue $\gamma : [0,1] \rightarrow E$ telle que $\gamma(0) = x$ et $\gamma(1) = y$.

6. Une composante connexe d'un graphe est un ensemble de noeuds accessibles les uns depuis les autres via des arêtes.

$\mathcal{CS}$, on fait appel à une fonction appelée *détecteur de collision* pour déterminer si elle appartient à $\mathcal{CS}_{free}$. On cherche ensuite à connecter les noeuds à l'aide de la *méthode locale*. La méthode locale est une fonction calculant une trajectoire entre deux configurations et vérifiant qu'elle est sans collision. La façon dont on essaye de connecter les noeuds du graphe dépend de la méthode probabiliste choisie. De même, suivant la méthode, il existe différents critères d'arrêt de construction du graphe. On peut, par exemple, décider d'arrêter la construction du graphe une fois que celui-ci contient un nombre donné de noeuds.

1.4.2 Recherche d'un chemin dans le graphe ou étape de recherche

Une fois le graphe construit, les configurations initiale et finale sont connectées au graphe avec la méthode locale puis on cherche dans le graphe un chemin qui les relie. Si une de ces configurations ne peut être reliée au graphe, celui-ci doit être enrichi. Pour cela, il faut alors poursuivre l'étape de construction. Dans le cas où configurations initiale et finale ont pu être reliées au graphe, il faut trouver quelle suite de noeuds et d'arêtes emprunter pour relier ces deux configurations. Cette étape correspond simplement à la recherche d'un chemin dans un graphe (figure 1.14). Différents algorithmes existent pour effectuer cette recherche. Si on a, au préalable, associé un coût positif ou longueur à chaque arête du graphe, on peut choisir de chercher le plus court chemin. Les algorithmes les plus utilisés sont l'algorithme de Dijkstra [Dij59][1] et l'algorithme A* [HNR68][2].

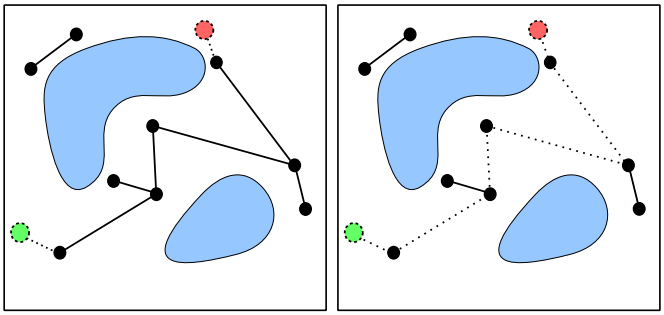


FIGURE 1.14 – Les configurations initiale et finale sont ajoutées au graphe puis on cherche dans le graphe un chemin qui les relie (dans cet exemple, il n'y a qu'un chemin possible).

[1] E.W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1 :269–271, 1959.

[2] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2) :100–107, 1968.

1.4.3 Optimisation du chemin ou étape de lissage

Une fois trouvé un chemin entre les configurations initiale et finale, une étape de lissage est effectuée dont le but est d'obtenir une trajectoire plus courte et plus régulière. Les chemins obtenus par l'étape de recherche ont généralement un aspect très tortueux en raison de l'utilisation de tirages aléatoires dans l'étape de construction du graphe. L'étape de lissage aussi peut se faire à l'aide de tirages aléatoires [LJTM94][3] : deux configurations sont tirées aléatoirement sur le chemin (figure 1.15), puis on cherche à relier directement ces deux configurations à l'aide de la méthode locale. Si le chemin obtenu est plus court, il remplacera le précédent. Cette étape de lissage est répétée un nombre donné de fois ou jusqu'à ce qu'elle n'apporte plus de raccourcissement supérieur à un seuil désiré.

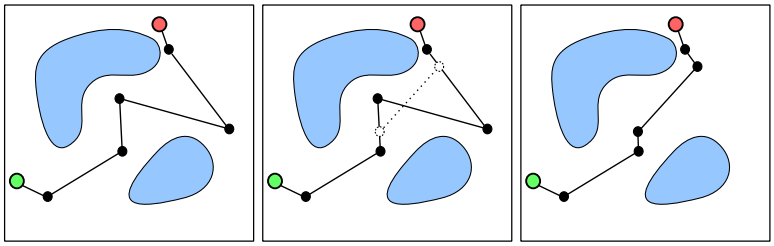


FIGURE 1.15 – Deux configurations sont tirées aléatoirement sur le chemin reliant les configurations initiale et finale, puis connectées directement.

Les différences existant entre les différentes méthodes probabilistes interviennent au niveau de la phase de construction du graphe. Nous présentons maintenant les méthodes principales.

1.5 La méthode PRM

Il s'agit d'une méthode multi-requête qui est aussi une des premières méthodes probabilistes. Elle fût également introduite sous le nom PPP. Ces méthodes sont multi-requêtes car le graphe construit durant la phase d'apprentissage peut être réutilisé pour différents problèmes de planification, tant que les obstacles ne changent pas. L'algorithme 1 décrit le fonctionnement de la méthode PRM. A chaque itération de l'algorithme, une configuration q_{rand} est tirée au hasard, puis sa validité est testée à l'aide du détecteur de collision. Si la configuration q_{rand} appartient à $\mathcal{CS}_{free}$, elle constitue un nouveau noeud du graphe et on essaye de le connecter à un des autres noeuds du

[3] J.-P. Laumond, P.E. Jacobs, M. Taix, and R.M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 10(5) :577–593, 1994.

graphe. Pour tester la possibilité de connecter deux noeuds, on utilise la méthode locale. V est l'ensemble des noeuds du graphe et E l'ensemble de ses arêtes. n_{max} est un critère d'arrêt, qui correspond au nombre de noeuds qu'on estime nécessaire pour que le graphe forme une bonne couverture de $\mathcal{CS}_{free}$ (voir partie 1.5.3).

Algorithme 1 : Algorithme de l'étape d'apprentissage de la méthode PRM

$V = \emptyset$
$E = \emptyset$
tant que $nb_{noeuds} < n_{max}$ **faire**
    $q \leftarrow$ configuration tirée au hasard
    si $q \in \mathcal{CS}_{free}$ **alors**
        $nb_{noeuds} = nb_{noeuds} + 1$
        $V = V \cup \{q\}$
        $V_q =$ un ensemble de noeuds voisins de q
        pour chaque $q' \in V_q$ *par ordre de distance croissant de* q **faire**
            si *la méthode locale trouve un chemin valide entre* q *et* q' **alors**
                $E = E \cup \{q, q'\}$
        fin
fin

Une étape de l'algorithme consiste à déterminer l'ensemble V_q des noeuds voisins du noeud/configuration q. Il s'agit de choisir un ensemble de noeuds du graphe parmi ceux qui auront le plus de chance de pouvoir être connectés à q par la méthode locale. La détermination de cet ensemble V_q peut se faire de différentes manières : prendre les k plus proches voisins de q (avec $k \in \mathbb{Z}^*$ à choisir empiriquement) parmi les noeuds du graphe, prendre les k noeuds du graphe ayant le plus d'arêtes, etc.

1.5.1 La méthode visibility-PRM

Un des inconvénients majeurs de la méthode PRM est qu'elle conduit à des graphes qui peuvent contenir un très grand nombre de noeuds inutiles, c'est-à-dire qui n'apportent aucune information pertinente sur la topologie de l'espace des configurations. En effet, à chaque itération de l'algorithme, la nouvelle configuration générée sera systématiquement ajoutée au graphe à partir du moment où elle appartient à $\mathcal{CS}_{free}$. La méthode visibility-PRM [SLN00][1] permet de remédier à cet inconvénient en exploitant la notion de visibilité. Deux configurations sont dites visibles si la méthode locale trouve un chemin pour les relier. On peut alors associer à chaque configuration q un

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

domaine de visibilité, qui est l'ensemble des configurations vues par q. Ce domaine dépend des obstacles et de la méthode locale choisie. Sur la figure 1.16, les régions grises représentent l'ensemble des configurations non visibles par le point/configuration pour deux méthodes locales différentes. Le principe de visibility-PRM est de n'ajouter un

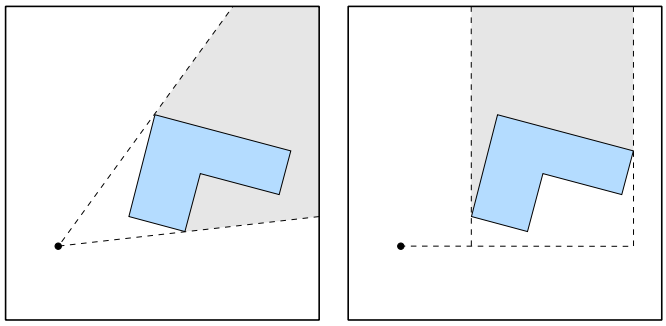


FIGURE 1.16 – Domaines de visibilité (zones blanches) en présence d'un obstacle pour deux méthodes locales différentes : (a) chemin linéaire (b) chemin Manhattan, qui correspond ici à un déplacement horizontal suivi d'un déplacement vertical.

noeud au graphe que s'il ne "voit" aucun noeud du graphe existant ou s'il voit au moins deux noeuds de deux composantes connexes différentes. Les noeuds du premier type sont appelés *gardiens*, ceux du second *connecteurs*. Les gardiens seront juste ajoutés au graphe et les connecteurs seront en plus connectés à un noeud de chacune des composantes connexes qu'ils voient, pour fusionner ces dernières (figure 1.17). Hormis la

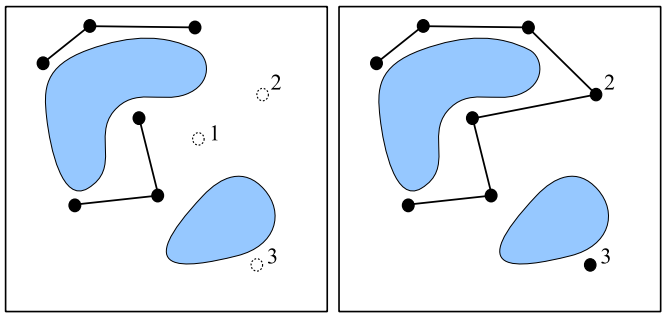


FIGURE 1.17 – Imaginons que l'on tire successivement les configurations 1, 2 et 3. 1 n'est ni gardien ni connecteur et sera rejetée. 2 est un connecteur et 3 un gardien. Ces deux dernières configurations sont ajoutées au graphe.

réduction du nombre de noeuds du graphe, un autre avantage de la méthode visibility-PRM est qu'on peut utiliser, comme critère d'arrêt de construction du graphe, le paramètre $ntry$ qui comptabilise le nombre de configurations qu'on a dû tirer aléatoirement

avant de trouver un nouveau gardien. Le rapport $\frac{1}{ntry}$ donne une estimation de la fraction du volume non couvert de $\mathcal{CS}_{free}$ sur son volume total. Tout comme la méthode PRM, visibility-PRM est multi-requête.

1.5.2 La méthode RRT

La méthode RRT [LaV98, LK00][1,2] est une méthode mono-requête très efficace. Il en existe différentes variantes : recherche avec un seul arbre [7], recherche bidirectionnelle avec deux arbres équilibrés, RRT rebound. Nous présentons la méthode RRT rebound bidirectionnelle, qui est celle utilisée dans la suite de cette thèse.
L'algorithme 2 détaille son fonctionnement. Le principe de cette méthode est de construire deux arbres. Un des arbres se développe à partir de la configuration initiale et l'autre à partir de la configuration finale. A chaque itération, une configuration q_{rand} est tirée au hasard dans $\mathcal{CS}$. On choisit d'abord celui des deux arbres qui est le plus "petit" selon un critère tel que le nombre total de noeuds ou la somme des longueurs de toutes les arêtes. On appellera arbre 1 l'arbre le plus petit et arbre 2 le plus grand. Dans l'arbre 1, on cherche q_{near}, le noeud le plus proche de q_{rand} selon une fonction distance adaptée au problème. On utilise la méthode locale pour calculer un chemin reliant q_{near} à q_{rand}. Si ce chemin existe, on ajoute q_{rand} à l'arbre 1. S'il n'existe pas, on connecte q_s, qui est la dernière configuration valide trouvée durant le calcul du chemin de q_{near} à q_{rand}. q_s est la configuration obtenue juste avant que le chemin bute sur l'obstacle, la configuration bloquante. Si $q_s = q_{near}$, l'égalité pouvant être considérée ici avec une certaine tolérance plutôt que strictement, c'est-à-dire si q_{near} est déjà tout contre l'obstacle, on tire un nouveau q_{rand}. Sinon, on effectue les mêmes opérations pour l'arbre 2. On obtient alors une configuration q'_s analogue de q_s pour l'arbre 2. Si $q_s = q'_s = q_{rand}$, les deux arbres sont connectés et on peut trouver un chemin reliant la configuration finale à la configuration initiale.
La figure 1.18 montre comment on obtient les différentes configurations mentionnées ci-dessus.

1.5.3 Quelques propriétés des méthodes probabilistes

Les méthodes probabilistes ont rencontré un grand succès dans le domaine de la planification de mouvement et par conséquent suscité de nombreux travaux d'analyse de leurs performances et propriétés. Deux notions primordiales sont associées

7. Un arbre est un graphe qui ne comporte pas de cycles. Il ne peut donc y avoir plusieurs chemins reliant deux noeuds d'un arbre.

[1] S.M. LaValle. Rapidly-exploring random trees : a new tool for path planning. Technical Report 98-11, Computer Science Dept., Iowa State University, octobre 1998.

[2] S.M. LaValle and J.J. Kuffner. Rapidly-exploring random trees : Progress and prospects. *Algorithmic Foundations of Robotics (WAFR'00)*, 2000.

Algorithme 2 : Algorithme de la méthode RRT bidirectionnel

```
1  Arbre1 = {q_initial}
2  Arbre2 = {q_final}
3  pour i de 1 à i_max faire
4      si Arbre2 = arbre_le_plus_petit(Arbre1, Arbre2) alors
5          permuter Arbre1 et Arbre2
6      q_rand ← configuration tirée au hasard
7      q_near ← plus_proche_voisin(Arbre1, q_rand)
8      q_s ← configuration_bloquante(q_near, q_rand)
9      si q_s ≠ q_near alors
10         ajoute_noeud(Arbre1, q_s)
11         ajoute_arete(Arbre1, {q_near, q_s})
12         q'_near ← plus_proche_voisin(Arbre2, q_rand)
13         q'_s ← configuration_bloquante(q'_near, q_rand)
14         si q'_s ≠ q_near alors
15             ajoute_noeud(Arbre2, q'_s)
16             ajoute_arete(Arbre2, {q'_near, q'_s})
17         si q'_s = q_s alors
18             on peut relier q_initial et q_final
19             FIN
20 fin
21 ECHEC
```

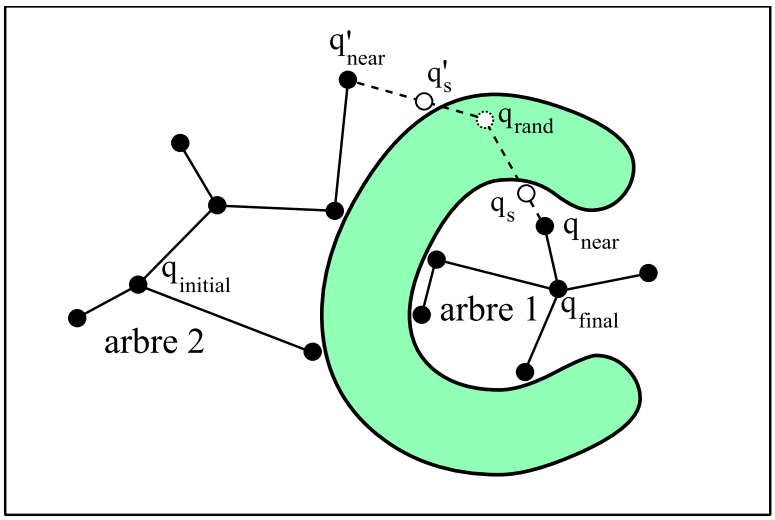


FIGURE 1.18 – Principe d'obtention des différentes configurations nécessaires au développement des arbres dans la méthode RRT rebound.

aux graphes probabilistes : la notion de couverture adéquate et celle de représentation adéquate.

La notion de couverture adéquate s'appuie sur celle de ϵ-goodness, introduite par Kavraki [KLMR95][1], qui permet d'évaluer la qualité du graphe en terme de couverture de $\mathcal{CS}_{free}$. L'ϵ-goodness de $\mathcal{CS}_{free}$ est défini par :

$$\epsilon(\mathcal{CS}_{free}) = \min_{q \in \mathcal{CS}_{free}} \left(\frac{\mu(V(q))}{\mu(\mathcal{CS}_{free})} \right) \tag{1.5}$$

où μ est une mesure, c'est-à-dire une fonction dont le but est d'étendre la notion de volume à un espace quelconque muni d'une métrique [LaV06][2] et $V(q)$ le domaine de visibilité de la configuration q. $\epsilon(\mathcal{CS}_{free})$ représente la fraction de $\mathcal{CS}_{free}$ vue par la configuration de $\mathcal{CS}_{free}$ qui a le plus petit domaine de visibilité. $\mathcal{CS}_{free}$ sera dit ϵ-good si $\epsilon(\mathcal{CS}_{free}) \neq 0$, autrement dit si toutes les configurations de $\mathcal{CS}_{free}$ voit une portion non nulle de cet espace. L'ensemble des noeuds d'un graphe fournit une couverture adéquate de $\mathcal{CS}_{free}$ si et seulement si la mesure du sous-ensemble de $\mathcal{CS}_{free}$ qui n'est visible d'aucun noeud est au plus $\frac{\epsilon(\mathcal{CS}_{free})}{2}\mu(\mathcal{CS}_{free})$. Plus $\epsilon(\mathcal{CS}_{free})$ sera grand, moins il faudra de noeuds pour obtenir une couverture adéquate de $\mathcal{CS}_{free}$. Les auteurs de [BKL+96][3] ont exprimé la probabilité de trouver une solution en fonction du nombre de noeuds du graphe et de $\epsilon(\mathcal{CS}_{free})$. Malheureusement, il est généralement très difficile voire impossible de déterminer $\epsilon(\mathcal{CS}_{free})$ puisqu'il faudrait pour cela connaître la structure de $\mathcal{CS}_{free}$. Cependant, leurs résultats montrent que bien que les méthodes probabilistes de type PRM ne garantissent pas l'obtention d'une couverture adéquate, la probabilité de ne pas l'obtenir décroît exponentiellement avec le nombre de noeuds engendrés.

La deuxième notion importante est celle de représentation adéquate. Elle est définie si $\mathcal{CS}_{free}$ est ϵ-good. Dans ce cas, un graphe G est une représentation adéquate de $\mathcal{CS}_{free}$ si ses noeuds fournissent une couverture adéquate de $\mathcal{CS}_{free}$ et s'il n'existe pas deux composantes connexes de G dans la même composante connexe de $\mathcal{CS}_{free}$. Si G est une représentation adéquate de $\mathcal{CS}_{free}$, alors à chaque composante connexe de $\mathcal{CS}_{free}$ correspond une composante connexe de G et une seule. On dit alors que G capture bien la connexité de $\mathcal{CS}_{free}$. Un résultat important [HKL+98][4] concernant les propriétés des méthodes probabilistes de type PRM est que la probabilité qu'une

[1] L. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot motion planning. *ACM Symposium on Theory of Computing*, pages 353–362, 1995.

[2] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Consultable en ligne : http ://planning.cs.uiuc.edu/.

[3] J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, and P. Raghavan. A random sampling scheme for robot path planning. *Proceedings of the International Symposium on Robotics Research*, pages 249–264, 1996.

[4] D. Hsu, L. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. *Robotics : The algorithmic perspective*, pages 141–143, 1998.

de ces méthodes génère un graphe ne capturant pas la connexité de $\mathcal{CS}_{free}$ décroît exponentiellement avec le nombre de noeuds de ce graphe.

Les méthodes probabilistes ont été utilisées avec succès pour résoudre des problèmes de planification de tâches de manipulation avec un bras robot, qui présentent des similitudes avec les problèmes de planification pour la manipulation dextre. Leur application se prête très bien à ce type de problème car ils mettent en jeu des systèmes possédant un grand nombre de degrés de liberté, ce qui rend l'emploi des méthodes exactes impossible. Nous présentons maintenant les approches qui ont été proposées dans la littérature, concernant la manipulation à l'aide d'un robot "simple", dont certaines idées nous ont servi pour notre méthode de planification pour la manipulation dextre.

1.6 La planification de mouvement pour les tâches de manipulation

La planification de tâches de manipulation concerne la génération automatique d'une suite de mouvements d'un robot permettant de déplacer un objet dans un environnement encombré d'obstacles. Il s'agit ici de manipulation de type *pick and place* c'est-à-dire de manipulation se faisant par une suite de saisie et de dépôt de l'objet (l'objet n'est pas maintenu tout au long de l'opération de manipulation) avec un préhenseur de type "pince". Ce qui distingue ce problème de la plupart des problèmes de planification de mouvement est que le robot a la possibilité de modifier son espace des configurations, en fonction de la façon dont il saisira l'objet et de l'endroit où il le déposera. Cette distinction a conduit à l'introduction de sous-espaces et de chemins particuliers [ASL89, ALS94][5,6].

1.6.1 Les sous-espaces $GRASP$ et $PLACEMENT$

Le système étudié se compose d'un robot $\mathcal{R}$, d'un objet $\mathcal{O}$, déplaçable par ce même robot, et d'obstacles fixes. $\mathcal{CS}_{free}$ est l'espace des configurations considérées comme valides. Dans beaucoup de problèmes de planification de mouvement, la condition de validité d'une configuration est de ne pas entraîner d'interpénétration des corps du système. Dans le cas de la manipulation, deux autres contraintes interviennent :

– l'objet, quand il n'est pas tenu par le robot, doit être dans une position stable, autrement dit posé de façon stable sur un support tel que le sol

[5] R. Alami, T. Siméon, and J.-P. Laumond. A geometric approach to planning manipulation tasks. the case of discrete placements and grasps. *Fifth International Symposium on Robotics Research*, 1989.

[6] R. Alami, J.-P. Laumond, and T. Siméon. Two manipulation planning algorithms. *Algorithmic Foundations of Robotics (WAFR'94)*, 1994.

– l'objet ne peut se déplacer seul ; tout mouvement de l'objet est induit par un mouvement du robot

Ainsi la seule contrainte géométrique de non-collision n'est pas suffisante pour définir $\mathcal{CS}_{free}$. Pour prendre en compte les contraintes énoncées plus haut, deux sous-espaces, dont l'union sera $\mathcal{CS}_{free}$, sont introduits.

– $PLACEMENT$, aussi noté $\mathcal{CP}$, est l'ensemble de toutes les configurations telles que l'objet est dans une position stable
– $GRASP$, aussi noté $\mathcal{CG}$, est l'ensemble de toutes les configurations telles que l'objet est saisi par le robot ; une configuration est dans $GRASP$ si les configurations respectives de l'objet et de la partie préhensile du robot vérifient une certaine relation géométrique ; si le robot et l'objet sont tous les deux polyédriques, par exemple, on peut considérer qu'une configuration appartient à $GRASP$ si elle implique un contact ponctuel entre ces deux corps ; il est possible de considérer des prises appartenant à un ensemble fini ou non [ALS94][1] ; seuls sont présentés ici les travaux concernant le deuxième cas, plus général

La figure 1.19 montre des exemples de configurations dans $\mathcal{CG}$ et $\mathcal{CP}$, dans le cas d'un système planaire composé d'un robot de forme triangulaire et d'un objet rectangulaire. Pour ce système, on considère que les configurations de prise sont celles qui correspondent à un placement relatif du robot et de l'objet tel que le robot établisse un contact ponctuel avec son sommet d'angle le plus aigu sur une des faces de l'objet. L'ensemble

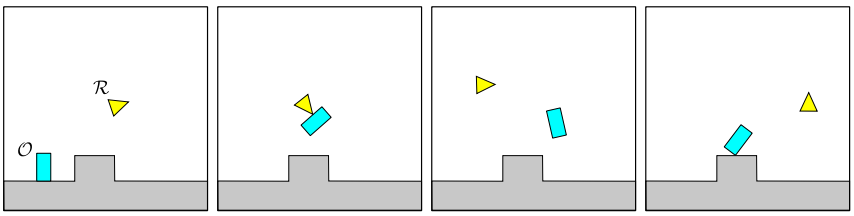


FIGURE 1.19 – Différentes configurations : (a) dans $PLACEMENT$ (b) dans $GRASP$ (c) et (d) invalides.

des configurations valides $\mathcal{CS}_{free}$ est l'union de $\mathcal{CG}$ et $\mathcal{CP}$. L'intersection $\mathcal{CG} \cap \mathcal{CP}$ est aussi intéressante car elle représente l'ensemble des configurations où l'objet est à la fois saisi et dans un placement stable. Ce sont les seules configurations dans lesquelles le robot peut saisir ou relâcher l'objet.

De la même façon que la nature des tâches de manipulation impose des contraintes sur les configurations valides, elle impose des contraintes sur les chemins admissibles.

[1] R. Alami, J.-P. Laumond, and T. Siméon. Two manipulation planning algorithms. *Algorithmic Foundations of Robotics (WAFR'94)*, 1994.

1.6.2 Les chemins de transit, de transfert et de manipulation

On peut définir deux types de chemin élémentaires :
- les *chemins de transit* correspondent à des déplacements du robot seul tandis que l'objet est immobile et dans une configuration stable. Un chemin de transit est un chemin particulier dans $\mathcal{CP}$
- les *chemins de transfert* correspondent à des déplacements du robot tenant l'objet. Un chemin de transfert est un chemin particulier dans $\mathcal{CG}$

La figure 1.20 montre des exemples de chemins de transit et de transfert, pour le système planaire décrit plus haut. Il est important de noter que deux configurations ne

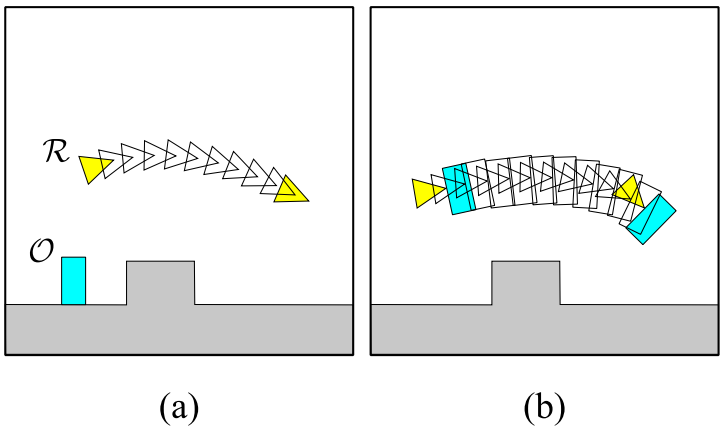


FIGURE 1.20 – chemins de transit (a) et de transfert (b).

peuvent être reliées par un chemin de transfert que si elles partagent une même configuration de prise et par un chemin de transit que si elles partagent un même placement de l'objet.
A partir des définitions des chemins de transit et de transfert, on définit un chemin de manipulation comme étant une suite finie de chemins de transit et de transfert. Le but de la planification d'une tâche de manipulation consiste à trouver un chemin de manipulation entre deux configurations données. Pour cela, il est nécessaire de construire un graphe de manipulation.

1.6.3 Construction du graphe de manipulation et propriété de réduction

Les seules configurations qui respectent à la fois les contraintes de placement et de mouvement sont celles du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ qui représente le lieu de connexion

des chemins de transit et des chemins de transfert. Pour résoudre le problème de manipulation, il faut donc explorer ce sous-espace. Pour cela, on capture la connexité de $\mathcal{CG} \cap \mathcal{CP}$ dans une structure de graphe. Ce graphe est appelé graphe de manipulation. Sa construction se fait en alternant les deux étapes suivantes :

- on explore $\mathcal{CG} \cap \mathcal{CP}$ à l'aide de chemins inclus dans ce sous-espace
- on essaye de fusionner les composantes connexes du graphe à l'aide de chemins de transit/transfert

L'exploration de $\mathcal{CG} \cap \mathcal{CP}$ se fait en construisant un graphe selon une technique de type PRM en utilisant comme méthode locale des chemins linéaires. De tels chemins correspondent à une variation simultanée de la configuration de l'objet et de la configuration de la prise. La figure 1.21(a) montre un tel chemin. Ces chemins violent la contrainte stipulant que les mouvements de l'objet sont induits par ceux du robot.

Cependant, une propriété très intéressante permet de justifier l'utilisation de ces chemins. Il s'agit de la propriété de réduction, introduite et démontrée dans [ALS94][1] et qui interviendra également dans la méthode du chapitre 3. Cette propriété établit qu'un chemin dans $\mathcal{CG} \cap \mathcal{CP}$ peut être décomposé en une suite finie de chemins de transfert et de transit, si le robot évite tout contact avec les objets le long de ce chemin. La figure 1.21 montre comment on peut transformer un chemin dans $\mathcal{CG} \cap \mathcal{CP}$, le long duquel changent à la fois le placement de l'objet et la prise sans que le robot entre en collision avec un obstacle, en une suite de chemins de transfert et de transit. Le nombre d'éléments de la suite transfert-transit dépend évidemment de l'encombrement de l'environnement autour du robot quand il parcourt le chemin dans $\mathcal{CG} \cap \mathcal{CP}$. Plus les obstacles éventuels sont proches, plus il faudra de chemins de transfert et de transit pour décomposer le chemin.

Pour fusionner les composantes connexes du graphe, deux configurations sont choisies dans deux composantes connexes distinctes et on essaye de les relier par un chemin de transfert suivi d'un chemin de transit, ou l'inverse. On peut choisir de privilégier l'étape d'exploration ou l'étape de fusion selon le problème.
Une fois les différentes composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$ explorées et éventuellement reliées à l'aide de chemin de transit-transfert, il sera possible de déterminer l'existence d'un chemin de manipulation entre les configurations initiale et finale et, le cas échéant, de trouver cette solution. La figure 1.22 montre la solution d'un problème de manipulation pour le système planaire ayant servi d'exemple jusqu'ici. Les configurations initiale et finale sont reliées à des configurations dans $\mathcal{CG} \cap \mathcal{CP}$ via des chemins de transit. Sur la figure, les chemins dans ce sous-espace ne sont pas décomposés, pour réduire le nombre d'étapes à représenter.

[1] R. Alami, J.-P. Laumond, and T. Siméon. Two manipulation planning algorithms. *Algorithmic Foundations of Robotics (WAFR'94)*, 1994.

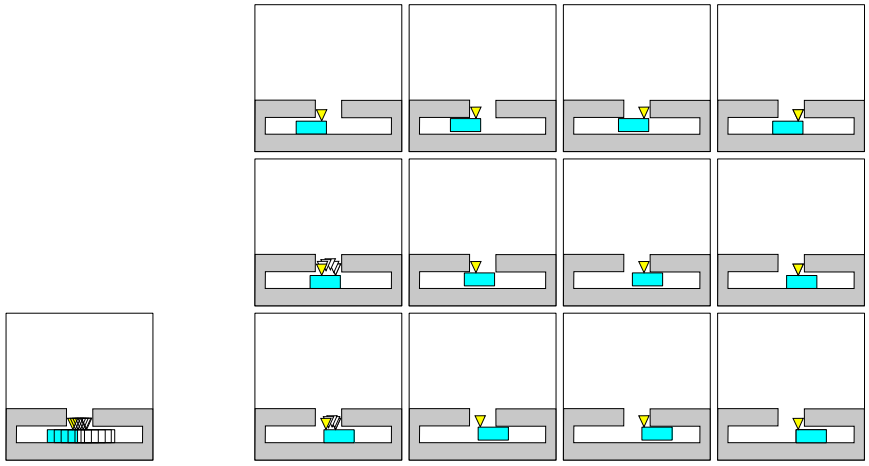


FIGURE 1.21 – Un chemin dans $\mathcal{CG} \cap \mathcal{CP}$ (a) et sa décomposition en suite transit-transfert (b).

1.6.4 Algorithmes de construction du graphe de manipulation

Les premières techniques de planification de tâches de manipulation concernaient des systèmes planaires simples comme celui présenté plus haut. Par conséquent, des méthodes de planification exactes pouvaient être employées comme la décomposition cellulaire [ALS94][1], contrairement à ce qu'il est possible de faire avec des systèmes plus complexes. Parmi ces systèmes, on trouve les bras manipulateurs, qui possèdent généralement un nombre important de degrés de liberté. Une autre spécificité de ce type de manipulateur est que leur base est toujours en contact avec le sol et qu'ils forment, par conséquent, une chaîne cinématique fermée avec l'objet saisi et le sol, pour les configurations dans $\mathcal{CG} \cap \mathcal{CP}$. La caractérisation de l'ensemble des configurations vérifiant la fermeture de chaîne est un problème encore non résolu aujourd'hui, sauf pour des systèmes planaires à liaison sphériques [TM02, LT05][2,3]. Pour toutes ces raisons,

[2] J.C. Trinkle and R.J. Milgram. Complete path planning for closed kinematic chains with spherical joints. *International Journal of Robotics Research*, 21(9) :773–789, septembre 2002.

[3] G.F. Liu and J.C. Trinkle. Complete path planning for planar closed chains among point obstacles. *Proceedings of Robotics : Science and Systems*, juin 2005.

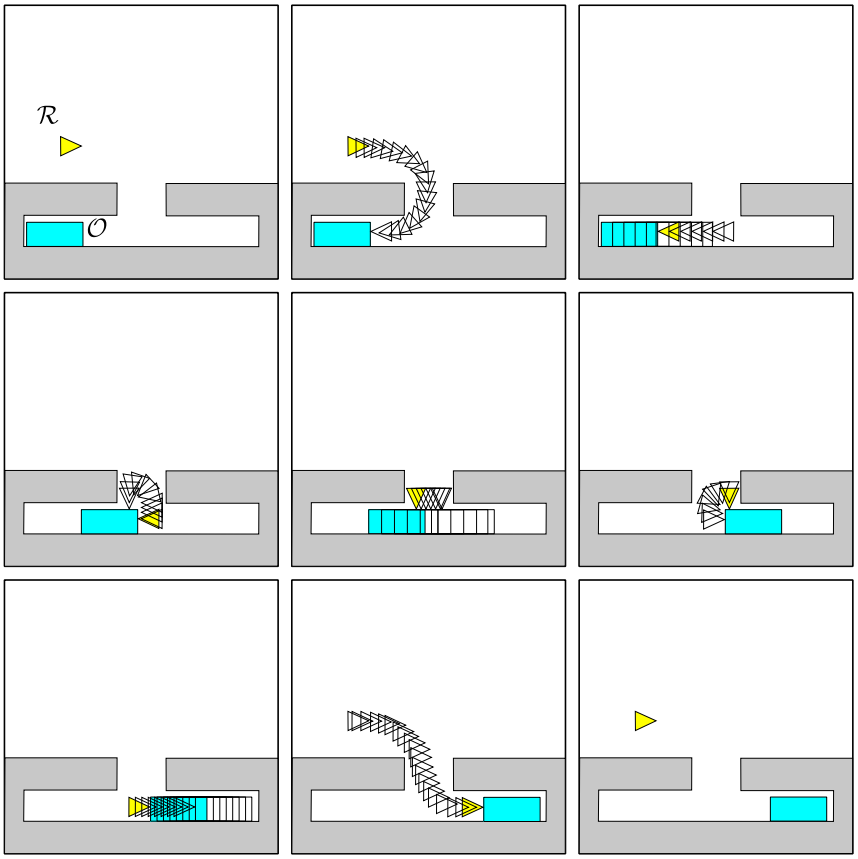


FIGURE 1.22 – Exemple de solution d'un problème de manipulation.

les auteurs de [SSC02, Sah03, SCSL03][1,2,3] ont opté pour les méthodes probabilistes pour planifier des tâches de manipulation avec un bras robot. Ils proposent d'explorer les différentes composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$ à l'aide de méthode de type PRM ou visibility-PRM. A chaque itération, l'algorithme de planification essaye, suivant une certaine probabilité qui dépend du problème, de relier ces différentes composantes via des chemins de transfert-transit. La figure 1.23 montre le résultat obtenu pour le problème qui consistait à sortir une barre d'une cage.

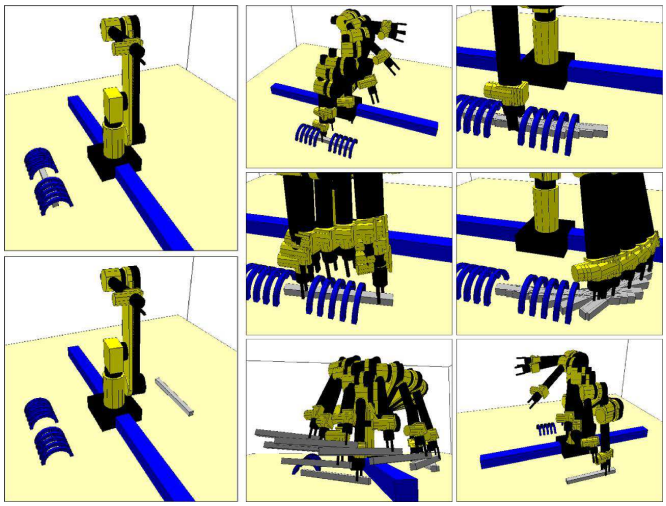


FIGURE 1.23 – Un exemple d'utilisation des méthodes probabilistes pour la manipulation d'une barre par un bras. Les configurations initiale et finale du problème sont montrées à gauche de la figure. Ce problème est intéressant car l'objet doit être déposé et ressaisi de nombreuses fois (tiré de [SSC02]).

1.6.5 Application à la manipulation dextre

Malheureusement, la théorie de planification pour la manipulation résumée dans la partie 1.6 ne peut pas être utilisée telle quelle pour la manipulation dextre. Une différence fondamentale est que l'objet doit toujours être tenu par la main. Par conséquent, le sous-espace $PLACEMENT$ devient sans signification. Une autre différence fondamentale est qu'il n'existe pas un "type" de prise unique mais plusieurs,

[1] A. Sahbani, T. Siméon, and J. Cortés. A probabilistic algorithm for manipulation planning under continuous grasps and placements. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'2002)*, 2002.

[2] A. Sahbani. Planification de tâches de manipulation en robotique par des approches probabilistes. *Thèse de doctorat de l'Université Paul Sabatier, Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS)*, 2003.

[3] T. Siméon, J. Cortés, A. Sahbani, and J.-P. Laumond. A general manipulation task planner. *Algorithmic Foundations of Robotics (WAFR'03)*, V :311–328, 2003.

suivant le nombre de doigts tenant l'objet. Il apparaît ainsi que $GRASP$ comportera de nombreuses composantes connexes devant chacune être explorée. D'autres spécificités sont à considérer comme la paramétrisation des prises ou la présence des nombreuses chaînes cinématiques fermées formées par les doigts et l'objet. Koga et Latombe [KL94][1] ont proposé une méthode de planification, utilisant le même formalisme que celui décrit plus haut, pour la manipulation à l'aide de plusieurs bras robotisés mais elle ne peut être appliquée à la manipulation dextre car un doigt ne peut maintenir rigidement un objet comme le peut un bras muni d'une pince. Il est donc nécessaire de proposer une formulation spécifique du problème de la manipulation dextre.

1.7 Conclusion

Ce chapitre a détaillé les différentes techniques existantes pour la planification de tâches de manipulation dextre et analysé leurs principaux défauts. L'analyse de ces défauts nous a conduit à replacer la planification pour la manipulation dextre dans le contexte, bien plus large, de la planification de mouvement. Certaines techniques de planification de mouvement pour la manipulation avec un bras manipulateur ont notamment été rappelées. Elles sont très efficaces mais ne peuvent être appliquées telles quelles pour la manipulation à l'aide d'une main mécanique. Notre souhait de s'inspirer de ces techniques nous a incité à adapter la formulation du problème. Cette nouvelle formulation est l'objet du chapitre suivant.

[1] Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA'94)*, 2 :945–952, 1994.

Chapitre 2

Formulation du problème et Espace des Configurations

Ce chapitre propose une nouvelle formulation du problème de la planification de tâches de manipulation dextre. Cette formulation concerne la description des différents éléments composant le système de manipulation formé par la main robotisée et l'objet à manipuler, la représentation utilisée pour décrire les états de ce système et enfin les différentes contraintes auxquelles le système de manipulation est soumis. En plus de ces définitions et hypothèses, une nouvelle structuration de l'espace des configurations est présentée. Elle repose sur l'introduction de sous-espaces particuliers que sont les espaces des configurations de prise. Leurs propriétés utiles, notamment leurs relations d'inclusion, sont détaillées. La dernière partie du chapitre s'intéresse aux chemins élémentaires respectant les contraintes de la manipulation.

2.1 Modélisation du système main+objet

2.1.1 Espace des configurations

Le système étudié se compose d'une main et d'un objet. L'objet est un solide rigide ; il possède donc six degrés de liberté. Une configuration ou pose de l'objet est la combinaison d'une position et d'une orientation et peut être représentée par un vecteur $\mathbf{p} = (x, y, z, \alpha, \beta, \gamma)$. $(x, y, z) \in \mathbb{R}^3$ est la position et (α, β, γ) une paramétrisation de l'orientation de l'objet. On peut, par exemple, choisir la représentation des angles d'Euler ou celle des quaternions (dans ce cas, on aura un vecteur de dimension quatre de norme unitaire donc toujours trois paramètres indépendants). L'espace des configurations de l'objet est noté $\mathcal{CS}_{objet}$. $\mathcal{CS}_{objet} = \mathbb{R}^3 \times \mathbb{RP}^3$ où $\mathbb{RP}^3$ est l'espace projectif réel [1].

La main robotisée se compose d'une paume, solide indéformable, à laquelle sont rattachés n doigts, dont les phalanges sont elles-aussi rigides. La paume sera supposée fixe ; par conséquent, tous les mouvements de l'objet seront induits par des mouvements des doigts uniquement. Le vecteur de configuration de la main est composé des vecteurs de configuration des n doigts. Chaque doigt i a m_i articulations indépendantes donc m_i degrés de liberté. Si on note $\theta_{i,k}$ la valeur du k-ième paramètre articulaire du doigt i, le vecteur de configuration de ce doigt est $\mathbf{a_i} = (\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,m_i}) \in \mathbb{S}^{m_i}$ où $\mathbb{S}$ est le cercle [2]. Le vecteur de configuration de la main est $\mathbf{a} = (\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n})$. L'espace des configurations de la main est noté $\mathcal{CS}_{main} = \{\mathbb{S}^{m_1} \times \mathbb{S}^{m_2} \times \ldots \times \mathbb{S}^{m_n}\}$. Il s'ensuit que l'espace des configurations du système main+objet est $\mathcal{CS} = \mathcal{CS}_{main} \times \mathcal{CS}_{objet}$.

2.1.2 Espace des configurations libres

Les configurations libres sont les configurations sans collision, c'est-à-dire les configurations n'entraînant pas de contact non autorisé entre des corps du système, ni de dépassement des butées. Les seuls contacts autorisés sont les contacts entre les bouts des doigts et l'objet, lorsque celui-ci est tenu par la main, et les contacts entre deux corps formant une articulation. Les butées sont des limitations des valeurs que peuvent prendre les éléments du vecteur de configuration et sont généralement imposées par la mécanique du système. Ainsi, les angles des articulations des doigts peuvent être contraints d'appartenir à un certain intervalle si leur construction ne leur permet pas de faire des tours complets. L'espace des configurations libres est $\mathcal{CS}_{free}$.

1. L'espace projectif réel est équivalent à l'ensemble des droites de $\mathbb{R}^4$ passant par l'origine. Il existe une équivalence entre les éléments de $\mathbb{RP}^3$ et l'ensemble des rotations dans l'espace.

2. Le cercle $\mathbb{S}$ est $[0, 2\pi]\,/\sim$, c'est-à-dire l'intervalle $[0, 2\pi]$ tel que 0 et 2π sont considérés équivalents.

2.1.3 Les configurations de prise

La main devant toujours tenir l'objet au cours d'une tâche de manipulation, la notion de prise est fondamentale. Une prise est géométriquement une configuration telle qu'un nombre donné de doigts sont en contact avec la surface de l'objet. Ces contacts doivent se faire entre deux surfaces. Autrement, s'il y a interpénétration de deux volumes, c'est-à-dire que le volume de leur intersection n'est pas nul, on considère qu'il y a collision. Pour simplifier le problème, nous restreignons les zones de contact souhaitables sur les doigts. Nous considérons, pour l'instant [3], que les contacts sont seulement autorisés au bout des doigts et que la surface de contact autorisée sur le bout des doigts est suffisamment petite pour pouvoir être assimilée à un point, ce qui est le cas si les doigts sont suffisamment pointus. Cette hypothèse permet de simplifier la paramétrisation des prises et de ne pas tenir compte de l'effet du roulement des doigts sur la surface de l'objet. Ainsi, hormis lorsqu'un doigt se repositionne pendant une changement de prise, la position d'un point de contact sera fixe sur la surface de l'objet et sur celle du doigt. La figure 2.1 montre à quoi correspond cette simplification en pratique.

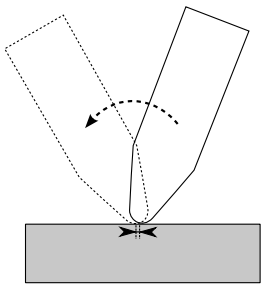


FIGURE 2.1 – Si les doigts sont suffisamment "pointus", on peut négliger l'effet du roulement sur la position des contacts.

L'hypothèse sur l'absence de roulement des contacts émise, on peut représenter une prise à k doigts par un vecteur $\mathbf{g}_k = (u_1, v_1, \ldots, u_k, v_k) \in \mathbb{R}^{2k}$ où (u_i, v_i) sont les coordonnées de la position du i-ème contact sur la surface de l'objet. Cette notation implique que la surface de l'objet doit être paramétrée, ce que nous supposons pour l'instant [4]. Nous parlerons de *doigt libre* pour un doigt qui ne participe pas à la prise.

Connaissant la pose de l'objet et la position d'un point sur sa surface, il est possible d'en déduire les configurations qui permettront à un doigt donné d'établir un contact avec l'objet en ce point, à condition que ce dernier soit atteignable par le doigt. Ces configurations s'obtiennent à l'aide du modèle géométrique inverse du doigt. Si le modèle géométrique inverse donne plusieurs solutions, il est toujours possible de le

3. Le chapitre 5 propose une extension du problème au cas de contacts avec roulement.
4. La partie 3.3 traite de la paramétrisation de la surface de l'objet.

restreindre de sorte qu'il n'en donne qu'une. Cette restriction peut se faire arbitrairement ou selon un critère quelconque tel que la maximisation de la distance des angles des liaisons par rapport à leurs butées respectives ou un critère de qualité de la prise. En pratique, les doigts des mains robotisées comportent rarement plus de trois degrés de liberté et leurs modèles géométriques donnent des solutions uniques. Supposons alors que les modèles géométriques des doigts sont à solution unique. Il en résulte qu'on peut connaître la configuration d'un doigt participant à la prise à partir de la position du contact associé sur la surface de l'objet et de la pose de l'objet. Par conséquent, la configuration du système *objet saisi par la main avec k doigts* est complètement déterminée par la connaissance de $\mathbf{p}$, $\mathbf{g}_k$ et des $\mathbf{a}_i$ des doigts libres.

Il est clair qu'une condition nécessaire de stabilité d'une configuration de $\mathcal{CS}_{free}$ est qu'elle soit une configuration de prise puisque l'objet non tenu est forcément instable –l'éventuel placement de l'objet sur un support quelconque n'étant pas ici considéré. Intuitivement, il est également clair qu'une configuration de prise n'est pas nécessairement stable. Il apparaît maintenant indispensable d'expliciter la définition et les conditions de stabilité d'une prise. Ces sujets sont abordés dans la partie suivante.

2.1.4 Stabilité d'une prise

Pour qu'une trajectoire planifiée pour le système main+objet soit réalisable, une condition nécessaire est que la prise soit stable tout au long du mouvement, autrement dit que la main tienne toujours fermement l'objet. Il existe deux critères de stabilité concernant la façon dont est saisi un objet : la *fermeture de forme* et la *fermeture de force*.

Avant d'expliciter ces critères, les modèles de contact entre deux solides les plus couramment employés sont rappelés. Ils seront utilisés dans la description des deux critères.

2.1.4.1 Les modèles de contact

Les modèles de contact entre deux solides rigides décrivent le type d'interaction mécanique de ces deux solides. Dans le cas de la manipulation dextre, les seuls contacts intéressants sont les contacts entre les doigts et l'objet. Dans la partie 2.1, nous avons émis l'hypothèse que les contacts sont ponctuels. Il est possible de définir toutes sortes de modèles pour deux solides en contact ponctuel. Cependant, deux d'entre eux sont le plus souvent utilisés, en raison de leur bon compromis entre simplicité et exactitude : le modèle de contact ponctuel avec frottement et le modèle de contact *soft finger* avec approximation elliptique.

Contact ponctuel avec frottement Soit un solide A en contact avec un solide B en un point $\mathbf{p}$ et $\mathbf{n}$ la normale au plan tangent défini entre les deux surfaces au

point $\mathbf{p}$ (figure 2.2). Le modèle de contact ponctuel avec frottement suppose que le contact ne peut transmettre qu'une force et pas de moment et que la force de frottement exercée par A sur B, $\mathbf{f}$, est dirigée vers B. Par ailleurs, pour qu'il n'y ait pas de glissement au niveau du point de contact $\mathbf{p}$, la force de contact doit respecter une contrainte qui, géométriquement, correspond à son appartenance à un cône appelé cône de frottement d'axe $\mathbf{n}$ et de demi-angle d'ouverture $\arctan(\mu)$ où μ est le coefficient de frottement statique de Coulomb. La valeur de μ dépend de la nature des matériaux mis en contact et se détermine par la mesure (par exemple, de l'ordre de 0.2 pour un contact acier/acier, 0.65 pour bois/bois, 1 pour pneu/route sèche, 0.7 pour pneu/route mouillée, etc.). Les contraintes auxquelles sont soumises les forces de contact, pour qu'il n'y ait pas glissement, s'écrivent ainsi :

$$\mathbf{f} \cdot \mathbf{n} > 0 \tag{2.1}$$

$$\mu f_n > \sqrt{f_{t_1}^2 + f_{t_2}^2} \tag{2.2}$$

où f_{t_1} et f_{t_2} sont les composantes tangentielles de $\mathbf{f}$ et f_n sa composante normale.

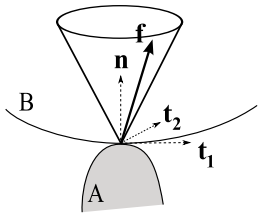


FIGURE 2.2 – Dans le cas d'un contact ponctuel avec frottement et en l'absence de glissement, la force de contact reste dans le cône de frottement.

Contact *soft finger* Le second modèle est le modèle de contact *soft finger* avec approximation elliptique. Ce modèle convient mieux au cas d'objets légèrement déformables, pour lesquels on suppose néanmoins le contact ponctuel, puisqu'il considère qu'un moment peut être transmis autour de la normale au point de contact. Les contraintes sur les efforts de contact en l'absence de glissement deviennent alors :

$$\mathbf{f} \cdot \mathbf{n} > 0 \tag{2.3}$$

$$f_n > \sqrt{\tfrac{1}{\mu^2}(f_{t_1}^2 + f_{t_2}^2) + \tfrac{1}{\mu_t^2} m^2} \tag{2.4}$$

où m est la valeur du moment exercé selon la normale au point de contact et μ_t un coefficient de frottement additionnel.

2.1.4.2 Stabilité de la prise et propriété de fermeture de forme et de fermeture de force

La stabilité d'une prise dépend de sa capacité à exercer, sur l'objet, les forces qui le maintiendront immobile ou lui feront suivre un mouvement voulu. L'étude de la stabilité de mouvements de manipulation dextre est un problème dynamique, qui se résout à l'aide du modèle dynamique du système main+objet, en l'intégrant le long de la trajectoire correspondante. Il faut alors connaître explicitement les forces que doivent exercer les doigts sur l'objet et, par conséquent, les valeurs des couples qui doivent s'appliquer sur les liaisons des doigts. Cela ajoute énormément de complexité aux calculs des chemins locaux de l'algorithme de planification, puisqu'il faut calculer les efforts à appliquer à l'objet, en déduire les couples à exercer sur les articulations et enfin intégrer les équations du mouvement. C'est pourquoi nous faisons l'hypothèse que le système main+objet est quasi-statique, c'est-à-dire qu'on peut considérer les trajectoires du système comme une succession d'états d'équilibre. Cela revient à négliger l'influence des effets inertiels dans le mouvement de l'objet. Cette hypothèse trouve sa justification dans le fait que les mouvements de manipulation dextre sont généralement lents, ou peuvent l'être sans inconvénient, et mettent en jeu des objets de masses relativement réduites. Dans ce cas, la stabilité peut être étudiée pour une configuration donnée sans avoir besoin d'introduire les vitesses des différents corps. Deux critères de stabilité peuvent alors être utilisés : la fermeture de forme et la fermeture de force.

La fermeture de forme La notion de fermeture de forme a été introduite par Reuleaux [Reu76][1], où elle était associée à la capacité d'un dispositif de saisie à empêcher les mouvements de l'objet tenu, en n'utilisant que des contacts unilatéraux sans frottement, c'est-à-dire des contacts ne pouvant exercer une force que dans la direction et le sens de la normale au point de contact et uniquement vers l'objet. Une prise assure la fermeture de forme si elle empêche tout mouvement de l'objet saisi, même si les contacts sont sans frottement. Reuleaux a montré que quatre contacts, au moins, sont nécessaires pour assurer la fermeture de forme d'une prise planaire. La figure 2.3 montre deux exemples de prise planaire d'un rectangle, vérifiant et ne vérifiant pas la fermeture de forme. Dans l'espace, Somov [Som00][2] a montré qu'il faut au minimum sept contacts. Par ailleurs, il n'existe pas de prise vérifiant la fermeture de forme pour tout objet présentant une surface de révolution, quelque soit le nombre de contacts. En effet, le mouvement de rotation autour de l'axe de révolution de la surface ne peut être bloqué. Une sphère ne peut ainsi être tenue de manière stable selon le critère de fermeture de forme.

[1] F. Reuleaux. Kinematics of machinery. 1876. Consultable en ligne : http ://historical.library.cornell.edu/kmoddl/.

[2] P. Somov. Über gebiete von schraubengeschwindigkeiten eines starren körpers bei verschiedener zahl von stützflächen. *Zeitschrift für Mathematik und Physik*, 45, 1900.

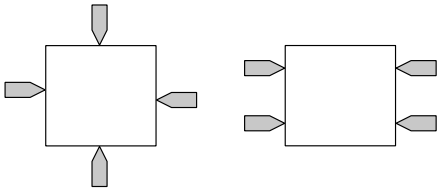


FIGURE 2.3 – Deux exemples de prise dans le plan vérifiant (a) et ne vérifiant pas (b) la fermeture de forme.

La fermeture de force Contrairement à la fermeture de forme, la propriété de fermeture de force fait intervenir les frottements des contacts. Une prise assure la fermeture de force si elle peut résister à n'importe quelle perturbation, en force et moment, exercée sur l'objet, en appliquant les forces de contact appropriées tout en respectant les contraintes de non glissement des points de contact. De nombreux travaux se sont intéressés à cette propriété (par exemple, [Bic95, Liu99][3,4]).

Cette propriété peut être visualisée géométriquement, si on se place dans le cas de contacts de type ponctuel avec frottement et qu'on linéarise les cônes de frottement. Dans ce cas, un cône de frottement devient un polyèdre et une force de contact lui appartenant s'écrit comme une combinaison linéaire à coefficients positifs des vecteurs définissant le cône linéarisé (figure 2.4). Le cône du contact d'indice i est ainsi transformé en cône polyédrique ayant n_c côtés. Si on note $\mathbf{s_{ij}}$ le j-ème vecteur-arête du cône

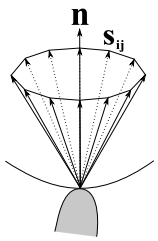


FIGURE 2.4 – Le cône de frottement linéarisé.

polyédrique de longueur unité, la force $\mathbf{f_i}$, exercée au niveau du contact et incluse dans le cône, s'écrit :

[3] A. Bicchi. On the closure properties of robotic grasping. *The International Journal of Robotics Research*, 14(4) :319–33, août 1995.

[4] Y-H. Liu. Qualitative test and force optimization of 3D frictional form-closure grasps using linear programming. *IEEE Transactions on Robotics and Automation*, 15(1) :163–173, 1999.

$$\mathbf{f_i} = \sum_{j=1}^{n_c} \lambda_{ij} \mathbf{s_{ij}}, \quad \lambda_{\mathbf{ij}} \geq \mathbf{0} \tag{2.5}$$

Chaque point de contact, d'indice i, est repéré par rapport au centre de gravité G de l'objet par un vecteur position $\mathbf{r_i}$. Le contact exerce alors sur l'objet un moment $\mathbf{r_i} \times \mathbf{f_i}$ en G, en plus de la force $\mathbf{f_i}$, ce qu'on peut écrire sous la forme d'un torseur d'effort :

$$\mathbf{w_i} = \begin{pmatrix} \mathbf{f_i} \\ \mathbf{r_i} \times \mathbf{f_i} \end{pmatrix} = \sum_{\mathbf{j=1}}^{\mathbf{n_c}} \lambda_{\mathbf{ij}} \begin{pmatrix} \mathbf{s_{ij}} \\ \mathbf{r_i} \times \mathbf{s_{ij}} \end{pmatrix} \tag{2.6}$$

Le torseur appliqué à l'objet est la somme des torseurs des différents contacts :

$$\mathbf{w} = \sum_{i=1}^{n} \mathbf{w_i} = \sum_{i=1}^{n} \sum_{j=1}^{n_c} \lambda_{ij} \begin{pmatrix} \mathbf{s_{ij}} \\ \mathbf{r_i} \times \mathbf{s_{ij}} \end{pmatrix} \tag{2.7}$$

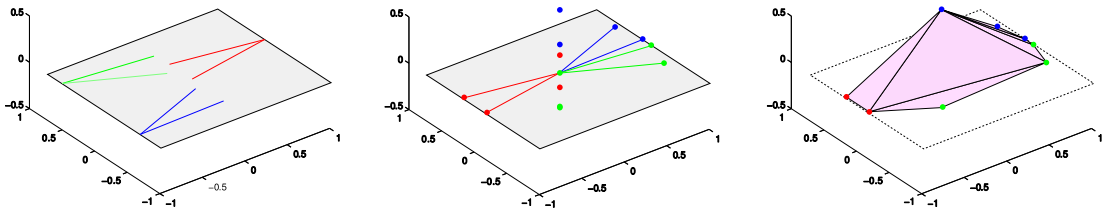
Une approximation de l'ensemble des torseurs que peut générer la prise avec des forces de contact appartenant aux cônes de frottement de longueur 1 est alors l'ensemble des torseurs contenus dans l'enveloppe convexe de l'ensemble des $\mathbf{w_i}$ [FC92][1], notée $H(W)$. La qualité de l'approximation dépend du nombre d'arêtes choisi pour linéariser les cônes de frottement. La prise peut exercer n'importe quel effort sur l'objet, c'est-à-dire qu'elle respecte la fermeture de force, s'il existe un voisinage de G, qui est strictement inclus dans $H(W)$. Géométriquement, cette caractéristique peut se visualiser dans le cas d'une prise dans le plan, puisque les $\mathbf{w_i}$ sont alors de dimension 3 (figure 2.5). Si on considère un voisinage sphérique, alors la fermeture de force est assurée si $H(W)$ contient une sphère de rayon non nul, centrée sur G. On peut, par ailleurs, se ramener à une étude plane dans le cas d'une prise à trois doigts dans l'espace [LLC03][2]. Un critère de qualité peut être ajouté à la notion de fermeture de force, en utilisant par exemple le rayon de la plus grande boule centrée sur G et incluse dans $H(W)$.

2.2 Espace des configurations

Les seules configurations pouvant être utilisées pour la construction d'une trajectoire dans le cadre spécifique de la manipulation dextre sont celles correspondant à une prise de l'objet. Pour manipuler un objet, il est souvent nécessaire, en raison des limitations des domaines d'accessibilité des doigts, de changer la position des contacts

[1] C. Ferrari and J. Canny. Planning optimal grasps. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'92)*, 3 :2290–2295, mai 1992.

[2] J.-W. Li, H. Liu, and H.-G. Cai. On computing three-finger force-closure grasps of 2-D and 3-D objects. *IEEE Transactions on Robotics and Automation*, 19(1) :155–161, février 2003.



(a) (b) (c)

FIGURE 2.5 – Visualisation géométrique de la propriété de fermeture de force dans le cas d'un système plan : cônes de frottement des contacts (a), forces de contact situés sur les cônes (traits pleins) et leurs moments (points alignés verticalement) par rapport au point (0,0,0) (b), enveloppe convexe des forces et moments (c). Ici, l'enveloppe convexe contient une sphère de rayon non nul centrée sur le point (0,0,0) donc la prise vérifie la fermeture de force.

sur l'objet. Le repositionnement d'un contact peut se faire par glissement sur la surface de l'objet mais se fera de façon plus générale en rompant le contact puis en le rétablissant en un autre point. Différents types de prises seront donc employés, correspondant chacun à une combinaison de doigts tenant l'objet. Il apparaît donc justifié d'associer à ces types de prise, un sous-espace particulier. Nous introduisons ainsi l'espace GS_k (GS pour *Grasp Subspace*) :

$$GS_k = \left\{ \begin{array}{c} (q_m, q_o) \in \mathcal{CS}_{main} \times \mathcal{CS}_{objet} \ / \\ \text{l'objet est saisi par une combinaison quelconque de k doigts} \end{array} \right\} \tag{2.8}$$

Comme il peut y avoir plusieurs combinaisons possibles pour les k doigts tenant l'objet, il est utile d'introduire les espaces $GS_k^i,\ i \in [\![1; C_n^k]\!]$, sous-espaces de GS_k :

$$GS_k^i = \left\{ \begin{array}{c} (q_m, q_o) \in \mathcal{CS}_{main} \times \mathcal{CS}_{objet} \ / \\ \text{l'objet est saisi par une combinaison donnée de k doigts} \end{array} \right\} \tag{2.9}$$

On a alors la relation $GS_k = \bigcup_{i \in [\![1;C_n^k]\!]} GS_k^i$. Comme les configurations acceptables du système main+objet appartiennent nécessairement à ces sous-espaces, il en sera de même pour toute solution éventuelle d'un problème de planification de tâche de manipulation dextre. La connaissance des caractéristiques de ces sous-espaces est donc primordiale pour la résolution d'un tel problème.

2.2.1 Les espaces GS_k

2.2.1.1 Stabilité des configurations de GS_k

Les configurations valides du système main+objet doivent appartenir à un des GS_k mais toutes les configurations de ces sous-espaces ne sont pas nécessairement valides.

Une configuration de GS_k peut entraîner une collision ou être instable. Par ailleurs, certains de ces sous-espaces ne contiennent aucune configuration stable. Par exemple, quel que soit le modèle de contact choisi, les configurations de GS_0 et GS_1 sont instables et l'existence de configurations stables dans GS_2 dépend du type des contacts doigt-objet ; une prise à deux doigts peut être stable pour des contacts de type *soft finger* mais pas pour des contacts ponctuels obéissant à la loi de frottement de Coulomb (on ne peut pas bloquer la rotation autour de l'axe formé par les deux points de contact).

2.2.1.2 Relation d'inclusion et d'intersection des GS_k

Une prise à $(k+1)$ doigts étant un cas particulier de prise à k doigts, correspondant à la situation où un doigt ne participant pas à la prise se positionne sur la surface de l'objet, on a la relation d'inclusion suivante :

$$GS_k \subset GS_{k-1},\ \forall k \tag{2.10}$$

Par ailleurs, on a la relation :

$$\bigcap_{i \in [\![1;C_n^k]\!]} GS_k^i = GS_n,\ \forall k \tag{2.11}$$

Cette propriété est importante car elle montre le rôle particulier de GS_n, qui est l'intersection de toutes les espaces de prises. La figure 2.6 illustre ces différentes relations dans le cas d'une main à quatre doigts pour GS_3 et GS_4.

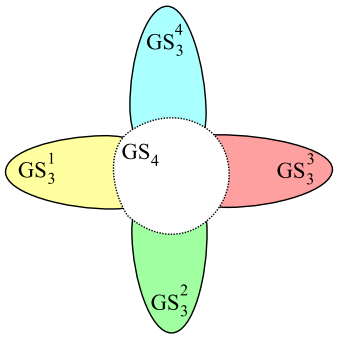


FIGURE 2.6 – Les sous-espaces GS_3 et GS_4 pour une main à quatre doigts .

2.2.1.3 Connexité des sous-espaces GS_k

Connaître la connexité des GS_k est indispensable pour savoir quels types de configurations peuvent être reliées par des chemins. La connexité d'un sous-espace renseigne sur l'existence d'une trajectoire continue reliant toute paire de configurations

de cet espace. Or, la continuité des mouvements des corps du système main+objet calculés par un algorithme de planification est impérative. Des mouvements discontinus des différents corps sont en effet physiquement impossibles. La connaissance de la connexité des GS_k permettra donc de savoir quand il est indispensable de changer de type de prise. Prenons l'exemple de la figure 2.7. Elle représente deux configurations appartenant à GS_4 mais à des GS_4^i différents, pour une main à cinq doigts. Il est

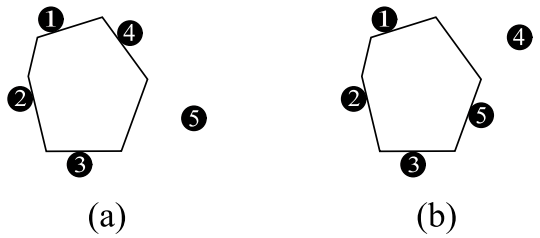


FIGURE 2.7 – Deux configurations de GS_4 ne pouvant être reliées sans passer par un autre type de prise.

évident qu'il n'existe pas de trajectoire continue, reliant ces deux configurations, qui ne quitte pas GS_4. Il faudra passer soit par une prise à trois doigts soit par une prise à cinq doigts, autrement dit par $GS_3 \backslash GS_4$ (GS_3 "moins" GS_4) ou GS_5. La figure 2.8 montre comment connecter ces deux configurations, via des configurations intermédiaires dans GS_5 ou $GS_3 \backslash GS_4$. Par conséquent, les différents $GS_k^i \backslash GS_{k+1}$ ne sont

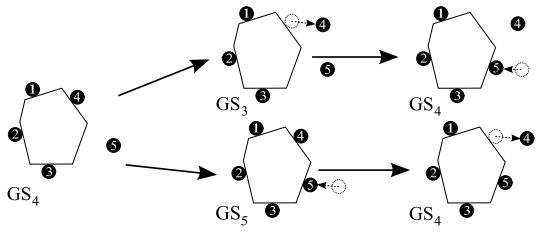


FIGURE 2.8 – Passage d'une classe de prise à quatre doigts à une autre pour une main à cinq doigts : la continuité des mouvements des doigts impose le passage intermédiaire à une prise à cinq ou trois doigts.

pas connexes. et un sous-espace $GS_k^i \backslash GS_{k+1}$ peut comporter plusieurs composantes connexes, même en l'absence d'obstacles ou d'autres contraintes. Les $GS_k^i \backslash GS_{k+1}$ se composeront éventuellement de différentes composantes connexes en présence d'obstacles ou d'autres contraintes. La figure 2.9 tente d'illustrer les différents sous-espaces GS_k et de montrer leurs différentes composantes connexes, pour une main à quatre doigts. Il y a un seul type de prise à quatre doigts, quatre types de prises à trois doigts (quatre GS_3^i) et six types de prises à deux doigts (six GS_2^i). Comme il est impossible de

représenter tous les sous-espaces, seule une partie des prises à deux doigts sont dessinées. On voit sur la figure que, par exemple, $GS_3^2 \backslash GS_4$ et $GS_3^3 \backslash GS_4$ sont disjoints mais qu'un chemin passant par GS_4 peut relier deux configurations de ces sous-espaces de même qu'un chemin passant par GS_2^2. Connaissant mieux la structure de l'espace des

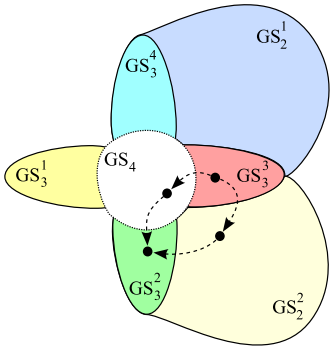


FIGURE 2.9 – Différents GS_k des prises de deux à quatre doigts, pour une main à quatre doigts.

configurations de notre système, il faut maintenant s'intéresser aux types de chemins à employer pour relier des configurations des différents sous-espaces.

2.2.2 Les différents types de chemin

Une fois connus les espaces dans lesquels il existe des trajectoires continues, il reste à choisir des types de chemin adaptés au problème. Deux types fondamentaux de chemin local peuvent être considérés, qui correspondent à des sous-tâches de manipulation élémentaires : la reconfiguration de prise et le déplacement de l'objet. Nous appellerons le premier *chemin de ressaisie* et le second, par analogie avec les techniques existantes concernant la manipulation (voir partie 1.6), *chemin de transfert*.

Les chemins de ressaisie Pendant un chemin de ressaisie (figure 2.10), l'objet est immobile et toujours maintenu par le nombre minimum de doigts permettant la stabilité de la prise, qui dépend du modèle de contact choisi. Par exemple, il faudra toujours qu'au moins trois doigts tiennent l'objet, dans le cas de contacts ponctuels avec frottement, pour que la prise puisse être stable. Les autres doigts pourront rompre leur contact pour se repositionner sur la surface de l'objet. Un chemin de ressaisie permet de changer la prise en maintenant la pose de l'objet constante. Le chemin de ressaisie correspond au passage d'un sous-espace GS_k^i à un sous-espace GS_{k-1}, puis au retour à GS_k^i, pour chacun des doigts qui doit changer de point de contact. Passer par GS_{k-1} revient à passer dans un espace de dimension supérieure. Chaque sous-espace GS_k a une dimension $2k + \sum_{i=1}^{n-k} m_i + 6$, puisque chacun des k doigts participant à la

prise possède deux degrés de liberté –la position du contact sur la surface de l'objet–, chacun des $(n-k)$ doigts libres garde ses m_i degrés de liberté et l'objet a six degrés de liberté. Ainsi, si c'est le doigt i_0 qui rompt son contact, le système gagne $(m_{i_0}-2)$ degrés de liberté.

Les chemins de transfert Pendant un chemin de transfert (figure 2.10), l'objet change de configuration tandis que la prise reste inchangée. La trajectoire de l'objet,

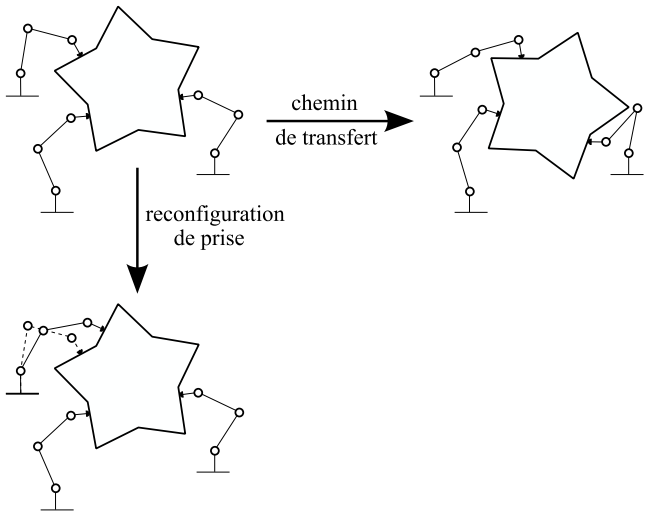


FIGURE 2.10 – Exemples de chemins de transfert et de ressaisie pour une main à 4 doigts dans le plan.

au cours d'un chemin de transfert, peut être choisie simplement comme étant l'ensemble des configurations reliant linéairement les configurations de départ et d'arrivée $\mathbf{p_1}$ et $\mathbf{p_2}$. Les configurations de l'objet comprises entre $\mathbf{p_1}$ et $\mathbf{p_2}$ sont alors interpolées linéairement pour la position et selon une interpolation linéaire sphérique pour l'orientation [Sho85][1]. Si on a $\mathbf{p_1} = (\mathbf{x_1}, \mathbf{h_1})$ et $\mathbf{p_2} = (\mathbf{x_2}, \mathbf{h_2})$ où $\mathbf{x_1} = (x_1, y_1, z_1)$ et $\mathbf{x_2} = (x_2, y_2, z_2)$ sont les positions de l'objet dans les configurations $\mathbf{p_1}$ et $\mathbf{p_2}$ respectivement et $\mathbf{h_1}$ et $\mathbf{h_2}$ des quaternions unitaires représentant les orientations de l'objet dans les configurations $\mathbf{p_1}$ et $\mathbf{p_2}$ respectivement, alors les configurations intermédiaires $\mathbf{p}(\alpha) = (\mathbf{x}(\alpha), \mathbf{h}(\alpha))$, $\alpha \in [0;1]$ sont obtenues selon :

$$\mathbf{x}(\alpha) = \alpha \mathbf{x}_1 + (1-\alpha)\mathbf{x}_2 \tag{2.12}$$

$$\mathbf{h}(\alpha) = \frac{sin((1-\alpha)\theta)}{sin(\theta)}\mathbf{h}_1 + \frac{sin(\alpha\theta)}{sin(\theta)}\mathbf{h}_2 \tag{2.13}$$

[1] K. Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH '85 : Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, New York, NY, USA, 1985. ACM Press.

avec $\theta = \arccos(\mathbf{h}_1 \cdot \mathbf{h}_2)$.

Les configurations des doigts sont calculées à partir de leurs modèles géométriques inverses, ce qui est possible car on connaît les positions des contacts sur l'objet et la pose de celui-ci. Un chemin de transfert est un type de chemin particulier dans GS_k^i.

Les chemins de transfert et de ressaisie, pris individuellement, ne permettent pas d'exécuter de vraies tâches de manipulation. En effet, deux configurations appartenant à GS_k^i peuvent être connectées par un chemin de transfert seulement si la prise est la même dans les deux configurations et par un chemin de ressaisie seulement si la pose de l'objet est la même dans les deux configurations. C'est pourquoi, il faudra combiner ces chemins, ce qui donnera un *chemin de manipulation dextre*.

Les chemins de manipulation dextre Un chemin de manipulation dextre est une suite de chemins de transfert et de ressaisie. Ces derniers seront suivis alternativement dans le chemin de manipulation et auront leur début et leur fin communes. Le chemin de manipulation sera donc formulé de la façon suivante :

$$\text{chemin de manipulation} = (l_1, l_2, l_3, \ldots, l_{2k}) \tag{2.14}$$
$$avec \begin{cases} l_1, l_3, \ldots, l_{2k-1} \text{ chemins de transfert} \\ l_2, l_4, \ldots, l_{2k} \text{ chemins de ressaisie} \\ \forall i \in \{1, 2, \ldots, 2k-1\}, \;\; l_i(1) = l_{i+1}(0) \end{cases}$$

Nous pouvons maintenant donner notre définition du problème de la planification de tâches de manipulation dextre.

Problème :

Le problème de la planification d'une tâche de manipulation dextre consiste à trouver un chemin de manipulation connectant deux configurations données de GS_n, tel que toutes les prises de la trajectoire soient stables, c'est-à-dire assurent la fermeture de force.

2.3 Conclusion

Ce chapitre a présenté notre façon de considérer le problème de la planification de tâches de manipulation dextre ainsi que tous les outils nécessaires à la mise au point d'une méthode de résolution de ce problème. En particulier ont été introduits les espaces des configurations de prise. Ces espaces jouent un rôle important car la méthode de planification proposée dans le chapitre suivant consiste à explorer ces espaces d'une façon qui tire partie de leur structure singulière.

Chapitre 3

Une Nouvelle Technique de Planification pour la Manipulation Dextre

Ce chapitre présente une nouvelle technique de planification pour la manipulation dextre, s'appuyant sur la structuration de l'espace des configurations proposée dans le chapitre 2 et notamment sur la définition de GS_n, le sous-espace des configurations de prise à n doigts. L'idée principale à l'origine de cette méthode est d'explorer préférentiellement GS_n, qui est le sous-espace de prise de plus petite dimension, pour accélérer la recherche d'une solution. Comme ce chapitre le montrera, explorer GS_n évite de calculer explicitement, au cours de la recherche, les trajectoires des mouvements de reconfiguration de prise, qui correspondent à des chemins dans des espaces de dimension supérieure ($GS_k, k < n$). Ces mouvements de reconfiguration de prise sont bien sûr nécessaires en raison des limites articulaires des doigts, de la présence éventuelle d'obstacles dans l'environnement ou d'instabilité dans la prise. Cependant, la méthode proposée limite le calcul de tels mouvements, en utilisant un type de chemin particulier dans GS_n, qui sera décrit dans la partie 3.1.1.

3.1 Approche proposée

Pour manipuler un objet, une main robotisée utilise une succession de mouvements de doigts. Ces mouvements peuvent mettre en jeu des prises à $2, 3, 4$, jusqu'à n doigts. Les prises doivent se faire avec un nombre minimum de doigts, qui dépend de la nature des contacts doigt-objet, mais le nombre de doigts peut varier d'une prise à une autre. Les solutions d'un problème de planification sont donc des trajectoires composées de configurations de prises à $2, 3, 4, \ldots, n$ doigts, selon les différentes combinaisons possibles. A chacune des combinaisons de doigts est associée un sous-espace de l'espace des configurations. Les solutions appartiennent donc à $\bigcup_{k\in[\![2;n]\!]} GS_k$, l'union de tous les sous-espaces de prise.

L'approche la plus exhaustive, pour résoudre un problème de planification, consiste à explorer tout l'espace des configurations. Ici, cela correspondrait à explorer l'union des sous-espaces de prise. Cependant, une telle démarche conduirait à des temps de résolution très élevés, en raison de la très grande taille de l'espace à explorer. Nous proposons plutôt de privilégier la recherche d'une solution dans GS_n [SSP06, SSP07a][1,2]. Deux raisons justifient ce choix. La première raison est qu'en favorisant l'exploration de GS_n, on privilégie les prises les plus stables, qui sont celles faisant intervenir le plus grand nombre de doigts. On a donc moins de chance de rencontrer d'instabilité au cours de l'exploration, ce qui facilitera cette dernière. Par ailleurs, la solution trouvée sera plus robuste qu'une solution employant des prises avec moins de doigts, car les prises utilisées seront plus stables. La seconde raison est que GS_n, parmi les sous-espaces des configurations de prise, est celui de plus petite dimension, car il est le plus contraint. Son exploration sera donc plus rapide et, avec elle, la résolution du problème de planification.

Jusqu'à présent, aucune approche basée sur l'exploration de GS_n n'avait été proposée, pour une raison évidente, qui est l'absence de chemins cinématiquement réalisables, dans cet espace, qui permettrait d'en relier deux configurations quelconques. En effet, deux configurations de GS_n pouvant correspondre, à la fois, à des poses de l'objet et des positions des contacts différentes, relier ces deux configurations signifie modifier indépendamment la prise et la configuration de l'objet, ce qui est irréalisable physiquement, puisque le mouvement de l'objet doit être induit par celui des doigts (l'objet ne peut pas bouger sans que les doigts bougent, par exemple). L'originalité de l'approche proposée dans ce mémoire est de passer outre cette impossibilité en généralisant et en exploitant une propriété, jusqu'alors limitée au problème de manipulation

[1] J.-P. Saut, A. Sahbani, and V. Perdereau. A global approach for dexterous manipulation planning using paths in n-fingers grasp subspace. *Proceedings of the 9th IEEE International Conference on Automation, Robotics and Vision (ICARCV)*, pages 2019–2024, décembre 2006.

[2] A. Sahbani, J.-P. Saut, and V. Perdereau. An efficient algorithm for dexterous manipulation planning. *Proceedings of the 4th IEEE International Multi-Conference on Systems, Signals & Devices (SSD2007)*, I : Conference on Systems Analysis & Automatic Control, mars 2007.

par un robot planaire composé d'un seul corps. Cette propriété est la propriété de réduction, introduite par Alami *et al.* [ALS94][3], déjà évoquée dans la partie 1.6.

Avant de présenter plus en détail le principe de la méthode de planification proposée, il faut expliquer la modélisation adoptée pour explorer GS_n. Cette modélisation concerne, à la fois, les configurations et les chemins permettant de les relier.

3.1.1 Représentation des configurations et des chemins dans GS_n

Il sera nécessaire, pour explorer GS_n, de caractériser les configurations de ce sous-espace et d'étudier la possibilité de relier ou non ces configurations par des chemins, ces derniers restant à définir. C'est pourquoi, nous nous intéressons maintenant à la modélisation des configurations de GS_n et, plus loin, aux chemins dans GS_n.

Modélisation des configurations de GS_n Le sous-espace GS_n est composé des configurations de prise à n doigts. Dans ces configurations, l'objet et les doigts forment des chaînes cinématiques fermées. Un point de contact peut, en effet, être vu comme le point de fermeture de la chaîne cinématique formée par deux doigts saisissant l'objet (figure 3.1). Déplacer les points de contact sur la surface de l'objet

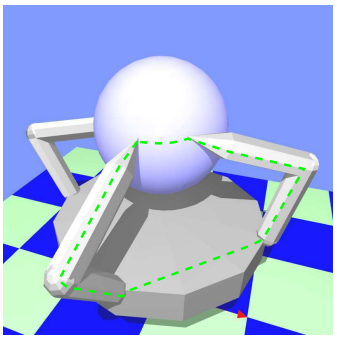


FIGURE 3.1 – Une des chaînes cinématiques fermées, formée par deux doigts et l'objet saisi (ligne verte en pointillés).

permet de modifier la prise, tout en maintenant ces contacts, donc la fermeture des chaînes. Les paramètres de ce déplacement sont des paramètres de changement continu de la prise. En faisant varier ces paramètres pour une configuration donnée de GS_n, on obtient un déplacement de la configuration du système à l'intérieur de GS_n. Les paramètres de positionnement des points de contact peuvent être vus comme des degrés

[3] R. Alami, J.-P. Laumond, and T. Siméon. Two manipulation planning algorithms. *Algorithmic Foundations of Robotics (WAFR'94)*, 1994.

de liberté *virtuels*. Pour un doigt contraint de maintenir un contact sur l'objet, ils vont remplacer tout ou partie de ses degrés de liberté originaux. Les degrés de liberté virtuels sont illustrés sur la figure 3.2. Dans cet exemple, chaque doigt libre a trois degrés de liberté (trois liaisons rotoïdes). Si la configuration appartient à GS_n, la configuration de chaque doigt est imposée par la position de son point de contact. Les deux degrés de liberté virtuels, correspondant au positionnement du point de contact, remplacent alors les trois degrés de liberté initiaux de chaque doigt. Ces degrés de liberté sont appelés virtuels car ils n'ont de sens que parce qu'on contraint la configuration du système main+objet à appartenir à GS_n. Une configuration de GS_n sera représentée par un

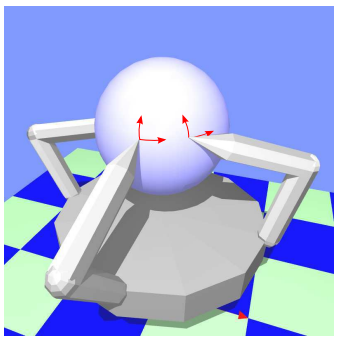


FIGURE 3.2 – Les degrés de liberté virtuels correspondent au déplacement continu des contacts sur l'objet (flèches rouges sur la surface de l'objet).

vecteur contenant la pose (position et orientation) de l'objet et les positions des points de contact des doigts sur la surface de l'objet (n paires de coordonnées). Si les doigts ont plus de trois degrés de liberté, il faudra que le vecteur de configuration contienne également les paramètres permettant de connaître la configuration complète des doigts.

Grâce à cette représentation, l'exploration de GS_n va se transformer en problème de planification de mouvement pour chaînes cinématiques fermées. Cette représentation permet notamment de définir et de calculer des chemins dans GS_n.

Les chemins dans GS_n Un chemin dans GS_n est une suite continue de configurations de prise à n doigts. Le long d'un tel chemin, la pose de l'objet et la configuration de prise changent en même temps, sans qu'il soit nécessaire de lever et repositionner les doigts pour modifier les positions des points de contact de la prise. Il suffit, pour calculer un chemin dans GS_n, de faire varier, simultanément, la pose de l'objet et les paramètres associés aux degrés de liberté virtuels, décrits plus haut. Les chemins les plus simples à calculer et les plus courts, en terme de déplacement de l'objet et de déplacement des points de contacts sur sa surface, sont alors les chemins linéaires (figure 3.3). L'intérêt de ces chemins apparaît immédiatement si on observe la figure 3.4, qui montre, dans le cas d'une main à quatre doigts, le nombre minimum de configura-

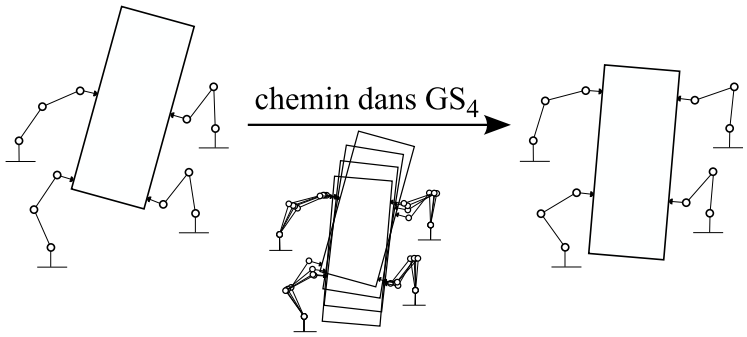


FIGURE 3.3 – Un chemin linéaire dans GS_4 pour une main à quatre doigts dans le plan.

tions intermédiaires nécessaires pour simplement passer d'une prise à quatre doigts à une autre, si les quatre contacts changent de position. Chaque doigt doit alors se repositionner, ce qui impose l'utilisation d'un chemin passant par GS_3. En restant dans GS_4, on peut relier directement les deux configurations sans avoir à résoudre de problèmes de planification supplémentaires ni introduire de configurations intermédiaires.

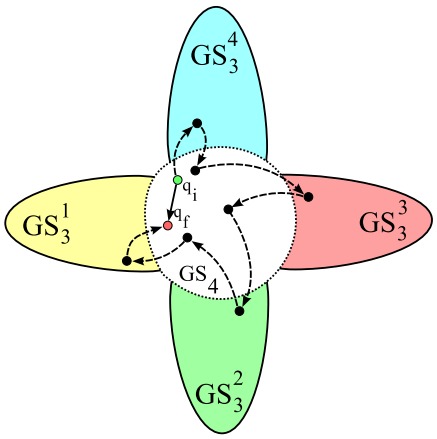


FIGURE 3.4 – Une reconfiguration de prise par une suite de chemins de ressaisie (chemin en pointillé) passant par les différents $GS_3^i,\ i \in [\![1;4]\!]$ et la même en restant dans GS_4 (chemin en trait plein).

En réalité, les chemins dans GS_n ne sont pas cinématiquement réalisables, car les contacts et l'objet ne peuvent se déplacer indépendamment. Heureusement, la propriété de réduction établie et prouvée par Alami *et al.*[ALS94] (voir partie 1.6.3) pour un système plan simple peut être étendue et ainsi remédier à ce problème.

3.1.2 La propriété de réduction étendue pour une main robotisée

Les chemins dans GS_n ne sont pas, en eux-mêmes, faisables d'un point de vue cinématique . Cependant, en étendant la propriété de réduction, on montre qu'un chemin dans GS_n sans collision peut être transformé en une suite finie de chemins de transfert et de ressaisie qui sont, eux, cinématiquement réalisables. La propriété de réduction telle qu'elle avait été définie n'est pas directement applicable à notre problème. Ici, le robot n'est pas constitué d'un seul mais de plusieurs corps, reliés par des articulations. Il doit donc éviter les collisions internes en plus de celles qui peuvent avoir lieu avec des obstacles de l'environnement. De plus, les représentations des configurations de l'objet, du robot et de la prise de [ALS94] ne sont pas adaptées à notre problème. En effet, pour le système de [ALS94], ces configurations étaient toutes représentées par un vecteur dans $\mathbb{R}^2$, y compris la configuration de la prise (position relative du robot et de l'objet). Ici, la représentation des prises est différente puisqu'elle doit prendre en compte les positions de chaque contact. Il faut donc étendre la propriété de réduction au cas de la manipulation dextre. Nous démontrons que l'extension de cette propriété au problème formulé dans le chapitre 2 est possible.

Soit l'hypothèse :

Hypothèse H : le robot (la main) évite tout contact interdit, c'est-à-dire tout contact avec les obstacles de l'environnement, tout contact entre deux de ses corps qui ne sont pas liés par une articulation [1] et tout contact avec l'objet hormis aux points de prise :

$$\exists \epsilon > 0 \ / \ B(q_{main}, \epsilon) \subset \mathcal{CS}_{main,free} \ ^2 \tag{3.1}$$

La propriété de réduction s'énonce alors :

Propriété de réduction généralisée :

Sous l'hypothèse H, un chemin dans GS_n, reliant deux configurations quelconques de cet espace, peut être décomposé en une suite finie de chemins de transfert et de ressaisie (aussi appelée chemin de manipulation).

Démonstration :

Soient a et b deux configurations de GS_n, reliées par un chemin p dans cet espace. p est, par définition, une fonction continue

$$p : [0,1] \ \rightarrow \ GS_n$$

telle que $p(0) = a$ et $p(1) = b$.

On note q_o la configuration de l'objet et q_g la configuration de la prise . La connaissance

1. Les collisions non voulues entre des corps liés par une articulation sont évitées grâce aux butées sur les articulations.

2. $B(q_{main}, \epsilon)$ est la boule de centre q_{main} et de rayon ϵ, avec la métrique suivante : $\rho(a,b) = \sum_{i=1}^{m} min(|a_i - b_i|, 2\pi - |a_i - b_i|)$ où $(a,b) \in \mathcal{CS}_{main} \times \mathcal{CS}_{main}$ et $m = \sum_{i=1}^{n} m_i$ est le nombre de liaisons de la main (voir partie 2.1). La norme induite est $\|x\| = \rho(x, 0)$.

de q_o et q_g permet de calculer, de façon unique, q_r, la configuration de la main (voir 2.1).
Soit la fonction f permettant de déterminer la configuration de la main à partir de celles de l'objet et de la prise :

$$\begin{aligned} f : \mathcal{CS}_{objet} \times \mathcal{CS}_{prise} &\rightarrow \mathcal{CS}_{main} \\ (q_o, q_g) &\mapsto q_r \end{aligned} \tag{3.2}$$

$\mathcal{CS}_{objet}$ est l'espace des configurations de l'objet seul ($\mathcal{CS}_{objet} = SE(3)$).
$\mathcal{CS}_{prise}$ est l'espace des configurations de la prise, caractérisée par les positions de n points sur la surface de l'objet. $\mathcal{CS}_{prise}$ dépend de la topologie de la surface de l'objet ; par exemple, si cette surface est topologiquement équivalente à une sphère $\mathcal{CS}_{prise} = \mathbb{S}^{2n}$. $\mathcal{CS}_{main}$ est l'espace des configurations de la main robot, caractérisées par un vecteur de paramètres articulaires.
Soient p_r la projection de p sur $\mathcal{CS}_{main}$, p_o la projection de p sur $\mathcal{CS}_{objet}$ et p_g la projection de p sur $\mathcal{CS}_{prise}$. Soit $c = p(t)$ une configuration quelconque sur le chemin p. D'après l'hypothèse H, $p_r(t)$ appartient à un ouvert de $\mathcal{CS}_{main,free}$. Il existe alors une boule ouverte $B_\epsilon \subset \mathcal{CS}_{main,free}$ centrée sur $p_r(t)$ et de rayon $\epsilon > 0$.

$f(q_o, q_g)$ est calculé à partir des modèles géométriques inverses des doigts, qui reçoivent comme entrées les positions des contacts, celles-ci dépendant de la pose de l'objet et des positions des contacts sur la surface de l'objet. Ces modèles géométriques sont des fonctions considérées continues. Par conséquent, f est aussi continue, ce qui signifie que $\forall (q_o^0, q_g^0) \in \mathcal{CS}_{objet} \times \mathcal{CS}_{prise}$:

$$\begin{aligned} \exists \eta > 0,\ tel\ que \|q_o - q_o^0\| < \eta\ et \|q_g - q_g^0\| < \eta, \\ \Rightarrow \|f(q_o, q_g) - f(q_o^0, q_g^0)\| < \epsilon \end{aligned} \tag{3.3}$$

où les normes utilisées sont des normes associées aux espaces correspondants ($SE(3)$, $\mathcal{CS}_{prise}$ et $\mathcal{CS}_{main}$). Par ailleurs, la continuité de p entraîne directement celles de p_o et p_g puisque la trajectoire de l'objet est continue le long de p et que les trajectoires des points de contact sur la surface de l'objet sont aussi continues le long de p, par construction des chemins dans GS_n.
Par conséquent, on a la relation de continuité suivante :

$$\begin{aligned} \exists \eta > 0,\ tel\ que\ \forall (\tau, \sigma) \in]t-\eta, t+\eta[\times]t-\eta, t+\eta[, \\ f(p_o(\tau), p_g(\sigma)) \in B_\epsilon \end{aligned} \tag{3.4}$$

Soient $c_1 = p(\tau_1)$ et $c_2 = p(\tau_2)$ deux configurations quelconques de p telles que $(\tau_1, \tau_2) \in]t-\eta; t+\eta[\times]t-\eta; t+\eta[$.
Prouvons que c_1 et c_2 peuvent être jointes par un chemin de transfert suivi d'un chemin

de ressaisie.
Soit le chemin p_1 :

$$\begin{array}{rccc} p_1 : & [\tau_1, \tau_2] & \rightarrow & \mathcal{CS}_{objet} \times \mathcal{CS}_{prise} \\ & \tau & \mapsto & (p_o(\tau), p_g(\tau_1)) \end{array} \tag{3.5}$$

p_1 est un chemin de transfert de $(p_o(\tau_1), p_g(\tau_1))$ à $(p_o(\tau_2), p_g(\tau_1))$.
D'après 3.4 on a :

$$\forall \tau \in [\tau_1, \tau_2], f(p_o(\tau), p_g(\tau_1)) \in B_\epsilon \tag{3.6}$$

donc les positions du robot correspondant au chemin p_1 appartiennent à B_ϵ et p_1 est sans collision.
Soit le chemin p_2 :

$$\begin{array}{rccc} p_2 : & [\tau_1, \tau_2] & \rightarrow & \mathcal{CS}_{objet} \times \mathcal{CS}_{prise} \\ & \tau & \mapsto & (p_o(\tau_2), p_g(\tau)) \end{array} \tag{3.7}$$

p_2 est un chemin de ressaisie de $(p_o(\tau_2), p_g(\tau_1))$ à $(p_o(\tau_2), p_g(\tau_2))$.
D'après 3.4 on a :

$$\forall \tau \in [\tau_1, \tau_2], f(p_o(\tau_2), p_g(\tau)) \in B_\epsilon \tag{3.8}$$

donc les positions du robot correspondant au chemin p_2 appartiennent à B_ϵ et p_2 est sans collision.
c_1 et c_2 peuvent donc être reliées par p_1 suivi de p_2.
Comme le chemin p_r en tant qu'ensemble (c'est-à-dire l'image de $[0, 1]$ par la fonction p_r) est un compact inclus dans un sous-ensemble ouvert de $\mathcal{CS}_{main,free}$, il existe un nombre fini de boules ouvertes de $\mathcal{CS}_{main,free}$ dont l'union contient p_r (il existe un revêtement fini de $[0, 1]$). On peut effectuer la transformation locale décrite ci-dessus sur chacune des parties du revêtement de p_r. Ainsi le chemin p entre a et b peut se décomposer en un nombre fini de séquences transfert-ressaisie (ou ressaisie-transfert).

La propriété de réduction nous sert à justifier l'utilisation de chemins dans GS_n pour explorer cet espace : bien que physiquement irréalisables, ils peuvent être décomposés en une suite finie de chemins réalisables. Si on trouve, pour un problème de planification donné, une trajectoire comportant des chemins dans GS_n, il suffira de décomposer ces derniers pour obtenir une trajectoire réalisable. La décomposition ne sera faite que pour la trajectoire solution. Ainsi, pour tous les chemins testés au cours de l'exploration de GS_n, et qui n'interviennent pas dans la solution, la décomposition ne sera pas effectuée. On minimisera donc le nombre de calculs de reconfigurations de prise, puisqu'ils ne seront faits qu'une fois une solution trouvée et seulement pour cette solution. Les détails concernant la décomposition des chemins dans GS_n sont donnés dans la partie 3.2.4.

Nous avons vu comment calculer des configurations et des chemins dans GS_n et justifié l'utilisation de ces derniers. Il devient donc possible d'envisager l'exploration de GS_n. Nous montrons maintenant comment mener cette exploration et comment l'exploiter dans le cadre de notre méthode de planification.

3.1.3 Principe général de la méthode de planification

Le coeur de la méthode de planification que nous proposons est l'exploration de GS_n. Cependant, des obstacles peuvent séparer ce sous-espace en plusieurs composantes connexes. Ces obstacles, dans l'espace des configurations, peuvent être des obstacles dans l'environnement ou des instabilités de la prise. Pour trouver une solution à un problème de planification, il faudra "contourner" ces obstacles, en changeant de sous-espace. Nous choisissons de passer par GS_{n-1}. Concrètement, ces changements de sous-espaces correspondent à des levers de doigts (un à la fois). Ces levers de doigts permettront d'éviter les obstacles de l'environnement ou de changer les points de contact de la prise de façon discontinue, pour passer une zone de prises instables. Ces mouvements de reconfiguration de prise se feront par l'intermédiaire de chemins de transfert-ressaisie. Nous proposons donc d'employer la méthode suivante [SSP07b][1] :

1. explorer GS_n en construisant un graphe dans cet espace, le but de ce graphe étant de représenter la connexité de GS_n
2. fusionner les composantes connexes du graphe obtenu à l'aide de chemins transfert-ressaisie ou ressaisie-transfert, qui passent par GS_{n-1}
3. décomposer, dans la trajectoire solution trouvée, les chemins dans GS_n en une suite de chemins de transfert-ressaisie

Ce principe est illustré sur la figure 3.5 dans le cas d'une main à quatre doigts. Sur la figure 3.5, des obstacles sont représentés. Ils correspondent à l'impossibilité de relier certaines configurations de GS_4 par un chemin inclus dans ce sous-espace. Pour relier des configurations séparées par un obstacle dans GS_4, il sera indispensable d'emprunter un chemin dans GS_3, autrement dit de changer la prise de l'objet par lever(s) de doigt(s). C'est ce qui explique pourquoi l'étape de fusion des composantes connexes du graphe est nécessaire.

La partie suivante s'intéresse à la mise en oeuvre de la méthode de planification proposée, en détaillant ses algorithmes et les choix techniques que nous avons faits.

[1] J.-P. Saut, A. Sahbani, and V. Perdereau. Dexterous manipulation planning using probabilistic roadmaps in continuous grasp subspaces. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, novembre 2007.

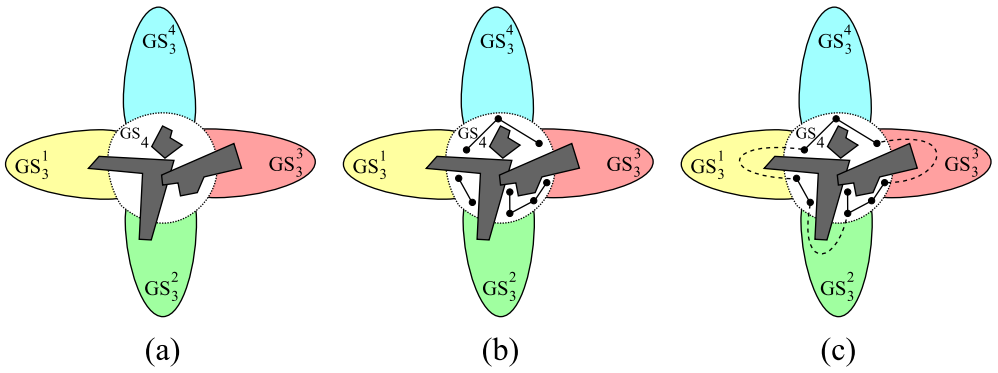


FIGURE 3.5 – Principe de la méthode de planification pour une main à quatre doigts. A cause de la présence d'obstacles (zones grises polygonales), l'espace GS_4 a plusieurs composantes connexes (a). Un graphe est construit pour explorer GS_4 (b). Les composantes connexes du graphe sont fusionnées à l'aide de chemins de transfert-ressaisie (en pointillés), qui ont des configurations dans GS_3 (c).

3.2 La méthode de planification pour la manipulation dextre

Pour explorer GS_n, nous avons choisi de construire un graphe sur le modèle des méthodes probabilistes exposées au chapitre 1, selon une version mono-requête. Cela implique en fait deux choix. Le premier d'avoir opté pour une méthode basée sur un échantillonnage de l'espace de recherche (GS_n et $GS_{n-1} \backslash GS_n$) (construction d'un graphe) ; le second d'avoir distingué, parmi les différents types d'échantillonnage possibles, un échantillonnage aléatoire. Le choix des méthodes par échantillonnage se justifie par l'impossibilité d'obtenir une représentation algébrique de l'espace des configurations et des obstacles. Cette impossibilité est due à la complexité du système étudié, qui s'explique par la présence de plusieurs chaînes cinématiques fermées et la grande dimension de $\mathcal{CS}$. Des travaux ont déjà traité de la détermination algébrique de $\mathcal{CS}$ pour des chaînes cinématiques fermées ([TM02, LT05][1,2]). Les configurations qui vérifient la fermeture de chaîne doivent vérifier un ensemble d'équations polynomiales. L'ensemble des configurations solutions de ces équations constitue une variété algébrique ([LaV06][3]), qu'on ne peut déterminer que dans certains cas. L'ajout d'obstacles rend la détermination de ces variétés très difficile et il n'existe aujourd'hui aucune méthode générale complète de planification de mouvement pour une chaîne

[1] J.C. Trinkle and R.J. Milgram. Complete path planning for closed kinematic chains with spherical joints. *International Journal of Robotics Research*, 21(9) :773–789, septembre 2002.

[2] G.F. Liu and J.C. Trinkle. Complete path planning for planar closed chains among point obstacles. *Proceedings of Robotics : Science and Systems*, juin 2005.

[3] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Consultable en ligne : http ://planning.cs.uiuc.edu/.

cinématique fermée en présence d'obstacles. Seul a pu être traité le cas d'une chaîne planaire en présence d'obstacles ponctuels ([LT05][2]). Le problème est encore plus complexe en ce qui concerne la manipulation dextre puisqu'elle fait intervenir un système composé de plusieurs chaînes fermées et que ces chaînes peuvent s'ouvrir et se fermer selon les situations.

Une fois décidée l'utilisation d'un échantillonnage de $\mathcal{CS}$, il reste à choisir une technique d'échantillonnage. L'échantillonnage aléatoire est le mieux adapté puisque si on sait générer des suites déterministes de points ayant une bonne dispersion pour des espaces simples, il n'en est pas de même pour un espace produit cartésien de nombreux espaces, comme c'est le cas pour GS_n.

Le planificateur de tâches de manipulation proposé travaillera en trois phases :
- une phase d'apprentissage (construction du graphe)
- une phase de recherche
- une phase de transformation des chemins

Le graphe de manipulation dextre est construit lors de la phase d'apprentissage. Il dépend de la géométrie de la main robotisée, de l'objet et des obstacles statiques de l'environnement. Comme nous avons préféré une technique mono-requête, les configurations initiale et finale d'un problème sont ajoutées au graphe au début de l'exécution de l'algorithme et la construction du graphe s'arrête quand ces deux configurations appartiennent à une même composante connexe du graphe. Ensuite, lors de la phase de recherche, un algorithme de recherche dans un graphe est utilisé pour trouver un chemin de manipulation entre les configurations initiale et finale. Une fois le chemin de manipulation trouvé, une phase de transformation des chemins est lancée. Elle est composée de trois étapes. La première consiste à lisser les chemins dans GS_n, pour minimiser la longueur de la trajectoire de l'objet. Un algorithme probabiliste est utilisé pour effectuer ce lissage (voir partie 1.4.3). Ensuite, au cours de la deuxième étape, les chemins dans GS_n sont décomposés en un nombre fini de chemins de transfert et de ressaisie. Cette étape se fait à l'aide d'un algorithme dichotomique détaillé dans la partie 3.2.4. Enfin, les chemins de ressaisie du chemin de manipulation sont lissés de la même façon que les chemins dans GS_n durant la première étape, pour minimiser l'amplitude des mouvement de repositionnement des doigts sur l'objet.

Ces étapes sont décrites en détail dans les parties suivantes.

3.2.1 Construction du graphe de manipulation

La construction du graphe de manipulation se fait en alternant une phase d'exploration de GS_n et une phase de fusion des composantes connexes du graphe par des chemins de transfert-ressaisie. L'algorithme 3 présente les étapes de construction du graphe de manipulation dextre. q_{init} et q_{goal} sont respectivement la configuration initiale et la configuration finale désirée du système, $\mathcal{GM}$ le graphe de manipulation, $\mathcal{CC}$ l'ensemble des composantes connexes de $\mathcal{GM}$. l_1 et l_2 sont les premières composantes

connexes trouvées par l'algorithme, à l'issue de l'ajout de q_{init} et q_{goal} à $\mathcal{GM}$. α est un paramètre réel compris entre 0 et 1. Selon sa valeur, l'algorithme privilégiera l'exploration de GS_n (α proche de 1) ou les tentatives de fusion des composantes connexes du graphe à l'aide de chemins de transfert-ressaisie (α proche de 0).

Algorithme 3 : Algorithme de construction du graphe de manipulation

```
1  GM = {q_init, q_goal}
2  CC = {l_1, l_2}
3  α ∈]0; 1[
4  si l_1 ≡ l_2 alors
5      fin de l'algorithme
6  sinon
7      tant que l_1 ≠ l_2 faire
8          choisir aléatoirement x ∈]0; 1[
9          si x < α alors
10             explorerGS_n(GM,CC)
11         sinon
12             connecter_composantes(GM,CC)
13         fin
14     fin
15 fin
```

La fréquence des tentatives de connexion des composantes connexes de GS_n à l'aide de chemin de ressaisie dépend ainsi du paramètre α de l'algorithme 3. Le réglage de ce paramètre est donc délicat. Pour certains problèmes, la méthode donnera de meilleurs résultats si α est proche de 1, pour d'autres si α est proche de 0. La partie 4 reviendra sur l'influence de ce paramètre pour plusieurs exemples de problèmes de planification ainsi que sur son choix. Précisons déjà qu'on peut trouver un compromis en initialisant α avec une grande valeur et en le diminuant au fur et à mesure que le nombre de noeuds du graphe augmente.

3.2.2 Exploration des composantes connexes de GS_n

Pour explorer GS_n, il faut pouvoir générer aléatoirement des configurations et calculer des chemins dans ce sous-espace.

Génération des configurations de prise Pour générer aléatoirement des configurations de prise, l'idéal serait de disposer d'une expression analytique de l'ensemble des configurations vérifiant les fermetures de chaînes, qui, en fonction d'un jeu de paramètres, donnerait le vecteur de configuration du système main+objet. Il suffirait alors

de tirer aléatoirement ces paramètres pour avoir une configuration de prise. Malheureusement, comme cela a déjà été évoqué dans la partie 3.2, obtenir une telle expression s'avère difficile dans le cas général d'un manipulateur parallèle et semble impossible, dans notre situation, étant donné la nature particulière des liaisons doigt-objet —ce sont des liaisons prismatiques doubles dont les orientations des axes mobiles dépendent de la surface de l'objet au niveau du point de contact. C'est pourquoi, nous avons opté pour une méthode de décomposition de la chaîne cinématique fermée en un ensemble chaîne active-chaîne passive. Nous avons pour cela adapté l'algorithme RLG (*Random Loop Generator*) de [Cor03][1] à notre problème. Le but de l'algorithme RLG est de générer des configurations respectant des fermetures de chaînes. Pour cela, les chaînes fermées sont coupées en deux sous-chaînes : une active et une passive. La chaîne active peut elle-même être composée de plusieurs chaînes. La chaîne passive est choisie telle que son extrémité ait une mobilité complète —c'est-à-dire que son domaine d'accessibilité soit un volume pour un système spatial ou une surface pour un système plan— et qu'on connaisse son modèle géométrique inverse. Les paramètres articulaires θ_i de la sous-chaîne active sont déterminés les uns après les autres, en remontant de la base de la chaîne à son extrémité. Le i-ème paramètre est choisi aléatoirement dans un intervalle I_i calculé de sorte que l'extrémité de la sous-chaîne active soit dans une estimation de la zone accessible à la sous-chaîne passive. Une fois que toute la chaîne active est connue, on trouve la configuration de la chaîne passive à l'aide de son modèle géométrique inverse. La figure 3.6 illustre le principe de l'algorithme RLG.

Dans le cas de la main saisissant un objet, comme il y a plusieurs chaînes en parallèle, qui ont toutes l'objet en commun, les chaînes seront en fait décomposées en deux parties passives et une partie active. L'objet doit impérativement faire partie des chaînes actives. Les doigts sont choisis comme chaînes passives. Dans l'algorithme RLG, les liaisons sont soit rotoïdes soit prismatiques. Ce n'est pas le cas des liaisons formées par les contacts doigt-objet puisque la direction de translation des contacts sur la surface de l'objet n'est pas fixe mais dépend de cette surface. L'algorithme de génération de configuration de prise diffère donc légèrement de RLG.

Les grandes lignes de l'algorithme de génération de configurations de GS_n sont les suivantes

1. génération aléatoire d'une configuration de l'objet dans une estimation grossière de la zone atteignable par les doigts de la main
2. génération aléatoire de chaque point de contact sur une estimation de la partie de la surface de l'objet accessible au doigt correspondant
3. application du modèle géométrique inverse des doigts
4. test de stabilité de la prise

[1] J. Cortés. Motion planning algorithms for general closed-chain mechanisms. *Thèse de doctorat de l'Institut National Polytechnique de Toulouse*, 2003.

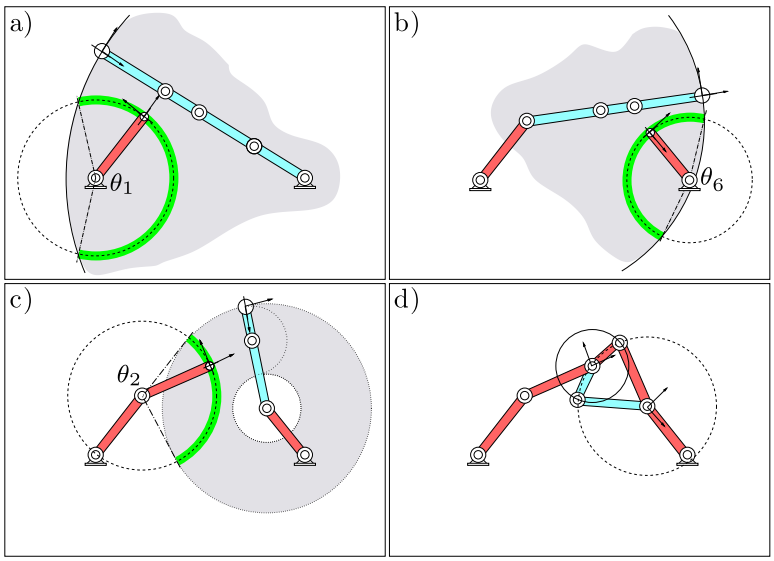


FIGURE 3.6 – Dans cet exemple planaire tiré de [Cor03], la sous-chaîne active (en rouge) est composée des liaisons θ_1, θ_2 et θ_6. Les θ_i de la chaîne active sont calculés du plus proche de la base au plus éloigné. Ils sont choisis aléatoirement dans un intervalle tel que l'extrémité de la sous-chaîne active soit dans l'estimation du domaine accessible de la chaîne passive (zones grises). La configuration de la chaîne passive est calculée à l'aide de son modèle géométrique inverse (d).

L'estimation de la zone atteignable par les doigts de la main est une sphère centrée sur la paume de la main dont le rayon est choisi en fonction des dimensions de celle-ci. Une fois la pose de l'objet choisie, il faut déterminer des points de contact. Plutôt que de les choisir au hasard n'importe où sur la surface de l'objet, on les choisit dans l'intersection de la surface de l'objet et de l'estimation du domaine accessible de chaque doigt. Cette estimation est une sphère centrée sur le doigt correspondant. Il faut donc pouvoir calculer l'intersection entre une sphère et l'objet. Nous avons considéré et implémenté dans notre planificateur deux types d'objet : la sphère et le polyèdre, ce dernier permettant de représenter n'importe quelle forme d'objet. Il est facile de calculer l'intersection entre le volume d'accessibilité et l'objet si celui-ci est une sphère. Si l'objet est un polyèdre, on utilise une représentation sous forme d'arbre de volumes englobants, ce qui permet de réduire considérablement le temps de calcul de l'intersection entre le polyèdre et la sphère d'accessibilité. Ce type de représentation est employé par les détecteurs de collision. Celui que nous avons utilisé ([GLM96][1]) utilise des parallélépipèdes comme volumes englobants (*Oriented Bounding Box* ou OBB). Au polyèdre, composé d'un ensemble de triangles, est associé un arbre binaire contenant des parallélépipèdes, caractérisés par leurs dimensions, leur position et leur

[1] S. Gottschalk, M.C. Lin, and D. Manocha. OBBTree : A hierarchical structure for rapid interference detection. *Proceedings of ACM Siggraph'96*, 1996.

orientation, englobant au plus près la surface du polyèdre (figure 3.7). La racine de l'arbre est une boîte englobant tout l'objet et ses feuilles sont les triangles du polyèdre. En testant récursivement l'intersection des boîtes avec les sphères des volumes

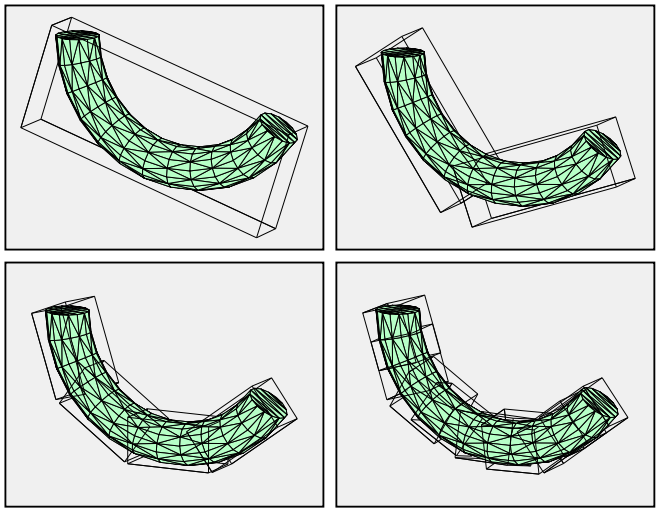


FIGURE 3.7 – Une hiérarchie de boîtes englobantes est construite pour approximer plus ou moins finement la forme de l'objet.

d'accessibilité des doigts, on peut déterminer si la surface de l'objet est accessible aux doigts et si oui quels triangles le sont. Parmi ces triangles, certains sont inaccessibles à cause de leur orientation. Les doigts, du fait du nombre nécessairement limité de leur degrés de liberté, ont en effet une direction privilégiée. On peut éliminer des parties de la surface de l'objet car les doigts ne peuvent les atteindre sans provoquer de collision avec l'objet. On choisit alors des points de contact au hasard sur les triangles restants. La figure 3.8 illustre le principe du calcul de l'estimation des zones atteignables par les doigts sur la surface de l'objet. Sur l'exemple de gauche, on voit que l'estimation du domaine d'accessibilité d'un des doigts n'intersecte pas la surface de l'objet ; il est donc inutile de chercher une prise pour cette pose de l'objet. Grâce à l'utilisation de l'arbre d'OBB, il est extrêmement rapide de déterminer si l'objet est accessible à un doigt ou pas. Sur l'exemple de droite, on voit qu'il est important de réduire la zone de la surface de l'objet dans laquelle on va choisir des points de contact puisque que seule une partie de la surface de l'objet est accessible. Une fois qu'on a choisi un point de contact, on vérifie qu'il est bien accessible en appliquant le modèle géométrique du doigt correspondant. S'il n'a pas de solution, on génère un nouveau point.

Les différents points de contact calculés donnent une prise dont il faut déterminer la stabilité. Étant donné que cette prise sera le point de départ de mouvements de manipulation de l'objet, elle doit permettre la reconfiguration de prise. Nous souhaitons donc que chacun des doigts puisse se lever sans rompre la stabilité de la prise. Pour une

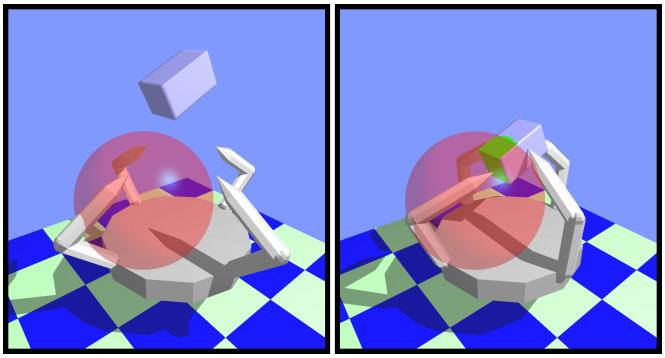


FIGURE 3.8 – La sphère rouge représente l'estimation du domaine d'accessibilité d'un des doigts et la partie illustrée en vert de la surface de l'objet représente l'estimation de la partie accessible de la surface de l'objet pour ce doigt. C'est sur cette partie qu'il faut choisir le point de contact du doigt.

prise donnée, on teste alors la stabilité de chaque prise obtenue en levant un des doigts. Par exemple, dans le cas où $GS_n = GS_4$, on testera la stabilité de quatre prises à trois contacts. En choisissant aléatoirement les points de contact sur les parties de la surface de l'objet accessible aux doigts, si la main a peu de doigts par rapport au nombre minimal de contacts nécessaires pour assurer la stabilité d'une prise, la probabilité de trouver une prise vérifiant la stabilité des sous-prises associées est extrêmement faible. C'est pourquoi, une heuristique est utilisée pour choisir la prise. Les doigts sont associés par paires, aléatoirement. Le point de contact $\mathbf{p_c}$ du premier doigt d'une paire est choisi aléatoirement sur les parties de la surface de l'objet accessible. On calcule ensuite la normale à la surface de l'objet en $\mathbf{p_c}$ et en quel point autre que $\mathbf{p_c}$ cette normale intersecte la surface de l'objet. Cette intersection existe forcément à partir du moment où la surface de l'objet est fermée. Si le point obtenu est accessible, il sera le point de contact de l'autre doigt de la paire, sinon on choisit un nouveau point de contact pour le premier doigt. La figure 3.9 montre les types de prise qu'on obtiendra pour un objet de forme parallélépipédique, si la main a quatre doigts.

Calcul des chemins dans GS_n La calcul de chemins linéaires dans GS_n ne pose pas de problème particulier si la surface de l'objet est paramétrée. Le cas où la surface de l'objet n'a pas de paramétrisation est traité dans la partie 3.3. Si la surface de l'objet est paramétrée, le chemin dans GS_n reliant deux configurations données est obtenu en faisant varier linéairement la configuration de l'objet [3] et les paramètres de position des contacts. Les paramètres articulaires des doigts sont calculés à l'aide des modèles géométriques inverses de ces derniers. Une fois la solution d'un problème de planification trouvée, les chemins linéaires dans GS_n faisant partie de cette solution

3. L'évolution de la pose de l'objet est calculée comme expliqué dans le paragraphe sur les chemins de transfert de la partie 2.2.2.

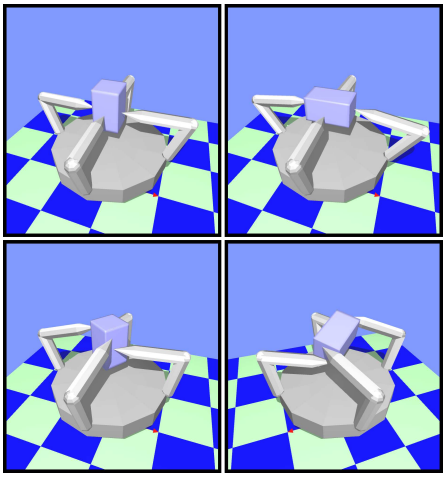


FIGURE 3.9 – Différents types de prise générées pour un objet parallélépipédique.

seront décomposés en suite transfert-ressaisie et, par conséquent, des contacts seront rompus. Comme, pendant la construction des chemins dans GS_n, on ne sait pas encore quel doigt va se lever ni à quel moment au cours de la trajectoire, la stabilité des chemins dans GS_n sera vérifiée de la même façon qu'on teste la stabilité des prises au cours de l'étape de génération des configurations dans GS_n, c'est-à-dire en testant la stabilité des n "sous-prises" obtenues en supprimant un contact.

3.2.3 Fusion des composantes connexes du graphe par des chemins de transfert-ressaisie

La fusion des composantes connexes du graphe de manipulation dextre, par des chemins de transfert-ressaisie est, avec l'exploration de GS_n, l'autre tâche importante de la méthode de planification proposée. Cette partie décrit la technique employée.

Il faut choisir deux noeuds de deux composantes connexes du graphe. C'est en reliant ces deux noeuds, via un chemin de transfert-ressaisie, que les composantes seront fusionnées. Pour commencer, les composantes connexes à fusionner sont choisies au hasard. On choisit ensuite un noeud dans chacune de ces composantes, de sorte à minimiser la distance entre les deux configurations de l'objet associées aux noeuds. Ainsi, si les configurations associées aux noeuds sont $q_1 = (\mathbf{p}_1, \mathbf{g}_{k_1})$ et $q_2 = (\mathbf{p}_2, \mathbf{g}_{k_2})$, on va choisir les noeuds dans leur composante respective de façon que la distance entre $\mathbf{p}_1$ et $\mathbf{p}_2$ soit minimale. On fait ce choix car deux configurations auront plus de chance de pouvoir être connectées si l'objet a des poses proches dans les deux configurations, étant donné que les chemins de transfert ne permettent généralement que des petits mouvements de l'objet. On choisit de ne pas faire intervenir la prise car il

n'y a *a priori* pas de fonction distance, applicable aux paramètres de prise, qui puisse être liée à la facilité de connexion de deux configurations par un chemin de transfert-ressaisie. La distance qu'on cherche à minimiser en choisissant les noeuds à relier est donc la distance entre deux poses de l'objet. Une pose de l'objet étant le produit cartésien d'une position et d'une orientation, cette distance est la somme pondérée de la distance entre deux points de $\mathbb{R}^3$ et de la distance entre deux points de $SO(3)$. Nous avons choisi, pour la distance entre deux points de $\mathbb{R}^3$, la distance euclidienne (L_2) et, pour la distance entre deux points de $SO(3)$, représentés à l'aide de deux quaternions $h_1 = (a_1, a_2, a_3, a_4)$ et $h_2 = (b_1, b_2, b_3, b_4)$, la fonction distance suivante :

$$d_{SO(3)}(h_1, h_2) = min(\rho(h_1, h_2), \rho(h_1, -h_2)) \tag{3.9}$$

où

$$\rho(h_1, h_2) = cos^{-1}(a_1b_1 + a_2b_2 + a_3b_3 + a_4b_4) \tag{3.10}$$

La distance entre deux configurations de l'objet est donc :

$$d(p_1, p_2) = a\ L_2(position_1, position_2) + b\ d_{SO(3)}(orientation_1, orientation_2) \tag{3.11}$$

Les poids a et b sont à choisir selon que l'on souhaite privilégier l'écart entre position ou orientation dans le calcul de la distance. Nous avons choisi $a = 1$ et $b = 0.8$.
Maintenant que nous avons les deux configurations q_1 et q_2, il faut les relier via une séquence minimale de chemins de transfert et de ressaisie. Il existe deux possibilités, nécessitant toutes deux l'ajout d'un noeud/configuration intermédiaire au graphe de manipulation :

- $q_1 \overset{ressaisie}{\longrightarrow} (\mathbf{p}_1, \mathbf{g}_{k_2}) \overset{transfert}{\longrightarrow} q_2$
- $q_1 \overset{transfert}{\longrightarrow} (\mathbf{p}_2, \mathbf{g}_{k_1}) \overset{ressaisie}{\longrightarrow} q_2$

Comme les mouvements permis par un chemin de transfert sont très limités et qu'un chemin de transfert n'est qu'un cas particulier de chemin dans GS_n, il est préférable d'utiliser la méthode de connexion décrite par l'algorithme 4, qui cherche à utiliser le plus possible de chemins linéaires dans GS_n.

La fonction $teste\_connexion\_chemin\_GS_n()$ essaye de relier les deux configurations passées en argument par un chemin dans GS_n et retourne la dernière configuration valide le long de ce chemin, c'est-à-dire la dernière configuration du chemin avant qu'il y ait collision ou instabilité, s'il y a échec de la connexion. Pour calculer les chemins de ressaisie, on utilise la méthode de planification RRT ([LK00][1]), successivement, pour chacun des doigts à repositionner. Le mouvement à planifier est celui d'un système composé d'une chaîne cinématique simple, qui aura un faible nombre de degrés de liberté pour la plupart des mains robots —le plus souvent, trois. Pour

[1] S.M. LaValle and J.J. Kuffner. Rapidly-exploring random trees : Progress and prospects. *Algorithmic Foundations of Robotics (WAFR'00)*, 2000.

Algorithme 4 : Connexion des composantes connexes de GS_n par des chemins de ressaisie

Entrées : graphe $\mathcal{G} = \bigcup_i \mathcal{CC}_i$ $q_1 = (\mathbf{p}_1, \mathbf{g}_{k_1}) \in \mathcal{CC}_{i_1}$ et $q_2 = (\mathbf{p}_2, \mathbf{g}_{k_2}) \in \mathcal{CC}_{i_2}$
$q_{s_1} = teste\_connexion\_chemin\_GS_n(q_1, q_2)$
$q_{s_2} = teste\_connexion\_chemin\_GS_n(q_2, q_1)$
si $q_{s_1} == q_{s_2}$ **alors**
    $connecte\_par\_chemin\_GS_n(q_{s_1}, q_1)$
    $connecte\_par\_chemin\_GS_n(q_{s_1}, q_2)$
    fusionne $\mathcal{CC}_{i_1}$ et $\mathcal{CC}_{i_2}$
si $teste\_connexion\_chemin\_ressaisie(q_{s_1}, q_{s_2}) == TRUE$ **alors**
    $connecte\_par\_chemin\_GS_n(q_{s_1}, q_1)$
    $connecte\_par\_chemin\_GS_n(q_{s_2}, q_2)$
    $connecte\_par\_chemin\_ressaisie(q_{s_1}, q_{s_2})$
    fusionne $\mathcal{CC}_{i_1}$ et $\mathcal{CC}_{i_2}$

appliquer la méthode RRT, il faut choisir une fonction distance pour la recherche de plus proche voisin dans les arbres. On choisit la même que celle de $\mathcal{CS}_{main}$ dans la partie 3.1.2. Comme seule la configuration d'un doigt peut changer, on la réduit à $\rho(a, b) = \sum_{i=1}^{m_i} min(|a_i - b_i|, 2\pi - |a_i - b_i|)$ où m_i est le nombre de liaisons du doigt et les a_i et b_i sont les valeurs des angles des liaisons du doigt dans les configuration a et b respectivement.

Dans certaines situations, les configurations q_1 et q_2 ne peuvent être reliées selon le principe de la fonction $teste\_connexion\_chemin\_GS_n()$ à moins que la pose de l'objet ne soit extrêmement proche dans les deux configurations. De telles situations peuvent survenir à cause d'un obstacle, d'une rupture de la stabilité de la prise ou être dues à la forme de l'objet. La figure 3.10 illustre ce problème. Pour y remédier, l'ajout

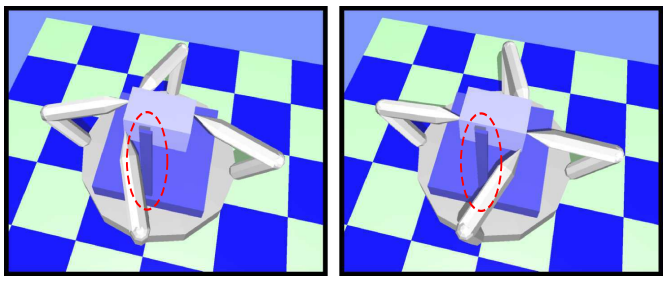


FIGURE 3.10 – Dans cet exemple, pour relier des configurations avec le doigt de part et d'autre de l'obstacle (entouré par l'ellipse rouge en pointillés), il faut que l'objet ait une pose très proche dans les deux configurations et soit bien centré par rapport à l'obstacle. Ces configurations ont très peu de chance d'être tirées au hasard, c'est pourquoi elles sont ajoutées à l'aide d'une heuristique.

de certaines configurations peut se révéler particulièrement utile. Ces configurations sont de la forme (p_i, g_{k_i}) où p_i est le milieu du segment $[p_1, p_2]$ et g_{k_i} est choisi aléatoirement. L'idée est de générer plusieurs configurations pour lesquels la pose de l'objet est très proche (en l'occurrence identiques) mais la prise différente. A chaque tentative de fusion de composantes connexes du graphe, on ajoute ainsi quelques configurations selon ce principe.

3.2.4 Décomposition des chemins dans GS_n en séquence transfert-ressaisie

On obtient une trajectoire solution au problème de planification quand les configurations initiale et finale du problème appartiennent à une même composante connexe du graphe. Cette trajectoire comporte des chemins dans GS_n, qui ne sont pas cinématiquement réalisables. Il faut les décomposer en une suite cinématiquement faisable, constituée de chemins de transfert et de ressaisie.

Soient $q_1 = (p_1, g_1)$ et $q_2 = (p_2, g_2)$ deux configurations reliées par un chemin dans GS_n. On cherche une séquence de chemins de transfert et de ressaisie permettant de relier q_1 à q_2. Pour calculer cette séquence, on utilise l'algorithme dichotomique 5.

Algorithme 5 : transforme_chemin(q_1,q_2)

```
si connexion_possible_par_Transfert_Ressaisie(q1,q2)==true alors
    décomposition réussie
    fin de l'algorithme
sinon
    si ||p1 − p2|| ≥ ε alors
        q_int ← configuration_intermediaire(q1,q2)
        transforme_chemin(q1,q_int)
        transforme_chemin(q_int,q2)
    sinon
        ECHEC
        fin de l'algorithme
    fin
fin
```

La fonction $transforme\_chemin()$ reçoit deux configurations q_1 et q_2 reliées par un chemin p dans GS_n et tente de les relier par un chemin de transfert suivi d'une reconfiguration de la prise de l'objet. En cas de réussite, la décomposition est terminée. Sinon, une configuration intermédiaire q_{int}, appartenant à p et comprise entre q_1 et q_2, est tirée et la fonction $transforme\_chemin()$ est appelée pour (q_1, q_{int}) et (q_{int}, q_2).

Un petit nombre réel positif ϵ est posé. Si la fonction $transforme\_chemin()$ est appelée pour deux configurations telles que la norme de la différence entre les deux configurations de l'objet associées est inférieure à ϵ, alors on considère que la décomposition a échoué. Une telle situation signifie que l'algorithme de décomposition ne converge pas ou converge trop lentement, autrement dit que le chemin dans GS_n n'était pas sans collision ou "trop proche" d'une collision. Ce dernier cas entraînerait un trop grand nombre de chemins de transfert-ressaisie.

3.2.5 Lissage des chemins

A cause de l'utilisation de tirages aléatoires dans l'étape de construction du graphe de manipulation, les chemins solutions trouvés sont irréguliers et souvent inutilement longs. C'est pourquoi on procède à une opération de lissage des chemins. Les chemins concernés sont les chemins dans GS_n et les chemins de ressaisie. Le lissage est effectué à l'aide de tirages aléatoires comme décrit dans la partie 1.4.3. Le principe de ce lissage est de choisir aléatoirement deux configurations q_1 et q_2 le long du chemin à lisser p, puis d'essayer de connecter directement ces deux configurations. Comme les chemins utilisés sont linéaires, le nouveau chemin entre q_1 et q_2 sera plus court que l'ancien et, de même, le nouveau p sera plus court. Dans le cas des chemins dans GS_n, les chemins seront plus courts en terme de déplacement de l'objet et, dans le cas des chemins de ressaisie, ils seront plus courts en terme de déplacement des doigts lors de leur mouvement de repositionnement sur la surface de l'objet. Cette opération est répétée tant qu'elle apporte un raccourcissement du chemin p supérieur à un certain seuil, à fixer par l'utilisateur.

Il est intéressant de remarquer qu'après décomposition d'un chemin dans GS_n en séquence de chemins de transfert-ressaisie et lissage des chemins de ressaisie obtenus, les trajectoires des doigts lorsqu'ils se repositionnent sur la surface de l'objet durant une reconfiguration de prise, suivent précisément le contour de l'objet et, en quelque sorte, glissent sur sa surface. Les prises apparaissent alors comme se faisant toujours avec n doigts.

3.3 Paramétrisation de la surface des objets

Nous avons vu que pour explorer GS_n, il est important de pouvoir connecter deux configurations de ce sous-espace et donc de calculer des chemins le long desquels la prise varie continûment. Dans les travaux de [SSC02][1], traitant de la manipulation à

[1] A. Sahbani, T. Siméon, and J. Cortés. A probabilistic algorithm for manipulation planning under continuous grasps and placements. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'2002)*, 2002.

l'aide d'un bras robotisé, les auteurs ont eu eux aussi à modéliser une variation continue de prise. Ils ont simplifié le problème en considérant le cas d'une pince à mors parallèles saisissant une barre parallélépipédique et en supposant que la prise ne se fait que sur une des paires de faces opposées de la barre. Ainsi, seuls trois degrés de liberté étaient associés à la prise (deux en translation et un en rotation). Dans le cas de la manipulation dextre, le problème est plus complexe. Comme la prise est caractérisée par la position des points de contact sur la surface de l'objet, il est nécessaire d'avoir une paramétrisation de cette surface pour pouvoir faire changer la prise de façon continue et ainsi calculer des chemins dans GS_n. Nous avons jusqu'à présent supposé que les surfaces des objets étaient paramétrées. Bien qu'il existe des familles de surfaces paramétrées souvent utilisées pour modéliser des objets, notamment en synthèse d'images par ordinateur (super-ellipsoïdes, super-toroïdes, super-quadriques, super-cylindres, etc.), il n'est généralement pas possible de trouver une paramétrisation de la surface d'un objet quelconque. De plus, la forme de l'objet sera déterminée en pratique à l'aide d'un dispositif de reconstruction 3D tel qu'un système multi-caméras ou un système de télémétrie laser. La surface de l'objet sera donc connue à travers un ensemble de points ou de faces et non pas à l'aide d'une expression analytique. La façon la plus générale de considérer la surface de l'objet est donc de l'approximer par un polyèdre, que l'on supposera uniquement composé de facettes triangulaires. Ce n'est pas une limitation très gênante car on peut approximer n'importe quelle surface, avec la précision que l'on veut, par un polyèdre —au prix d'un plus grand besoin en mémoire et en calcul. Il existe des techniques pour trouver une paramétrisation sphérique pour tout polyèdre dont la surface est de genre 0 [4]. La surface du polyèdre peut alors être paramétrée par des coordonnées de type sphérique (angles longitudinal et latitudinal). Ces techniques sont utilisées en synthèse d'image pour de nombreuses applications (*mapping* d'une texture 2D sur la surface d'un polyèdre, modification du maillage de la surface (*remeshing*), *morphing* entre deux objets, etc.) [FH05][1]. Leur implémentation n'est pas aisée et sort du cadre de notre sujet. C'est pourquoi nous nous sommes intéressés à une famille beaucoup plus restreinte de polyèdres qui nous a cependant permis de planifier des tâches de manipulation pour différentes formes d'objet. Il s'agit des polyèdres étoilés.

Les polyèdres étoilés Un polyèdre étoilé est un polyèdre dont le volume définit une région étoilée. Une région étoilée est un ensemble E tel qu'il existe $o \in E$ vérifiant

4. Une surface est de genre 0 si toute courbe fermée simple sans points communs que l'on peut tracer à l'intérieur de cette surface la déconnecte, c'est-à dire que le complémentaire de cette courbe n'est pas connexe. Concrètement, tout découpage fermé de la surface la coupera en deux. La sphère est une surface de genre 0 et toute surface sans trou est aussi de genre 0.

[1] M.S. Floater and K. Hormann. Surface parameterization : a tutorial and survey. pages 157–186, 2005.

$\forall p \in E, op \in E$. Une conséquence de cette propriété est que toute demi-droite partant de o coupe la surface de E en un point et un seul. Connaissant o, on peut donc associer de façon unique un point de la surface du polyèdre à une direction dans l'espace. Cela revient à disposer d'une paramétrisation de la surface de l'objet. Connaissant une direction d, on peut calculer quel triangle du polyèdre intersecte la demi-droite de direction d passant par o et en quel point. De même, connaissant un point p de la surface du polyèdre, la paramétrisation correspondante sera la direction de op. Pour échantillonner uniformément la surface du polyèdre, un échantillonnage uniforme de direction n'est pas approprié puisque certaines zones de la surface seront coupées par plus de rayons que d'autres. A la place, on choisit aléatoirement des triangles du polyèdre avec un biais sur leur aire et on choisit ensuite un point sur leur surface, la surface d'un triangle étant facilement paramétrable. Le biais sur l'aire des triangles est important car la densité de triangles n'est généralement par la même sur toute la surface de l'objet, car les régions courbées sont approximées avec un plus grand nombre de triangles que les régions plates. Les polyèdres étoilés permettent de représenter bon nombre d'objets, y compris non convexes, ce qui rend leur usage intéressant.

Le fait qu'on ait besoin que la surface de l'objet à manipuler soit paramétrée, pour utiliser notre méthode, n'est donc pas particulièrement contraignant. Son application pratique est ainsi possible pour de très nombreuses formes d'objet.

3.4 Extension de l'approche

3.4.1 Problématique

La méthode proposée plus haut travaille en explorant les différentes composantes connexes de GS_n, via des chemins spécifiques à cet espace, et en connectant ces composantes avec des séquences ressaisie-transfert. Elle réduit, par conséquent, l'espace de recherche puisqu'elle n'explore alors que GS_n et GS_{n-1} (ce dernier avec les chemins de ressaisie). Cette approche privilégie les configurations qui ont le plus de contacts donc le plus de chance d'être stables et permet d'accélérer la recherche d'une solution, mais elle ne considère que des mouvements de manipulation alternant prises à n doigts et prises à $(n-1)$ doigts. Il peut néanmoins exister des problèmes qui nécessitent des mouvements alternant d'autres types de prise. Par exemple, pour une main à cinq doigts, certaines situations peuvent requérir une alternance de prises à trois doigts et de prises à quatre doigts et d'autres de prises à quatre doigts et de prises à cinq doigts. Le premier cas apparaît si l'objet doit être amené loin du domaine d'accessibilité d'un des doigts. Le second quand certaines configurations de l'objet ne permettent pas de trouver de prises stables à moins de quatre doigts. Il faut alors explorer d'autres sous-espaces, qui correspondent à des prises avec moins de $(n-1)$ doigts, pour ne pas perdre de solutions.

La partie suivante explique comment adapter la méthode de planification à de telles situations.

3.4.2 Une méthode plus générale

La méthode la plus générale, celle qui assure de trouver toutes les solutions, doit prendre en compte tous les types d'alternance de prise possibles. Par conséquent, tous les sous-espaces de prise GS_k contenant des configurations de prise stables doivent être explorés. L'exploration de chacun des GS_k peut se faire de la même façon que la méthode de la partie 3.2 explore GS_n. Les graphes construits dans les différents GS_k sont fusionnés à l'aide de chemins de transfert-ressaisie. Cette méthode peut s'avérer très complexe si la main comporte un grand nombre de doigts, car alors les possibilités de choix des doigts participant à la prise de l'objet sont très nombreuses. Comme GS_k a au moins C_n^k composantes connexes (voir partie 2.2.1.3), le nombre de composantes connexes des sous-espaces de prise peut être énorme. Pour une main à dix doigts et pour des prises de quatre à dix doigts, il y a 848 ($\sum_{k\in[\![4;10]\!]} C_n^k$) choix possibles pour les doigts participant à la prise. Explorer les espaces de prise associés aboutit certainement à des temps de calcul prohibitifs. C'est pourquoi, nous avons choisi, dans la méthode de la partie 3.2, de restreindre l'exploration à GS_n et GS_{n-1}. En pratique, cependant, les mains robotisées n'ont jamais plus de cinq doigts et, dans ce cas, il devient possible de considérer toutes les prises possibles. La partie suivante explique comment s'y prendre pour une main à cinq doigts.

3.4.3 Application à une main à cinq doigts

On s'intéresse au cas d'une main à cinq doigts pour des contacts ponctuels avec frottement. Les seuls espaces de prise à explorer sont ceux à trois, quatre et cinq doigts. L'idée va être d'explorer les composantes connexes de GS_4 avec des chemins dans ces sous-espaces et de les connecter par des chemins de transfert-ressaisie dans GS_3 ou dans GS_5. Un graphe est construit dans chacun des $GS_4^i, i \in [\![1;5]\!]$. Pour chacun des GS_4^i, les composantes connexes du graphe sont fusionnées via des chemins dans GS_3 (levers de doigts) ou des chemins dans GS_5 (chemins décrits dans la partie 3.1.1). Ensuite, les graphes des différents GS_4^i sont fusionnés à leur tour via ces mêmes types de chemin. Tous les graphes peuvent être éventuellement fusionnés à l'aide de chemins dans GS_5, mais seuls ceux dont les GS_4^i ont trois doigts, participant à la prise, en commun peuvent éventuellement être fusionnés à l'aide de chemins de ressaisie dans GS_3 (voir la partie 2.2.1.3 sur la connexité des GS_k). La figure 3.11 illustre ce principe. Seule une partie des GS_3^i sont représentés par souci de clarté.

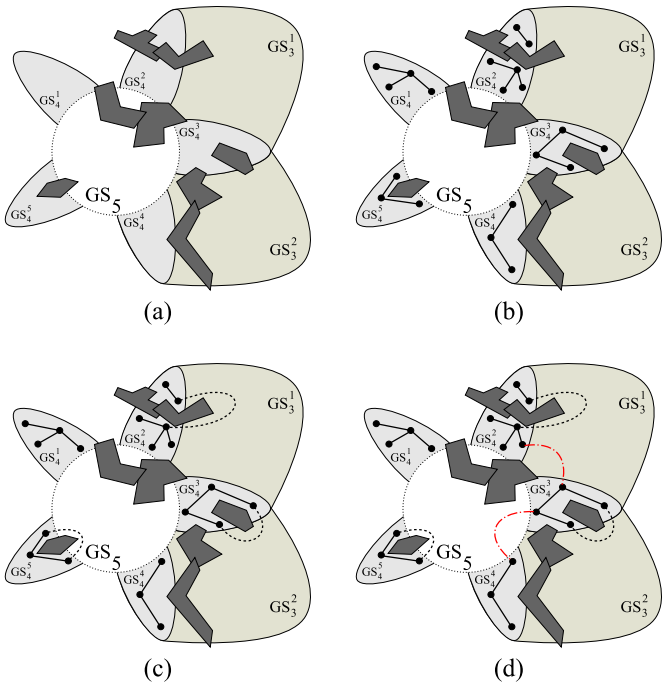


FIGURE 3.11 – Un graphe est construit pour explorer chaque GS_4^i (b). On essaye ensuite de fusionner les composantes connexes de chacun des graphes via des reconfigurations de prise ou des chemins dans GS_5 (c). Enfin, les différents graphes sont fusionnés à leur tour, via ces mêmes types de chemin (d).

L'algorithme 6 construit et fusionne les diverses parties du graphe. Un nouveau paramètre intervient. C'est le paramètre β, qui, suivant sa valeur va privilégier l'exploration de chacun des GS_4^i ou leur connexion. L'autre nouveauté par rapport à la méthode de la partie 3.2 concerne les connexions des différents graphes, réalisées par les fonctions $connecter\_composantesGS_4^j()$ et $fusionner\_graphes\_GS_4^i()$. Précédemment, les connexions se faisaient uniquement dans GS_3, alors qu'ici elles peuvent se faire également en passant dans GS_5. Cela amène deux questions : comment choisir entre la connexion par GS_3 et celle par GS_5 ? comment relier deux configurations de GS_4 via GS_5 ?

Algorithme 6 : Algorithme de construction du graphe de manipulation pour une main à cinq doigts avec l'approche généraliste

$\mathcal{GM} = \{q_{init}, q_{goal}\}$
$\mathcal{CC} = \{l_1, l_2\}$
$\alpha \in]0;1[$
$\beta \in]0;1[$
si $l_1 \equiv l_2$ **alors**
    fin de l'algorithme
sinon
    tant que $l_1 \neq l_2$ **faire**
        choisir aléatoirement $x \in]0;1[$
        si $x < \beta$ **alors**
            choisir aléatoirement $j \in [\![1;5]\!]$
            choisir aléatoirement $y \in]0;1[$
            si $y < \alpha$ **alors**
                $explorerGS_4^j(q_{init},q_{goal},\mathcal{GM},\mathcal{CC})$
            sinon
                $connecter\_composantesGS_4^j(\mathcal{GM},\mathcal{CC})$
            fin
        sinon
            $fusionner\_graphes\_GS_4^i(\mathcal{GM},\mathcal{CC})$
        fin
    fin
fin

3.4.3.1 Choix de l'espace de connexion

Un changement de type de prise peut être nécessaire pour trois raisons qui sont la présence d'un obstacle dans l'environnement, la rupture de la stabilité de la prise et

la limitation des mouvements des doigts. Dans le premier cas, il semble logique que la solution soit de rompre des contacts car un doigt libre a une plus grande mobilité. Dans le second cas, il semble logique d'augmenter le nombre de contacts sur l'objet pour rendre la prise plus stable. Dans le troisième cas, il est plus difficile de savoir quelle stratégie est la plus intéressante. Pour choisir l'espace de connexion de deux composantes de GS_4, plutôt que de s'en remettre à un tirage aléatoire, on peut donc essayer de se baser sur ces deux observations. Il n'est pas forcément facile de mesurer l'encombrement d'une configuration de prise mais on trouve de nombreux critères de stabilité de prise dans la littérature. Si on veut relier deux configurations de GS_4 appartenant à des composantes connexes différentes du graphe, on prendra en compte un critère de stabilité. Plus une de ces deux configurations est proche de l'instabilité, plus on favorisera une tentative de connexion passant par GS_5.

3.4.3.2 Connexion via GS_5

La nouveauté de la connexion des configurations de GS_4 en passant par GS_5 concerne le positionnement du doigt qui crée le cinquième contact. Si on veut relier deux configurations q_1 et q_2 appartenant au même GS_4^i, ce doigt supplémentaire n'est en contact avec l'objet ni dans q_1 ni dans q_2 (figure 3.12-a). Il faut donc lui choisir un point de contact. Pour cela, on prend un point de la surface de l'objet situé dans le domaine d'accessibilité du doigt, pour les configurations initiale et finale de l'objet.

Si q_1 et q_2 n'appartiennent pas au même GS_4^i, le doigt qui crée le cinquième contact participe à la prise dans la configuration q_2 (figure 3.12-b). La position finale de ce doigt est donc imposée par q_2 et sa position initiale est choisie dans son domaine d'accessibilité.

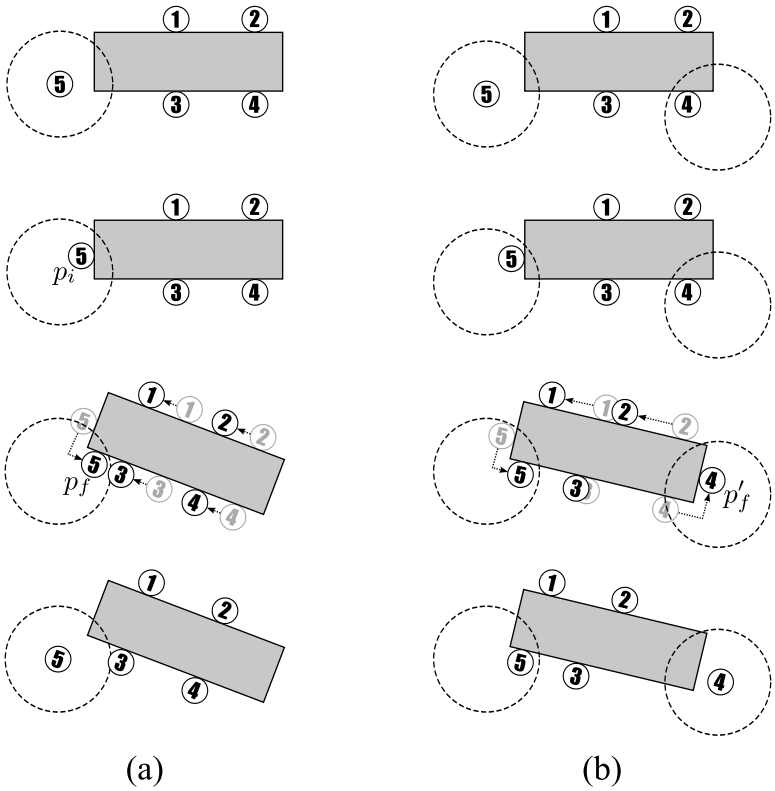


FIGURE 3.12 – Si un des contacts n'apparaît ni dans la configuration initiale ni dans la configuration finale, on choisit, au hasard, ses positions initiale et finale sur l'objet (p_i et p_f), dans le domaine d'accessibilité (cercle en pointillés) du doigt correspondant (a). On procède de même pour un contact qui ne se fait que dans la configuration initiale, en choisissant sa position finale sur l'objet dans domaine d'accessibilité (p'_f) (b).

3.5 Conclusion

Dans ce chapitre une nouvelle méthode de planification de mouvement pour la manipulation dextre a été présentée. Son intérêt vient du fait qu'elle privilégie une exploration directe de GS_n au lieu de calculer systématiquement des trajectoires de changement de prise, comme le font les méthodes existantes. Ces méthodes calculent explicitement les reconfigurations de prise, en résolvant à chaque fois un sous-problème de planification, car il faut calculer une trajectoire sans collision pour chaque doigt se repositionnant sur la surface de l'objet. Le calcul de ces reconfigurations de prise a non seulement un coût en temps mais aussi en mémoire, puisqu'il requiert l'introduction et le stockage de nombreuses configurations intermédiaires. Par ailleurs, les mouvements de l'objet étant très limités le long des chemins locaux de type transfert-ressaisie ou ressaisie-transfert, il faudra un échantillonnage très dense pour que les configurations-échantillons puissent être jointes par ce type de chemin.

En utilisant des chemins dans GS_n, qui ne sont décomposés en suite de changements de prise qu'une fois une solution trouvée, la méthode proposée évite ces deux inconvénients des méthodes traditionnelles. Le cas où le calcul des mouvements de reconfiguration de prise est nécessaire au cours de la recherche d'une solution, est traité par l'étape de fusion des composantes connexes du graphe d'exploration de GS_n par des chemins de transfert-ressaisie. Cette méthode ne considère que des alternances de prises à n et $(n-1)$ doigts. C'est pourquoi, une approche plus générale a été proposée, qui utilise toujours des chemins linéaires dans les espaces des configurations de prises mais ne limite plus l'exploration à GS_n.

Le chapitre suivant concerne l'application de notre méthode. Il présente des résultats obtenus en simulation en utilisant la méthode de planification qui explore uniquement GS_n.

Chapitre 4

Résultats de Simulation

Ce chapitre concerne la validation expérimentale de la technique de planification présentée dans le chapitre 3. Les expérimentations ont été faites en simulation et concernent des taches de manipulation diverses, différant par la forme de l'objet manipulé, ses configurations initiale et finale et la présence ou l'absence d'obstacles dans l'environnement. Ce chapitre commence par présenter la plate-forme de simulation développée et utilisée durant cette thèse. Puis, les résultats des simulations et les performances de notre planificateur sont détaillés et analysés.

4.1 Présentation de la plate-forme de simulation

Dans le cadre de cette thèse a été développée une plate-forme logicielle permettant de tester des algorithmes de planification sur des systèmes robotiques. Elle comprend plusieurs modules, qui interagissent et sont représentés sur la figure 4.1. Le module géométrie contient les représentations de tous les corps. Un certain nombre de primitives sont disponibles (sphères, boîtes, cylindres, tores) en plus de la représentation sous forme de polyèdres (liste de points et triangles définis à partir de ces points). Tous les corps ont cependant une représentation sous forme de polyèdre pour le modèle requis par la détection de collision. Le module géométrie gère également les calculs de type modèle géométrique pour les corps articulés et contient le sous-module de détection de collision. Le sous-module de détection de collision utilise la bibliothèque de

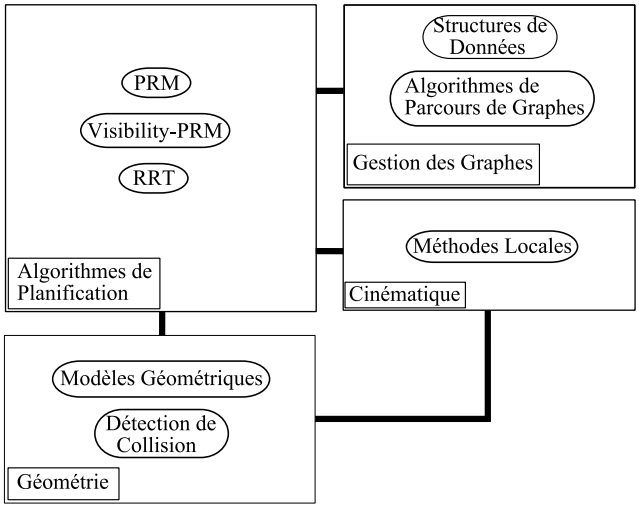


FIGURE 4.1 – Structure du programme de simulation.

fonctions PQP [1], écrite en C++ et développée à l'UNC (University of North Carolina). Elle utilise une modélisation géométrique des objets sous forme de liste de triangles, sans contrainte particulière sur leur arrangement. A l'initialisation, chaque objet est récursivement divisé en boîtes englobantes pouvant prendre n'importe quelle orientation (Oriented Bounding Boxes (OBB), [GLM96][1]) (figure 4.2). Un arbre binaire est créé pour chaque modèle et ses noeuds sont les OBB obtenues à la décomposition. Pour tester la collision entre deux objets, il suffit alors de tester récursivement les collisions entre les OBB des arbres des objets. Le module cinématique concerne le calcul

1. Proximity Query Package, http ://www.cs.unc.edu/ geom/SSV

[1] S. Gottschalk, M.C. Lin, and D. Manocha. OBBTree : A hierarchical structure for rapid interference detection. *Proceedings of ACM Siggraph'96*, 1996.

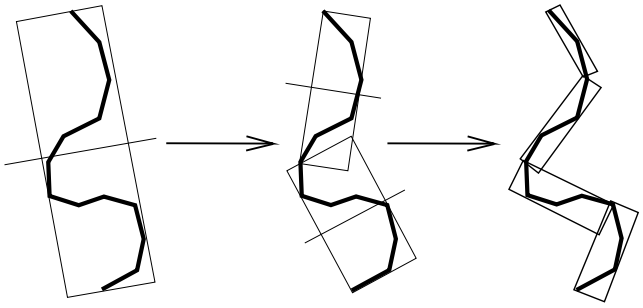


FIGURE 4.2 – Principe des OBB : à chaque itération, on coupe chaque OBB en deux et on calcule les OBB des parties de l'objet comprises dans chacune des moitiés obtenues (tiré de [GLM96]).

des chemins des méthodes locales respectant les contraintes cinématiques du système considéré. Un autre module prend en charge toute la gestion de données nécessaire à la création, au développement et à la fusion des graphes. Il intègre également des algorithmes de recherche de chemin dans un graphe. Enfin, le module de planification contient les différents algorithmes de planification (PRM, visibility-PRM et RRT) ainsi que les algorithmes spécifiques de la méthode de planification proposée dans cette thèse.

Comme les valeurs numériques fournies par le logiciel ne sont pas exploitables telles quelles par l'utilisateur et qu'il est préférable de ne pas s'en remettre uniquement aux tests logiciels pour vérifier la validité des solutions trouvées, il est important de disposer d'une interface graphique permettant de visualiser les trajectoires obtenues. C'est pourquoi le planificateur a été doté d'une interface graphique, réalisée avec la bibliothèque de fonctions d'affichage 3D OpenGL [2], qui permet à la fois de visualiser les trajectoires mais aussi de définir des problèmes de planification en créant de nouveaux environnements et en indiquant visuellement les configurations initiale et finale désirées.

4.2 Résultats

Nous présentons dans cette partie les résultats obtenus pour la simulation de diverses tâches de manipulation dextre. Il a fallu choisir une géométrie pour la main simulée ainsi qu'un modèle pour les contacts entre les doigts et l'objet. Nous avons opté pour le modèle de contact le plus couramment employé, à savoir le modèle du contact ponctuel avec frottement (voir partie 2.1.4.1). Ce modèle choisi, la main doit avoir au moins quatre doigts car une prise stable nécessite au minimum trois contacts et il faut pouvoir lever un doigt pour le repositionner. Comme c'est la seule condi-

2. http ://www.opengl.org

tion sur le nombre minimum de doigts, nous avons décidé de travailler avec une main à quatre doigts. L'ajout de doigts supplémentaires permet de tenir l'objet en un plus grand nombre de points de contact donc d'améliorer la stabilité de la prise mais rend l'implémentation logicielle de la méthode de planification plus compliquée. En pratique, beaucoup des mains existant dans les laboratoires de recherche ont quatre doigts ([BLMV04][1]) ce qui permet de simplifier leur mécanique et de réduire les coûts de fabrication. Certaines mains ont cinq doigts mais vraisemblablement dans le but d'accroître leur ressemblance avec la main humaine. Nous avons opté pour la géométrie représentée sur la figure 4.3. Cette géométrie offre l'avantage de bien répartir dans l'espace les domaines d'accessibilité des doigts et d'utiliser le même modèle géométrique pour tous les doigts.

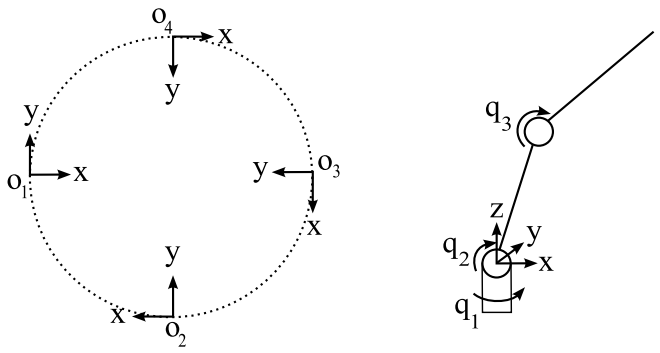


FIGURE 4.3 – La main a quatre doigts à trois degrés de liberté, répartis symétriquement autour de la paume.

Nous présentons maintenant plusieurs exemples de tâches de manipulation dextre ainsi que les résultats produits par notre planificateur. Comme la méthode proposée utilise des algorithmes de planification probabilistes, les temps de calcul et le graphe construit dépendent du tirage aléatoire ayant servi à l'échantillonnage des configurations. Le résultat d'un seul calcul n'est donc pas pertinent pour évaluer les performances de la méthode. Il faut s'intéresser à des performances statistiques. Pour cela, les calculs des exemples sont effectués un grand nombre de fois avec des tirages aléatoires différents et les temps de calcul sont moyennés. Parmi les grandeurs physiques à préciser, figure seulement le coefficient de frottement des contacts. Nous choisissons un coefficient de 0.8, qui semble raisonnable comparé à ceux de matériaux ordinaires (par exemple, bois sur métal ≈ 0.7, verre sur verre ≈ 0.9, pneu sur route sèche ≈ 1). La masse de l'objet ainsi que son inertie n'interviennent pas puisque nous avons choisi de ne pas prendre en compte les limites des couples articulaires des doigts.

[1] L. Biagiotti, F. Lotti, C. Melchiorri, and G. Vassura. How far is the human hand ? a review on anthropomorphic robotic end-effectors. *Internal Report - DEIS (University of Bologna)*, 2004.

La machine de calcul utilisée est un PC équipé d'un processeur Intel Core2Duo (bi-processeur cadencé chacun à 2.3GHz) et de 2Go de RAM. Comme le planificateur n'a pas été programmé de façon à tirer partie des possibilités de calcul parallèle, son exécution utilise un seul processus. Ainsi, seul un des deux processeurs travaille et seule la moitié de la puissance de calcul de la machine est exploitée. Il est donc théoriquement possible de diviser les temps de calcul obtenus par un facteur proche de 2. Les temps de résolution donnés ne prennent pas en compte la décomposition des chemins dans GS_4 ni leur lissage. Ce premier processus est déterministe pour un chemin donné et prend un temps de l'ordre de quelques secondes (de 1 à 5 secondes environ pour les exemples présentés). Le lissage utilise des tirages aléatoires et se fait après avoir trouvé une solution et n'intervient donc pas sur les temps de calcul. La durée de cette étape dépend de la qualité de lissage souhaitée. Généralement, quelques secondes seront suffisantes pour avoir des trajectoires solution très douces. Les valeurs du paramètre α retenues dans les tableaux de résultats sont celles qui ont donné les meilleures performances pour notre planificateur. La partie 4.3 revient sur l'influence de ce paramètre sur les performances de la méthode de planification.

4.2.1 Réorientation d'une sphère

Le premier exemple est la réorientation d'une sphère. La simplicité de cette forme et de sa paramétrisation en ont fait l'exemple le plus fréquent dans la littérature. La configuration initiale du système et sa configuration finale désirée sont reproduites sur la figure 4.4. La sphère doit être réorientée d'un angle π autour d'un axe horizontal. Le paramètre α est choisi constant et égal à 0.9.

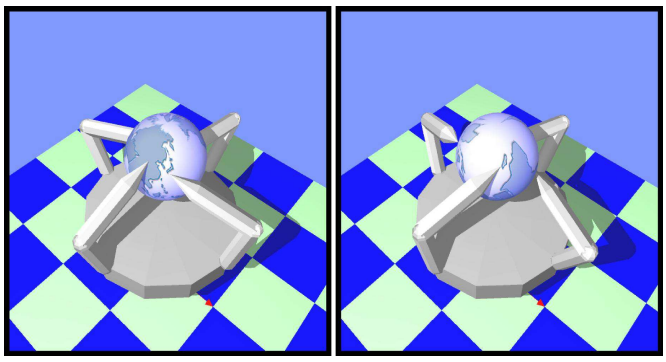


FIGURE 4.4 – Les configurations initiale et finale du premier exemple.

La figure 4.5 montre quelques configurations d'une solution trouvée par le planificateur.

La principale originalité de la méthode proposée est d'explorer préférentiellement GS_n à l'aide de chemins dans cet espace, ce qui évite d'avoir à déterminer les mou-

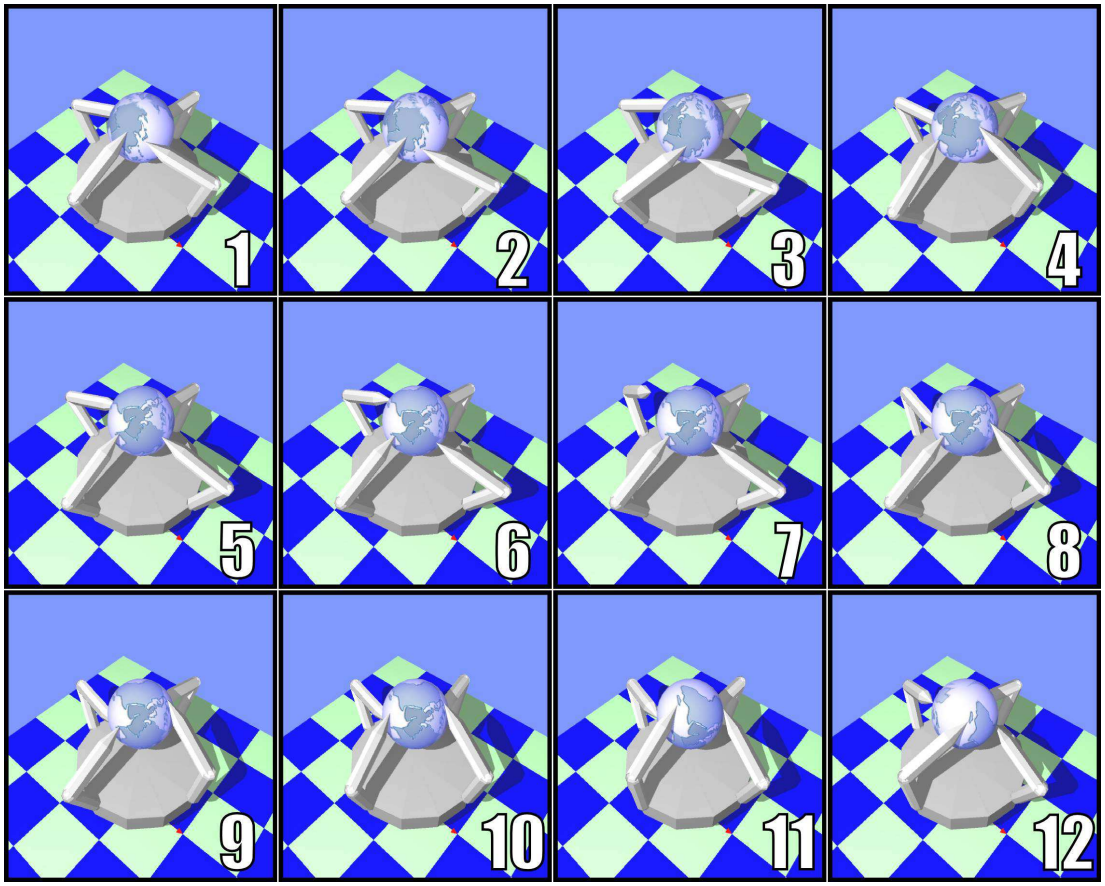


FIGURE 4.5 – Trajectoire solution du problème de réorientation d'une sphère.

vements de reconfiguration de prise au cours de l'exploration. Pour vérifier que l'exploration de GS_n à l'aide de ces chemins particuliers apporte un avantage en terme de rapidité, nous avons testé les résultats de la résolution de cet exemple avec une méthode n'utilisant que des chemins de type transfert-ressaisie et ressaisie-transfert, que nous appellerons "méthode classique". La méthode classique est juste une méthode PRM ou visibility-PRM dont la méthode locale est une séquence simple transfert-ressaisie ou ressaisie-transfert. Le tableau 4.1 rassemble quelques informations sur la résolution du problème avec les deux méthodes. Le nombre de noeuds générés est le nombre de

méthode	méthode proposée			méthode classique		
	min	moyenne	max	min	moyenne	max
temps de résolution (s)	0.4	5	21	53	580	2141
nombre de noeuds générés	1	41	186	90	858	2733

TABLE 4.1 – Résultats obtenus pour l'exemple de la réorientation d'une sphère avec les deux méthodes, pour 200 tests.

configurations valides générées en vu d'être ajoutées au graphe, qui comporte initialement deux noeuds correspondant aux configurations initiale et finale. On constate une très grande différence en terme de temps de calcul entre les deux méthodes. L'exploration de GS_4 permet de réduire considérablement le nombre d'échantillons nécessaires à la résolution du problème. Beaucoup moins de connexions doivent être testées et le calcul d'une connexion dans GS_4 est plus rapide que celui d'une connexion de type transfert-ressaisie puisque la dimension de l'espace de recherche associé est plus petite.

4.2.2 Translation d'une sphère en présence d'un obstacle

Le deuxième exemple concerne toujours la manipulation d'une sphère, cette fois en présence d'un obstacle, qui rend obligatoire un lever de doigt bien précis. L'espace GS_4 est ainsi coupé en deux par la présence de l'obstacle. La sphère doit être translatée horizontalement. Le paramètre α est choisi constant et égal à 0.3.

La figure 4.7 représente les étapes clefs d'une solution trouvée par le planificateur. Le tableau 4.2 fournit des résultats obtenus avec notre méthode et la méthode classique. Le nombre de tests a été fortement réduit (seulement 15 itérations) pour cette dernière méthode, étant donné la longueur des temps de résolution associés. Les résultats permettent donc plus de constater la différence de rapidité des deux méthodes que de comparer leurs performances rigoureusement. Pour la même raison, les deux derniers exemples du chapitre, nettement plus difficiles, n'ont été testés qu'avec notre méthode.

Le tableau 4.2 montre comment, sur cet exemple, l'écart de performance entre les deux méthodes est encore plus flagrant. Cet exemple est particulièrement difficile à

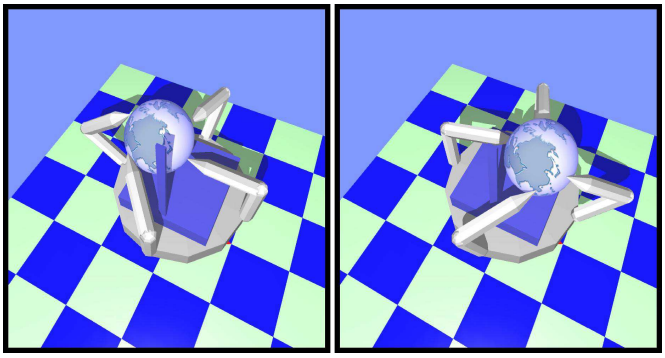


FIGURE 4.6 – Les configurations initiale et finale de l'exemple du déplacement d'une sphère en présence d'un obstacle.

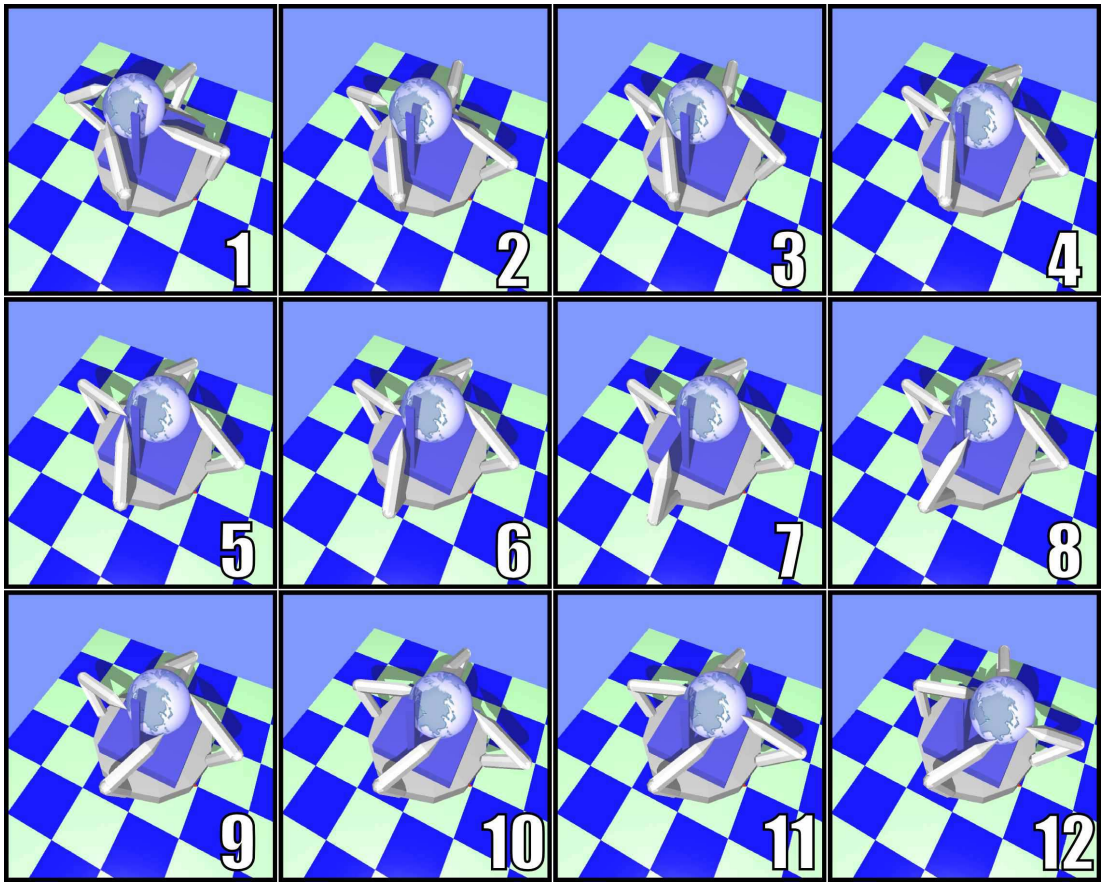


FIGURE 4.7 – Trajectoire solution du problème de translation d'une sphère en présence d'un obstacle.

méthode	méthode proposée			méthode classique		
	min	moyenne	max	min	moyenne	max
temps de résolution (s)	0.7	2.4	57	4839	15548	37160
nombre de noeuds générés	6	90	250	1805	6386	15477

TABLE 4.2 – Résultats obtenus pour l'exemple de la translation d'une sphère en présence d'un obstacle avec les deux méthodes, (200 tests pour la méthode proposée et 15 pour la méthode classique).

résoudre pour la méthode classique car deux configurations très précises doivent être tirées : objet centré sur l'obstacle avec le doigt devant l'éviter placé de part et d'autre de l'obstacle dans les deux configurations (images numérotées 4 et 9 sur la figure 4.7). Lors de la tentative de connexion des différentes composantes connexes de GS_4 à l'aide de chemin de ressaisie, les configurations ajoutées automatiquement par la méthode proposée correspondent à de telles situations, ce qui permet de résoudre ce genre de difficultés.

4.2.3 Réorientation d'un parallélépipède

Le troisième exemple est la réorientation d'un parallélépipède (une boîte, figure 4.8) d'un angle π autour d'un axe horizontal. Contrairement aux exemples précédents, l'objet présente des arêtes prononcées. Comme il n'est pas possible de passer d'une prise à une autre de manière continue, sans rompre la stabilité de la prise, quand les doigts sont sur des faces différentes de la boîte, GS_4 aura nécessairement plusieurs composantes connexes. La résolution du problème sera donc logiquement plus longue que dans le cas d'un objet lisse et régulier comme la sphère. Le paramètre α est choisi initialement égal à 0.5 puis décrémenté de 0.05 à chaque tentative de connexion des composantes connexes du graphe de manipulation via des chemins de ressaisie jusqu'à 0.3.

La figure 4.9 reproduit les étapes importantes d'une trajectoire solution calculée. Le tableau 4.3 réunit des informations sur la résolution du problème, notamment la comparaison avec la méthode classique. Une fois de plus, notre méthode obtient de meilleurs résultats.

méthode	méthode proposée			méthode classique		
	min	moyenne	max	min	moyenne	max
temps de résolution (s)	1.2	57	204	54	763	5702
nombre de noeuds générés	14	155	433	138	1165	5703

TABLE 4.3 – Résultats obtenus pour l'exemple de la réorientation d'une boîte avec les deux méthodes, pour 200 tests.

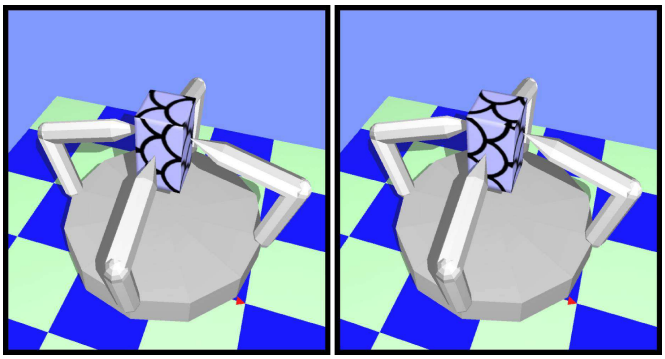


FIGURE 4.8 – Les configurations initiale et finale de l'exemple 3.

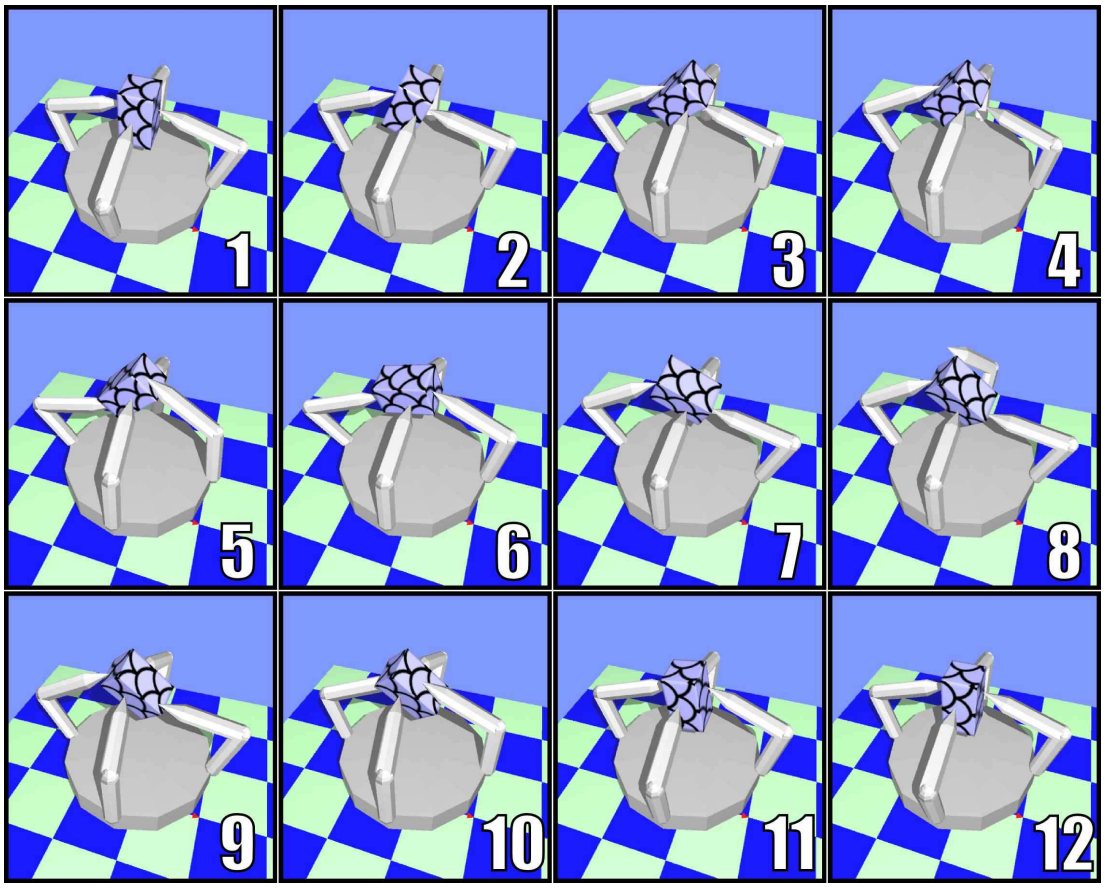


FIGURE 4.9 – Trajectoire solution au problème de réorientation d'une boîte.

4.2.4 Retournement d'un crayon

Le quatrième problème est intéressant car il ne peut être résolu par une méthode classique calculant d'abord la trajectoire de l'objet seul avant de calculer celle des doigts. L'objet à manipuler —un crayon— a une forme cylindrique très allongée. Il est d'abord tenu verticalement par une de ses extrémités. Le robot doit parvenir à une configuration finale où l'objet est toujours vertical mais tenu par l'autre extrémité (figure 4.10). Dans cet exemple, les configurations initiale et finale de l'objet sont liées par un simple mouvement de rotation mais il est nécessaire de translater l'objet au cours de la manipulation pour qu'il reste accessible à la main. Cette contrainte est prise implicitement en compte par la méthode de planification. Le paramètre α est choisi constant et égal à 0.6.

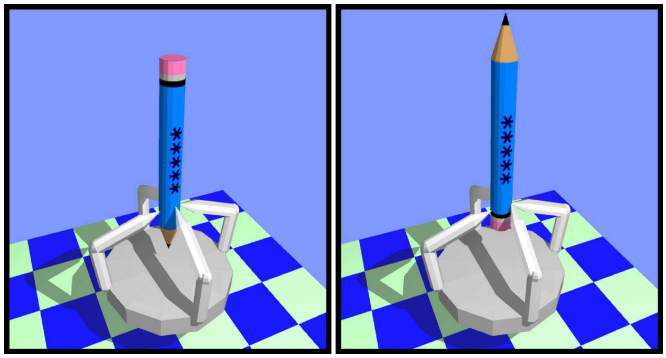


FIGURE 4.10 – Les configurations initiale et finale pour la réorientation d'un crayon.

La figure 4.11 expose quelques configurations d'une solution calculée par le planificateur. Le tableau 4.4 donne quelques informations sur la résolution du problème. Les temps de calcul sont nettement plus longs que dans les exemples précédents. Cela s'explique par la dimension plus réduite des zones où la prise de l'objet peut varier continûment sans rompre la stabilité de la prise, ce qui implique l'existence de très nombreuses composantes connexes de GS_4. Par ailleurs, générer des prises stables, avec une technique basée sur des tirages aléatoires, s'avère plus difficile pour un objet très allongé que pour un objet compact et symétrique comme la sphère et le parallélépipède des exemples précédents.

4.2.5 Insertion et vissage d'une ampoule

Nous présentons maintenant un exemple pour lequel le calcul de la trajectoire de l'objet nécessiterait à lui seul l'utilisation d'une technique de planification de mouvement. Il s'agit pour la main d'introduire une ampoule électrique dans une prise de forme appropriée (figure 4.12). Le paramètre α est choisi constant et égal à 0.8. Les

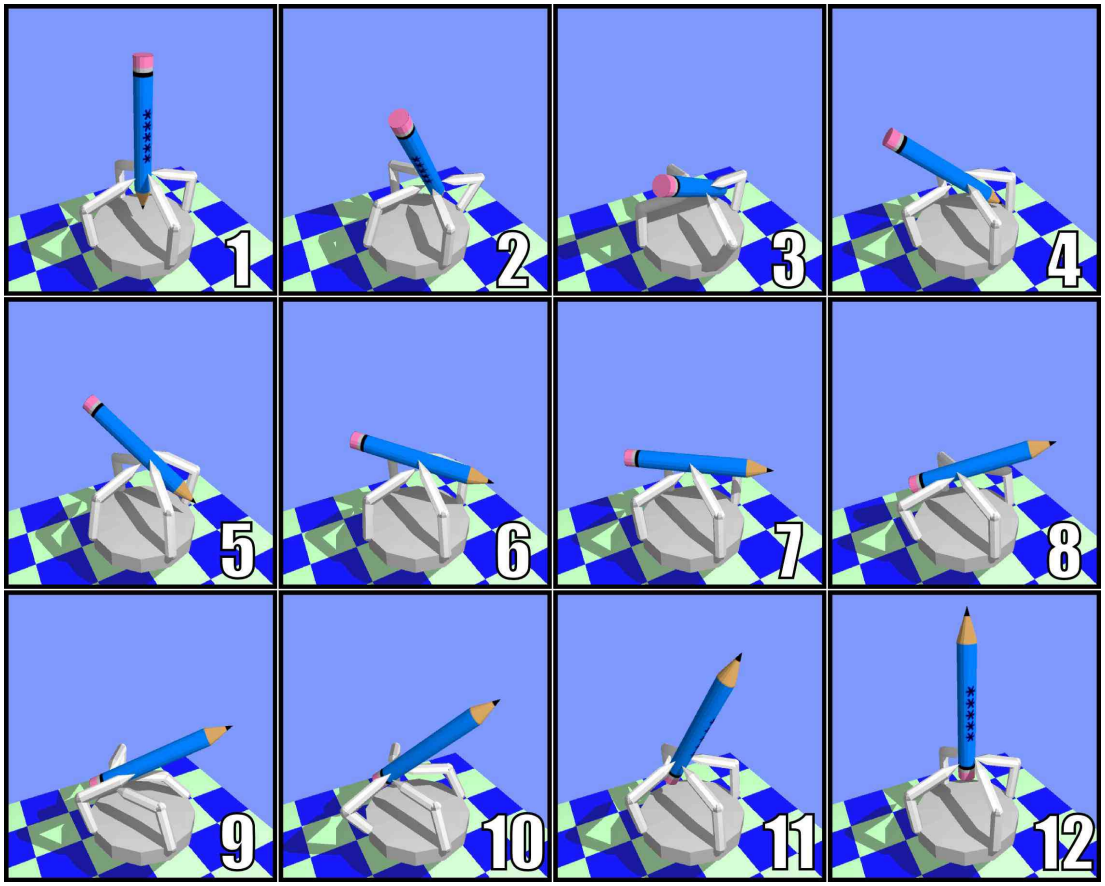


FIGURE 4.11 – Trajectoire solution du problème de réorientation d'un crayon.

méthode	méthode proposée		
	min	moyenne	max
temps de résolution (s)	87	699	2423
nombre de noeuds générés	381	1896	6731

TABLE 4.4 – Résultats obtenus pour l'exemple de la réorientation d'un crayon.

configurations initiale et finale du problème sont illustrées sur la figure 4.12. La figure 4.13 montre diverses étapes d'une solution calculée par le planificateur. Le tableau 4.5 regroupe quelques informations sur la résolution du problème. Les temps de calcul sont bien plus longs que pour les exemples précédents. Cela n'a rien de surprenant car la résolution de ce problème requiert le calcul d'un chemin dans un passage très étroit, ce qui pose problème aux méthodes de planification de mouvement en général et particulièrement aux méthodes probabilistes. Les résultats obtenus restent donc tout à fait satisfaisants.

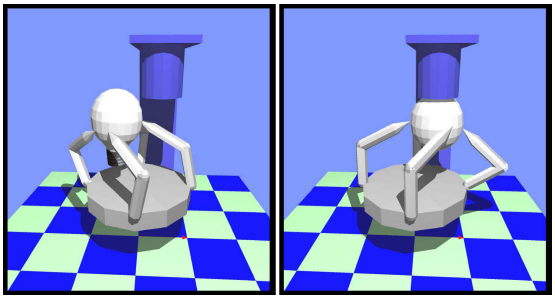


FIGURE 4.12 – Les configurations initiale et finale de l'exemple du remplacement d'une ampoule.

méthode	méthode proposée		
	min	moyenne	max
temps de résolution (s)	18	1019	5545
nombre de noeuds générés	28	836	4369

TABLE 4.5 – Résultats obtenus pour l'exemple de l'ampoule, pour 200 tests.

4.2.6 Temps de calcul des prises

Le temps de calcul des prises est un paramètre très important car, avec le calcul des chemins et les appels au détecteur de collision associés, le calcul des prises fait partie

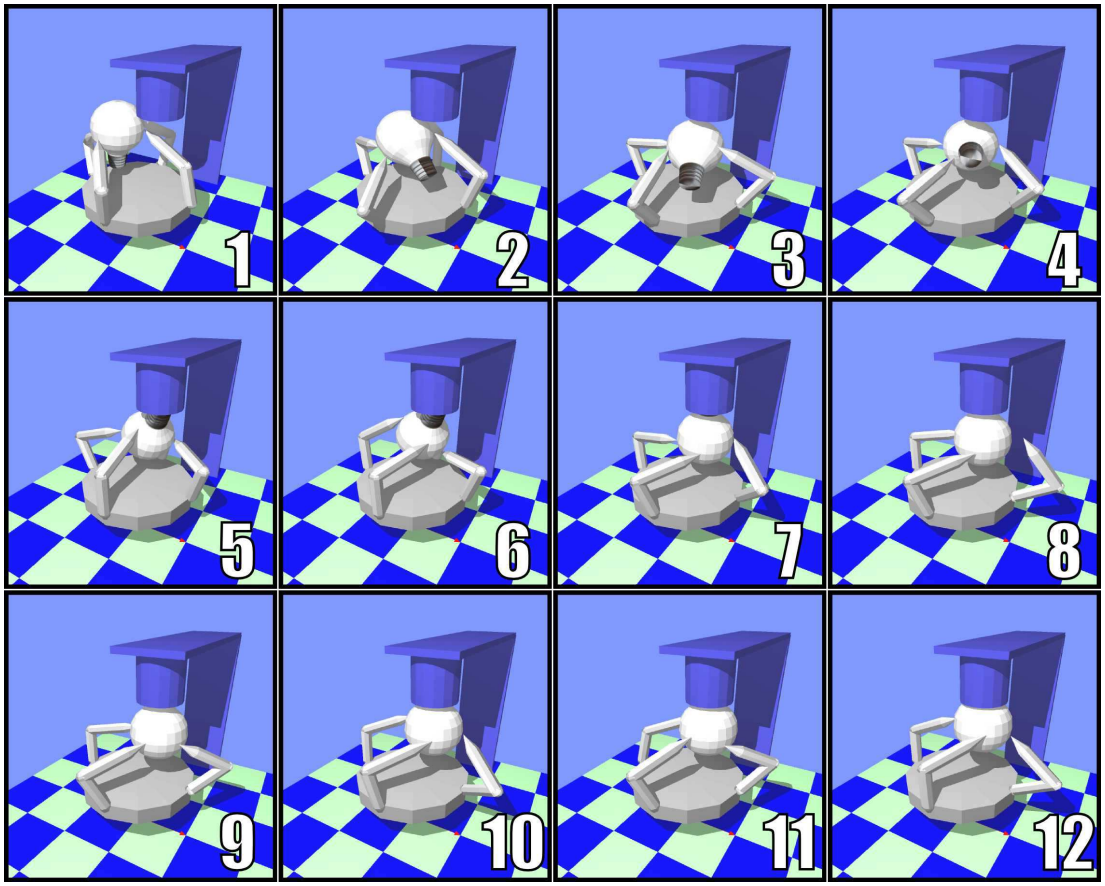


FIGURE 4.13 – Trajectoire solution du problème d'insertion d'une ampoule.

des tâches les plus coûteuses en temps de calcul. Les prises sont calculées de deux façons : par la technique de génération aléatoire intervenant dans l'étape d'extension du graphe dans GS_n (partie 3.2.2) et par la technique d'ajout de prises intervenant dans l'étape de fusion des composantes connexes du graphe par des chemins de transfert-ressaisie (partie 3.2.3). Comme l'étape de fusion est généralement beaucoup moins fréquente que l'étape d'extension du graphe, la seconde technique de génération de configuration de prise est beaucoup moins employée que la première. Par ailleurs, la seconde technique dépend du graphe à un moment donné, ce qui n'est pas le cas de la première. Il semble donc plus judicieux de s'intéresser aux performances de la première méthode qu'à celles de l'autre. Le tableau 4.6 regroupent les temps de calcul moyens de l'algorithme de génération de configurations de prise dans les conditions des exemples présentés plus haut, pour 2000 exécutions et rappelle les temps de résolution moyens de ces exemples. Ces exemples font plus ou moins appel à l'algorithme

exemple	temps de calcul moyen d'une configuration de prise	temps de résolution moyen du problème de planification
sphère	0.06s	5s
sphère avec obstacle	0.08s	2.4s
boîte	0.06s	57s
crayon	0.39s	699s
ampoule	0.49s	1019s

TABLE 4.6 – Temps de calcul de l'algorithme de génération aléatoire d'une configuration de prise de la partie 3.2.2, moyennés sur 2000 tests.

de la partie 3.2.2, suivant la valeur du paramètre α, mais il apparaît tout de même que les temps de résolution augmentent avec le temps de calcul des configurations de prise. Indépendamment de la méthode de planification, les temps de résolution peuvent donc être améliorés si la technique de génération des prises est perfectionnée ou si on choisit un modèle de contact facilitant la stabilité des prises (modèle *soft finger* au lieu de ponctuel avec frottement) ou une main pourvue de plus de doigts.

4.3 Influence du paramètre α

Nous avons jusqu'à présent pris des valeurs de α donnant de bons résultats. Nous rappelons que le paramètre α est un réel appartenant à l'intervalle $]0; 1[$ et que plus α est proche de 1, plus l'algorithme de planification favorise l'exploration de GS_n, alors que plus α est proche de 0 plus l'algorithme de planification favorise les tentatives de fusion des composantes connexes du graphe explorant GS_n, à l'aide de chemins de transfert-ressaisie. L'importance de ce paramètre amène à se poser la question de son

influence sur les performances du planificateur. Nous présentons trois courbes mesurant l'évolution du temps de résolution de trois exemples de la partie précédente en fonction de α. On constate logiquement que dans le premier exemple (figure 4.14) le

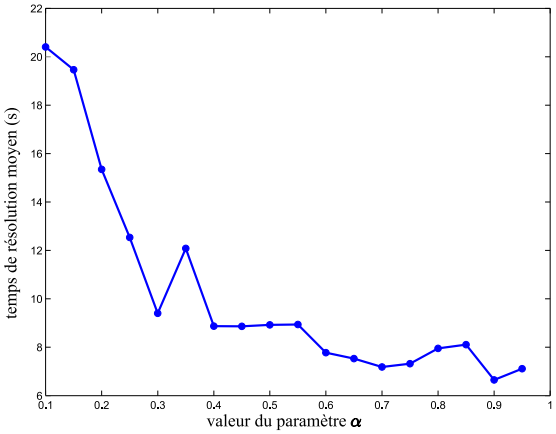


FIGURE 4.14 – Évolution du temps de résolution du premier exemple en fonction du paramètre α (200 tests par valeur de α) .

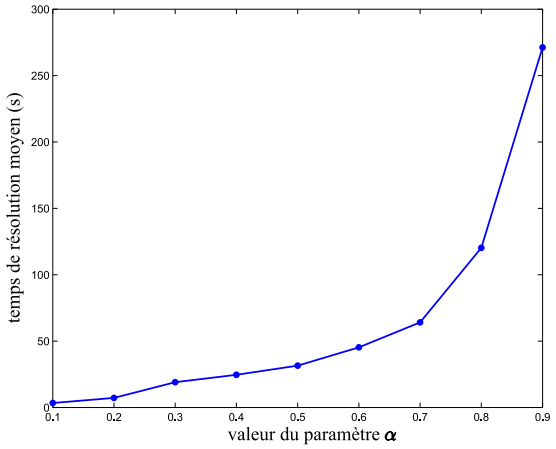


FIGURE 4.15 – Évolution du temps de résolution du deuxième exemple en fonction du paramètre α (100 tests par valeur de α).

temps de résolution augmente quand α diminue. Comme GS_4 possède une seule composante connexe, il est inutile et plus long de chercher des chemins dans un espace de dimension supérieure comme cela est fait à chaque tentative de connexion par une chemin de ressaisie. Dans le cas du deuxième exemple (figure 4.15), il est indispensable de chercher des connexions à l'aide de chemin de ressaisie. Comme l'exploration de

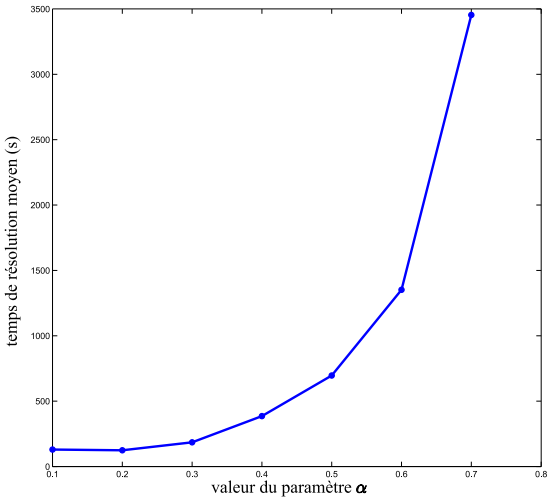


FIGURE 4.16 – Évolution du temps de résolution du troisième exemple en fonction du paramètre α (100 tests par valeur de α).

GS_4 se fait rapidement, c'est l'étape de connexion des composantes connexes de GS_4 qui va être déterminante. Plus α est grand plus la résolution prendra du temps et dans le cas extrême où α vaudrait 1, le planificateur ne trouverait pas de solution. Enfin, le troisième exemple nécessite plusieurs connexions de composantes connexes du graphe de manipulation via des chemins de ressaisie. La courbe représentant l'évolution du temps de résolution du problème de manipulation en fonction de α présente donc la même allure que dans le deuxième exemple. Le paramètre α devra donc être choisi d'autant plus petit que l'environnement gène le mouvement des doigts mais également que les domaines où la prise peut varier continûment sont petits, comme ce sera le cas pour un objet très anguleux. Cependant, la génération de configurations dans GS_n et l'exploration de cet espace doivent également être assurées. C'est pourquoi il est intéressant de trouver un compromis en prenant une valeur de α initialement grande et en la diminuant en suivant un critère d'évolution du graphe : par exemple, le nombre de noeuds du graphe ou le nombre d'échecs dans la tentative d'ajout d'un noeud au graphe explorant GS_n. Dans l'exemple de la réorientation du parallélépipède, c'est ainsi que l'on obtient les meilleurs résultats.

4.4 Analyse

Comme les divers exemples présentés le montrent, la méthode de planification proposée donne des résultats très intéressants, en ce qui concerne les temps de calcul et la possibilité de résoudre des problèmes complexes. Le gain apporté par l'exploration di-

recte de GS_n est très important voire considérable. Cela s'explique logiquement par le fait que deux configurations du système "main+objet" ne peuvent être connectées par une séquence transfert-ressaisie ou ressaisie-transfert que si à la fois les deux poses de l'objet et les positions des doigts sur l'objet sont très proches. Par conséquent, l'exploration de l'espace des configurations se fera très lentement avec ces types de chemin. Au contraire, les chemins dans GS_n permettent de connecter même des configurations très différentes. De plus, les connexions par ce type de chemin sont beaucoup plus rapides à tester puisqu'aucun sous-problème de planification n'intervient comme c'est le cas pour le calcul des chemins de ressaisie. Enfin, l'ajout de configurations particulières joue un rôle très important dans la résolution de certains problèmes comme celui du deuxième exemple. Ces configurations ont une probabilité extrêmement faibles d'être générées par un tirage aléatoire et conduisent à des temps de calcul prohibitifs pour des méthodes classiques.

Notre méthode comporte cependant quelques défauts. Le principal est le manque d'optimisation de certaines solutions calculées. L'étape de lissage, qui se fait avant la décomposition pour lisser les chemins dans GS_n et après pour lisser les mouvements des doigts libres pendant les reconfigurations de prise, permet une amélioration limitée de la trajectoire calculée. Dans certains cas, l'objet aura une trajectoire trop longue, ce qui allongera inutilement les opérations de manipulation. Ce problème du manque de possibilité d'optimisation des solutions est en fait inhérent à l'utilisation des méthodes de type PRM. Comme ces méthodes construisent des graphes non cycliques, il n'existera qu'un seul chemin dans le graphe reliant les configurations initiale et finale. Nous avons privilégié la rapidité dans la mise au point de notre méthode mais si on souhaitait favoriser les possibilités d'optimisation des trajectoires, au détriment de la rapidité, il suffirait d'utiliser une méthode de construction de graphe autorisant la création de cycles. Une fois les configurations initiale et finale connectées à une même composante connexe du graphe de manipulation, il serait alors possible, après une phase de lissage, de choisir parmi plusieurs chemins lequel donne la trajectoire la plus courte à l'objet.

Un autre défaut concerne la nécessité de choisir le paramètre α. Cependant, cet inconvénient est en partie surmonté si on le modifie comme décrit dans la partie 4.3. Il serait peut-être également possible de prendre en compte des informations sur la surface de l'objet : si sa courbure présente des variations très importantes, alors sa forme est très irrégulière et de nombreuses connexions seront nécessaires en dehors de GS_n ce qui imposera le choix d'un α petit.

Enfin, il est important de noter que les exemples de simulation correspondent à des situations difficiles puisque la main étudiée ne comporte que le nombre minimum de doigts et que le modèle de contact employé ne favorise pas la stabilité des prises. Avec plus de doigts en contact et des contacts assurant une meilleure stabilité, ce qui rapprocherait des caractéristiques de la manipulation humaine, les solutions seraient trouvées bien plus rapidement.

4.5 Conclusion

Ce chapitre avait pour but de démontrer l'intérêt de la méthode de planification présentée dans le chapitre 3. Pour cela, divers problèmes représentant des situations variées ont été soumis au planificateur. Celui-ci a été capable de les résoudre dans des temps très satisfaisants compte tenu de la complexité de certains d'entre eux. Ce chapitre a également présenté une étude expérimentale de l'influence du seul paramètre à régler dans l'algorithme de planification et s'est terminé par une analyse des résultats obtenus.

Le chapitre suivant propose une extension de la méthode de planification au cas où le roulement des points de contact est pris en compte.

Chapitre 5

Planification pour des Contacts avec Roulement

Dans les chapitres précédents, les effets du roulement des contacts avaient été négligés. Cela permettait de ne pas tenir compte de l'interaction entre les surfaces de l'objet et des bouts des doigts et impliquait que la position des contacts, relativement à l'objet et aux bouts des doigts, restait la même tant qu'il n'y avait pas de rupture de contact. De plus, le choix des points de contact sur les doigts était évité puisqu'on considérait qu'ils ne pouvaient s'établir qu'en un seul point. Cette hypothèse simplifie nettement la description et la résolution du problème de planification mais réduit le domaine d'applicabilité de la formulation et de la méthode de planification des chapitres 2 et 3.

Ce chapitre propose une extension à cette formulation et à cette méthode, qui permet de gérer le roulement des doigts sur l'objet. Pour commencer, les contraintes cinématiques qu'impose le roulement des contacts sont présentées puis le reste du chapitre explique comment modifier la méthode de planification pour qu'elle prenne en compte ces contraintes.

5.1 Le roulement d'un point de contact

5.1.1 Cinématique du roulement

Cette partie est consacrée à la présentation des conditions du roulement sans glissement des contacts entre les doigts et l'objet. La théorie présentée ici provient principalement de [CHS89][1].
Soient un objet et un des doigts d'une main articulée. A l'objet et au doigt sont attachés respectivement un repère $\mathcal{R}_o$ et un repère $\mathcal{R}_d$ (figure 5.1).

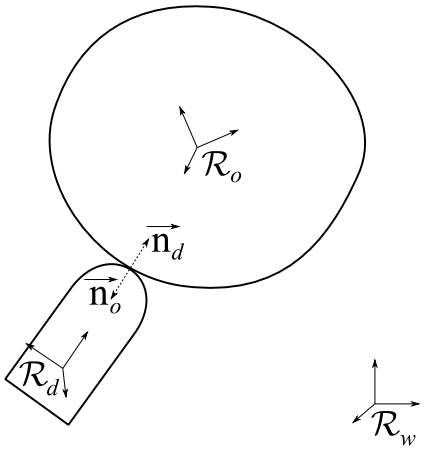


FIGURE 5.1 – Repères des contacts.

La position d'un point de la surface de l'objet, dans le repère $\mathcal{R}_o$, est notée $c_o \in \mathbb{R}^3$ et celle d'un point de la surface du doigt, dans le repère $\mathcal{R}_d$, est notée $c_d \in \mathbb{R}^3$. Soit $\mathcal{R}_w$, un repère galiléen de référence . Le centre de $\mathcal{R}_o$ exprimé dans $\mathcal{R}_w$ est noté $x_o \in \mathbb{R}^3$ et le centre de $\mathcal{R}_d$ exprimé dans $\mathcal{R}_w$ est noté $x_d \in \mathbb{R}^3$. La transformation permettant de passer de $\mathcal{R}_w$ à $\mathcal{R}_o$ est caractérisée par une translation de vecteur x_o et une rotation représentée par une matrice de rotation $R_o \in SO(3)$. De même, la transformation permettant de passer de $\mathcal{R}_w$ à $\mathcal{R}_d$, est composée d'une translation de vecteur x_d et d'une rotation de matrice $R_d \in SO(3)$. Les coordonnées de c_o dans $\mathcal{R}_w$ s'obtiennent alors selon :

$$x_o + R_o c_o \tag{5.1}$$

De même, les coordonnées de c_d dans $\mathcal{R}_w$ s'obtiennent selon :

$$x_d + R_d c_d \tag{5.2}$$

[1] A. Cole, J. Hauser, and S. Sastry. Kinematics and control of multifingered hands with rolling contact. *IEEE Transactions on Automatic Control*, 34(4) :398–404, avril 1989.

Pour qu'il y ait contact entre le doigt et l'objet, il faut que les positions des points sur les surfaces du doigt et de l'objet soient identiques. On doit donc avoir :

$$x_o + R_o c_o = x_d + R_d c_d \tag{5.3}$$

Une autre condition sur le contact est que les plans tangents aux surfaces de contact respectives de l'objet et du doigt coïncident, tout en ayant des normales opposées. Cette condition s'écrit :

$$R_o n_o = -R_d n_d \tag{5.4}$$

où n_o et n_d sont les normales aux points de contact sur les surfaces respectives de l'objet et du doigt (figure 5.1).

Il existe également des contraintes sur les vitesses. La première est due à la condition de non pénétration de deux corps maintenant un contact ponctuel. Elle impose que la vitesse relative instantanée des deux corps au point de contact soit nulle dans la direction commune à leurs normales respectives. Une seconde contrainte sur les vitesses est due à l'absence de glissement entre les deux solides au niveau du contact. Cela se traduit par le fait que la composante tangentielle de la vitesse relative instantanée du point de contact sur chacun des deux corps doit être nulle. De ces deux dernières contraintes, on déduit qu'il doit y avoir égalité de la vitesse relative instantanée des deux corps au point de contact. Exprimons la vitesse instantanée d'un point donné de la surface de l'objet, dans le repère $\mathcal{R}_w$, en dérivant sa position :

$$\dot{x}_o + \dot{R}_o c_o = v_o + \omega_o \times R_o c_o \tag{5.5}$$

où $v_o = \dot{x}_o$ est la vitesse de translation de l'objet et $\omega_o \in \mathbb{R}^3$ son vecteur vitesse de rotation instantanée. On écrit de même la vitesse instantanée d'un point de la surface du doigt, dans le repère $\mathcal{R}_w$:

$$\dot{x}_d + \dot{R}_d c_d = v_d + \omega_d \times R_d c_d \tag{5.6}$$

où $v_d = \dot{x}_d$ est la vitesse de translation de l'objet et $\omega_d \in \mathbb{R}^3$ son vecteur vitesse de rotation instantanée. La condition sur les vitesses au point de contact est alors :

$$v_o + \omega_o \times R_o c_o = v_d + \omega_d \times R_d c_d \tag{5.7}$$

On peut résumer, à partir des équations 5.3, 5.4 et 5.7, les différentes contraintes que doit vérifier un point de contact :

$$\begin{cases} x_o + R_o c_o = x_d + R_d c_d \\ R_o n_o = -R_d n_d \\ v_o + \omega_o \times R_o c_o = v_d + \omega_d \times R_d c_d \end{cases} \tag{5.8}$$

Dès lors qu'un point est un point de contact, il vérifie les deux premières équations de 5.8 et la seule contrainte à respecter est alors la contrainte de vitesse qui est de

dimension 3. Pour calculer l'évolution des points de contact au cours du temps, ces différentes contraintes devront être intégrées. Elles ne peuvent pas être intégrées telles quelles puisqu'elles sont seulement valides localement, pour un point donné de la surface de chacun des corps ; il faut tenir compte de la forme des surfaces des doigts et de l'objet. Les auteurs de [CHS89][1] proposent alors d'utiliser une paramétrisation de ces surfaces (qui doivent donc être paramétrées).

Soient $\xi_o \in \mathbb{R}^2$ et $\xi_d \in \mathbb{R}^2$ des paramétrisations de la surface de l'objet et d'un doigt, respectivement. On a $c_o = c_o(\xi_o)$ et $c_d = c_d(\xi_d)$. Si on dérive l'équation 5.3, on obtient alors :

$$\dot{x}_o(t) + \omega_o \times R_o c_o(t) + R_o \dot{c}_o(t) = \dot{x}_d(t) + \omega_d \times R_d c_d(t) + R_d \dot{c}_d(t) \tag{5.9}$$

soit, en omettant la variable t pour simplifier l'écriture :

$$v_o + \omega_o \times R_o c_o + R_o \dot{c}_o = v_d + \omega_d \times R_d c_d + R_d \dot{c}_d \tag{5.10}$$

En soustrayant l'équation 5.7 dans l'équation 5.10, on a :

$$R_o \dot{c}_o = R_d \dot{c}_d \tag{5.11}$$

On peut alors se servir de la paramétrisation de c_o et c_d, pour arriver à :

$$R_o \frac{\partial c_o}{\partial \xi_o} \dot{\xi}_o = R_d \frac{\partial c_d}{\partial \xi_d} \dot{\xi}_d \tag{5.12}$$

En dérivant 5.4, on a :

$$\omega_o \times R_o n_o + R_o \frac{\partial n_o}{\partial \xi_o} \dot{\xi}_o + \omega_d \times R_d n_d + R_d \frac{\partial n_d}{\partial \xi_d} \dot{\xi}_d = 0 \tag{5.13}$$

En regroupant les équations 5.12 et 5.13, on obtient :

$$\begin{bmatrix} R_o \frac{\partial c_o}{\partial \xi_o} & -R_d \frac{\partial c_d}{\partial \xi_d} \\ R_o \frac{\partial n_o}{\partial \xi_o} & R_d \frac{\partial n_d}{\partial \xi_d} \end{bmatrix} \begin{bmatrix} \dot{\xi}_o \\ \dot{\xi}_d \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ (R_o n_o \times) & (R_d n_d \times) \end{bmatrix} \begin{bmatrix} \omega_o \\ \omega_d \end{bmatrix} \tag{5.14}$$

L'égalité 5.14 permet de calculer $\dot{\xi}_o$ et $\dot{\xi}_d$ connaissant ω_o et ω_d. Jusqu'ici, aucune contrainte cinématique sur le mouvement des deux corps n'a été considérée, excepté au point de contact. Si l'objet n'a pas de contraintes cinématiques qui lui sont propres ce n'est pas le cas des doigts, qui font partie de chaînes cinématiques contraignant leurs déplacements.

[1] A. Cole, J. Hauser, and S. Sastry. Kinematics and control of multifingered hands with rolling contact. *IEEE Transactions on Automatic Control*, 34(4) :398–404, avril 1989.

5.1.2 Contrainte sur le mouvement des doigts

Pour comprendre que les contraintes précédentes n'imposent pas entièrement le mouvement des doigts, il est plus facile de raisonner sur un exemple simple. Intéressons-nous alors au roulement entre deux disques dans le plan. Le doigt est alors considéré comme un corps planaire pouvant se déplacer librement. Les positions des points de contact sur la surface de l'objet et sur la surface du doigt peuvent être paramétrées par des angles ξ_o et ξ_d, respectivement. On a alors :

$$c_o = r_o \begin{bmatrix} cos(\xi_o) \\ sin(\xi_o) \end{bmatrix} \tag{5.15}$$

$$c_d = r_d \begin{bmatrix} cos(\xi_d) \\ sin(\xi_d) \end{bmatrix} \tag{5.16}$$

où r_o et r_d sont les rayons de l'objet et du doigt respectivement. L'équation 5.12 devient :

$$R_o r_o \begin{bmatrix} -sin(\xi_o) \\ cos(\xi_o) \end{bmatrix} \dot{\xi}_o = R_d r_d \begin{bmatrix} -sin(\xi_d) \\ cos(\xi_d) \end{bmatrix} \dot{\xi}_d \tag{5.17}$$

Par ailleurs, on a la relation suivante entre les vecteurs tangents aux deux surfaces, qui sont colinéaires mais de sens opposés :

$$R_o \begin{bmatrix} -sin(\xi_o) \\ cos(\xi_o) \end{bmatrix} = -R_d \begin{bmatrix} -sin(\xi_d) \\ cos(\xi_d) \end{bmatrix} \tag{5.18}$$

Les deux vecteurs sont opposés car les normales aux surfaces le sont aussi.
L'équation 5.17 devient :

$$r_o \dot{\xi}_o = -r_d \dot{\xi}_d \tag{5.19}$$

L'équation 5.4 de l'égalité, au signe près, des normales des surfaces au point de contact, ne peut plus s'écrire avec des matrices de rotation mais plus simplement, en notant θ_o et θ_d les angles représentant les orientations de l'objet et du doigt, respectivement :

$$\theta_o + \xi_o = \theta_d + \xi_d + (2k+1)\pi, \quad k \in \mathbb{Z} \tag{5.20}$$

En dérivant cette équation on obtient :

$$\omega_o + \dot{\xi}_o = \omega_d + \dot{\xi}_d \tag{5.21}$$

avec $\omega_o = \dot{\theta}_o$ et $\omega_d = \dot{\theta}_d$. On arrive finalement, en tenant compte de 5.19, à :

$$\begin{cases} r_o \dot{\xi}_o = -r_d \dot{\xi}_d \\ (1 + \frac{r_o}{r_d}) \dot{\xi}_o = \omega_d - \omega_o \end{cases} \tag{5.22}$$

Les équations 5.22 ne sont pas suffisantes pour déterminer l'évolution des points de contact si on connaît seulement la vitesse de rotation de l'objet. Les doigts doivent

cependant obéir à une contrainte cinématique supplémentaire. Cette contrainte est due aux chaînes cinématiques formées par les doigts. Le point de contact doit être atteignable par le doigt, c'est-à-dire qu'il doit être une solution du modèle géométrique direct :

$$x_d + R_d c_d = MGD(q) \tag{5.23}$$

où $q \in \mathbb{R}^m$ est le vecteur des paramètres articulaires du doigt, qui comporte m articulations. En dérivant cette équation, on obtient une contrainte sur la vitesse du point de contact, qui doit être compatible avec le modèle cinématique du doigt :

$$v_d + \omega_d \times R_d c_d = J(q)\dot{q} \tag{5.24}$$

où J est la matrice jacobienne du doigt.
Ces contraintes peuvent être complexes puisqu'elles sont susceptibles de faire apparaître des produits de fonctions trigonométriques. C'est pourquoi l'intégration analytique de 5.8 associé à 5.23 et 5.24 n'est généralement pas possible. En revanche, elle peut bien sûr être réalisée numériquement. La méthode utilisée est décrite dans la partie 5.2.2 et n'intervient pas dans l'algorithme de planification lui-même. Par contre, maintenant que les contacts se font entre deux surfaces et non plus entre une surface et un point, la formulation du problème doit être modifiée. C'est le sujet de la partie suivante.

5.2 Prise en compte du roulement des contacts

5.2.1 Nouvelle définition des vecteurs de configuration

Dans la partie 2.1, nous avions supposé que les contacts ne pouvaient se faire qu'en un seul point de la surface des doigts et que pour une configuration donnée de l'objet, un contact doigt-objet était déterminé entièrement par la connaissance de sa position sur la surface de l'objet. Ces hypothèses ne sont plus valides dans le cas où le contact se fait entre deux surfaces, ce qui conduit à reformuler légèrement le problème. Comme dans la partie 2.1, le système est toujours composé d'une main mécanique à n doigts et d'un objet rigide. Le vecteur de configuration de la main est toujours $(a_1, a_2, \ldots, a_n)$ où $a_i = (\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,m_i}) \in \mathbb{R}^{m_i}$ et $\theta_{i,k}$ est le k-ième paramètre articulaire du doigt i, qui comporte m_i liaisons. Le vecteur de configuration de l'objet est $\mathbf{p} = (x, y, z, \alpha, \beta, \gamma)$ où (x, y, z) est sa position et (α, β, γ) une paramétrisation de son orientation. Le vecteur utilisé jusqu'à présent pour représenter la prise de l'objet contenait seulement les coordonnées des points de contact sur la surface de l'objet. Ce vecteur n'est plus suffisant pour décrire la prise maintenant que la position des contacts sur le bout des doigts n'est plus fixe. Comme la prise est toujours définie pour une configuration donnée de l'objet, les contacts vus de l'objet sont toujours uniquement caractérisés par leur position sur sa surface. Il faut de même caractériser

la position du contact sur le doigt. Nous supposons que pour chaque doigt la surface où les contacts sont susceptibles de se faire est paramétrée. Cette hypothèse est réaliste car la forme des doigts de la main est choisie par ses concepteurs et connue au départ, au contraire de celle de l'objet. Si le doigt i est en contact avec l'objet, la position de ce contact sur la surface du doigt i est alors $(u_{d_i}, v_{d_i}) \in \mathbb{R}^2$. Une fois connue la position du point sur la surface d'un doigt, il reste à déterminer son orientation, soit trois paramètres de rotation. Deux d'entre eux sont imposés d'emblée car les directions des normales des surfaces du doigt et de l'objet au point de contact doivent être égales (équation 5.4). Il reste alors à fixer l'angle autour de la normale. Appelons cet angle ψ_i, pour le doigt i. Le contact est alors entièrement caractérisé par la connaissance de $(u, v) \in \mathbb{R}^2$ et $(u_{d_i}, v_{d_i}, \psi_i) \in \mathbb{R}^2 \times \mathbb{S}$. (figure 5.2). Il est à noter que dans le cas très fréquent d'un doigt à trois degrés de liberté, les paramètres u_{d_i}, v_{d_i} et ψ_i sont imposés par la configuration de l'objet et la position du point de contact sur sa surface.

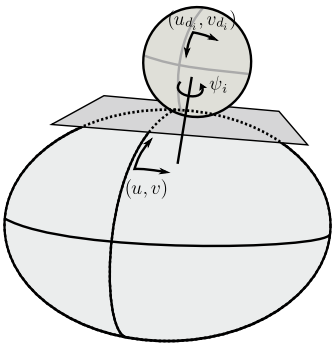


FIGURE 5.2 – Paramètres caractérisant les position relatives du doigt et de l'objet, au point de contact.

Pour le reste, la méthode de planification décrite dans le chapitre 3 est inchangée sauf en ce qui concerne le calcul des chemins de transfert, qui est traité dans la partie suivante. La construction du graphe dans GS_n avec des chemins dans GS_n et le principe de la fusion des composantes connexes du graphe à l'aide de chemins de transfert-ressaisie restent les mêmes.

5.2.2 La nouvelle méthode locale pour les chemins de transfert

La différence fondamentale qui intervient dans le calcul des chemins de transfert, quand on prend en compte le roulement des contacts, est que la prise n'est plus constante le long d'un chemin de transfert, comme c'était le cas précédemment. Cela amène trois interrogations :

1. la propriété de réduction est-elle toujours valide ?

2. le vérification de la fermeture de force pour un chemin dans GS_n assure-t-elle la fermeture de force après la décomposition en séquence transfert-ressaisie ?
3. comment calculer les positions des points de contact au cours d'un chemin de transfert ?

5.2.2.1 Validité de la propriété de réduction

La propriété de réduction est toujours valide car l'hypothèse nécessaire au respect de cette propriété est elle-même toujours vérifiée (le long d'un chemin dans GS_n, le robot n'est en collision avec aucun obstacle, ni en collision interne). La décomposition pourra donc toujours se faire même si c'est en un très grand nombre de chemins de transfert et de ressaisie.

5.2.2.2 Test de fermeture de force pour un chemin dans GS_n

La réponse à la deuxième question n'est pas toujours positive. Elle ne l'est que si on considère qu'il existe pour chaque contact un voisinage sur la surface de l'objet, centré sur ce point de contact et tel qu'un déplacement du point de contact dans ce voisinage n'entraîne pas d'invalidation de la fermeture de force. Cela est illustré sur la figure 5.3. Avant la décomposition d'un chemin dans GS_n, le contact suit un chemin AB sur la surface de l'objet. Après décomposition du chemin en chemins de transfert-ressaisie, le chemin que suivra le point de contact sera différent et dépendra du roulement relatif entre la surface de l'objet et celle du doigt. La prise sera alors différente des prises obtenues en parcourant le chemin AB et leurs équivalents pour chaque contact, pour lesquelles on avait testé la fermeture de force. S'il existe un voisinage, autour du chemin AB, tels que les points de contact lui appartenant garantissent toujours la fermeture de force, alors la décomposition pourra se faire sans supprimer cette propriété. Concrètement, cela revient à supposer que toutes les prises correspondant aux configurations du chemin dans GS_n respectent la fermeture de force même si elle sont légèrement modifiées, et peut être comparé à l'introduction d'une incertitude sur la position des contacts pour le test de fermeture de force comme dans [ZQ05][1].

5.2.2.3 Calcul de l'évolution des points de contact

Pour calculer l'évolution des points de contact, comme dans le cas de la méthode locale de la partie 2.2.2, on part d'un mouvement donné de l'objet pour calculer les mouvements correspondants des doigts. Le mouvement de l'objet impose les vitesses instantanées des différents points de contact. Les mouvements des doigts doivent alors

[1] Y. Zheng and W.-H. Qian. Coping with the grasping uncertainties in force-closure analysis. *The International Journal of Robotics Research*, 24(4) :311–327, avril 2005.

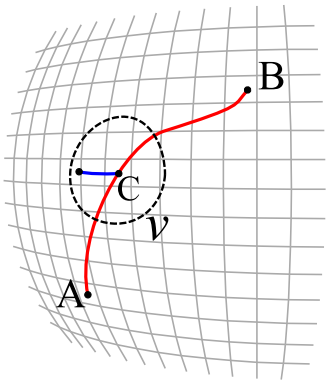


FIGURE 5.3 – Le long d'un chemin dans GS_n, le contact se déplace de A à B (chemin rouge). Si on veut décomposer le chemin en séquence transfert-ressaisie, le contact roulera, par exemple, à partir du point C sur la surface de l'objet (chemin bleu), en quittant le chemin AB.

respecter les contraintes sur les vitesses aux points de contact du système d'équation 5.8. Ces contraintes doivent être intégrées pour calculer l'évolution des points de contact. Elles ne peuvent pas être intégrées telles quelles puisqu'elle sont valides localement, pour un point donné de la surface de chacun des corps ; il faut tenir compte de la forme des surfaces des doigts et de l'objet. Comme il est préférable de ne pas émettre d'hypothèse sur la forme de l'objet, on souhaite se passer d'une paramétrisation de sa surface. Nous décrivons maintenant comment nous avons choisi d'intégrer de façon approchée les différentes contraintes intervenant au niveau du contact.

On cherche à relier, par un chemin de transfert, deux configurations de l'objet $\mathbf{p_i} = (x_{o,i}, R_{o,i})$ et $\mathbf{p_f} = (x_{o,f}, R_{o,f})$. Nous introduisons une variable temps t, qui n'est pas considérée dans son sens physique puisque le système est quasi-statique. La représentation du temps doit plutôt être vue comme implicite (elle n'intervient que dans le fait que les configurations se suivent dans un ordre précis). On considère ainsi que le chemin de transfert entre $\mathbf{p_i}$ et $\mathbf{p_f}$ est parcouru en un temps Δt. Le chemin de transfert est parcouru à vitesse constante par l'objet : $v = \frac{x_{o,f} - x_{o,i}}{\Delta t}$ est sa vitesse linéaire et V_R la matrice telle que $R_{o,f} = R_{o,i} + V_R \Delta t$. $\mathbf{p_o}(t) = (x_o(t), R_o(t))$ est la configuration de l'objet à l'instant t et $c_o(t)$ la position du contact dans le repère de l'objet, permettant de calculer sa position $c(t)$ dans un repère global fixe, à partir de la pose de l'objet selon $c(t) = x_o(t) + R_o(t)c_o(t)$. La position du contact dans le repère du corps du doigt sur lequel s'effectue le contact est $c_d(t)$. L'algorithme 7 calcule les points de contact à l'instant $t + dt$ (le vecteur c' est une variable de calcul).

$\mathbf{q}$ est la configuration du doigt qui amène le point $c_d(t)$ de sa surface au point c dans le repère global. La fonction $calcul\_nouveaux\_contacts()$ calcule, à partir des configurations de l'objet et du doigt, une correction de ces configurations qui assure

Algorithme 7 : Algorithme de calcul de l'évolution des points de contact le long d'un chemin de transfert

1 $\mathbf{p_o}(t+dt) = (x_o(t+dt), R_o(t+dt)) = (x_o(t) + vdt, R_o(t) + V_R dt)$
2 $c' = x_o(t+dt) + R_o(t+dt)c_o(t)$
3 $\mathbf{q} = modele\_geometrique\_inverse(c', c_d(t))$
4 $(c_o(t+dt), c_d(t+dt)) = calcul\_nouveaux\_contacts(\mathbf{p_o}, \mathbf{q})$

un contact ponctuel entre les surfaces de ces deux corps. Le principe de la méthode est illustré par la figure 5.4, avec un pas exagérément agrandi pour mieux visualiser.

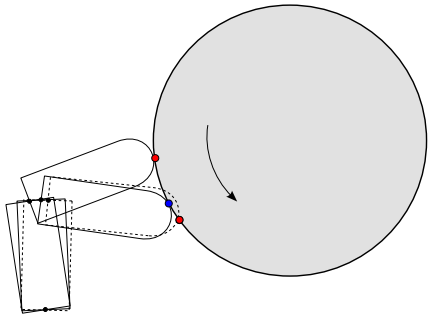


FIGURE 5.4 – On calcule une nouvelle position du doigt correspondant à une vitesse relative nulle des points de contact sur la surface des deux corps (configuration en pointillé). La position du doigt est ensuite corrigée pour empêcher l'interpénétration des deux surfaces ou maintenir le contact. Le nouveau contact est en bleu.

La position "corrigée" du doigt est calculée de la façon suivante selon que le doigt et l'objet sont en collision ou ont rompu le contact :

- si les deux corps ne sont plus en contact, les nouveaux points de contact c_o et c_d sur les surfaces respectives de l'objet et du doigt, sont choisis tels que la distance $c_o c_d$ soit minimale. Ce calcul peut se faire en approximant les surfaces par des polyèdres. Un détecteur de collision (comme celui de [GLM96][1]) fournit alors la solution.
- si les deux corps s'interpénètrent, on calcule la position moyenne des points de contacts (toujours en utilisant un détecteur de collision et des modèles polyédriques) puis on cherche le point qui en est le plus proche sur la surface de l'objet et celui qui en est le plus proche sur la surface du doigt. Ce sera la nouvelle paire de contacts.

Cela est illustré sur la figure 5.5. Cette méthode n'est qu'une approximation mais plus

[1] S. Gottschalk, M.C. Lin, and D. Manocha. OBBTree : A hierarchical structure for rapid interference detection. *Proceedings of ACM Siggraph'96*, 1996.

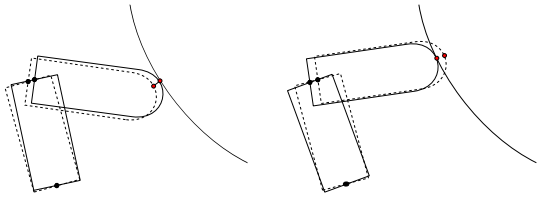


FIGURE 5.5 – Principe de la "correction" de la configuration du doigt dans le cas où il y a rupture du contact (a) et interpénétration des volumes (b).

le pas de calcul est petit plus sa précision est grande. Elle permet surtout de se passer d'une paramétrisation sous forme analytique de la surface de l'objet.

A cause de la dérive sur la position des points de contact, due aux approximations de calcul du roulement, les trajectoires suivies réellement par les doigts et l'objet lors des chemins de transfert seront légèrement différentes des trajectoires planifiées. Il faudra donc effectuer un recalage régulier de la configuration du système réel sur les configurations calculées pendant la planification. Ce recalage sera effectué, en pratique, par le système d'asservissement nécessaire au suivi des trajectoires.

5.3 Résultats de simulation

Nous avons testé la méthode de planification étendue sur les exemples du chapitre 4. Les performances du planificateur ne sont pas changées. Seuls changent le temps de décomposition des chemins et bien sûr l'allure des solutions trouvées. Les figures 5.6 et 5.7 montrent des étapes de trajectoires solutions obtenues pour les tâches de réorientation d'une sphère et d'une boîte, pour des contacts avec roulement.

Le tableau 5.1 regroupe des résultats sur la résolution de ces deux problèmes par le planificateur. Ces résultats montrent que le modèle de contact n'affecte pas notre

exemple	exemple 1			exemple 2		
	min	moyenne	max	min	moyenne	max
temps de résolution (s)	0.6	3.4	9.2	2.75	57.8	308.3
nombre de noeuds générés	4	39	113	33	559	2261

TABLE 5.1 – Résultats obtenus pour les deux exemples (sur 200 tests).

approche et que le planificateur est capable de résoudre des problèmes même en tenant compte du roulement des contacts.

Cependant, comme la méthode ne cherche pas à optimiser les mouvements des doigts pour tirer partie du roulement le mieux possible, les doigts ne se replacent pas

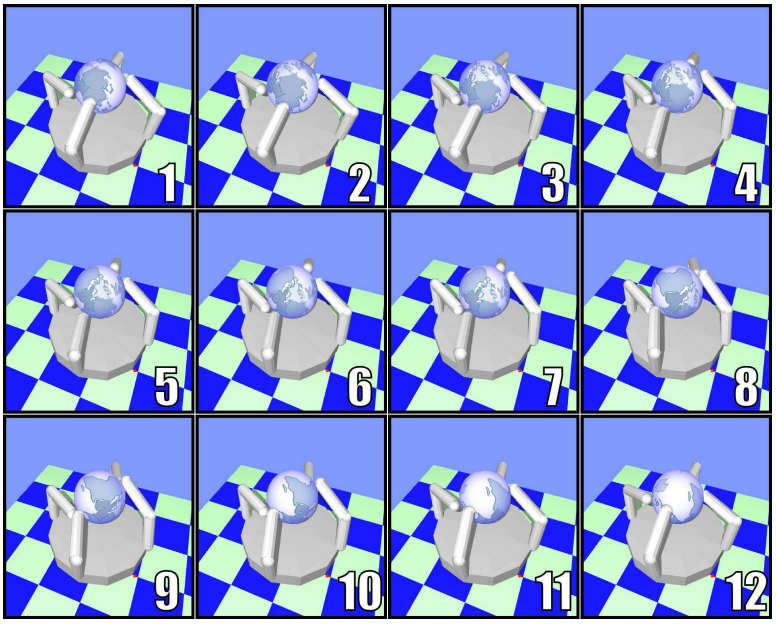


FIGURE 5.6 – Trajectoire solution du problème de réorientation d'une sphère avec prise en compte du roulement des contacts.

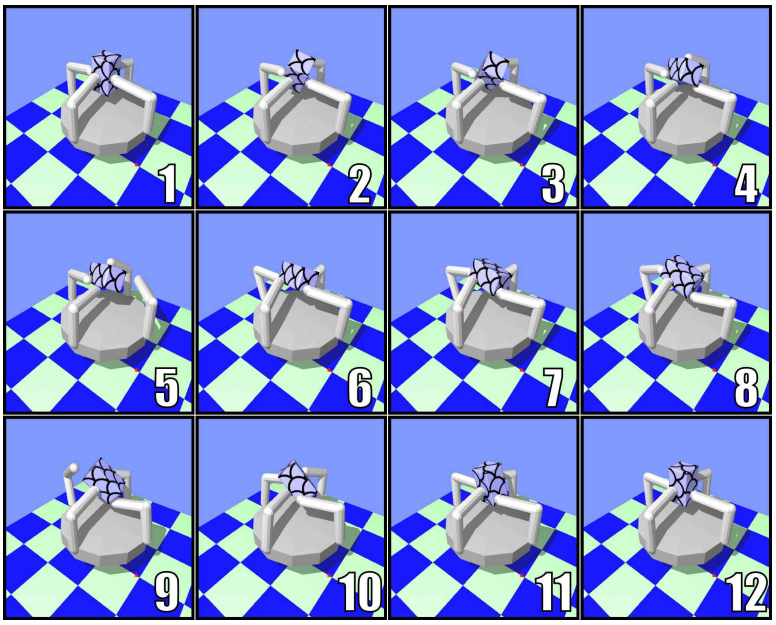


FIGURE 5.7 – Trajectoire solution du problème de réorientation d'une boite avec prise en compte du roulement des contacts.

de manière à utiliser le roulement pour déplacer l'objet le plus possible sans reconfiguration de prise. Pour minimiser le nombre de reconfigurations de prise, l'idéal est de faire en sorte que chaque point de contact se déplace le plus possible sur le bout hémisphérique du doigt avant qu'il n'atteigne un des bords de l'hémisphère. Chaque repositionnement de doigt doit donc se faire de sorte que la nouvelle position du contact soit le plus loin possible dans la direction opposée à celle du mouvement dû au roulement. Ce principe n'étant pas pris en compte par la méthode de planification, on constate que les doigts se replacent beaucoup plus que nécessaire, ce qui rend, dans certaines situations, la tâche de manipulation exagérément longue. Ce manque d'optimisation concerne l'étape de décomposition des chemins dans GS_n. Une solution à ce défaut pourrait être de développer la technique de décomposition des chemins pour qu'elle minimise le nombre de reconfigurations de prise en prenant en compte le mieux possible le déplacement des points de contact, qui intervient le long des chemins de transfert.

5.4 Conclusion

Les différentes contraintes cinématiques de roulement des points de contact doigt-objet ont pu être intégrées dans la formulation et la méthode de planification des chapitres 2 et 3. Pour cela, la description des points de contact est modifiée pour prendre en compte le fait que les contacts peuvent s'établir n'importe où sur les surfaces des bouts des doigts et de l'objet. Le calcul des chemins de transfert a lui aussi été modifié pour considérer l'évolution de la position des contacts due au mouvement relatif de roulement des surfaces de contact. Des résultats ont également été présentés pour deux exemples de tâches de manipulation.

Conclusion Générale

Cette thèse s'est intéressée à la planification des tâches de manipulation effectuées par une main robotisée. Il s'agissait de mettre au point un système de calcul automatique des trajectoires que doivent suivre les doigts et l'objet manipulé, pour passer d'une configuration initiale à une configuration finale données.

La méthode que nous avons proposée s'appuie sur une formulation originale du problème de planification, basée sur l'étude de la connexité des espaces des configurations de prise. Ces espaces sont explorés par l'intermédiaire de graphes probabilistes. En particulier, un graphe est construit pour explorer GS_n, l'espace des configurations de prise à n doigts, n étant le nombre de doigts de la main. Les arêtes de ce graphe sont des chemins linéaires dans GS_n. Utiliser de tels chemins permet d'éviter le calcul des mouvements de reconfiguration de prise et donc de réduire les temps de calcul et l'espace mémoire requis par la construction du graphe. Ces chemins ne sont pas cinématiquement réalisables puisque la pose de l'objet et la position des contacts ne peuvent changer indépendamment mais leur utilisation est rendue possible par la généralisation de la propriété de réduction introduite par Alami *et al.* [ALS94]. Les mouvements de changement de prise qui requièrent d'être explicitement calculés au cours de la construction du graphe, sont pris en compte lors d'une étape de fusion des composantes connexes du graphe. Ces fusions sont réalisées à l'aide de chemins élémentaires respectant la cinématique de la manipulation coordonnée. Ces chemins sont appelés *chemins de ressaisie* et *chemins de transfert*. Une fois que les configurations initiale et finale appartiennent à une même composante connexe du graphe, les chemins dans GS_n sont décomposés en une suite de mouvements de déplacement de l'objet et de reconfiguration de la prise (chemins de transfert et de ressaisie), cinématiquement réalisables. Pour assurer la stabilité des chemins construits, un critère de stabilité de la prise (fermeture de force) est vérifié le long des chemins, lors de leur construction.

Pour valider cette approche, une plate-forme de simulation a été développée et a permis de planifier différentes tâches de manipulation dextre avec une main à quatre doigts. Le planificateur offre des performances très intéressantes en terme de temps de

calcul et a permis de résoudre des problèmes complexes tels qu'aucun résultat pour des problèmes de difficulté équivalente n'avait jamais été présenté jusqu'à présent.

La méthode proposée s'applique à n'importe quel type de main, quel que soit son nombre de doigts mais, comme elle explore uniquement GS_n et GS_{n-1}, elle peut manquer des solutions si la main robotisée et le modèle des contacts doigt-objet permet la prise avec un nombre différent de doigts. Pour remédier à cela, nous avons proposé une méthode légèrement différente qui s'applique à une main à cinq doigts et consiste à construire un graphe pour explorer chacune des cinq composantes connexes de GS_4 à l'aide de chemins linéaires dans cet espace et à tenter de fusionner les différents graphes à l'aide de chemins linéaires dans GS_5 ou de chemins de transfert-ressaisie (dans GS_3).

Enfin, une variante de la méthode proposée a été développée pour prendre en compte le roulement relatif des surfaces de contact au cours de la manipulation de l'objet. Les différentes modifications nécessaires, concernant la représentation des prises et le calcul de chemins de transfert, ont été présentées en détail.

Perspectives

Le travail présenté possède quelques limitations dont le traitement pourrait faire l'objet de développements futurs. Une de ces limitations concerne la modélisation des contacts entre les doigts et l'objet. Nous avons supposé que les surfaces de contacts sont indéformables, comme cela est généralement admis dans la littérature. Cependant, cette restriction sur les contacts réduit fortement l'ensemble des prises possibles de l'objet. D'abord parce que les contacts entre les doigts et les arêtes et autres parties pointues de l'objet sont impossibles car ils seraient beaucoup trop instables. Ensuite parce que l'utilisation de contacts ponctuels rend les prises beaucoup moins stables que si elles étaient effectuées à l'aide de contacts déformables. La réduction de l'ensemble des prises rend la manipulation plus difficile et ralentit la résolution d'un problème de planification. De même, limiter la zone de contact des doigts à leurs extrémités peut poser des difficultés, particulièrement si les domaines de travail des doigts de la main utilisée ne sont pas répartis uniformément dans l'espace. Il serait intéressant que les contacts puissent se faire n'importe où sur la main, à l'image de la manipulation pratiquée par l'homme, qui fait parfois participer la paume de la main ou l'intérieur des doigts aux prises. Enfin, on peut également mentionner la gestion du glissement parmi les améliorations possibles liées aux contacts. Utiliser le glissement des points de contact permettrait d'enrichir la palette de mouvements de manipulation du robot.

Une autre limitation porte sur la validation expérimentale de la méthode de planification. Toutes les expériences présentées dans ce mémoire ont été réalisées en simulation. Passer à des expérimentations réelles permettrait une meilleure validation et donnerait peut-être des idées d'amélioration de la méthode. Cela permettrait également d'associer la planification aux autres travaux de recherche concernant la manipulation

dextre, pour tester leurs performances respectives en situation réelle. La planification s'insérerait ainsi dans une chaîne complexe comprenant la reconnaissance de l'objet, le mouvement d'approche de la main, la stratégie de prise de l'objet permettant de choisir les configurations initiale et finale désirées et, une fois la planification effectuée, le calcul et l'optimisation des consignes d'effort puis l'exécution des mouvements, contrôlée par un système d'asservissement.

Indépendamment des développements liés aux limitations de la méthode de planification proposée, il est possible d'envisager d'autres applications de ce travail de thèse. On pourrait, par exemple, imaginer intégrer la méthode de planification à un système plus complexe autorisant la manipulation à deux mains. Cela apporterait de nouvelles capacités de manipulation, adaptées à des tâches telles que l'assemblage ou le travail avec deux outils. Une application plus directe consisterait à adapter le planificateur au problème de la locomotion d'un robot à pattes. La marche est en effet très proche de la manipulation dextre. De la même façon que la méthode proposée permet de réduire les calculs des mouvements de reconfiguration de prise dans le cas de la manipulation, les calculs des levers de patte pourraient se limiter à l'enjambement des obstacles et au calcul de la trajectoire solution.

Annexe : Techniques d'échantillonnage

Échantillonnage déterministe

De nombreuses techniques existent pour échantillonner l'espace des configurations de façon déterministe. La plus simple est certainement la construction d'une grille ou d'un treillis, ce dernier pouvant être vu comme une grille dont les axes ne sont pas orthogonaux. Si l'espace à échantillonner est $[0,1]^d \subset \mathbb{R}^d$, les points de la grille de résolution l, composée de 2^{ld} points, sont définis par

$$P_l^d = \{(\frac{i_1}{2^l}, \ldots, \frac{i_d}{2^l}), i \in \mathbb{Z}, 0 \leq i \leq 2^l - 1\} \tag{25}$$

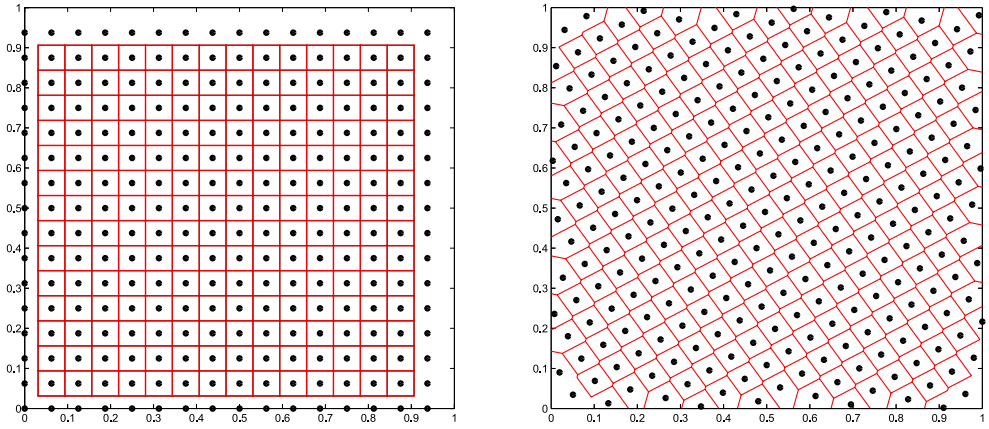
Pour un treillis de N points, on a :

$$P_N^d = \{\frac{i}{N}, i_{\alpha_1}, \ldots, i_{\alpha_{d-1})}, i \in \mathbb{Z}, 0 \leq i \leq N - 1\} \tag{26}$$

où les $(\alpha_k), k = 1, \ldots, d-1$ sont $(d-1)$ nombres positifs irrationnels distincts. La figure 8 montrent à quoi ressemblent ces échantillonnages dans le cas où $d = 2$, ainsi que leurs diagrammes de Voronoï [1]. On constate immédiatement leurs qualités en terme de structure et d'uniformité. Par contre, leur qualité incrémentale est très mauvaise. Tant qu'on n'a pas généré la suite complète, les points ne seront pas répartis uniformément.

Des suites d'échantillonnage ont été mises au point pour bénéficier d'une meilleure qualité incrémentale. Les points de ces suites sont générés sans qu'il y ait besoin de fixer le nombre total de points de la suite au préalable. Ces suites sont parfois appelées quasi-aléatoires car, bien qu'elles soient déterministes, la répartition de leurs points

1. Le diagramme de Voronoï d'un ensemble de points $P = (p_1, p_2, \ldots, p_n)$ d'un espace X est l'ensemble des cellules de Voronoï de ces points. La cellule de Voronoï d'un point p_i est l'ensemble des points de X plus proches de p_i que de n'importe quel autre point de P.



(a) (b)

FIGURE 8 – Échantillonnage de $[0,1]^2$ sur 256 points selon une grille (a) et un treillis (b). En rouge, les diagrammes de Voronoï de chacun des ensembles de points.

peut donner l'impression d'être aléatoire. La plus connue est certainement la suite de Halton [Hal60][1] qui est construite à partir de d entiers $p_1, p_2, \ldots, p_d$ choisis tels qu'aucun d'entre eux ne soit un multiple d'un des autres. On choisit habituellement les d premiers nombres premiers. Pour calculer le i-ème élément de la suite, on écrit i dans chacune des bases des $p_j, j = 1, \ldots, d$. On aura ainsi :

$$i = a_0 + p_j a_1 + p_j^2 a_2 + \ldots + p_j^d a_{d-1} \tag{27}$$

On définit :

$$r_{p_j} = \frac{a_0}{p_j} + \frac{a_1}{p_j^2} + \ldots + \frac{a_{d-1}}{p_j^d} \tag{28}$$

Le i-ème échantillon est alors :

$$(r_{p_1}, r_{p_2}, \ldots, r_{p_d}) \tag{29}$$

Une modification de cette méthode est la suite de Hammersley [Ham60][2] qui requiert la connaissance du nombre de points selon le premier axe. L'échantillon i est alors

$$(\frac{i}{N}, r_{p_1}, r_{p_2}, \ldots, r_{p_{d-1}}) \tag{30}$$

La figure 9 montrent les 256 premiers échantillons, toujours pour $d = 2$. Les points se répartissent de façon moins uniforme que précédemment et ne présentent pas de

[1] J.H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating mulitdimensional integrals. *Numerische Mathematik*, 2 :84–90, 1960.

[2] J.M. Hammersley. Monte-Carlo methods for solving multivariable problems. *Annals of the New York Academy of Science*, 86 :844–874, 1960.

structure exploitable pour la détermination rapide des points voisins de chaque point. En revanche, ces suites présentent une bonne qualité incrémentale.

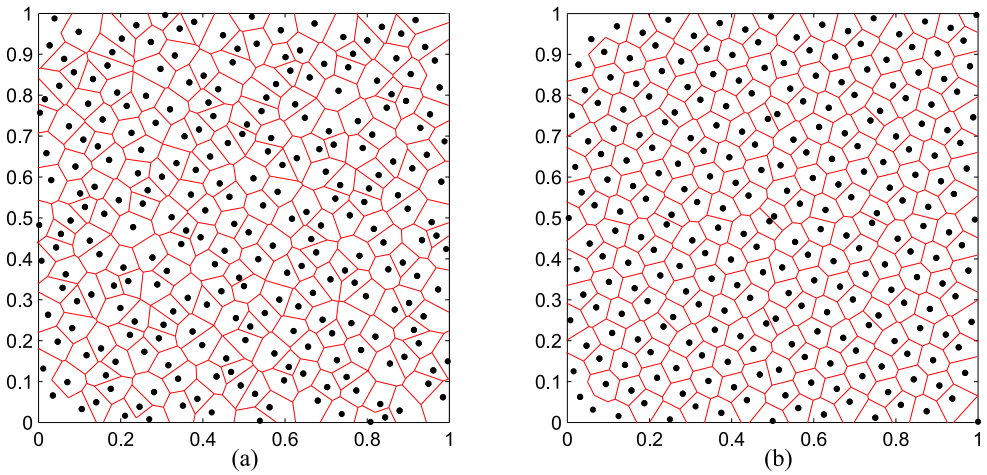


FIGURE 9 – Échantillonnage de $[0, 1]^2$ sur 256 points selon les suites de Halton (a) et de Hammersley (b). En rouge, les diagrammes de Voronoï de chacun des ensembles de points.

Plus récemment, les auteurs de [LL03][3] ont proposé une méthode de construction incrémentale de treillis qui combine les avantages des méthodes présentées plus haut, au détriment cependant d'une augmentation du temps de calcul des échantillons. Les différentes techniques mentionnées ci-dessus permettent de calculer des suites de points uniformes dans $\mathbb{R}^d$. Leur application à d'autres espaces topologiques n'est pas toujours possible. En robotique, les espaces auxquels on s'intéresse sont généralement des produits cartésiens de nombreux espaces donc de la forme $E = E_1 \times E_2 \times \ldots \times E_n$. Même si on sait échantillonner uniformément chacun des E_i à l'aide des méthodes déterministes décrites plus haut, on ne saura pas forcément échantillonner uniformément E. Les échantillonnages aléatoires permettent de résoudre ce problème.

Échantillonnage aléatoire

L'échantillonnage aléatoire s'effectue en tirant chaque point de la suite à l'aide d'un générateur de nombres aléatoires. L'intérêt d'un tel échantillonnage est que si on peut générer des suites aléatoires uniformes de points $x_1, x_2, \ldots, x_n$ dans des espaces $E_1, E_2, \ldots, E_n$ on peut alors générer une suite aléatoire uniforme dans $E = E_1 \times E_2 \times \ldots \times E_n$ en prenant simplement les points $(x_1, x_2, \ldots, x_n)$. Il faut cependant noter que les suites de nombres aléatoires générés par des ordinateurs ne sont pas réellement

[3] S. R. Lindemann and S.M. LaValle. Incremental low-discrepancy lattice methods for motion planning. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2003)*, pages 2920–2927, septembre 2003.

aléatoires puisque générées de façon déterministe. Cela peut fausser la construction d'une suite pour un espace produit cartésien de plusieurs espaces. Des trois critères de qualité de la partie 1.3.3, seul le critère de qualité incrémentale est bon pour une suite aléatoire. La figure 10 montre une suite de 256 points de $[0,1]^2$ générés aléatoirement avec la fonction $rand$ du logiciel MATLAB. On remarque la grande différence de surface entre les plus petites cellules du diagramme de Voronoï et les plus grandes, à cause de la répartition non uniforme des points.

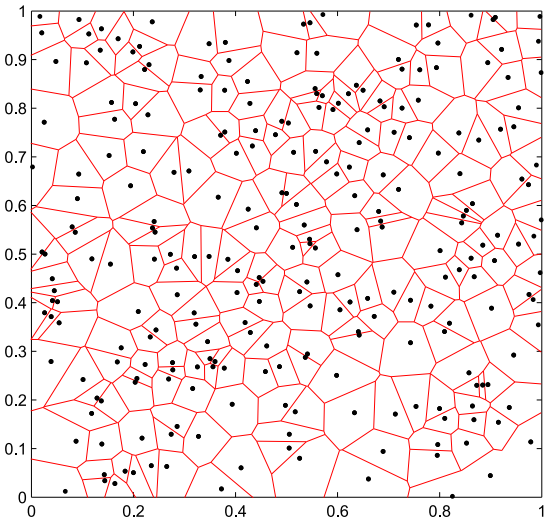


FIGURE 10 – Échantillonnage aléatoire de $[0,1]^2$ sur 256 points. En rouge, le diagramme de Voronoï de l'ensemble de points.

Bibliographie

[ALS94] R. Alami, J.-P. Laumond, and T. Siméon. Two manipulation planning algorithms. *Algorithmic Foundations of Robotics (WAFR'94)*, 1994.

[ASL89] R. Alami, T. Siméon, and J.-P. Laumond. A geometric approach to planning manipulation tasks. the case of discrete placements and grasps. *Fifth International Symposium on Robotics Research*, 1989.

[Bic95] A. Bicchi. On the closure properties of robotic grasping. *The International Journal of Robotics Research*, 14(4) :319–33, août 1995.

[BKL+96] J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, and P. Raghavan. A random sampling scheme for robot path planning. *Proceedings of the International Symposium on Robotics Research*, pages 249–264, 1996.

[BLMV04] L. Biagiotti, F. Lotti, C. Melchiorri, and G. Vassura. How far is the human hand ? a review on anthropomorphic robotic end-effectors. *Internal Report - DEIS (University of Bologna)*, 2004.

[CG98] M. Cherif and K. Gupta. 3D in-hand manipulation planning. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'98)*, pages 146–151, octobre 1998.

[CG99] M. Cherif and K. Gupta. Global planning for dextrous re-orientation of rigid objects : Finger tracking with rolling and sliding. *Rapport de recherche RR-3770, INRIA Rhône-Alpes*, septembre 1999.

[CHS89] A. Cole, J. Hauser, and S. Sastry. Kinematics and control of multifingered hands with rolling contact. *IEEE Transactions on Automatic Control*, 34(4) :398–404, avril 1989.

[Cor03] J. Cortés. Motion planning algorithms for general closed-chain mechanisms. *Thèse de doctorat de l'Institut National Polytechnique de Toulouse*, 2003.

[Dij59] E.W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1 :269–271, 1959.

[DV89] R.S. Desai and R.A. Voltz. Identification and versification of termination conditions in fine motion in presence of sensor errors and geometric uncertainties. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'89)*, pages 800–807, 1989.

[FC92] C. Ferrari and J. Canny. Planning optimal grasps. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'92)*, 3 :2290–2295, mai 1992.

[FH05] M.S. Floater and K. Hormann. Surface parameterization : a tutorial and survey. pages 157–186, 2005.

[GB02] B. Goodwine and J. Burdick. Motion planning for kinematic stratified systems with application to quasi-static legged locomotion and finger gaiting. *IEEE Transactions on Automatic Control*, 18(2) :209–222, 2002.

[GLM96] S. Gottschalk, M.C. Lin, and D. Manocha. OBBTree : A hierarchical structure for rapid interference detection. *Proceedings of ACM Siggraph'96*, 1996.

[Goo98] B. Goodwine. Stratified motion planning with application to robotic finger gaiting. *Proceedings of the IFAC World Congress*, 1998.

[Hal60] J.H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating mulitdimensional integrals. *Numerische Mathematik*, 2 :84–90, 1960.

[Ham60] J.M. Hammersley. Monte-Carlo methods for solving multivariable problems. *Annals of the New York Academy of Science*, 86 :844–874, 1960.

[HKL+98] D. Hsu, L. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. *Robotics : The algorithmic perspective*, pages 141–143, 1998.

[HLMT90] J. Hong, G. Lafferriere, B. Mishra, and X. Tan. Fine manipulation with multifinger hands. *Proceedings of the International Conference on Robotics and Automation (ICRA'90)*, pages 1568–1573, 1990.

[HLP02] I. Harmati, B. Lantos, and S. Payandeh. On fitted stratified and semi-stratified geometric manipulation planning with fingertip relocations. *The International Journal of Robotics Research*, 21(5-6) :489–510, juin 2002.

[HLT+00] L. Han, Z. Li, J.C. Trinkle, Z. Qin, and S. Jiang. The planning and control of robot dextrous manipulation. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2000)*, pages 263–269, 2000.

[HNR68] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2) :100–107, 1968.

[HT98] L. Han and J.C Trinkle. Dextrous manipulation by rolling and finger gaiting. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'98)*, pages 730–735, mai 1998.

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

[KL94] Y. Koga and J.-C. Latombe. On multi-arm manipulation planning. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA'94)*, 2 :945–952, 1994.

[KLMR95] L. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot motion planning. *ACM Symposium on Theory of Computing*, pages 353–362, 1995.

[KSLO96] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4) :566–580, juin 1996.

[LaV98] S.M. LaValle. Rapidly-exploring random trees : a new tool for path planning. Technical Report 98-11, Computer Science Dept., Iowa State University, octobre 1998.

[LaV06] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Consultable en ligne : http ://planning.cs.uiuc.edu/.

[LBL04] S.M. LaValle, M.S. Branicky, and S. R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics Research*, 23(7/8) :673–692, juillet/août 2004.

[LCS89] Z. Li, J. Canny, and S. Sastry. On motion planning for dexterous manipulation, part I : The problem formulation. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA'89)*, pages 775–780, 1989.

[Liu99] Y-H. Liu. Qualitative test and force optimization of 3D frictional form-closure grasps using linear programming. *IEEE Transactions on Robotics and Automation*, 15(1) :163–173, 1999.

[LJTM94] J.-P. Laumond, P.E. Jacobs, M. Taix, and R.M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 10(5) :577–593, 1994.

[LK00] S.M. LaValle and J.J. Kuffner. Rapidly-exploring random trees : Progress and prospects. *Algorithmic Foundations of Robotics (WAFR'00)*, 2000.

[LL03] S. R. Lindemann and S.M. LaValle. Incremental low-discrepancy lattice methods for motion planning. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2003)*, pages 2920–2927, septembre 2003.

[LLC03] J.-W. Li, H. Liu, and H.-G. Cai. On computing three-finger force-closure grasps of 2-D and 3-D objects. *IEEE Transactions on Robotics and Automation*, 19(1) :155–161, février 2003.

[LP83] T. Lozano-Pérez. Spatial planning : a configuration space approach. *IEEE Transactions on Computers*, 32(2) :108–120, 1983.

[LT05] G.F. Liu and J.C. Trinkle. Complete path planning for planar closed chains among point obstacles. *Proceedings of Robotics : Science and Systems*, juin 2005.

[Mon95] D.J. Montana. The kinematics of multi-fingered manipulation. *IEEE Transactions on Robotics and Automation*, 11(4) :491–503, août 1995.

[OF96] T. Omata and M.A. Farooqi. Regrasps by a multifingered hand based on primitives. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA'96)*, 3 :2774–2780, avril 1996.

[ON94] T. Omata and K. Nagata. Planning reorientation of an object with a multifingered hand. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA'94)*, 4 :3104–3110, mai 1994.

[Or94] M. Overmars and P. Švestkhjka. A probabilistic learning approach to motion planning. *Algorithmic Foundations of Robotics (WAFR'94)*, 1994.

[Reu76] F. Reuleaux. Kinematics of machinery. 1876. Consultable en ligne : http ://historical.library.cornell.edu/kmoddl/.

[Rus97] D. Rus. Coordinated manipulation of objects in a plane. *Algorithmica*, 19(1/2) :129–147, 1997.

[Rus98] D. Rus. In-hand dexterous manipulation of 3D piecewise-smooth objects. *International Journal of Robotics Research*, 1998.

[Sah03] A. Sahbani. Planification de tâches de manipulation en robotique par des approches probabilistes. *Thèse de doctorat de l'Université Paul Sabatier, Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS)*, 2003.

[SCSL03] T. Siméon, J. Cortés, A. Sahbani, and J.-P. Laumond. A general manipulation task planner. *Algorithmic Foundations of Robotics (WAFR'03)*, V :311–328, 2003.

[Sho85] K. Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH '85 : Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, New York, NY, USA, 1985. ACM Press.

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

[Som00] P. Somov. Über gebiete von schraubengeschwindigkeiten eines starren körpers bei verschiedener zahl von stützflächen. *Zeitschrift für Mathematik und Physik*, 45, 1900.

[SP03] A. Sudsang and T. Phoka. Regrasp planning for a 4-fingered hand manipulating a polygon. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'03)*, 2 :2671–2676, septembre 2003.

[SSC02] A. Sahbani, T. Siméon, and J. Cortés. A probabilistic algorithm for manipulation planning under continuous grasps and placements. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'2002)*, 2002.

[SSP06] J.-P. Saut, A. Sahbani, and V. Perdereau. A global approach for dexterous manipulation planning using paths in n-fingers grasp subspace. *Proceedings of the 9th IEEE International Conference on Automation, Robotics and Vision (ICARCV)*, pages 2019–2024, décembre 2006.

[SSP07a] A. Sahbani, J.-P. Saut, and V. Perdereau. An efficient algorithm for dexterous manipulation planning. *Proceedings of the 4th IEEE International Multi-Conference on Systems, Signals & Devices (SSD2007)*, I : Conference on Systems Analysis & Automatic Control, mars 2007.

[SSP07b] J.-P. Saut, A. Sahbani, and V. Perdereau. Dexterous manipulation planning using probabilistic roadmaps in continuous grasp subspaces. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, novembre 2007.

[Sve96] P. Svestka. On probabilistic completeness and expected complexity of probabilistic path planning. *Technical Report*, 1996.

[TH91] J.C. Trinkle and J. Hunter. A framework for planning dexterous manipulation. *Proceedings of the IEEE Conference on Robotics and Automation (ICRA'91)*, pages 775–780, avril 1991.

[TM02] J.C. Trinkle and R.J. Milgram. Complete path planning for closed kinematic chains with spherical joints. *International Journal of Robotics Research*, 21(9) :773–789, septembre 2002.

[XL05] J. Xu and Z. Li. Kinematic modelling of multifingered hand's finger gaits as hybrid automaton. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, pages 3252–3257, août 2005.

[YSY03] M. Yashima, Y. Shiina, and H. Yamaguchi. Randomized manipulation planning for a multi-fingered hand by switching contact modes. *Procee-*

dings of the 2003 IEEE International Conference on Robotics and Automation (ICRA'2003), septembre 2003.

[YY02] M. Yashima and H. Yamaguchi. Dynamic motion planning whole arm grasp systems based on switching contact modes. *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA'2002)*, mai 2002.

[Zef02] M. Zefran. A feedback strategy for dextrous manipulation. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2002)*, 3 :2479–2484, 2002.

[ZQ05] Y. Zheng and W.-H. Qian. Coping with the grasping uncertainties in force-closure analysis. *The International Journal of Robotics Research*, 24(4) :311–327, avril 2005.

[ZTM96] H. Zhang, K. Tanie, and H. Maekawa. Dextrous manipulation planning by grasp transformation. *Proceedings of the IEEE Conference on Robotics Automation (ICRA'96)*, avril 1996.

Publications

Conférences Internationales

J.-P. Saut, C. Rémond, V. Perdereau, M. Drouin. Online Computation of Grasping Forces in Multi-Fingered Hands. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Canada, Août 2005.

J.-P.. Saut, A., Sahbani, V. Perdereau. A Global Approach for Dexterous Manipulation Planning Using Paths in n-fingers Grasp Subspace. *Proceedings of the 9th IEEE International Conference on Automation, Robotics and Vision (ICARCV)*, Singapour, Décembre 2006.

A., Sahbani, J.-P.. Saut, V. Perdereau. An Efficient Algorithm for Dexterous Manipulation Planning. *Proceedings of the 4th IEEE International Multi-Conference on Systems, Signals & Devices (SSD2007),Volume I : Conference on Systems Analysis & Automatic*, Hammamet, Tunisie, Mars 2007.

J.-P.. Saut, A., Sahbani, V. Perdereau. Dexterous Manipulation Planning Using Probabilistic Roadmaps in Continuous Grasp Subspaces. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, États-Unis, Novembre 2007.

Conférences Nationales

J.-P.. Saut, A., Sahbani, V. Perdereau. Planification de Mouvement pour la Manipulation Fine d'Objets Rigides. *MAnifestation des JEunes Chercheurs en STIC (MajecSTIC)*, Lorient, Novembre 2006.

J.-P.. Saut, A., Sahbani, V. Perdereau. Planification de Mouvement pour la Manipulation Dextre d'Objets Rigides. *Journées Nationales de la Recherche en Robotique (JNRR), session poster*, Obernai, Octobre 2007.

Table des figures