Thèse

préparée au

Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

en vue de l'obtention du

Doctorat de l'Université Paul Sabatier de Toulouse

Spécialité :

Robotique - Informatique

par

Anis SAHBANI

Planification de tâches de manipulation en robotique par des approches probabilistes

---

Soutenue le 17 Février 2003, devant le jury :

Président :	**Raja CHATILA**	**LAAS-CNRS, Toulouse**
Rapporteurs :	**Bruno ARNALDI**	**INRIA, Rennes**
	Emmanuel MAZER	**INRIA, Rhône-Alpes**
Examinateurs :	**Jean-Paul LAUMOND**	**LAAS-CNRS, Toulouse**
	Claude LAURGEAU	**École des Mines, Paris**
Directeur de thèse :	**Thierry SIMÉON**	**LAAS-CNRS, Toulouse**

LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex 4

À la mémoire de Filipe

À mes parents

Avant Propos

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS. Je remercie les directeurs successifs du LAAS, Jean-Claude Laprie et Malik Ghallab, de m'avoir accueilli au sein de leur laboratoire.

Je remercie également Raja Chatila, Directeur de Recherche au CNRS et responsable du groupe Robotique et Intelligence Artificielle, de m'avoir permis de travailler dans son groupe de recherche et d'avoir accepté de présider ce jury de thèse.

Je tiens à dire toute ma reconnaissance à Bruno Arnaldi, professeur à l'INSA de Rennes, Emmanuel Mazer, directeur de recherche à l'INRIA Rhône-Alpes, pour avoir accepté d'être rapporteurs de cette thèse ainsi que pour leur lecture pertinente du manuscrit.

Merci également à Jean-Paul Laumond, directeur de recherche au LAAS-CNRS, Claude Laurgeau, directeur du Centre CAO & Robotique de l'école des Mines de Paris, pour avoir accepté de faire partie du jury de ma thèse. Je voudrais également leur témoigner toute ma reconnaissance pour leurs conseils ainsi que leurs réflexions sur ce manuscrit.

Je tiens à remercier vivement Thierry Siméon, chargé de recherche au LAAS-CNRS, pour avoir encadré ces travaux. Je ne saurais m'empêcher d'exprimer à mon directeur de thèse ma réelle reconnaissance et ma grande gratitude pour sa patience, sa disponibilité et son soutien moral dont il a fait preuve le long de ces trois années de thèse.

Je remercie l'ensemble du groupe RIA, permanents, doctorants et stagiaires. Mes remerciements vont également à toutes les personnes qui, à un moment ou à un autre, m'ont apporté leur aide, et dont il est impossible de dresser ici une liste exhaustive.

Je ne saurais oublier de remercier tous ceux qui parviennent à me supporter au quotidien, Paulo, Juan, Julien, Jib, Tahar, Asfour, Slim, Olfa, Farah, ... Merci pour votre amitié et vos encouragements.

Enfin merci à Eve, pour sa compréhension, sa patience et pour avoir supporté mes états d'âme.

Table des matières

Introduction générale **11**

1 État de l'art des méthodes probabilistes et des planificateurs de tâches de manipulation **15**

1.1 Introduction 15

1.2 Planification de mouvement 16

1.2.1 Les méthodes probabilistes 17

1.2.2 Variantes 20

1.2.3 Quelques propriétés des méthodes probabilistes 28

1.2.4 Applications 34

1.2.5 Vers des approches déterministes 36

1.3 Planification de tâches de manipulation 37

1.3.1 Formulation du problème 38

1.3.2 Extension à la manipulation multi-robots 40

1.3.3 Extension de l'algorithme *"Fil d'Ariane"* 42

1.3.4 Approche par *"contraintes progressives"* 44

1.3.5 Extension des méthodes probabilistes : *"Fuzzy PRM"* 45

1.4 Conclusion 46

2 Formulation Géométrique et Approche Adoptée **49**

2.1 Introduction 49

2.2 Structuration de l'espace de recherche 50

2.2.1 Préliminaires 50

2.2.2 Contraintes de placement 51

2.2.3 Contraintes de mouvement 53

2.2.4 Chemin de manipulation 56

2.3 Le problème de planification de tâches de manipulation 58

2.3.1 Cas de zones de placements et de prises discrètes 60

2.3.2 Cas de zones de placements et de prises continues 63

2.4 Propriété de réduction 66

2.5 Conclusion . . . 68

3 Planification de tâches de manipulation sous contraintes discrètes 69

3.1 Introduction . . . 69

3.2 Graphe de manipulation et réseaux probabilistes . . . 70

3.2.1 Le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ . . . 70

3.2.2 Graphe de manipulation . . . 70

3.3 Approche par graphe dynamique . . . 75

3.3.1 Approche proposée . . . 75

3.3.2 Choix effectués . . . 77

3.3.3 Exemples de problème de manipulation sous contraintes discrètes . . . 78

3.4 Conclusion . . . 83

4 Planification de tâches de manipulation sous contraintes continues 85

4.1 Introduction . . . 85

4.2 Approche proposée . . . 86

4.2.1 Principe . . . 86

4.2.2 Composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$ . . . 87

4.2.3 Connexions par des chemins de transit et de transfert . . . 94

4.3 Algorithme de manipulation . . . 98

4.3.1 Construction du graphe de manipulation . . . 98

4.3.2 Recherche du chemin de manipulation . . . 103

4.3.3 Optimisation et lissage du chemin de manipulation . . . 105

4.4 Conclusion . . . 106

5 Résultats et Analyses 107

5.1 Introduction . . . 107

5.2 Présentation de la plate-forme logicielle Move3D . . . 107

5.2.1 Architecture de Move3D . . . 107

5.2.2 Extension de l'architecture au problème de manipulation . . . 109

5.3 Exemples de problèmes de manipulation . . . 111

5.3.1 Description d'un problème de manipulation . . . 111

5.3.2 Le bras manipulateur et la cage . . . 111

5.3.3 Le bras manipulateur avec multiples orientations de prise/pose de l'objet . . . 115

5.3.4 Manipulateur holonome mobile . . . 117

5.3.5 Pont Roulant . . . 119

5.4 Analyse de la sensibilité de l'algorithme en fonction d'une variation des paramètres . 122

5.5 Conclusion . . . 123

Conclusion générale 125

Références bibliographiques **127**

Liste des tableaux

3.1 Caractéristiques des graphes calculés 79

5.1 Caractéristiques des graphes de manipulation construits. 113
5.2 Comparaison entre les temps de résolution du problème selon la technique utilisée pour capturer la connectivité du sous-espace $\mathcal{CG} \cap \mathcal{CP}$. 113
5.3 Influence de la technique utilisée pour le calcul du graphe initial sur les temps de résolution. . 114
5.4 Caractéristiques du graphe de manipulation. 116
5.5 Caractéristiques du graphe de manipulation. 121

Table des figures

1.1 Les trois étapes de *PRM* : a) Phase d'apprentissage. b) Phase de recherche. c) Phase de lissage. 17
1.2 a) Graphe calculé en permettant une distance de pénétration dans les obstacles. b) Modification du graphe en repoussant les configurations non-valides. 21
1.3 a) Échantillonnage Uniforme. b) Échantillonnage Gaussien. 21
1.4 Domaines de visibilité d'une configuration pour la méthode locale Linear (gauche) et Manhattan (droite). 22
1.5 Deux graphes issus du même échantillonnage : PRM classique à droite et visibilité PRM à gauche. 24
1.6 Planification pour un bras manipulateur tiré de [Bohlin 00]. 25
1.7 Évolution des balises dans l'espace libre tiré de [Ahuactzin 94]. 26
1.8 Évolution d'un arbre couvrant l'espace libre par la méthode RRT. 27
1.9 Exemple de problème de planification de mouvement résolu par RRT. 28
1.10 Une méthode locale vérifiant la propriété topologique générale. 30
1.11 Une méthode locale vérifiant la propriété topologique locale lipschitzienne. 30
1.12 La β-lookout d'une configuration q. 32
1.13 Un passage étroit dans un espace ε-*good*. 33
1.14 Deux *"benchmarks"* avec des robots *"free-flying"* 34
1.15 Échantillonnage aléatoire vs. quasi-aléatoire (tiré de [Branicky 01]). 37
1.16 Un exemple de problème de manipulation résolu dans [Alami 95]. 39
1.17 Un autre type de problème de manipulation traité dans [Alami 95]. 40
1.18 Un exemple de coopération entre trois bras manipulateurs de type PUMA 560 tiré de [Koga 95]. 41
1.19 Un exemple d'acteur virtuel manipulant un échiquier tiré de [Koga 95]. 42
1.20 Un autre exemple d'acteur virtuel manipulant plusieurs objets déplaçables tiré de [Kuffner 99]. 42
1.21 Exemple de problème de manipulation résolu dans [Ahuactzin 98]. 43
1.22 Un bras manipulateur à 3 degrés de liberté manipulant deux disques dans un environnement 2D (tiré de [Barraquand 94]). 44
1.23 Bras manipulateur déplaçant un objet en forme de L [Nielsen 00]. 46

2.1 Placement stable de l'objet déplaçable (la barre) : placement non-valide (gauche) vs. valide(droite). 51

2.2 Deux orientations différentes pour poser l'objet déplaçable sur la même zone de pose (le sol). 52
2.3 Un chemin calculé dans le sous-espace *PLACEMENT*. 53
2.4 Différentes prises de l'objet déplaçable par le bras manipulateur. 54
2.5 Un exemple 2D avec prise continue. 55
2.6 Un chemin dans le sous-espace *GRASP*. 56
2.7 Chemin de transit (gauche) vs. chemin de transfert(droite). 57
2.8 Structure de l'espace de recherche et chemin de manipulation. 58
2.9 Graphe de manipulation dans le cas de problèmes de manipulation sous contraintes discrètes. 61
2.10 Un exemple 2D de problème de manipulation sous contraintes discrètes [Siméon 89]. 62
2.11 Connexions entre différentes tranches d'espace de configuration. 63
2.12 Graphe de manipulation dans le cas de problèmes de manipulation sous contraintes continues. 64
2.13 Un exemple 2D de problème de manipulation sous contraintes continues. 65
2.14 Le graphe de manipulation calculé. 65
2.15 Illustration de l'hypothèse1. 66
2.16 Le cas d'un grand nombre de chemins de transit/transfert. 68

3.1 a) Exemple de problème de manipulation sous contraintes discrètes. b) Les deux configurations formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ entrées du problème. 71
3.2 Structure du "méta-graphe" capturant la connectivité de l'espace de configurations valides ($\mathcal{CG} \cup \mathcal{CP}$). 72
3.3 Les trois graphes calculés par la méthode probabiliste (*Visibility-PRM*). 73
3.4 Le "méta-graphe" construit à partir des trois graphes calculés (gauche) permettant le calcul du chemin de manipulation solution du problème posé (droite). 73
3.5 Evolution du nombre de configurations dans $\mathcal{CG} \cap \mathcal{CP}$ en fonction du nombre d'objets déplaçables présents dans l'environnement et de placements stables définis. 74
3.6 Le graphe marqué en fonction du placement des objets déplaçables. 76
3.7 Recherche de chemin dans le graphe marqué : a) il n'existe pas de solution, b) un chemin est trouvé, c) une mise à jour locale des arêtes bloquantes. 76
3.8 Exemples de problème de manipulation avec deux objets déplaçables. 78
3.9 Les entrées du problème (placements intermédiaires et prises). 78
3.10 Les graphes calculés pour le robot en considérant uniquement les obstacles statiques et en utilisant respectivement *PRM* (gauche) et *Visibilité-PRM* (droite) 79
3.11 Le chemin de manipulation calculé. 80
3.12 Un exemple industriel de problème de manipulation. 81
3.13 Le graphe initial calculé par : *PRM* (à gauche) et par *Visibilité-PRM* (à droite). 82
3.14 Pont-roulant transportant le capot d'une turbo-pompe dans une centrale nucléaire (modèle EDF). 83

4.1 Exemple de chaîne fermée formée par le système global avec un chemin calculé dans $\mathcal{CG} \cap \mathcal{CP}$ (la barre se déplace sur le sol tout en glissant entre les mors de la pince). . . 88
4.2 La chaîne cinématique fermée formée par le robot et l'objet déplaçable. 89
4.3 Deux chaînes fermées différentes selon la classe de prise 90
4.4 Exemples de problèmes de planification de mouvement pour des mécanismes fermés. 91
4.5 Capturer la connectivité de $\mathcal{CG} \cap \mathcal{CP}$ dans un réseau probabiliste. 92
4.6 Un graphe calculé dans $\mathcal{CG} \cap \mathcal{CP}$ en utilisant la technique de visibilité (à gauche) et deux configurations du système appartenant à deux composantes connexes différentes de $\mathcal{CG} \cap \mathcal{CP}$ pour un même placement de l'objet (à droite). 94
4.7 Structure du graphe de manipulation. 95
4.8 Un chemin de transit connectant deux nœuds partageant le même placement de l'objet déplaçable. 96
4.9 Un chemin de transfert connectant deux nœuds partageant la même prise. 97
4.10 Connexion par *"re-grasping"* entre deux nœuds N et N_{k,C_i} de $\mathcal{CG} \cap \mathcal{CP}$. 101
4.11 Un chemin calculé dans $\mathcal{CG} \cap \mathcal{CP}$ (à gauche) transformé en une séquence de trois chemins admissibles *transfert/transit/transfert* (droite). 104

5.1 Architecture de la plate-forme logicielle *Move3D*. 108
5.2 Extension aux problèmes de planification de tâches de manipulation. 110
5.3 Configurations initiale (à gauche) et finale (à droite) du système global. 111
5.4 Le graphe de manipulation calculé avec : a)*PRM* (gauche). b)*Visibilité-PRM* (droite). 112
5.5 Graphe initial pour le robot calculé avec : a) PRM. b) Visibilité-PRM. 113
5.6 La séquence de chemins transit/transfert composant le chemin de manipulation solution au problème. 114
5.7 Scène et configurations initiale et finale du système global. 115
5.8 Le graphe de manipulation construit pour ce problème. 116
5.9 La séquence de chemins transit/transfert composant le chemin de manipulation solution au problème. 117
5.10 Exemple de problème de manipulation avec un robot redondant. 118
5.11 Le graphe de manipulation calculé pour ce problème. 118
5.12 Caractéristiques du graphe de manipulation. 118
5.13 Le chemin de manipulation solution de ce problème. 119
5.14 Exemple de problème de logistique industrielle (projet MOLOG). 120
5.15 Le graphe de manipulation construit par le planificateur. 120
5.16 Le chemin de manipulation calculé pour ce problème. 121
5.17 Évolution du temps de construction du graphe de manipulation en fonction d'une variation de la probabilité de choisir l'une des stratégies d'expansion (expansion dans $\mathcal{CG} \cap \mathcal{CP}$ vs. connexions par transit/transfert). 122
5.18 Exemple d'une simulation d'interactions entre homme-humanoïde (tiré de [Chatila 02]). . . 126

Introduction générale

Le mythe fascinant de la machine créée par l'homme qui effectuerait toutes les tâches a nourri pendant longtemps l'imaginaire du grand public. Ce mythe symbolise la volonté de l'homme de faire exécuter des tâches fastidieuses par des machines appelées robots. L'évolution de ces machines est généralement classifiée suivant des "générations" reflétant les potentialités offertes par ces dernières. Les robots de première génération exécutent uniquement une suite de mouvements pré-enregistrés (chariot filoguidé). Ceux de la deuxième génération sont dotés de fonctions élémentaires de perception leur permettant de se diriger de manière simple dans leur environnement. Les développements récents de la robotique visent à accroître l'autonomie de ces systèmes. Cette notion d'autonomie représente une composante importante caractérisant les robots de la troisième génération.

Un système robotique agit par le mouvement dans un monde physique. Sa capacité de planification de mouvement apparaît ainsi comme une composante essentielle de l'autonomie d'un robot.

Planification de mouvement et robotique :
Dans sa version la plus simple, la planification de mouvement s'intéresse au calcul automatique de chemins sans collision pour un système mécanique (robot mobile, bras manipulateur, personnage animé, ...) évoluant dans un environnement encombré d'obstacles. La solution de ce problème, appelée trajectoire, se présente comme une suite continue de situations géométriques successivement occupées par le robot durant son déplacement.

Les premières solutions au problème de planification de mouvement remontent au début des années 1980 avec l'introduction de la notion *d'espace des configurations* [Lozano-Pérez 83]. Une configuration regroupe l'ensemble des paramètres permettant de localiser un système mécanique dans son environnement. La planification de mouvement pour un système est alors ramenée au problème de la planification de mouvement d'un point dans cet espace des configurations. De nombreuses approches ont par la suite été proposées pour la résolution de diverses instances de ce problème, visant à construire ou explorer efficacement cet espace. Face à la complexité combinatoire du problème, les approches probabilistes [Kavraki 96, Švestka 96] apparues depuis une dizaine d'années permettent aujourd'hui de résoudre de nombreux problèmes qui restent hors de portée des méthodes déterministes initialement développées, en particulier lorsqu'il s'agit d'explorer des espaces de recherche hautement dimensionnés. Leur champ

d'application dépasse aujourd'hui la seule robotique : elles intéressent des domaines aussi diversifiés que la *CAO*, l'animation graphique ou la biochimie [Latombe 99].

Tâches de manipulation :
Le problème de la planification de tâches de manipulation consiste à calculer la séquence de mouvements permettant à un (ou plusieurs) robot(s) de déplacer un ou plusieurs objets d'un placement initial à un placement but. Il ne se limite pas au niveau du calcul automatique des chemins sans collision pour un robot, mais intègre également le séquencement des différentes sous-tâches de prise et de pose d'objets.

Plusieurs travaux ont été effectués dans ce domaine, notamment dans le cadre des projets *HANDEY* [Mazer 87, Lozano-Pérez 87], *SHARP* [Laugier 88] et *SPARA* [Mazon 90, Violero 91], qui décomposaient le problème initial en actions élémentaires de "*pick and place*". Cependant, dans certains problèmes de planification, une simple requête de déplacement d'un robot peut nécessiter la génération d'une séquence complexe de manipulations d'objets afin de modifier la structure de l'environnement (ex : déplacement d'objets gênants). Une telle décomposition en sous-tâches ne peut être obtenue par ces systèmes. Elle nécessite une analyse plus complexe et plus fine de l'espace des configurations composite (espace de configurations des robots plus celui des objets déplaçables). Il s'agit en effet d'un problème plus riche que la seule planification de mouvement, et qui est plus directement lié à la notion de tâches en robotique : la manipulation.

Contexte des travaux :
Cette thèse fait suite aux travaux menés au LAAS au début des années 1990 qui avaient conduit à une approche originale pour la planification de tâches de manipulation dans des environnements encombrés d'obstacles [Alami 89, Siméon 89, Alami 95]. L'apport de cette approche, reposant sur une formulation géométrique du problème, était de produire la décomposition en sous-tâches élémentaires en même temps que la séquence de mouvements du robot. Cependant, les planificateurs proposés restent limités à des instances particulières du problème de manipulation (environnement 2D, robots polygonaux, ensemble de prises et de poses discret, ...). Les progrès algorithmiques de ces dernières années, en particulier le développement des approches probabilistes pour la planification de mouvement, permettent aujourd'hui de revisiter ce problème.

Les travaux présentés dans ce mémoire s'inscrivent dans le cadre du projet européen *MOLOG* (MOtion for LOGistics) [Siméon 01a]. Ce projet est dédié à l'intégration de techniques de planification dans des systèmes de *CAO* utilisés dans des opérations de logistique industrielle. Il s'agit d'automatiser la résolution des problèmes spécifiques qui se posent lors de la maintenance des installations nucléaires et qui nécessitent le transport d'équipements lourds au moyen de divers outils de manutention.

Contribution :

La contribution principale de cette thèse est une approche générique qui permet de traiter le cas de domaines continus de prise et de pose. Elle permet également d'attaquer des problèmes réels de manipulation (environnement 3D, systèmes mécaniques complexes, ...) posés par diverses applications (programmation de robots, logistique, animation graphique, ...). De plus, cette approche est capable de répondre aux besoins de ces applications en terme d'efficacité, de généralité et de robustesse algorithmique.

Plan du manuscrit :

Ce manuscrit est composé de 5 chapitres. Nous abordons dans la première partie du chapitre 1 le problème de planification de mouvement et nous présentons un état de l'art détaillé des approches probabilistes. En effet, la ligne forte qui se retrouve dans nos travaux est de privilégier des méthodes effectives de planification. Contrairement aux approches algorithmiques exactes, les méthodes probabilistes ont su se soustraire à la combinatoire du problème de planification de mouvement et présentent des caractéristiques intéressantes. La seconde partie de ce chapitre constitue une synthèse des différents travaux qui ont été proposés dans la littérature pour la planification de tâches de manipulation. Nous discuterons également les limitations présentées par ces différents algorithmes.

Le chapitre 2 reprend et étend la formulation géométrique du problème de manipulation introduite dans [Alami 89]. Nous revenons dans la première partie de ce chapitre sur la notion d'espaces de pose et de prise permettant de structurer l'espace de recherche. La structure de ces deux sous-espaces permet aussi de distinguer deux instances du problème de manipulation : le cas discret et le cas continu d'ensemble de prises et de poses. Nous discuterons les limitations de cette approche dans chacun de ces cas à travers deux exemples tirés de [Alami 95] et présentés à la fin de ce chapitre.

Le chapitre 3 présente une extension des approches probabilistes (appelée *approche par graphe dynamique*) permettant de traiter une large classe de problèmes de manipulation (environnement 3D, robots articulés, ...). Cette approche est cependant limitée à une instance particulière du problème de manipulation, à savoir le cas d'ensemble discret de prises et de poses. Elle représente une partie d'une approche plus générale, présentée dans le chapitre 4, permettant la résolution de problèmes de manipulation sous contraintes continues. L'approche par graphe dynamique repose sur l'utilisation des méthodes probabilistes pour le calcul d'un ensemble initial de chemins couplées avec un mécanisme "paresseux" de mise à jour de cet ensemble. Cette extension des techniques probabilistes permet de gérer plus efficacement plusieurs requêtes de planification de mouvement dans des environnements partiellement changeants (par le placement des différents objets déplaçables).

Dans le chapitre 4, nous proposons une approche inspirée du schéma de résolution présenté dans [Alami 89] et capable de traiter le cas de domaines continus de prises et de poses. Elle consiste à capturer dans un premier temps la connectivité d'une sous-variété particulière de l'espace des confi-

gurations composite dans une structure de graphe. Les connexions entre les différentes composantes connexes de ce graphe sont ensuite calculées en utilisant l'approche par graphe dynamique. Le chemin solution au problème de manipulation est cherché dans le graphe ainsi construit.

Le chapitre 5 présente l'architecture de *Move3D*, une plate-forme logicielle dans laquelle nous avons intégré les algorithmes de manipulation développés selon notre approche. Une analyse empirique des performances de l'algorithme est également présentée dans ce chapitre à travers les différents exemples de problèmes de manipulation résolus par le planificateur.

Chapitre 1

État de l'art des méthodes probabilistes et des planificateurs de tâches de manipulation

1.1 Introduction

La capacité de se mouvoir est une caractéristique essentielle permettant à un système robotique d'effectuer des tâches de manipulation. Il est donc clair que nous ne pouvons pas attaquer le problème de manipulation sans aborder la planification de mouvement.

Le problème de planification de mouvement est sans conteste celui qui a suscité le plus de travaux durant ces dernières années. Dans sa forme la plus simple, il est intrinsèquement difficile. La plupart des problèmes de planification de mouvement résistent aux approches algorithmiques exactes. Il est donc nécessaire de concevoir des approches de nature différente, permettant par exemple de calculer des solutions approchées ou basées sur des techniques d'exploration non-déterministes. La première partie de ce chapitre présente un état de l'art détaillé de ce type d'approches.

La planification de tâches de manipulation est un problème plus riche et combinatoirement plus complexe que la seule planification de mouvement. La plupart des approches développées se sont limitées à des instances particulières de ce problème, la seconde partie de ce chapitre leur est entièrement consacrée. Nous présentons une synthèse des différents travaux qui ont été effectués dans ce domaine et nous discutons leur limitation.

1.2 Planification de mouvement

Le problème de planification de mouvement consiste à calculer une trajectoire pour un robot entre une position initiale et une autre finale. Il peut être sommairement posé de la façon suivante : Étant donné un robot évoluant dans un environnement parsemé d'obstacles, trouver s'il existe un mouvement amenant ce robot d'une position à une autre, respectant ses contraintes cinématiques, tout en évitant les obstacles. L'algorithmique de la planification de mouvement débute avec l'introduction de la notion d'espace de configurations par Lozano-Pérez dans [Lozano-Pérez 83]. Il a substitué la recherche d'un chemin sans collision pour un robot évoluant parmi des obstacles par la recherche de chemin d'un point évoluant dans l'espace de configurations du robot. La planification de mouvement devient rapidement un domaine de recherche très actif, se focalisant sur quatre approches principales : la décomposition cellulaire [Schwartz 83], les méthodes de rétraction [O'Dùnlaing 82], les champs de potentiels [Khatib 86] et la construction de graphe (roadmaps) [Nilsson 69]. Dans sa version la plus simple, le problème est intrinsèquement difficile [Reif 79], sa complexité croit exponentiellement avec le nombre de degrés de liberté du robot [Canny 87].

Traditionnellement l'algorithmique du mouvement conduisait soit à des méthodes exactes et complètes, soit à des méthodes heuristiques. La complexité des premières rendait leur utilisation rédhibitoire dès qu'il s'agit de systèmes de dimension élevée. Les secondes souffraient des inconvénients liés à la non complétude.

Plus récemment, plusieurs approches non déterministes sont apparues, elles ne reposent plus sur une construction préalable d'une représentation exacte ou approchée de l'espace de configurations. Certaines de ces méthodes se basent sur l'exploration de l'espace de configurations admissibles, d'autres visent à capturer sa connectivité. Ces nouvelles méthodes de planification dites *"méthodes probabilistes"*, permettent de faire face à la complexité exponentielle du problème. Par ailleurs, elles vérifient une propriété de complétude probabiliste (garantissant en probabilité de trouver une solution en temps fini, lorsqu'elle existe). Certes cette propriété est moins forte que la complétude déterministe vérifiée par les méthodes exactes, mais ces nouvelles approches sont désormais moins sensibles à la dimension de l'espace de recherche et s'avèrent très efficaces en pratique.

Dans la section suivante, nous présenterons en détail ces nouvelles approches ainsi que les différentes variantes du schéma classique proposées dans la littérature. Pour plus de détails sur l'ensemble des contributions, notamment sur les autres méthodes, nous renvoyons les lecteurs au livre de référence dans ce domaine [Latombe 91b] ainsi qu'à deux autres ouvrages plus récents [Laumond 98] et [Gupta 98].

1.2.1 Les méthodes probabilistes

Ces méthodes ont été introduites en parallèle par Kavraki et Latombe sous le nom de *PRM* (Probabilistic Roadmap Planner) [Kavraki 95a, Kavraki 96] et Švestka et Overmars sous le nom de de *PPP* (Probabilistic Path Planner) [Švestka 96]. Depuis, plusieurs variantes de ces méthodes ont été proposées dans la littérature mais toutes se basent sur le même concept. La technique développée se décompose en trois phases : phase d'apprentissage, phase de recherche et une phase de lissage des chemins. La figure 1.1 illustre les trois étapes d'une méthode probabiliste classique permettant la résolution d'un problème de planification de mouvement.

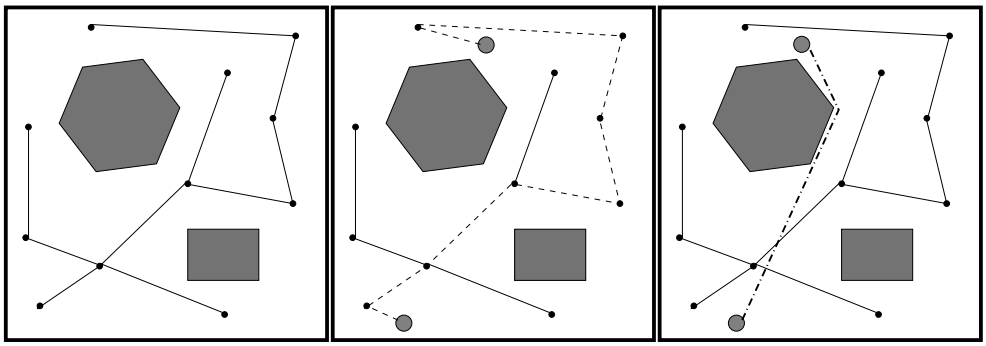


FIG. 1.1 – Les trois étapes de *PRM* : a) Phase d'apprentissage. b) Phase de recherche. c) Phase de lissage.

La phase d'apprentissage

Le principe sous-jacent aux méthodes probabilistes est simple. Il consiste à capturer la connectivité de l'espace libre par un graphe (ou un arbre), construit à partir de points choisis aléatoirement dans l'espace des configurations. Un ***détecteur de collision*** permet de vérifier si les points tirés sont des configurations admissibles (sans collision). Une ***méthode locale*** permet de tester l'existence d'un chemin faisable entre deux points donnés du graphe. Deux points sont ainsi reliés par un arc lorsque le chemin calculé est sans collision. La méthode locale utilisée prend en considération les différentes contraintes mécaniques et cinématiques du système. L'algorithme de construction du graphe $G = (N, A)$, où N est l'ensemble des nœuds du graphe et A l'ensemble de ses arêtes, est présenté ci-dessous.

Le graphe capturant la topologie de l'espace libre peut être construit dans une étape préalable afin de résoudre efficacement plusieurs requêtes de planification de mouvement. Il peut également être enrichi au fur et à mesure que le planificateur résout un problème.

Plusieurs points sont à détailler dans ce schéma global : Quelle méthode locale utiliser ? Comment choisir les nœuds à connecter ? Quelle mesure de distance adopter ? etc.

Algorithme 1 Phase d'apprentissage

1: $V \leftarrow \emptyset$; $E \leftarrow \emptyset$
2: **Tant Que** $nb_{noeuds} < N_{max}$ **Faire**
3: $c \leftarrow$ configuration générée aléatoirement dans C_{libre}
4: $V \leftarrow V \cup \{c\}$
5: $V_c \leftarrow$ un ensemble de voisins de c inclus dans V
6: **Pour Tout** v de V_c par ordre de distance croissante de c **Faire**
7: **Si** v et c sont connectés dans G **Alors**
8: **Si** la méthode locale utilisée trouve un chemin connectant v et c **Alors**
9: $A \leftarrow A \cup \{c, v\}$
10: **Fin Si**
11: **Fin Si**
12: **Fin Pour**
13: **Fin Tant Que**

N_{max} est le nombre maximum de nœuds générés à partir duquel on considère que le graphe est suffisant.

La méthode locale est imposée en général par les contraintes cinématiques du robot. Néanmoins, son choix peut avoir une influence importante sur les performances de l'algorithme [Laumond 00]. Elle est appelée un grand nombre de fois lors de la construction du graphe. Elle doit donc être rapide et efficace.

Le choix des nœuds à connecter au nouveau nœud généré aléatoirement (c) est également un paramètre important dans cette phase d'apprentissage. En effet, ce choix influencera la structure du graphe ainsi que les performances de l'algorithme. Ce choix dicte également le nombre d'appels à la méthode locale (qui peut être coûteuse en terme de temps de calcul). Généralement, un nœud est dit *voisin* du nœud généré c s'il est situé à une distance inférieure à un seuil d_{max}. Un nœud appartenant à la même composante connexe que c n'est donc pas forcément considéré comme un voisin de ce dernier. Par ailleurs, l'interdiction des cycles dans le graphe peut être envisagée afin de minimiser aussi le nombre d'appels au planificateur local.

La fonction distance utilisée devient donc un paramètre critique de l'algorithme puisque c'est à elle de définir l'ensemble des nœuds voisins du nouveau nœud c. Elle permet aussi, par la définition de cet ensemble, d'éviter des appels inutiles à la méthode locale. Elle devra refléter la chance de connexion de deux nœuds par la méthode locale.

La phase de recherche

Lorsque le graphe construit reflète la connectivité de l'espace libre, il peut être utilisé pour la résolution des différentes requêtes de planification. La phase de recherche consiste à relier dans un premier temps la configuration de départ et le but aux nœuds du graphe en utilisant la méthode locale. En cas

de succès, une recherche de chemin dans le graphe est effectuée. Ce chemin, quand il existe, est une séquence de chemins locaux. Un algorithme de type A^* peut être utilisé pour la recherche du chemin dans le graphe.

Si les deux configurations, départ et but, n'appartiennent pas à la même composante connexe, une phase d'enrichissement du graphe peut être effectuée [Kavraki 95a]. Cette phase est arrêtée une fois que les deux configurations sont dans la même composante connexe, ou si un critère d'arrêt est vérifié. Ce critère peut consister en un nombre maximal de nœuds rajoutés au graphe lors de cette phase.

Optimisation et lissage des chemins

En raison du caractère probabiliste du planificateur, le chemin solution n'est généralement pas lissé. Il peut être long et irrégulier. Une phase d'optimisation locale ou plutôt de lissage s'avère donc nécessaire. Une méthode simple, introduite dans [Laumond 94], consiste à découper aléatoirement le chemin à lisser en trois parties, et à essayer de remplacer chacune d'elles par le chemin local reliant leurs extrémités. Dans le cas où l'une des portions calculées est plus courte que la séquence initiale des chemins locaux, elle remplace la séquence initiale. Cette opération est répétée jusqu'à ce que le gain de lissage devienne négligeable ou que la durée fixée à la phase d'optimisation soit écoulée. L'algorithme de cette phase de lissage est le suivant :

Algorithme 2 Phase de lissage

1: $\gamma \leftarrow$ séquence de chemins locaux obtenue après la phase de recherche
2: **Tant Que** ($\mathcal{GAIN} \geq \varepsilon$) **et/ou** ($\mathcal{DUREE} < T_{max}$) **Faire**
3: $l_2 \leftarrow$ tiré aléatoirement dans $[0, 1]$
4: $l_1 \leftarrow$ tiré aléatoirement dans $[0, l_2]$
5: $\varsigma_1 \leftarrow \gamma(l_1 \times \alpha)$; $\alpha \in [0, 1]$
6: $\varsigma_2 \leftarrow \gamma((l_2 - l_1) \times \alpha + l_1)$; $\alpha \in [0, 1]$
7: $\varsigma_3 \leftarrow \gamma((1 - l_2) \times \alpha + l_2)$; $\alpha \in [0, 1]$
8: **Pour Tout** ς_i **Faire**
9: **Si** $\mathcal{P}(\varsigma_i(0), \varsigma_i(1)) \subset CS_{libre}$ **et** $\mathcal{L}(\mathcal{P}(\varsigma_i(0), \varsigma_i(1))) < \mathcal{L}(\varsigma_i)$ **Alors**
10: $\varsigma_i \leftarrow \mathcal{P}(\varsigma_i(0), \varsigma_i(1))$
11: **Fin Si**
12: **Fin Pour**
13: $\mathcal{GAIN} \leftarrow \mathcal{L}(\gamma) - \mathcal{L}(\bigcup_{i=1}^{3} \varsigma_i)/\mathcal{L}(\gamma)$
14: $\gamma \leftarrow \bigcup_{i=1}^{3} \varsigma_i$
15: **Fin Tant Que**

Où $\mathcal{P}(a, b)$ est le chemin local connectant les deux configurations a et b, $\mathcal{L}(\gamma)$ est la longueur du chemin γ.

Par ailleurs, l'absence de cycles dans le graphe représente un handicap pour les planificateurs proba-

bilistes dans l'obtention d'un chemin optimal. En effet, l'algorithme de lissage présenté ci-dessus vise la minimisation de la longueur du chemin solution mais ne garantit pas l'obtention d'un chemin optimal. Des travaux récents portant sur la modification de la phase d'apprentissage pour qu'elle fournisse un graphe capturant les différentes classes d'homotopies ont été présentés dans [Schmitzberger 02].

1.2.2 Variantes

De nombreuses variantes et extensions de ce schéma général ont par la suite été proposées afin de capturer plus efficacement la connectivité de l'espace libre. Certains auteurs ont mis en avant l'inconvénient de la stratégie d'échantillonnage uniforme de la méthode classique et ont présenté d'autres stratégies permettant une résolution plus efficace de problèmes difficiles, en particulier lorsqu'il s'agit de capturer la connectivité de régions libres de CS reliées par un passage étroit. D'autres ont travaillé sur la minimisation de la taille de graphes afin d'accélérer certaines phases ou de rajouter d'autres paramètres de contrôle de l'algorithme générique. L'objet de cette section est d'exposer les différentes extensions ainsi que d'autres variantes des méthodes probabilistes classiques présentées dans la littérature.

Différentes stratégies d'échantillonnages

Les méthodes probabilistes classiques (PRM et PPP) échantillonnent de façon uniforme l'espace libre. Ce type d'échantillonnage produit une couverture admissible de l'espace libre si les différents obstacles sont répartis de façon uniforme dans l'environnement. Malheureusement, ceci n'est pas toujours le cas. Dans certains cas, l'espace des configurations admissibles se compose d'espaces dégagés connectés par des passages étroits. Générer des nœuds dans ce type de passage est coûteux en terme de temps de calcul, ce qui se traduit aussi par un graphe de grande taille (un grand nombre de nœuds générés). De plus, la probabilité de générer une configuration dans un passage étroit, dans un temps petit (raisonnable), est quasiment nulle. De nombreux auteurs ont suggéré différentes méthodes permettant d'obtenir efficacement une couverture admissible des zones dites *"difficiles"*.

Une première approche [Kavraki 96], concentre l'échantillonnage aléatoire autour des composantes isolées du graphe afin de capturer plus efficacement les passages étroits. L'idée principale de cet algorithme est de garder le nombre d'échecs du planificateur local lors de la connexion entre certains nœuds. Quand ce nombre devient grand pour un nœud donné, ceci signifie que ce dernier se trouve dans une zone *"difficile"* d'accès. Une exploration de cette zone est alors favorisée.

Une autre approche, présentée dans [Švestka 97b, Amato 98], consiste à ajouter au graphe des nœuds générés en contact avec les obstacles. Elle repose sur une analyse explicite de contacts élémentaires afin de pouvoir échantillonner l'espace de contact. Dans [Hsu 98, Wilmarth 99a, Wilmarth 99b] la méthode exploite des calculs de distance de pénétration pour calculer en premier temps un graphe dont certains nœuds sont en collision. En effet, seules de faibles pénétrations sont permises. Le graphe

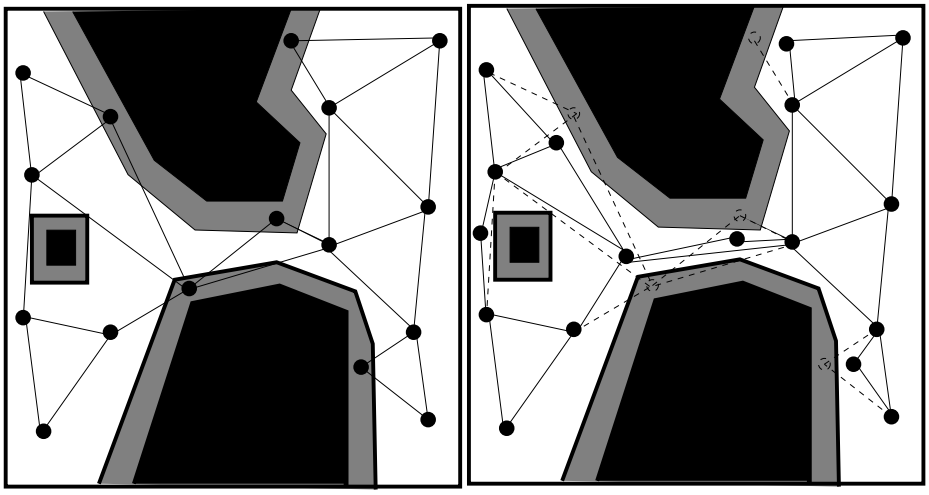


FIG. 1.2 – a) Graphe calculé en permettant une distance de pénétration dans les obstacles. b) Modification du graphe en repoussant les configurations non-valides.

est ensuite modifié en repoussant progressivement les configurations en collision vers l'espace libre (voir Figure 1.2). D'autres méthodes se basant sur le même principe ont été proposées dans la littérature [Ferbach 94, Ferbach 96]. Ce type de méthodes demande une étude géométrique complexe sur les obstacles. Elles sont de plus coûteuses en terme de temps de calcul car elles nécessitent un calcul des distances de pénétration. Elles demandent également un réglage de certains paramètres relatifs aux problèmes traités. Elles sont généralement limitées à des robots de structure simple (free-flying).

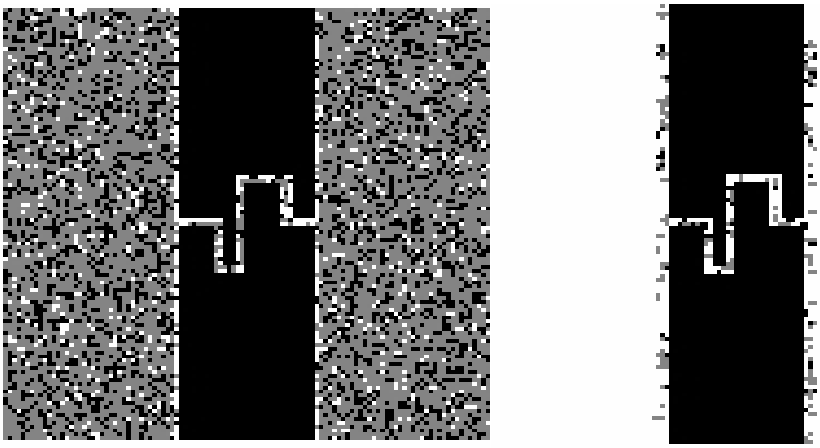


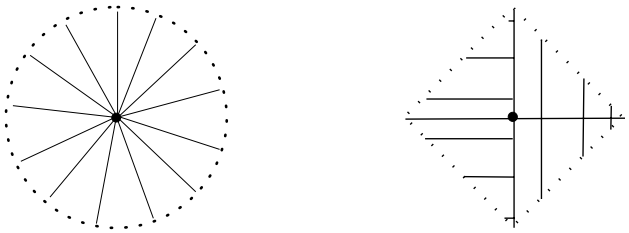
FIG. 1.3 – a) Échantillonnage Uniforme. b) Échantillonnage Gaussien.

Une méthode contournant la complexité de cette étude géométrique est la technique d'échantillonnage gaussien présentée dans [Boor 99] . La technique proposée tire aléatoirement des paires de configurations séparées par une distance selon une distribution gaussienne et ne conserve un échantillon situé dans l'espace libre que lorsque le second échantillon est en collision. Généralement, les régions

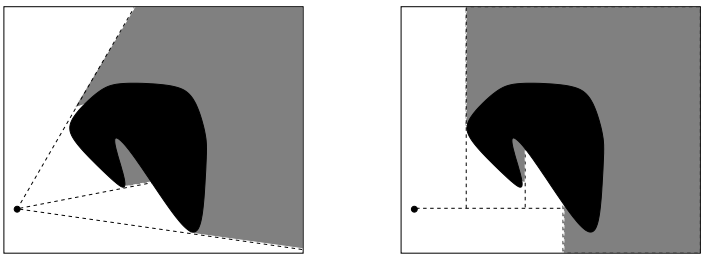
les plus difficiles à capturer sont les régions situées entre au moins deux obstacles, celles qui forment des passages étroits (figure 1.3). Cependant, l'échantillonnage gaussien génère des configurations le long des obstacles et pas seulement dans les passages étroits. Afin de remédier à cela, un tirage de trois configurations, au lieu d'uniquement deux, distantes de d selon une distribution gaussienne peut être envisagé. Dans ce cas, une configuration libre est conservée si et seulement si les deux autres sont interdites (en collision). Cette méthode d'échantillonnage gaussien est efficace pour des environnements présentant des passages étroits. Cependant, elle est très sensible au réglage des paramétres de la distribution gauussienne. L'utilisation de cette méthode est généralement limitée au robot de type *"free-flying"*.

Taille du graphe

D'autres approches visant à minimiser soit la taille du graphe construit lors de la phase d'apprentissage, soit le nombre d'appels au détecteur de collision lors de la même phase, ont été proposées dans la littérature.



a : Espace atteignable avec des chemins de longueurs fixes.



b : Espace visible pour une configuration en présence d'obstacle (les domaines grisés ne sont pas atteignables).

FIG. 1.4 – Domaines de visibilité d'une configuration pour la méthode locale Linear (gauche) et Manhattan (droite).

La première approche, appelée *"Visibility PRM"* et présentée dans [Nissoux 99a, Nissoux 99b, Siméon 00, Laumond 00], combine l'échantillonnage aléatoire avec une notion de visibilité. Cette technique permet de réduire la complexité du graphe et de contrôler sa construction. L'idée est de guider la couverture aléatoire de l'espace de configurations en exploitant la notion de visibilité de la méthode locale utilisée pour relier deux configurations. Pour un même robot et un même environnement, la forme des domaines de visibilité dépend fortement de la méthode locale choisie [Laumond 00]. Une

configuration q_1 est dite visible d'une configuration donnée q_2 par une méthode locale $\mathcal{L}$ si et seulement si le chemin local $\mathcal{L}(q_1, q_2)$ est sans collision ($\subset CS_{libre}$). Le domaine de visibilité d'une configuration q, pour une méthode locale $\mathcal{L}$ donnée, est l'ensemble des configurations visibles de q (noté $\mathcal{V}is_{\mathcal{L}}(q)$) :

$$\mathcal{V}is_{\mathcal{L}}(q) = \{q_i \in CS_{libre} / \mathcal{L}(q, q_i) \subset CS_{libre}\}. \tag{1.1}$$

La figure 1.4 montre les domaines de visibilité d'une configuration pour deux méthodes locales différentes : Linear et Manhattan. La première méthode locale relie deux configurations par un segment de droite dans l'espace des configurations. La seconde consiste à ne faire évoluer qu'un degré de liberté à la fois. Elle construit donc un chemin composé d'un segment horizontal et d'un autre vertical.

Algorithme 3 MRP-visi

1: $ntry \leftarrow 0$
2: **Tant Que** $(ntry < M)$ **Faire**
3: Tirer au hasard une configuration libre q
4: $guard \leftarrow \emptyset$; $nb_{connect} \leftarrow 0$; $G_{vis} \leftarrow \emptyset$
5: $found \leftarrow FAUX$
6: **Répéter**
7: **Pour Toute** composante C_i du graphe G **Faire**
8: **Pour Tout** nœud N_j de C_i **Faire**
9: **Si** $(q \in \mathcal{V}is_{\mathcal{L}}(N_j))$ **Alors**
10: $found \leftarrow VRAI$
11: $guard \leftarrow guard \cup N_j$; $nb_{connect} \leftarrow nb_{connect} + 1$; $G_{vis} \leftarrow G_{vis} \cup C_i$
12: **Fin Si**
13: **Fin Pour**
14: **Fin Pour**
15: **Jusqu'à** $(found = VRAI)$
16: **Si** $(nb_{connect} = 0)$ **Alors**
17: Créer la nouvelle composante connexe $\{q\}$; $ntry \leftarrow 0$
18: **Sinon**
19: **Si** $(nb_{connect} > 1)$ **Alors**
20: Créer les arêtes (q, N) pour tout N de $guard$
21: Fusionner toutes les composantes de G_{vis} et ajouter q à cette fusion
22: **Sinon**
23: $ntry \leftarrow ntry + 1$
24: **Fin Si**
25: **Fin Si**
26: **Fin Tant Que**

L'algorithme probabiliste (*MRP-visi*) présenté dans ces travaux génère itérativement deux types de

nœuds : les gardiens et les connecteurs. Les gardiens sont des nœuds qui ne se voient pas mutuellement. Leurs domaines de visibilité recouvrent l'espace libre. Les connecteurs sont des nœuds visibles par au moins deux gardiens. Ils permettent d'établir les connexions dans le graphe. Ils sont situés dans l'intersection des domaines de visibilité des gardiens qu'ils connectent. Ainsi, au lieu de garder toutes les configurations sans collision, l'algorithme utilise un pavage implicite de l'espace libre par ces domaines de visibilité pour ne retenir que des configurations qui améliorent la couverture de l'espace libre ou la connectivité du graphe. Par ailleurs, le domaine de visibilité d'un nœud n'est pas explicitement calculé. Le test d'appartenance d'une configuration au domaine de visibilité d'un nœud repose sur le succès ou l'échec de la méthode locale à relier par un chemin sans collision cette configuration à ce nœud.

Un paramètre $ntry$, important de l'algorithme, comptabilise le nombre d'échecs avant l'insertion d'un nouveau gardien. Le rapport $1/ntry$ donne une estimation de la fraction du volume non couvert de CS_{libre} sur son volume total. Ce paramètre représente l'un des points forts de cette approche. En effet, il permet de définir un critère d'arrêt explicite de l'algorithme dans sa phase d'apprentissage, critère absent des approches précédentes. En outre, il permet d'estimer la qualité du graphe en terme de couverture de l'espace libre.

La figure 1.5 montre deux graphes calculés à partir d'une même liste de configurations aléatoires. Le graphe de visibilité (figure de droite) est nettement plus compact que celui généré par la méthode probabiliste classique. La taille réduite du graphe est un second avantage pour cette approche, elle conduit à un gain significatif en temps de calcul. Il est à noter que cette taille dépend fortement de la méthode locale utilisée ainsi que de l'espace de configuration libre.

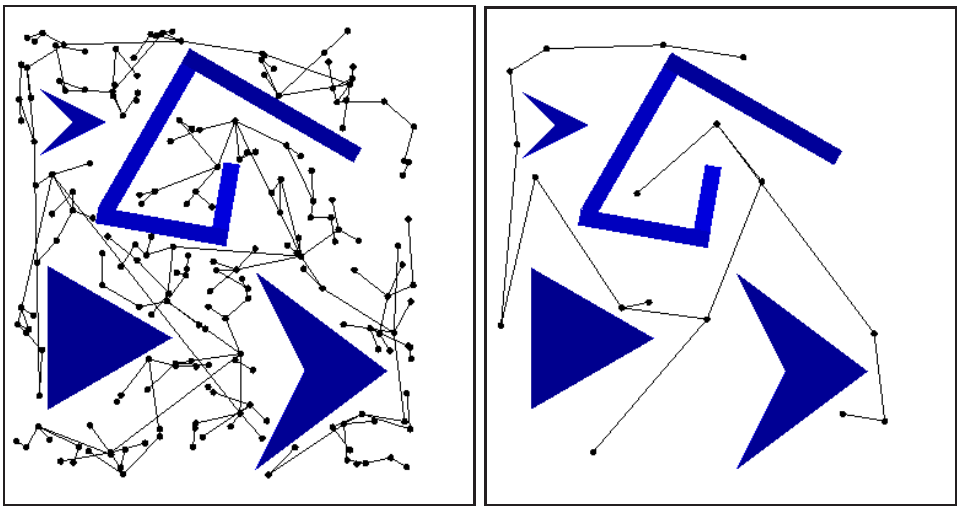


FIG. 1.5 – Deux graphes issus du même échantillonnage : PRM classique à droite et visibilité PRM à gauche.

Des tests de connexions plus efficaces :

Une autre variante des méthodes probabilistes est celle présentée par Bohlin et Kavraki dans [Bohlin 00]. Les auteurs proposent une méthode probabiliste qui minimise les appels au détecteur de collision afin de réduire le temps de calcul. Ils introduisent un mécanisme d'évaluation paresseuse lors de la vérification de la validité des chemins calculés par la méthode locale. Lors de la phase d'apprentissage, contrairement aux méthodes probabilistes classiques, le test de collision n'est vérifié qu'au niveau des nœuds. Ce n'est que lors de la phase de recherche que les arêtes sont testées sans-collision. Lors de cette phase, afin de répondre à une requête de planification, une recherche du chemin le plus court est lancée dans le graphe. Lorsque ce chemin existe, la validité (sans-collision) des arêtes le formant est testée. Dans le cas où le chemin trouvé est en collision, les arêtes non valides sont éliminées du graphe. Le graphe est ensuite enrichi par des nouveaux nœuds générés essentiellement dans la zone des arêtes supprimées. L'idée guidant cette méthode est que lors de la phase de recherche, seule une partie du graphe est explorée afin de répondre à une requête. De plus, la probabilité qu'une arête soit sans-collision est grande si les deux nœuds qu'elle connecte sont sans-collision, et si le chemin est court. Ainsi, lors de la phase d'apprentissage, les nœuds sont générés uniformément éloignés d'une petite distance afin d'augmenter la probabilité de validité des arêtes les connectant. Il est à noter que le même graphe est utilisé pour répondre aux différentes requêtes de planification de mouvement dans le même environnement. Plus le nombre de requêtes est grand, plus la probabilité que les arêtes formant le graphe soient sans-collision est grande.

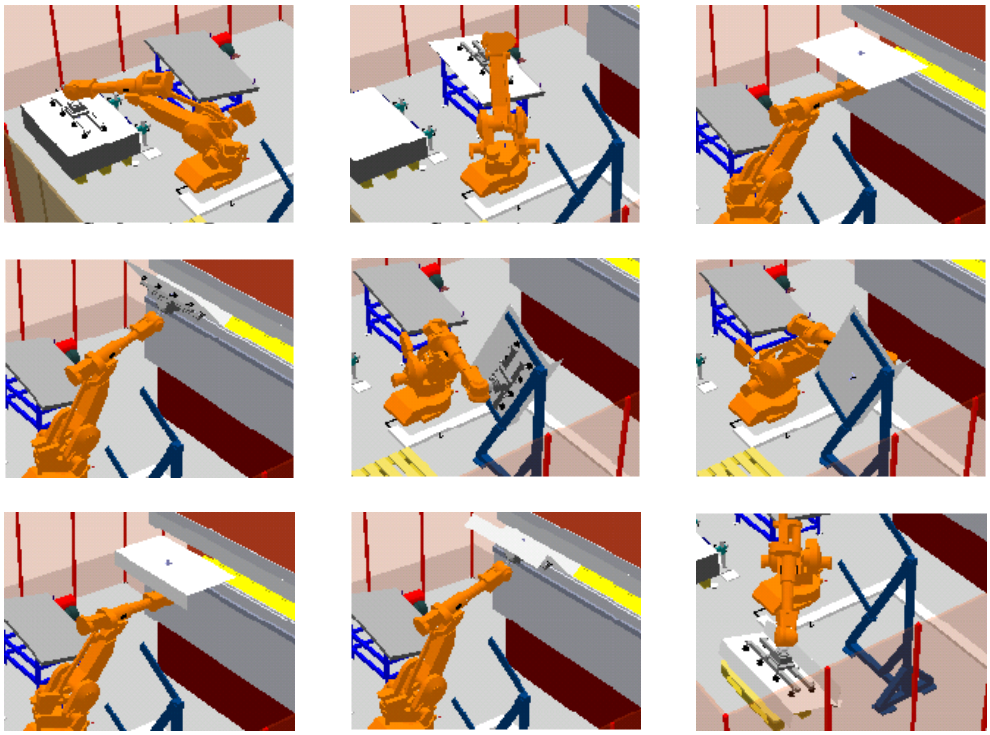


FIG. 1.6 – Planification pour un bras manipulateur tiré de [Bohlin 00].

Cette approche a été également appliquée à des problèmes de manipulation d'objet comme illustré dans la figure 1.6. Cependant, la décomposition du problème de manipulation en sous-tâches est considérée comme une donnée du problème.

Les méthodes probabilistes dites par *'diffusion'*

Les méthodes décrites dans cette section reposent, elles aussi, sur le même principe des méthodes probabilistes classiques. Cependant, le graphe construit dépend fortement des configurations initiale et but du problème de planification de mouvement posé. Pour le même environnement, le même robot et la même méthode locale, ces méthodes ne peuvent pas utiliser le même graphe construit lors d'une requête préalable pour résoudre un autre problème de planification.

La première, *RPP* (*Randomized Path Planner*) [Barraquand 91], construit un champ de potentiel sur l'espace des configurations dont l'unique minimum global est la configuration finale. La recherche combine alors deux types de mouvements : la descente le long du gradient du potentiel pour s'approcher du but et des mouvements aléatoires pour sortir des éventuels minima locaux.

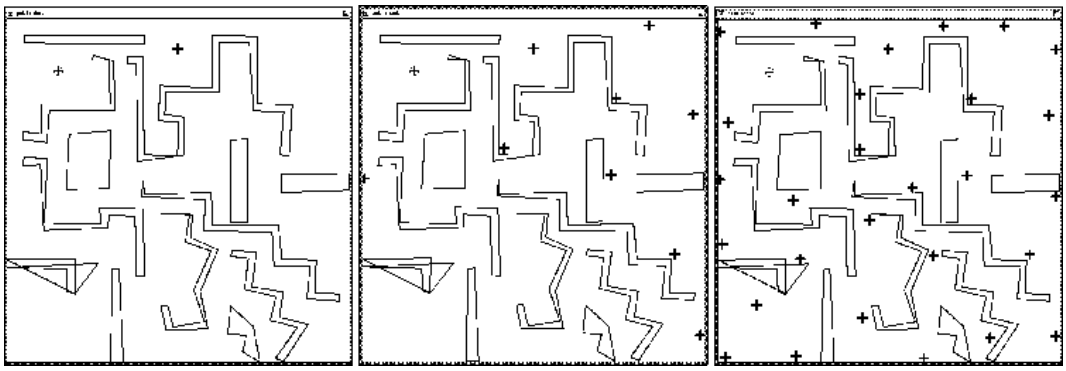


FIG. 1.7 – Évolution des balises dans l'espace libre tiré de [Ahuactzin 94].

L'algorithme *"Fil d'Ariane"*, décrit dans [Bessière 93, Ahuactzin 94, Mazer 98], utilise les algorithmes génétiques pour résoudre des problèmes de planification de trajectoire [Ahuactzin 92]. Les auteurs ne se placent pas, comme dans la plupart des planificateurs, dans l'espace des configurations mais plutôt dans l'espace des commandes. Ils définissent un plan comme un ensemble de commandes pour les actionneurs. Pour un robot holonome, un plan peut être défini par une séquence sous la forme $(\theta_1, l_1, \theta_2, l_2, ..., \theta_m, l_m)$, qui est interprétée comme : tourner de θ_1 puis avancer de l_1, tourner de θ_2 puis avancer de l_2, etc...

Le problème de planification de mouvement est ainsi transformé en un problème d'optimisation d'une fonctionnelle [1] sur cet espace de plans. La nature du problème ainsi que la méthode d'optimi-

[1] $F(\hat{x}) = \|\hat{q}_f - \hat{q}_{act}\|$, où $\hat{x}$ est le plan à trouver, $\hat{q}_f$ la configuration finale et $\hat{q}_{act}$ est la configuration actuelle.

sation utilisée peuvent engendrer des minima locaux différents de la configuration but. Afin de palier ce problème, les auteurs construisent un arbre de plans permettant l'exploration de tout l'espace libre. Cet arbre est obtenu en posant des balises dans l'espace accessible du robot. Du point de vue implémentation, l'algorithme *"Fil d'Ariane"* est composé de deux sous algorithmes *SEARCH* et *EXPLORE*. Le premier collecte des informations sur l'espace accessible depuis la position initiale en posant des balises dans l'espace de recherche et en mémorisant les chemins entre ces balises et la position initiale. Le second essaie de placer de nouvelles balises aussi loin que possible de celles déjà placées. La figure 1.7 illustre l'évolution des balises placées par l'algorithme.

Inspirés de la technique de *marches aléatoires* introduite dans [Motwani 95], Hsu et al. ont introduit dans [Hsu 97] une approche permettant de résoudre efficacement des problèmes de planification de mouvement lorsque l'espace des configurations est hautement dimensionné. L'idée principale de cette approche est de limiter la recherche dans un sous-espace significatif (judicieux selon le problème à résoudre) au lieu de lancer une exploration globale de l'espace des configurations. L'algorithme génère par échantillonnage aléatoire des configurations voisines de q_{init} et de q_{but} et répète itérativement cet échantillonnage dans le voisinage des configurations déjà connectées à q_{init} et q_{but} jusqu'à ce qu'un chemin soit trouvé.

La méthode RRT basique (*"Rapidly-exploring Random Trees"*), décrite dans [LaValle 98, LaValle 99a], représente un intermédiaire entre les méthodes probabilistes classiques et l'approche décrite dans le paragraphe précédent ([Hsu 97]). Elle consiste à faire évoluer un arbre partant de la configuration initiale afin d'atteindre la configuration but. A chaque itération, un nouveau nœud est aléatoirement généré et rajouté à l'arbre s'il n'est pas en collision. La figure 1.8 illustre la construction incrémentale d'un arbre par cette méthode. Le point au centre représente la configuration initiale.

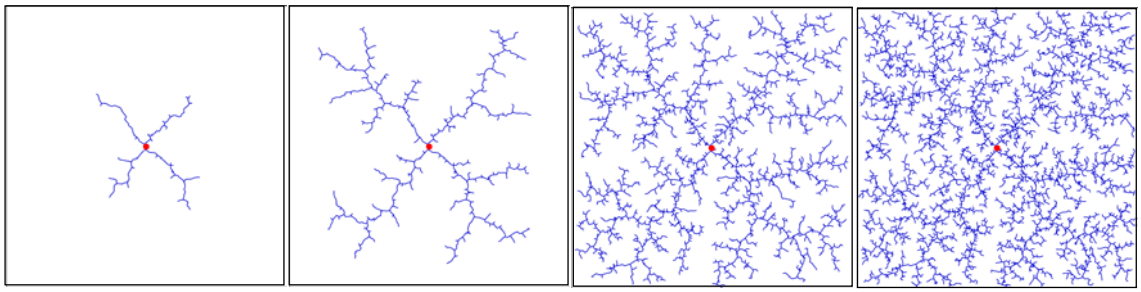


FIG. 1.8 – Évolution d'un arbre couvrant l'espace libre par la méthode RRT.

Plusieurs extensions et variantes de cette méthode ont été présentées dans [Kuffner 00d, LaValle 00b, LaValle 01]. Parmi ces variantes, l'algorithme *RRT-bidirectionnel* qui consiste à faire évoluer deux arbres à la fois ayant comme racine respective la configuration initiale et but. Les deux arbres progressent l'un vers l'autre selon une heuristique simple. La figure 1.9 montre le chemin solution ainsi

que l'arbre construit par cette variante pour un robot en forme de *'L'* dans un environnement 2D.

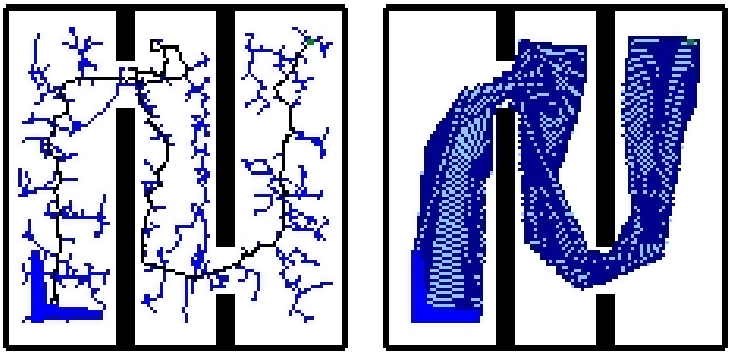


FIG. 1.9 – Exemple de problème de planification de mouvement résolu par RRT.

La méthode RRT ainsi que ses variantes ont montré des performances meilleures que les méthodes probabilistes classiques dans des cas contraints : lorsqu'il s'agit de connecter deux configurations dans un passage étroit. Elles sont généralement utilisées pour résoudre des problèmes d'assemblages mécaniques où la distance aux obstacles reste très faible autour de la trajectoire solution. Cependant, elles ne nécessitent pas la construction préalable d'un graphe. Leurs utilisations se limitent à fournir une réponse à une seule requête de planification de mouvement.

1.2.3 Quelques propriétés des méthodes probabilistes

Bien que les méthodes probabilistes aient trouvé un large succès dans la littérature ces dernières années, l'analyse de leur comportement reste limitée à quelques articles. Leur caractère aléatoire rendait cette analyse difficile. Cependant, certains résultats concernant la complétude de la méthode ainsi que la qualité du graphe en terme de couverture et de connexité ont été établis récemment.

Complétude probabiliste

Une méthode de planification de mouvement est dite complète en probabilité si la probabilité que la méthode trouve une solution (si elle existe) au problème posé tend vers 1 lorsque le temps de calcul tend vers l'infini. Si, de plus, l'espace des configurations admissibles est borné, le robot est localement commandable alors, si la méthode locale vérifie la propriété topologique générale, la méthode (PRM ou PPP) est complète en probabilité. La complétude en probabilité des méthodes probabilistes (PRM ou PPP) a été démontrée dans [Švestka 96] et [Švestka 97a]. Les propriétés ainsi que les théorèmes établis dans cette section concernent les méthodes probabilistes classiques (PPP et PRM).

Définition 1 *ε-accessibilité*
Soient $\mathcal{R}$ un robot localement commandable, $\mathcal{L}$ la méthode locale utilisée et $B_\varepsilon(q)$ la boule de rayon ε

et centrée en une configuration q de CS, le domaine ε-accessible *de q par $\mathcal{L}$ noté $\mathcal{R}_{\mathcal{L},\varepsilon}(q)$ est défini comme suit :*

$$\mathcal{R}_{\mathcal{L},\varepsilon}(q) = \{\tilde{q} \in B_\varepsilon(q) \mid \mathcal{L}(q,\tilde{q}) \subset B_\varepsilon(q)\}$$

Définition 2 *Propriété Topologique Locale Générale (TLG)*
Une méthode locale $\mathcal{L}$ pour un robot localement commandable $\mathcal{R}$ vérifie la propriété topologique locale générale si et seulement si :

$$\forall \varepsilon > 0,\ \exists\ \delta > 0 \ \ tel\ \ que\text{:}\ \ \forall\ q \in CS,\ B_\delta(q) \subset \mathcal{R}_{\mathcal{L},\varepsilon}(q)$$

L'existence d'une méthode locale vérifiant la propriété topologique générale est toujours garantie, au moins en théorie, par la commandabilité locale du robot. Cette propriété peut s'expliquer par le fait qu'à chaque configuration de CS_{libre}, il existe un voisinage dans lequel cette configuration peut se déplacer sans s'éloigner de ε de la configuration initiale. La figure 1.10 illustre un exemple de méthode locale vérifiant cette propriété. Le théorème ci-dessous montre que cette propriété est suffisante pour garantir la complétude en probabilité de la méthode probabiliste.

Théorème 1 *Complétude en probabilité des méthodes probabilistes [Švestka 96]*
Si $\mathcal{L}$ est une méthode locale vérifiant la propriété TLG, alors la méthode probabiliste utilisant $\mathcal{L}$ est complète en probabilité.

La démonstration de ce théorème a été présentée dans [Švestka 96]. La planification de mouvement pour des robots de type voiture, utilisant des méthodes locales vérifiant la propriété TLG a été également étudiée dans [Laumond 90].

Dans le cas d'une méthode locale symétrique et un robot localement commandable, une autre propriété moins contraignante que la TLG a été introduite dans [Švestka 96], garantissant la complétude en probabilité de la méthode probabiliste. Cette propriété présente l'avantage d'être facile à vérifier pour une méthode locale. Elle est appelée *propriété topologique locale lipschitzienne* (ou TLL).

Définition 3 *Propriété Topologique Locale Lipschitzienne (TLL)*
Une méthode locale symétrique $\mathcal{L}$ pour un robot localement commandable $\mathcal{R}$ vérifie la propriété TLL si et seulement si :

$$\forall \varepsilon > 0,\ \exists\ \delta_\varepsilon > 0 \ et\ r_\varepsilon : CS \to B_{\delta_\varepsilon}\ une\ fonction\ continue\ lipschitzienne\ ^2\ tels\ que :$$
$$\forall\ q \in CS,\ r_\varepsilon(q) \subset \mathcal{R}_{\mathcal{L},\varepsilon}(q)$$

[2] $f : A \to B$ est dite continue lipschitzienne si : $\exists$ une constante $d \mid \forall\ (a,b) \in A \times A : |f(a) - f(b)| \leq d \times |a - b|$

Contrairement à la TLG, cette propriété ne nécessite plus que la boule B_δ, continue dans le domaine ε-accessible de q, soit centrée en q. La relaxation de cette contrainte permet en quelque sorte une *"marge de manoeuvre"* plus grande pour la méthode locale $\mathcal{L}$. Une simple comparaison entre la figure 1.10 et 1.11, nous permet de mieux assimiler cette relaxation. La propriété TLL garantit également la complétude en probabilité de la méthode probabiliste utilisant une méthode locale symétrique et vérifiant cette propriété.

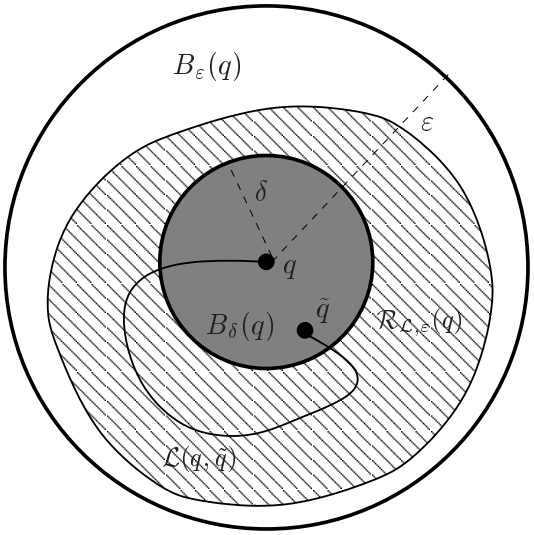


FIG. 1.10 – Une méthode locale vérifiant la propriété topologique générale.

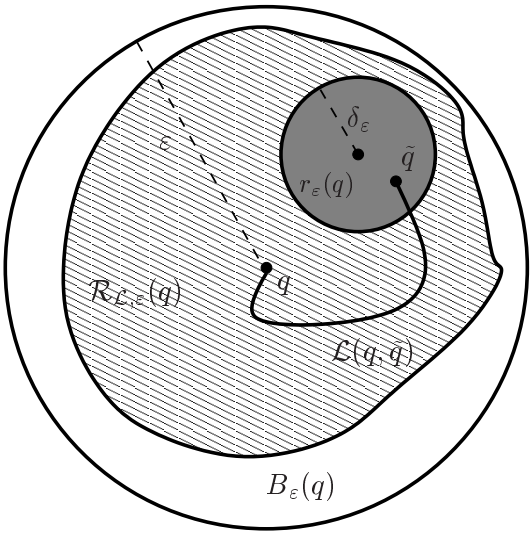


FIG. 1.11 – Une méthode locale vérifiant la propriété topologique locale lipschitzienne.

Théorème 2 *Complétude en probabilité des méthodes probabilistes [Švestka 96]*
Si $\mathcal{L}$ est une méthode locale symétrique vérifiant la propriété TLL, alors la méthode probabiliste utilisant $\mathcal{L}$ est complète en probabilité.

La complétude de la méthode RRT, décrite dans la section 1.2.2, a été également prouvée [P.Cheng 02]. Les auteurs ont suivi la même démarche que celle adoptée pour la démonstration de la complétude des méthodes probabilistes de type PRM.

Couverture du graphe : Notion de *l'ε-goodness*

La phase d'apprentissage des méthodes probabilistes classiques s'achève lorsque le nombre de nœuds générés a atteint une limite maximale fixée comme entrée du problème à résoudre. La notion d'*ε-goodness*, introduite par Kavraki et al. dans [Kavraki 95b, Kavraki 98, Barraquand 96], permet d'évaluer la qualité du graphe construit lors de cette phase en terme de couverture de l'espace libre. Notons par $V(S)$ le volume d'un ensemble S, et par $\mathcal{V}is_{\mathcal{L}}(q)$ le domaine d'accessibilité (visibilité) d'une configuration q pour une méthode locale $\mathcal{L}$.

Définition 4 *Configuration* ε-good
Soit $\varepsilon \in]0,1]$*, une configuration* q *est dite* ε-good *si :* $V(\mathcal{V}is_{\mathcal{L}}(q)) \geq \varepsilon \times V(CS_{libre})$

Ainsi, une configuration q de CS_{libre} est *ε-good* si elle *"voit"* au moins une fraction ε du volume total de CS_{libre}. L'espace libre CS_{libre} est dit *ε-good* si toute configuration de CS_{libre} est *ε-good.*

L'ensemble des nœuds du graphe fournit une couverture adéquate de l'espace libre si et seulement si le volume du sous-ensemble de CS_{libre} qui n'est visible d'aucun nœud est au plus $\frac{\varepsilon}{2}V(CS_{libre})$. Il est donc clair que plus ε est grand, plus il est facile de connecter un nouveau nœud au graphe construit lors de la phase d'apprentissage puisque chaque configuration voit une large portion de CS_{libre}. En effet, ε représente la plus petite fraction du volume de CS_{libre} visible pour n'importe quelle configuration du graphe : $\varepsilon = min\ \frac{V(\mathcal{V}is_{\mathcal{L}}(q))}{V(CS_{libre})}$.

Par ailleurs, pour un espace libre CS_{libre} vérifiant la propriété *d'ε-goodness*, si le graphe construit lors de la phase d'apprentissage comporte s nœuds alors le théorème suivant a été démontré (voir [Barraquand 96] pour la démonstration) :

Théorème 3 *Condition sur* s *nombre de nœuds du graphe* G
Soit CS_{libre} ε-good. *Soient* $\beta \in]0,1]$*, et* C *un réel positif assez grand* $\mid \forall\ x \in]0,1]$ *tel que :*
$(1-x)^{(\frac{C}{x})\log(\frac{4}{x\beta})} \leq \frac{x\beta}{4}$*, si* s *vérifie :*

$$s \geq \frac{C}{\varepsilon}(\log(\frac{1}{\varepsilon}) + log(\frac{4}{\beta}))$$

Alors la probabilité pour que le graphe construit fournit une couverture adéquate *de* CS_{libre} *est d'au moins* $1-\beta$.

D'après ce théorème, le nombre "s" de nœuds nécessaires pour obtenir une couverture adéquate de CS_{libre}, ne peut pas être explicitement calculé. Néanmoins, ce résultat montre que si une couverture adéquate n'est pas garantie par les méthodes probabilistes, la probabilité de ne pas l'obtenir décroît exponentiellement avec le nombre de nœuds engendrés.

Connexité du graphe : Notion de la β-*lookout*

Une fois la notion de couverture adéquate introduite, la connexité du graphe construit lors de la phase d'apprentissage peut être étudiée. Dans ce paragraphe, nous étudions la *"représentativité"* de ce graphe pour l'espace des configurations libres CS_{libre}.

Définition 5 *Représentation adéquate*
Soit CS_{libre} *un espace libre* ε-good*, un graphe* G *est une* **représentation adéquate** *de* CS_{libre} *si ses*

nœuds fournissent une couverture adéquate de CS_{libre} et il n'existe pas deux composantes connexes de G dans la même composante connexe de CS_{libre}

Si G est une représentation adéquate de CS_{libre} (qui est ε-*good*), chaque composante connexe de CS_{libre} a un volume inférieur à $\varepsilon V(CS_{libre})$. Ainsi, il y a au moins un nœud de G dans chaque composante de CS_{libre}. Puisque il n'y a pas deux composantes connexes de G dans la même composante connexe de CS_{libre}, on peut donc conclure qu'à chaque composante connexe de CS_{libre} correspond une composante connexe de G et une seule. G capture bien la connexité de CS_{libre}.

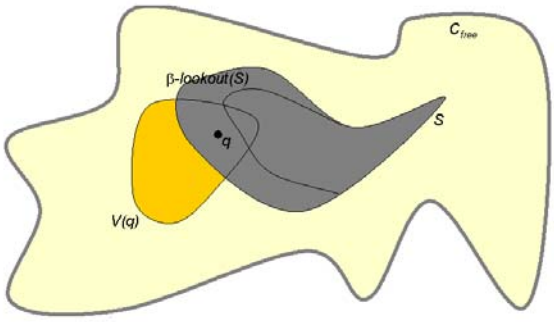


FIG. 1.12 – La β-lookout d'une configuration q.

Dans [Hsu 98], les auteurs ont introduit la notion de *"vue extérieure" (lookout)* d'un sous-ensemble S de CS_{libre} (figure 1.12) , comme étant l'ensemble des points de S pouvant voir une large portion de $CS_{libre} \backslash S$. Si cet ensemble est assez grand, il est facile de relier n'importe quel point de S à un point extérieur en générant aléatoirement des points dans S et $CS_{libre} \backslash S$.

Définition 6 *Vue extérieure d'un sous-ensemble*
Soient $\beta \in]0,1]$ une constante, et S un sous-ensemble d'une composante connexe E de CS_{libre}. La β-lookout *de S est définie comme suit :*

$$\beta\text{-lookout}(S) = \{q \in S \mid V(\mathcal{V}is_{\mathcal{L}}(q) \backslash S) \geq \beta \times V(E \backslash S)\}$$

A partir de la définition de la *"vue extérieure" (lookout)* d'un sous-espace, il est alors possible d'introduire la notion d'espace libre étendu (*expansive free space*).

Définition 7 *espace $(\varepsilon, \alpha, \beta) - Expansive$*
*Soient α, β et ε des constantes de $]0,1]$. L'espace libre CS_{libre} est dit $(\epsilon, \alpha, \beta)$-***Expansive** *si il est* ε-good, *et si $\forall\ S \subset CS_{libre}$ on a :*

$$V(\beta\text{-}Lookout(S)) \geq \alpha \times V(S)$$

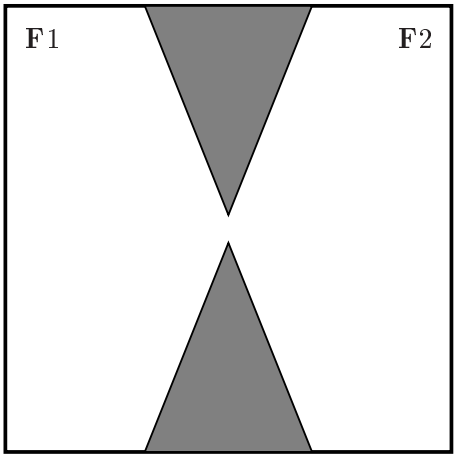


FIG. 1.13 – Un passage étroit dans un espace ε-*good*.

La notion d'espace libre étendu (*"Expansive"*) est directement liée à la difficulté d'une méthode probabiliste à relier des nœuds à travers des passages étroits. Dans le cas de l'exemple présenté dans la figure 1.13, la phase d'apprentissage conduisant à un graphe connexe est très coûteuse en terme de temps de calcul. Ceci est dû au fait que seul un petit ensemble de points de $F1$ "voit" une large fraction de $F2$. Ainsi, la probabilité pour que le planificateur génère un échantillon dans $F1$ qui "voit" un échantillon de $F2$ est petite.

Théorème 4 *Condition sur le nombre de nœuds s du graphe G*
Soient CS_{libre} un espace $(\epsilon, \alpha, \beta)$-Expansive, et $\xi \in]0, 1]$. Si s le nombre de nœuds du graphe calculé lors de la phase d'apprentissage est tel que :

$$s \geq \frac{16}{\varepsilon\alpha} log(\frac{8}{\varepsilon\alpha\xi}) + \frac{6}{\beta} + 4$$

Alors, la probabilité pour que la méthode probabiliste construit un graphe G telle qu'il n'existe pas deux composantes connexes de G se trouvant dans la même composante connexe de CS_{libre} est d'au moins $1 - \xi$.

Nous redirigeons les lecteurs vers [Hsu 98] pour la démonstration de ce théorème. Par ailleurs, ce théorème montre que la probabilité pour que la méthode probabiliste génère un graphe qui ne capture pas la connectivité de CS_{libre} décroît exponentiellement avec le nombre s de nœuds de G. De plus, le nombre de nœuds nécessaires pour obtenir une représentation adéquate de l'espace libre augmente modérément quand ε, α et β décroissent.

Il est à noter qu'aucun des théorèmes précédemment cités ne prend en considération la dimension n de l'espace libre. En effet, les définitions d'espace ε-good et espace étendu font référence uniquement aux volumes des sous-espaces S de CS_{libre}. Néanmoins, la dimension n de l'espace libre est implicitement prise en considération dans les paramètres ε, α et β.

1.2.4 Applications

Diverses applications ont accompagné le succès des méthodes probabilistes. L'application la plus simple de ces méthodes est celle de planifier le mouvement d'un objet *"free flying"* à 3 degrés de liberté (resp. 6) dans un environnement 2D (resp. 3D). Plusieurs exemples de références ou étalons ("benchmarks") ont été présentés dans la littérature afin d'évaluer les performances des différents algorithmes de planification développés.

FIG. 1.14 – Deux *"benchmarks"* avec des robots *"free-flying"*

La figure 1.14 illustre deux "*benchmarks"* de planification de mouvement. La complexité dans ces étalons réside dans la capture de la connectivité des passages étroits. L'exemple de droite (présenté par Kavraki) consiste à faire passer le robot à travers les passages étroits. Le deuxième exemple (introduit par Amato) est composé de deux tubes en forme "d'alpha" (d'où le nom de cet étalon : *alpha puzzle benchmark*). L'un des deux tubes constitue l'obstacle alors que l'autre représente le robot. Le problème de planification à résoudre consiste à trouver un chemin sans collision permettant la séparation de ces deux pièces.

Robots non-holonomes

Il y a maintenant quarante-cinq ans, Dubins [Dubins 57] a réussi à caractériser les plus courts chemins pour une particule évoluant dans le plan le long de chemins à courbures bornées parcourus à vitesse constante. Cette particule peut en fait être assimilée à un robot mobile de type voiture qui n'évoluerait qu'en marche avant. Depuis, il a fallu attendre jusqu'au début des années 90 pour résoudre le même problème dans le cas d'un robot de type voiture capable de faire également des marches arrière (Reeds et Shepp dans [Reeds 90]).

Très vite, des planificateurs de trajectoires en présence d'obstacles pour des robots non-holonomes sont apparus dans la littérature [Latombe 91a, Laumond 94]. Pour les méthodes probabilistes, il suffit de développer une méthode locale qui prend en compte les contraintes non-holonome de ce type de robot. Dans [Švestka 97a, Švestka 97b], les auteurs ont choisi d'utiliser comme méthode locale une séquence finie d'au plus trois courbes élémentaires : un arc de cercle ayant un rayon de

courbure minimale suivi par une ligne droite, et suivi par un autre arc de cercle. Ils ont montré dans [Švestka 97b], que cette méthode est assez efficace dans la résolution de problèmes de planification de mouvement pour des robots mobiles non-holonomes. La complétude des méthodes probabilistes utilisant ce type de méthode locale a été également prouvée. Par ailleurs, des extensions de ces travaux dans le cas des robots mobiles à remorque ont été présentées dans [Sekhavat 98, Lamiraux 00].

Animation graphique et acteurs virtuels

Les techniques de planification peuvent également être appliquées pour automatiser la production d'animations d'artefacts virtuels dans des logiciels d'animation graphique. Ce domaine de recherche a été propulsé par l'industrie du cinéma et du divertissement. La difficulté principale réside dans la définition de méthodes efficaces de contrôle d'acteurs virtuels et de les combiner avec un planificateur de trajectoires.

Dans [Kuffner 00a], les auteurs proposent de coupler un planificateur de trajectoires utilisant des méthodes probabilistes avec un contrôleur basé sur l'édition de capture de mouvement et garantissant un résultat réaliste. L'idée principale est de décomposer le problème initial en deux sous problèmes : le calcul d'un chemin et le suivi (réaliste) de ce chemin. Le premier sous problème est résolu en utilisant une approche probabiliste : la méthode *RRT*. Le chemin calculé est animé dans un deuxième temps lors de son exécution. L'absence de collision est garantie grâce à l'utilisation d'une forme englobante dans la phase de planification.

Dans [Pettré 02], les auteurs proposent de résoudre le problème de planification de mouvement réaliste en une seule étape. Ils utilisent une approche probabiliste de type *"Visibility-PRM"*. Les chemins formant les arêtes du graphe généré prennent en considération l'animation de l'acteur virtuel. En effet, la combinaison entre planification de mouvement et animation réaliste est assurée par la méthode locale utilisée. Cette dernière consiste à générer une trajectoire pour la racine du personnage (en utilisant des courbes de Bezier) et puis l'animer grâce à un contrôleur de marche basé sur des techniques de capture de mouvement.

La biochimie

Un autre champ d'application moins attendu est apparu très récemment suite au progrès des techniques de modélisation moléculaire. En effet, la géométrie joue un rôle important pour la compréhension de mécanismes mettant en jeu des protéines, en particulier pour la prédiction de leur structure. Les structures protéiques sont modélisées par des chaînes cinématiques pouvant atteindre plusieurs milliers de degrés de liberté [Teodoro 01]. La complexité de telles structures impose l'utilisation des méthodes probabilistes pour la planification de mouvement.

Dans [LaValle 00a, Apaydin 02], les auteurs présentent une extension des méthodes probabilistes

pour la planification de mouvement des molécules. Cette extension, appelée *SRS* ("Stochastic Roadmap Simulation"), calcule comme les méthodes probabilistes classiques un graphe. Les noeuds de ce graphe sont tirés aléatoirement et représentent une conformation de la molécule. Chaque arête, connectant un noeud i à un noeud j, est annotée par la probabilité de la molécule de passer de l'état ou de la configuration i à la configuration j. Cette probabilité dépend des niveaux d'énergie des configurations i et j. Une approche similaire a été présentée dans [Amato 02].

Autres applications

Les méthodes probabilistes ont été appliquées avec succès dans plusieurs autres domaines. Ces applications incluent la planification de mouvement pour les objets déformables [Holleman 98, Anshelevich 00, Lamiraux 01], la planification pour des chaînes cinématiques fermées [Yakey 99, Han 00, Cortés 02] et la planification de mouvement en présence de zones dangereuses à éviter [Sent 01]. Dans tous ces cas, il a suffit de trouver une représentation adéquate de l'espace de configuration et de construire une méthode locale répondant aux contraintes cinématiques et mécaniques du système. Ceci montre la souplesse ainsi que la généricité des méthodes probabilistes.

1.2.5 Vers des approches déterministes

Le succès des méthodes probabilistes dans la planification de mouvement s'explique par leur généricité et par leur capacité à attaquer des problèmes de grande dimension. L'efficacité de ces méthodes reste cependant mieux constatée que comprise. La modélisation formelle du comportement en moyenne de ces algorithmes est un problème difficile.

Des travaux très récents [Branicky 01, LaValle 02] tendent à montrer que l'échantillonnage aléatoire n'est pas forcément la stratégie optimale d'échantillonnage. L'idée est de combiner l'aléatoire avec des techniques déterministes, ou plutôt d'utiliser le "quasi-aléatoire". En effet, les générateurs de nombres aléatoires produisent une séquence déterministe de nombres calculés de manière à satisfaire un critère de densité de probabilité uniforme. Cette séquence de nombres déterministes ou plus exactement pseudo-aléatoire sera utilisée pour résoudre une tâche spécifique de planification, autant alors définir une séquence déterministe permettant de résoudre cette tâche plus efficacement. D'où l'idée explorée dans [Branicky 01, LaValle 02] d'utiliser des techniques "quasi-aléatoires" dont l'efficacité a déjà été démontrée dans plusieurs domaines, en particulier celui de l'optimisation avec des méthodes de type *Quasi-Monte-Carlo*.

La figure 1.15 montre deux graphes de taille identique (même nombre de nœuds) calculés avec le même algorithme à une différence près : l'échantillonnage de l'espace libre. Le graphe de droite est obtenu en utilisant un générateur quasi-aléatoire basé sur des points de l'ensemble de Hammersley [Matousek 99]. Ces points présentent une dispersion plus faible qu'un ensemble de points aléatoires, ce qui se traduit par une meilleure couverture de l'espace libre.

FIG. 1.15 – Échantillonnage aléatoire vs. quasi-aléatoire (tiré de [Branicky 01]).

Ces résultats récents permettent d'envisager une évolution possible vers des techniques déterministes offrant les avantages des méthodes probabilistes et bénéficiant d'une forme de complétude approchée plus forte que la complétude probabiliste.

1.3 Planification de tâches de manipulation

La planification de mouvement d'un robot évoluant parmi des obstacles fixes ainsi que plusieurs variétés du problème basique ont été largement étudiées au cours des vingt dernières années. Cependant, les recherches s'adressant au problème de planification de tâches de manipulation sont assez récentes.

Le premier article traitant de la problématique de planification de tâches de manipulation est celui de Wilfong [Wilfong 88]. Il s'intéresse au problème de planification de mouvement pour un robot en présence d'objets déplaçables dans des environnements polygonaux et convexes. Le robot peut éventuellement déplacer les objets afin d'atteindre son but. Le placement final des objets déplaçables n'apparaît pas forcément de façon explicite dans la définition du problème. L'auteur considère un polygone convexe en translation dans un environnement 2D. Le robot *"prend"* un objet lorsque l'une de ses arêtes coïncide avec l'une des arêtes de l'objet. Dans le cas où le placement final des objets déplaçables n'est pas spécifié, Wilfong montre que le problème est *NP dur*[3](i.e : il peut être résolu par un algorithme de complexité en temps polynomial sur une machine non déterministe [4]). Par ailleurs, si le placement final des objets est complètement spécifié, il montre qu'il s'agit d'un problème *P-ESPACE dur* (i.e : il peut être résolu par un algorithme de complexité en espace mémoire polynomiale sur une machine non déterministe). Dans le cas particulier d'environnement polygonal avec un seul objet déplaçable, il

[3]Un problème est dit *dur* pour une classe si tous les problèmes de la classe peuvent s'y ramener par une transformation polynomiale.

[4]Une machine qui disposerait d'instructions de choix pour dédoubler le programme en programme parallèle. La complexité d'un algorithme non déterministe correspond au temps pris par un algorithme qui aurait toujours effectué le bon choix.

a montré l'existence d'un algorithme en $O(n^3 \log^2 n)$, où n est le nombre de sommets de tous les objets présents dans l'environnement.

Dans la même période, Laumond et Alami ont proposé dans [Laumond 89] un algorithme en $O(n^4)$ pour la résolution d'un problème similaire : le cas d'un robot et d'un objet circulaire évoluant dans un environnement polygonal. Le nombre de prises de l'objet est supposé infini. Dans cet article, les auteurs introduisent la notion d'espace des actionneurs comme étant un sous-espace de CS (espace des configurations du robot + objets) associé aux paramètres de configuration sur lesquels il est possible d'agir directement. Ceci constitue une première étape d'une approche plus générique proposée par les mêmes auteurs avec Siméon dans [Alami 89] et un peu plus tard dans [Alami 95]. Les grandes lignes de cette approche seront présentées dans le paragraphe suivant. Une présentation plus détaillée intégrant notre contribution fera l'objet du chapitre 2.

1.3.1 Formulation du problème

Dans [Alami 89] et [Alami 95], les auteurs proposent une approche originale pour la planification de tâches de manipulation. Cette approche (détaillée dans le chapitre 2) est basée sur une formulation géométrique du problème. Ils considèrent qu'un chemin de manipulation est un chemin sans collision contraint dans l'espace des configurations composite ($CS = CR \times CO_1 \times ... \times CO_m$, où CR est l'espace de configurations du robot et CO_i est l'espace de configurations de l'objet M_i). Deux types de contraintes sont introduites : *les contraintes de placement* et *les contraintes de mouvement* des objets déplaçables.

A partir de ces deux contraintes, deux sous-espaces de l'espace de configurations composite sont introduits : l'espace $GRASP$ et l'espace $PLACEMENT$. Le premier correspond au sous-espace dans lequel le robot se déplace solidairement avec l'objet déplaçable manipulé sans entrer en collision avec les obstacles ainsi que les autres objets. Alors que dans le second, le robot se déplace seul sans entrer en collision ni avec les obstacles, ni avec les objets déplaçables qui occupent une configuration stable. Le problème de planification de tâches de manipulation apparaît ainsi comme la recherche d'un chemin contraint dans les différentes composantes connexes du sous-espace $GRASP \cap PLACEMENT$ et entre eux. Le *graphe de manipulation* est défini comme étant le graphe dont les nœuds sont les différentes composantes connexes de $GRASP \cap PLACEMENT$. Les arêtes de ce graphe sont des chemins de *transit* et/ou de *transfert*. De façon informelle, un chemin de *transit* est un chemin calculé pour le robot seul, alors que dans un chemin de *transfert* le robot transporte l'objet manipulé. Cette formulation géométrique sera présentée de façon plus détaillée dans le chapitre 2.

Une fois les sous-espaces $GRASP$ et $PLACEMENT$ spécifiés, les auteurs se placent dans le cas d'un ensemble fini de prises et de poses pour plusieurs objets déplaçables. Ils présentent un premier algorithme permettant la résolution d'une instance du problème général, à savoir le cas d'ensemble discret de poses et de prises. L'algorithme présenté est composé de deux modules : un planificateur de tâches et

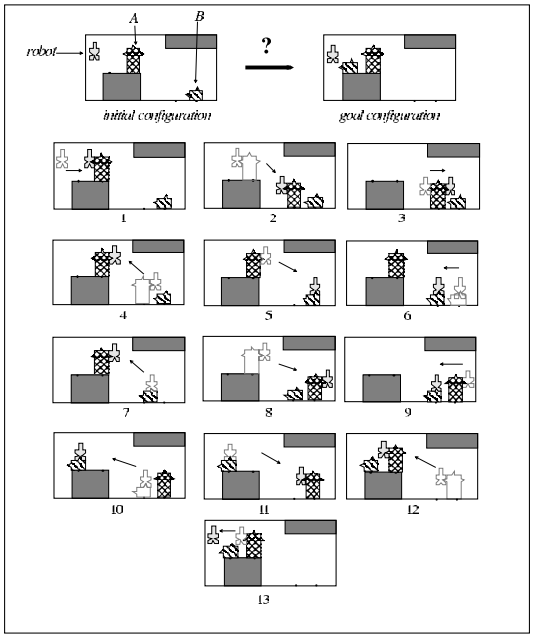


FIG. 1.16 – Un exemple de problème de manipulation résolu dans [Alami 95].

un planificateur de mouvement. Le planificateur de tâches génère les différentes composantes connexes de $GRASP \cap PLACEMENT$. Dans ce cas particulier, elles sont réduites à de simples nœuds du graphe de manipulation. Notons N_n le nombre de nœuds du graphe de manipulation, N_g le nombre de prises possibles pour l'ensemble des objets à déplacer et N_p le nombre de poses possibles de ces objets. On a alors : $N_n = N_g \times N_p$. Le planificateur de mouvement est utilisé pour connecter les différents nœuds générés par le premier module. La figure 1.16, tirée de cet article, montre un exemple de problème de manipulation résolu par cet algorithme. Dans cet exemple, à chaque objet est défini un ensemble de 4 placements stables. Le robot peut "prendre" l'objet A selon trois prises discrètes définies comme données du problème. Alors que pour l'objet B, deux prises discrètes lui sont associées.

Un deuxième algorithme est présenté dans [Alami 95] traitant un autre cas particulier du problème de manipulation. Les auteurs se placent dans le cas d'un robot polygonal en présence d'un seul objet déplaçable. Les deux acteurs de cet environnement sont des polygones convexes. La construction du graphe de manipulation s'effectue en deux étapes :

- Calculer les différentes composantes connexes du sous-espace $GRASP$.
- Connecter ces différentes composantes par des chemins de *transit*.

Le calcul des différentes composantes connexes du sous-espace $GRASP$ est basé sur une décomposition cellulaire de l'espace de configuration admissible composite. Une méthode de rétraction est ensuite appliquée sur cette décomposition afin d'obtenir une décomposition cellulaire de $GRASP$. La

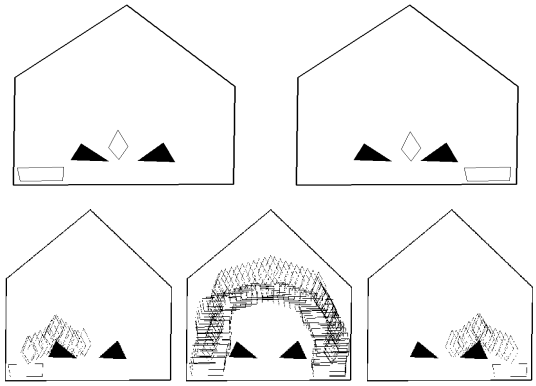


FIG. 1.17 – Un autre type de problème de manipulation traité dans [Alami 95].

connexion entre les différentes composantes connexes du sous-espace $GRASP$ par des chemins de transit est déduite après une étude de l'espace de configuration admissible du robot (l'objet déplaçable est supposé dans une configuration stable). Une fois le graphe de manipulation calculé, la recherche de chemin dans ce graphe est développée par un algorithme A^* couplé avec plusieurs heuristiques telles que la minimisation de : la longueur totale du chemin de manipulation, le nombre de prises et de poses, la longueur du chemin de l'objet déplaçable...etc. La figure 1.17 montre un exemple de problème de manipulation résolu par cet algorithme. Le chemin de manipulation a été généré en minimisant le nombre de prises et de poses de l'objet déplaçable.

1.3.2 Extension à la manipulation multi-robots

La même formulation géométrique présentée dans le paragraphe précédent a été adoptée par Koga et *al.* dans [Koga 94a], mais avec une approche un peu différente adaptée aux problèmes traités. Deux applications de cette approche à deux domaines différents ont été parallèlement développées en se basant sur cette approche : coopération de plusieurs bras manipulateurs pour effectuer une tâche de manipulation et acteurs virtuels manipulant des objets.

La coopération entre plusieurs bras manipulateurs afin d'effectuer une tâche de manipulation a été présentée dans [Koga 94a] ainsi que dans [Koga 94b]. Les auteurs proposent une méthode incrémentale pour la construction du graphe de manipulation. Ils se placent dans le cas d'un ensemble fini de prises. Ils définissent pour chaque bras manipulateur une position dite "non gênante". Un bras manipulateur qui ne participe pas au mouvement courant doit occuper cette position. La définition de l'ensemble de positions de poses stables de l'objet déplaçable dans l'environnement est légèrement adaptée au problème traité. En effet, pour changer de prise, l'un des bras manipulateurs tient l'objet alors que l'autre effectue le changement de prise. Ceci introduit une contrainte supplémentaire : un bras manipulateur effectuant une tâche de "*regrasping*" ne peut pas choisir la prise du bras tenant l'objet. L'approche proposée dans ces deux articles commence par planifier un chemin sans collision pour l'objet seul utilisant une méthode probabiliste de type *RPP*. Le chemin de l'objet est défini par

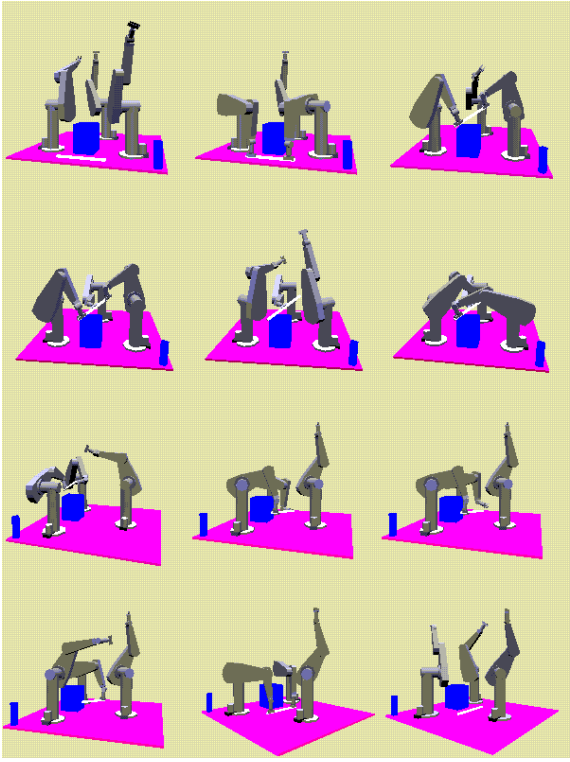


FIG. 1.18 – Un exemple de coopération entre trois bras manipulateurs de type PUMA 560 tiré de [Koga 95].

une liste de configurations adjacentes. La méthode probabiliste utilisée a été modifiée afin qu'à chaque configuration insérée dans le chemin corresponde au moins une prise parmi celles définies comme entrée du problème. Une fois le chemin de l'objet calculé, les robots suivent la trace des points de prises afin de garder la même prise le long du chemin. Quand une trace ne peut être suivie, un chemin de transit est calculé permettant ainsi le changement de prise et le suivi du chemin de l'objet. Cette approche a été étendue dans le cas de plusieurs bras manipulateurs de type PUMA 560. La figure 1.18, tirée de [Koga 95] illustre un exemple de chemin de manipulation généré par le planificateur décrit.

L'originalité de l'article [Koga 94a] réside dans la manière par laquelle les auteurs abordent le problème du modèle cinématique inverse d'un bras d'humanoïde (ou acteur virtuel). Ils se basent sur une étude neurophysiologique permettant la décomposition du problème général (complexe) en sous problèmes moins complexes et gérables. La figure 1.19, tirée de [Koga 95], montre un exemple de coopération entre un acteur virtuel et un bras manipulateur pour tourner un échiquier. La génération de ce chemin de manipulation est basée sur l'approche décrite dans le paragraphe précédent.

La deuxième application concerne la planification de tâches de manipulation pour des acteurs virtuels ou encore des humanoïdes. Dans [Kuffner 00c] et [Kuffner 00b], les auteurs proposent une méthode pour résoudre des tâches simples de manipulation pour des acteurs virtuels. Le calcul du modèle cinématique inverse utilise la méthode décrite dans [Koga 94a]. La nouveauté dans ce travail est

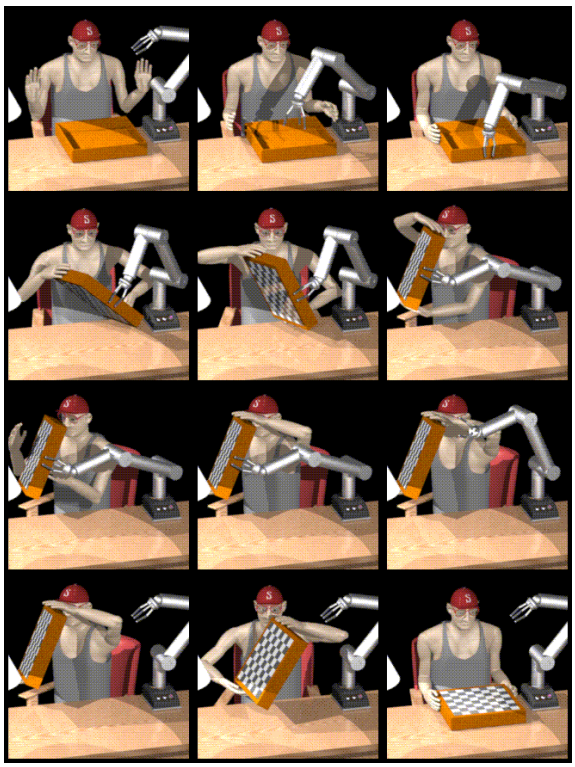


FIG. 1.19 – Un exemple d'acteur virtuel manipulant un échiquier tiré de [Koga 95].

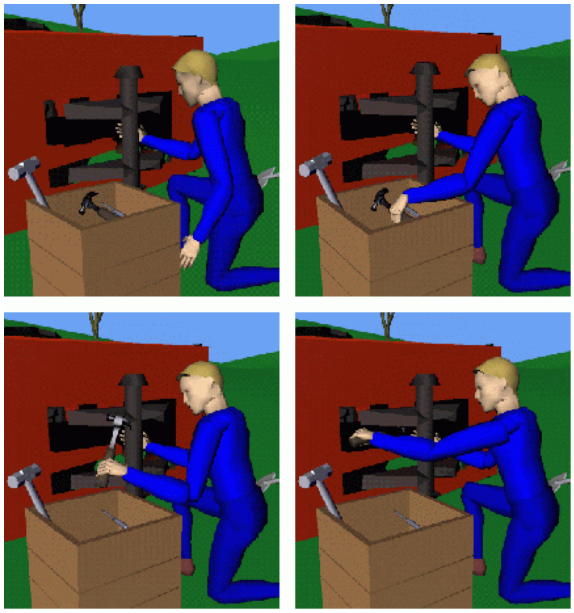


FIG. 1.20 – Un autre exemple d'acteur virtuel manipulant plusieurs objets déplaçables tiré de [Kuffner 99].

l'introduction de la perception ainsi que de la vision dans la planification d'une prise. L'humanoïde ou l'acteur virtuel vérifie, avant de planifier un mouvement de prise, si la cible est visible et s'il est possible de l'atteindre. Par ailleurs, la décomposition en sous-tâches élémentaires du problème de manipulation est considérée comme une entrée du problème. La figure 1.20, tirée de [Kuffner 99], illustre un exemple de tâches de manipulation effectuées par un acteur virtuel.

Pour conclure sur ces méthodes de planification de tâches de manipulation, il faut admettre qu'elles sont des méthodes pragmatiques qui souffrent comme toutes les méthodes heuristiques de problèmes liés à la non complétude. Cependant, elles permettent d'attaquer des problèmes réels et complexes de manipulation.

1.3.3 Extension de l'algorithme *"Fil d'Ariane"*

Dans [Ahuactzin 98] et [Ahuactzin 95], Ahuactzin et *al.* ont présenté une extension de l'algorithme du *"Fil d'Ariane"* permettant de résoudre des problèmes de planification de tâches de manipulation. Ils se sont basés sur la structuration de l'espace de recherche introduite dans [Alami 95]. La figure 1.21, tirée de [Ahuactzin 98] montre un chemin de manipulation produit par leur planificateur.

L'algorithme proposé, ACA_MANIP, comporte deux composantes : E_MANIP et S_MANIP. Le rôle de la première consiste à poser des balises dans le sous-espace $GRASP \cap PLACEMENT$ en partant

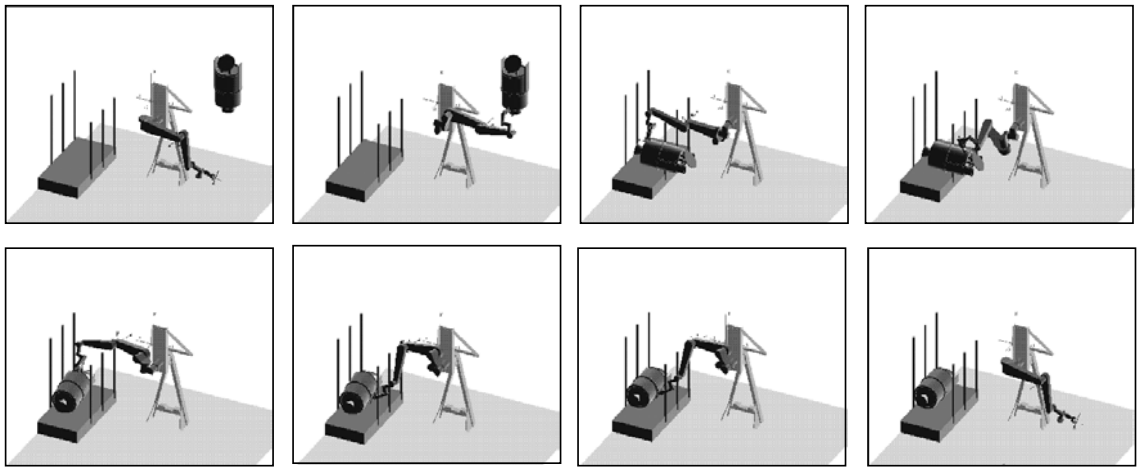


FIG. 1.21 – Exemple de problème de manipulation résolu dans [Ahuactzin 98].

de la configuration initiale. Ces balises représentent les configurations du robot (tenant l'objet) atteignables à partir de la configuration initiale et en favorisant les régions non explorées. Par construction, une nouvelle balise est atteignable d'au moins une des balises déjà placées via un chemin de manipulation. A chaque nouvelle balise placée, *S_ MANIP* teste l'existence d'un chemin de transfert connectant la balise courante à la configuration finale. En cas d'échec, *E_ MANIP* génère une nouvelle balise.
L'algorithme *S_ MANIP* est formulé comme étant un problème d'optimisation qui se présente de la façon suivante. Soit $X \subset \Re^n$, étant donnés un point initial $\hat{x_0} \in X$ et une zone but à atteindre $X_\bullet$, le problème se résume à trouver un chemin $\hat{\rho}$, s'il existe, connectant $\hat{x_0}$ à $X_\bullet$. Notons que $\hat{\rho}(0) = \hat{x_0}$ et que $\hat{\rho}(1) \in X_\bullet$. Ainsi *S_ MANIP* consiste à résoudre le problème suivant :

$$\begin{cases} Minimiser \quad c(\hat{\rho}(1), X_\bullet) \\ \hat{\rho} \in \Omega(X, \hat{x}) \end{cases} \tag{1.2}$$

où $\Omega(X, \hat{x})$ est l'ensemble des chemins dans X partant de $\hat{x} \in X$.
$c(\hat{\rho}(1), X_\bullet)$ est une fonction coût estimant la distance entre $\hat{x}$ et la région but $X_\bullet$.

Notons que pour un même environnement, changer la configuration initiale du problème de manipulation demande de reprendre toutes les itérations depuis le début. En effet, la configuration initiale sert comme point de référence pour l'algorithme *E_ MANIP*. Les différentes balises placées ainsi que les différents chemins mémorisés dépendent de cette configuration. Cet algorithme sert à répondre à une requête unique de planification de tâches de manipulation du fait que les balises construites pour cette requête ne peuvent être réutilisées pour une autre. Par ailleurs, ce planificateur traite des problèmes de planification de tâches de manipulation sous des contraintes discrètes. L'ensemble fini de prises et poses est défini par l'utilisateur. Cependant, cet algorithme a été appliqué avec succès sur

des problèmes de planification de tâches de manipulation pour des robots redondants pour lesquels la cardinalité du sous-espace $GRASP \cap PLACEMENT$ est infinie.

1.3.4 Approche par *"contraintes progressives"*

Une extension de l'approche par contraintes progressives à la planification de tâches de manipulation a été présentée dans [Barraquand 94] et [Ferbach 96]. Dans le cas d'un ensemble fini de prises et de poses , les auteurs définissent deux types de contraintes : les contraintes de prises et de non-prises. Ces contraintes sont similaires à celles introduites dans [Alami 95]. Partant de ces contraintes, ils transforment le problème de planification de tâches de manipulation en une série de sous problèmes dont les espaces libres décroissent et convergent vers celui du problème original. Les contraintes de prises sont introduites progressivement en utilisant PVDP (Progressive Variational Dynamic Programming) [Ferbach 94]. Le planificateur calcule dans un premier temps un chemin dans lequel l'objet peut léviter sans qu'il soit attaché au robot. Ce chemin est ensuite utilisé comme entrée de problèmes de plus en plus contraints dans lesquels l'objet doit se rapprocher de plus en plus du robot pour se déplacer.

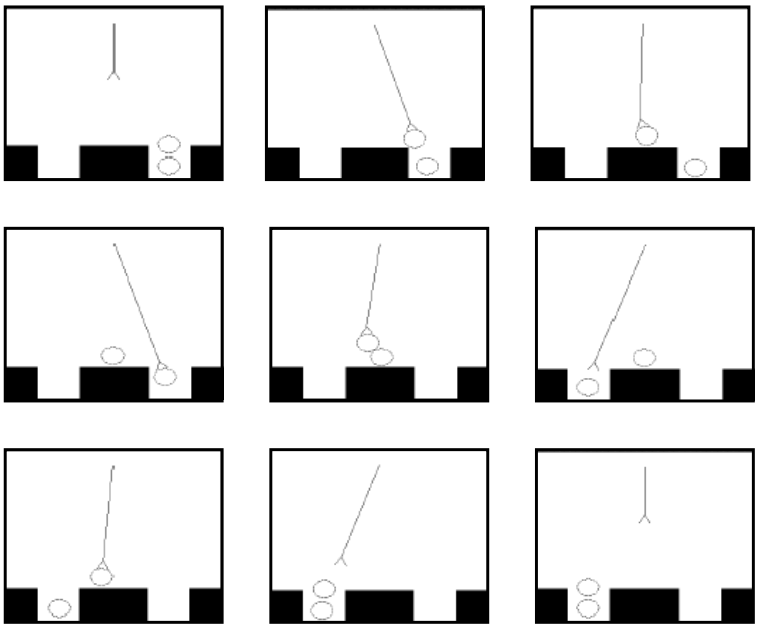


FIG. 1.22 – Un bras manipulateur à 3 degrés de liberté manipulant deux disques dans un environnement 2D (tiré de [Barraquand 94]).

La figure 1.22 illustre un exemple de problème de manipulation résolu dans [Barraquand 94] par la méthode de contraintes progressives. Les placements intermédiaires sont fournis par l'utilisateur.

1.3.5 Extension des méthodes probabilistes : *"Fuzzy PRM"*

Une autre approche présentée dans [Nielsen 00] consiste à étendre les méthodes probabilistes (*PRM*) afin de traiter les problèmes de manipulation. Dans cet article, les auteurs introduisent une nouvelle extension des méthodes probabilistes appelée *Fuzzy PRM* (ou méthode probabiliste floue). Cette nouvelle variante conduit à la construction d'un graphe dont les arêtes sont annotées par une probabilité estimant la faisabilité du chemin. Cette méthode comporte deux étapes : une phase d'apprentissage et une phase de recherche. Contrairement aux méthodes probabilistes classiques, cette variante commence par la phase de recherche. Cette phase se compose de trois étapes : une mise à jour des arêtes, une étape de recherche et une étape d'ordonnancement des arêtes selon des niveaux. La première étape consiste à rajouter les nœuds correspondant aux configurations initiale et finale (s'ils n'ont pas été déjà rajoutés) et les connecter au graphe. L'étape de recherche consiste à trouver le chemin le plus probable connectant les configurations initiale et finale. La probabilité d'un chemin ς est le produit des probabilités des arêtes constituant ce chemin : $p(\varsigma) = \prod_{i=1}^{n} p(e_i)$. La recherche de chemin utilise un algorithme de *Dijkstra* après avoir pondéré chaque arête du graphe par un poids de type : $w(e_i) = -log(p(e_i))$. Il est à noter qu'aucun test de validité de chemin (test de collision), au cours des étapes précédentes, n'a été effectué. Ce n'est que dans l'étape de l'ordonnancement des arêtes selon leur niveau que cette validation sera finalement testée. Les arêtes du graphe sont classées selon leur probabilité de non collision et un test de collision est lancé uniquement sur certaines configurations du chemin formant l'arête. Le nombre de configurations à tester dépend du niveau affecté à l'arête. La probabilité affectée à chaque arête dépend de la longueur l du chemin ainsi que du niveau (niv) qui lui est affecté :

$$p(e) = p(l, niv) = \begin{cases} [cosh(\lambda \frac{l}{2^{niv}})]^{-2^{niv}} & : \frac{l}{2^{niv}} > \varepsilon \\ 1 & : \frac{l}{2^{niv}} \leq \varepsilon \end{cases} \tag{1.3}$$

λ : Les auteurs utilisent une distribution de Poisson.

Une fois cette approche introduite, les auteurs proposent un algorithme en deux niveaux. Dans le premier niveau le problème de planification de tâches de manipulation est décomposé en plusieurs problèmes de planification de mouvement qui seront résolus au second niveau. Cette décomposition est assurée par la construction d'un graphe de manipulation utilisant l'approche *Fuzzy PRM* décrite ci-dessus. Le fait d'utiliser les méthodes probabilistes au second niveau présente un handicap pour cette approche. En effet, la complétude en probabilité de ces méthodes ne permet pas d'affirmer la non existence d'un chemin connectant deux configurations en un temps court. Afin de palier ce problème, les auteurs proposent de modifier le calcul de la probabilité marquant les arêtes en tenant compte du temps mis pour valider le chemin. La probabilité qu'une arête soit valide est exprimée de la façon suivante :

$$p(e) = 1 - \frac{t(e)}{t_global} \tag{1.4}$$

où $t(e)$ représente le temps mis par le deuxième niveau pour trouver le chemin de l'arête e, t_global est le temps total pour le second niveau du planificateur.

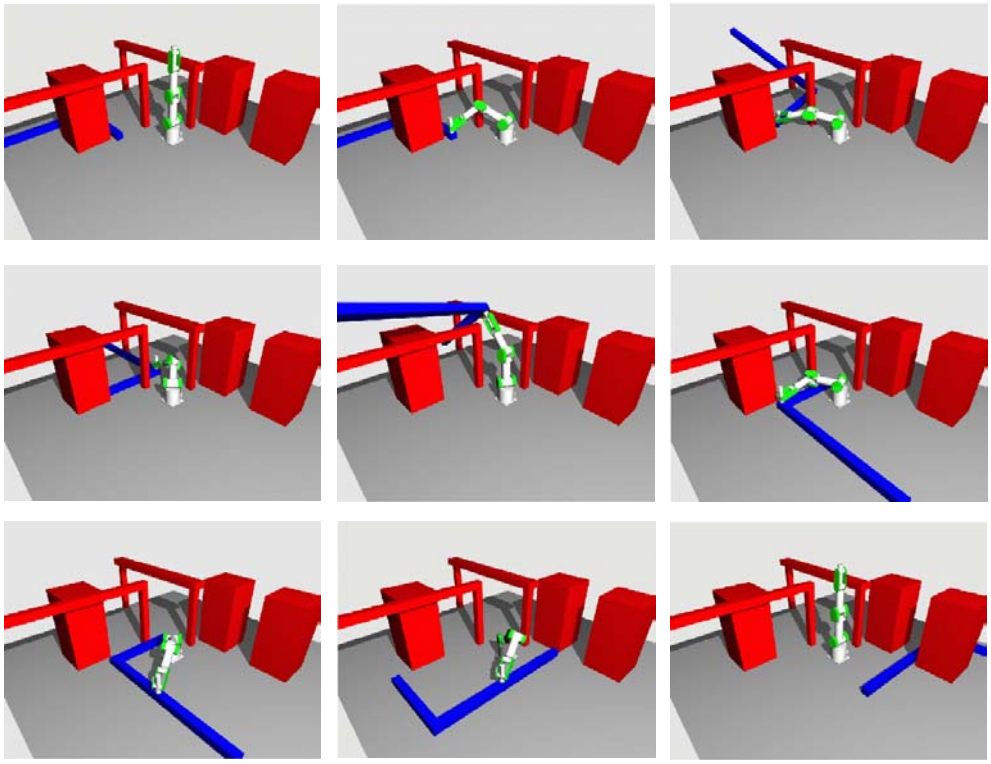


FIG. 1.23 – Bras manipulateur déplaçant un objet en forme de L [Nielsen 00].

L'approche présentée dans [Nielsen 00] ne résout qu'une instance du problème général de planification de tâches de manipulation. Les auteurs se placent dans le cas où le nombre de prises et de poses est fini. Celles-ci sont explicitement données par l'utilisateur comme entrée du problème.

1.4 Conclusion

Pour la planification de mouvement, l'apparition des approches probabilistes a permis de résoudre une large classe de problèmes difficiles. En particulier, la méthode probabiliste de type *"Visibility-PRM"* permet de capturer la connectivité de l'espace libre avec un graphe de taille réduite tout en ayant la même couverture de CS_{libre}, ce qui conduit à un gain en temps de calcul. Un autre avantage pour cette méthode est la présence d'un critère d'arrêt explicite de l'algorithme dans sa phase d'apprentissage, critère absent dans les différentes approches présentées. Ce critère permet également d'estimer la qualité du graphe en terme de couverture de l'espace libre.

D'autre part, concernant la planification de tâches de manipulation, les différents algorithmes proposés dans la littérature se limitent à des instances particulières du problème. Aucun article n'a traité le problème de manipulation sous sa forme la plus générale à savoir le cas de contraintes continues de prise et de placement. Le fait de considérer les contraintes discrètes comme une entrée du problème constitue en soi une partie de la solution fournie par l'utilisateur pour résoudre un problème de manipulation.

Notre contribution consiste à généraliser la formulation géométrique du problème de planification de tâches de manipulation présentée dans [Alami 89, Alami 95] et de l'étendre au cas de contraintes continues de prise et de pose. Cette approche est détaillée dans le chapitre suivant. Son extension au cas continu du problème de manipulation fait l'objet du chapitre 4. Dans le cas où les contraintes de prise et de pose sont discrètes, nous introduisons une extension des méthodes probabilistes basée sur la notion de *graphe dynamique*, permettant de résoudre des problèmes de manipulation dans ce cas particulier.

Chapitre 2

Formulation Géométrique et Approche Adoptée

2.1 Introduction

Le problème de planification de tâches de manipulation consiste à calculer automatiquement la séquence de mouvements permettant à un système robotique de déplacer un ou plusieurs objets vers une position finale complètement ou partiellement spécifiée. Ce problème ne se limite donc pas au niveau de la planification de mouvements sans collision pour un robot, mais il inclut également le séquencement des différentes sous-tâches de prise et de pose d'objets. L'extension des planificateurs de mouvements au problème de manipulation s'avère ainsi une tâche délicate. Par ailleurs, la plupart des planificateurs de tâches de manipulation présentés dans le chapitre précédent se limite à une instance particulière du problème général : à savoir le cas discret. La discrétisation concerne les placements stables des objets : ensemble des positions tampons dans lesquelles le robot est autorisé à placer l'un d'entre eux. La discrétisation porte également sur la description d'une prise (la manière de saisir l'objet).

Dans ce chapitre nous décrivons une formulation géométrique de ces problèmes de manipulation. Une grande partie du chapitre reprend et étend les idées introduites dans [Alami 89] (voir également le chapitre XI de [Latombe 91b]). Nous décrivons dans un premier temps les contraintes caractérisant un problème de manipulation et conduisant à la structuration ainsi qu'à la réduction de l'espace de recherche. Nous présentons ensuite l'approche adoptée pour la résolution de problème de planification de tâches de manipulation dans le cas de contraintes discrètes. Puis, nous discutons l'extension de cette approche au cas de problème de manipulation sous contraintes continues.

2.2 Structuration de l'espace de recherche

Nous nous plaçons pour la suite dans un espace de travail tridimensionnel, noté $\mathcal{W}$ contenant trois types d'objets : des obstacles statiques $(\mathcal{O}_1, ..., \mathcal{O}_k)$, un robot $\mathcal{R}$ et des objets déplaçables $(\mathcal{M}_1, ..., \mathcal{M}_r)$.

2.2.1 Préliminaires

Un robot $\mathcal{R}$ est un système mécanique constitué d'un ou plusieurs solides, reliés entre eux par des articulations. La position de $\mathcal{R}$ dans son espace de travail, appelée *configuration*, peut être représentée par un vecteur $(q_1, q_2, ..., q_n)$, où n est le nombre de degrés de liberté de $\mathcal{R}$. L'ensemble de ces positions est appelé *espace de configurations*, et noté $CS_{\mathcal{R}}$. De la même façon, nous définissons $CS_{\mathcal{M}_i}$ l'espace de configurations de l'objet déplaçable $\mathcal{M}_i$. Les objets déplaçables sont considérés comme étant des corps rigides ; $CS_{\mathcal{M}_i}$ est donc de dimension 6 (3 translations et 3 rotations).

L'espace composite de tout le système est $CS = CS_{\mathcal{R}} \times CS_{\mathcal{M}_1} \times ... \times CS_{\mathcal{M}_r}$. Une configuration de CS, appelée *configuration du système*, est de la forme $q_c = (q_{\mathcal{R}}, q_{\mathcal{M}_1}, ..., q_{\mathcal{M}_r})$, avec $q_{\mathcal{R}}$ [1] $\in CS_R$ et $q_{\mathcal{M}_i} \in CS_{\mathcal{M}_i}$. Certaines parties de l'espace de travail $\mathcal{W}$ sont occupées par des obstacles. Nous appelons *espace de configurations libres* CS_{libre} l'ensemble des configurations pour lesquelles l'intersection entre les corps du robot, les objets déplaçables et les obstacles statiques est vide. Plus formellement, l'espace de configurations libres est défini de la façon suivante :

$$\begin{aligned} CS_{libre} \;=\; & \{(q_{\mathcal{R}}, q_{\mathcal{M}_1}, ..., q_{\mathcal{M}_r}) \in CS \mid \forall\, i \in \{1, ..., k\},\ \forall\, j, l \in \{1, ..., r\},\ j \neq l,\ \mathcal{R}(q_{\mathcal{R}}) \cap \mathcal{O}_i = \emptyset, \\ & \mathcal{M}_j(q_{\mathcal{M}_j}) \cap \mathcal{O}_i = \emptyset,\ \ \mathcal{R}(q_{\mathcal{R}}) \cap \mathcal{M}_j(q_{\mathcal{M}_j}) = \emptyset \ \ et \ \ \mathcal{M}_j(q_{\mathcal{M}_j}) \cap \mathcal{M}_l(q_{\mathcal{M}_l}) = \emptyset\} \end{aligned}$$

Le problème de planification de mouvement pour le robot seul consiste à trouver une séquence de chemins sans collision amenant ce système d'une configuration initiale $q^i_{\mathcal{R}}$ à une configuration finale $q^f_{\mathcal{R}}$. Nous définissons un chemin comme suit :

Définition 8 *Un* **chemin** *est une fonction continue $\tau : [0, 1] \to CS$, connectant deux configurations $\tau(0)$ (la configuration initiale) et $\tau(1)$ (la configuration finale).*

Le problème de planification de tâches de manipulation ne se limite pas à connecter deux configurations du robot mais plutôt à trouver une solution permettant d'amener les objets déplaçables $(\mathcal{M}_1, ..., \mathcal{M}_r)$ d'un arrangement initial à arrangement final désiré. Dans le cas où les positions finales de tous les objets déplaçables sont complètement spécifiées, le problème de manipulation peut être formulé de la façon suivante : "partant d'une configuration initiale du système q^i_c, trouver un chemin pour atteindre q^f_c la configuration but du système". Un tel chemin est appelé *chemin de manipulation*.

[1] $q_{\mathcal{R}}$ est un vecteur de la forme $(q_1, q_2, ..., q_n)$, où n est le nombre de degré de liberté de $\mathcal{R}$, alors que $q_{\mathcal{M}_i}$ est un vecteur de la forme $(q_1, q_2, ..., q_6)$.

Le chemin de manipulation solution à notre problème, est un chemin particulier dans l'espace des configurations libres du système. Il s'agit d'un chemin contraint dans CS_{libre}. Il existe deux types de contraintes :

- Les contraintes de placement : chaque objet déplaçable doit occuper une position stable, lorsqu'il n'est pas saisi par le robot.
- Les contraintes de mouvement : chaque mouvement d'un objet déplaçable est induit par un mouvement du robot.

Nous formalisons ces deux types de contraintes dans la section suivante. Nous présentons également les restrictions induites par ces contraintes sur l'ensemble des configurations valides de l'espace libre.

2.2.2 Contraintes de placement

Toutes les configurations de CS_{libre} ne correspondent pas nécessairement à des configurations valides du point de vue physique (un objet ne peut pas "flotter"). Chaque objet déplaçable doit occuper une configuration **stable** (obéissant à la loi de gravité) lorsqu'il n'est pas déplacé par le robot. La figure 2.1 illustre la notion de configurations valides.

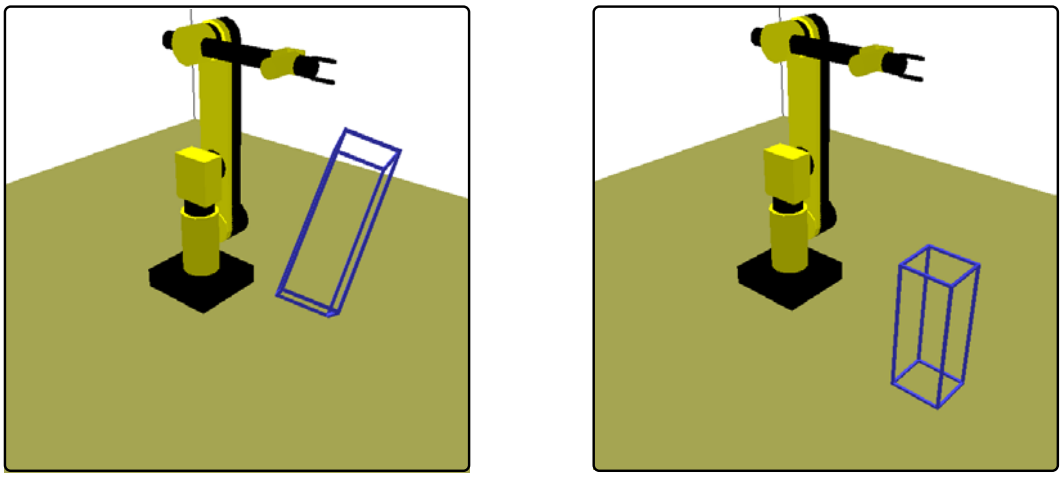


FIG. 2.1 – Placement stable de l'objet déplaçable (la barre) : placement non-valide (gauche) vs. valide(droite).

La stabilité de l'objet déplaçable $\mathcal{M}_i$ dans une configuration $q_{\mathcal{M}_i}$ dépend de la distribution des obstacles dans l'environnement et des configurations $(q_{\mathcal{M}_l})_{(l=1,\ldots,r),l\neq i}$ des autres objets déplaçables. Cette notion de stabilité peut être assurée par un contact face/face entre l'objet $\mathcal{M}_i$ et l'un des obstacles $\mathcal{O}_j$ ou encore entre cet objet et un autre objet déplaçable. Notons par $\mathcal{P}_i$ l'ensemble des classes de placements stables définies pour l'objet $\mathcal{M}_i$ et par $\mathcal{P} = \bigcup_{i=1}^{r} \mathcal{P}_i$ l'ensemble de placements stables de tous les objets. Dans le cas discret, $\mathcal{P}_i = \{p_i^1, \ldots, p_i^k\}$, où p_i^j correspond à un placement stable de

l'objet $\mathcal{M}_i$ ($\mathcal{P}_i$ est composé ainsi d'un nombre fini de placements stables de l'objet $\mathcal{M}_i$). Dans le cas de contraintes de placements continues, $\mathcal{P}_i = \bigcup(q_{place})$, où q_{place} modélise un ensemble de degrés de liberté de l'objet $\mathcal{M}_i$ sur une zone de placement donnée. Dans le cas d'un placement plan, q_{place} peut être paramétrisé par le triplet (x, y, θ) correspondant à deux translations et une rotation dans le plan de la zone de placement.

Notons que pour une même zone de placement stable (discrète ou continue), l'objet déplaçable peut être placé selon plusieurs orientations. La figure 2.2 montre deux orientations stables différentes de l'objet déplaçable sur la même zone de placement (le sol).

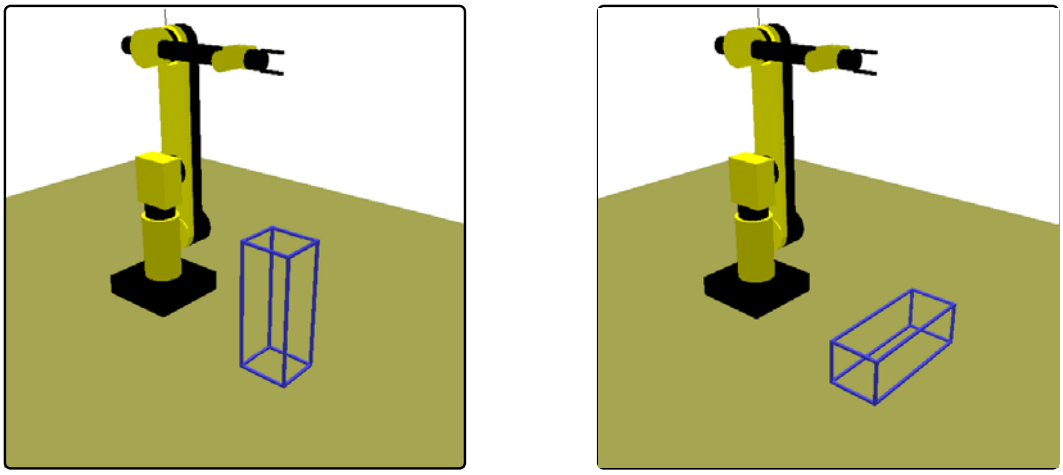


FIG. 2.2 – Deux orientations différentes pour poser l'objet déplaçable sur la même zone de pose (le sol).

Remarque :
Le but de ces travaux n'est pas de calculer l'ensemble de placements stables $\mathcal{P}_i$ d'un objet déplaçable dans un environnement donné. Ce type de calcul ainsi que l'étude de la stabilité d'un arrangement donné d'objets sous des contraintes de gravité est un problème difficile qui a été étudié dans un nombre réduit de papiers ([Kriegman 97],[Boneschanscher 88]). Dans la suite, nous considérons que ces ensembles de placements stables des objets sont donnés en entrée du problème de planification.

Du point de vue géométrique, les contraintes de placement réduisent l'espace des configurations admissibles CS_{libre} à un sous-espace contenant toutes les configurations valides qui vérifient également la stabilité des objets déplaçables. Ce sous-espace est appelé l'espace $PLACEMENT$ et noté $\mathcal{CP}$.

Définition 9 *L'espace* **PLACEMENT** *($\mathcal{CP}$) est l'ensemble des configurations du système dans CS_{libre} dans lesquelles tous les objets déplaçables occupent une position statiquement stable. La stabilité d'un objet $\mathcal{M}_i$ peut être accomplie par contact avec les obstacles statiques de l'environnement ou avec les*

autres objets déplaçables.

Plus formellement, le sous-espace *PLACEMENT* ($\mathcal{CP}$) peut être défini par :

$$\mathcal{CP} = \{q_c = (q_{\mathcal{R}}, q_{\mathcal{M}_1}, ..., q_{\mathcal{M}_r}) \in CS_{libre} \mid q_{\mathcal{R}} \in (CS_{\mathcal{R}})_{libre} \ et \ \forall i \in \{1, ..., r\}, \ q_{\mathcal{M}_i} \in \mathcal{P}_i\}$$

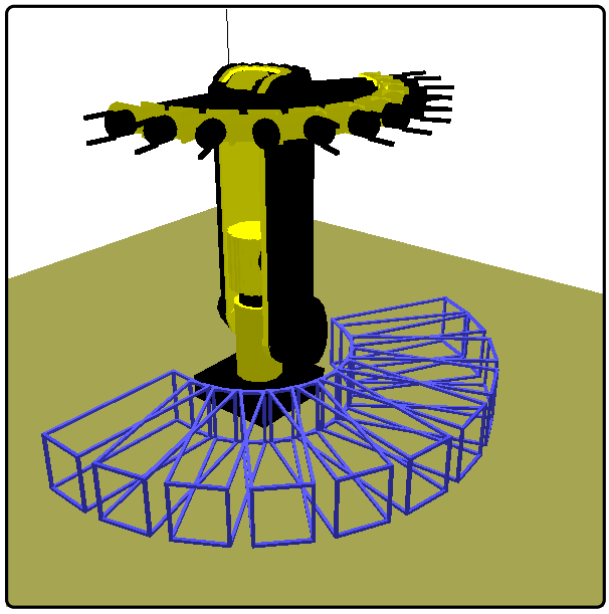
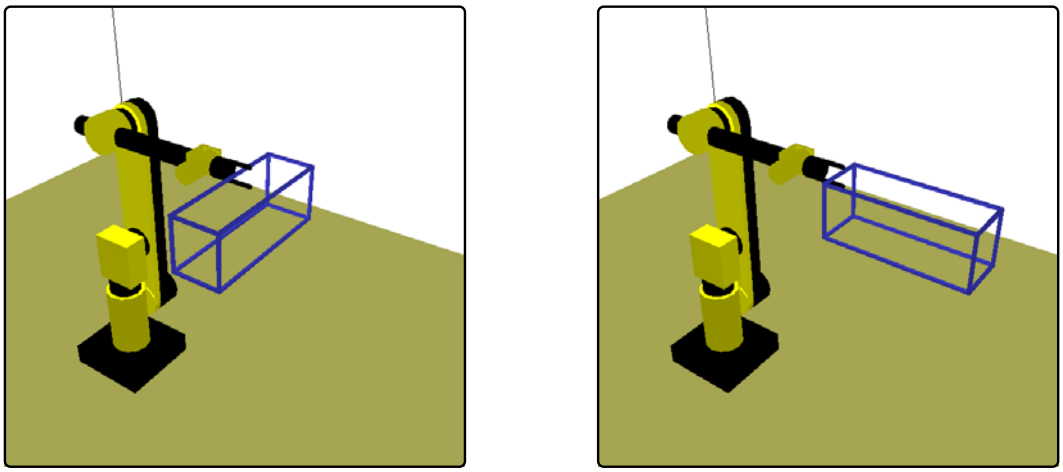


FIG. 2.3 – Un chemin calculé dans le sous-espace *PLACEMENT*.

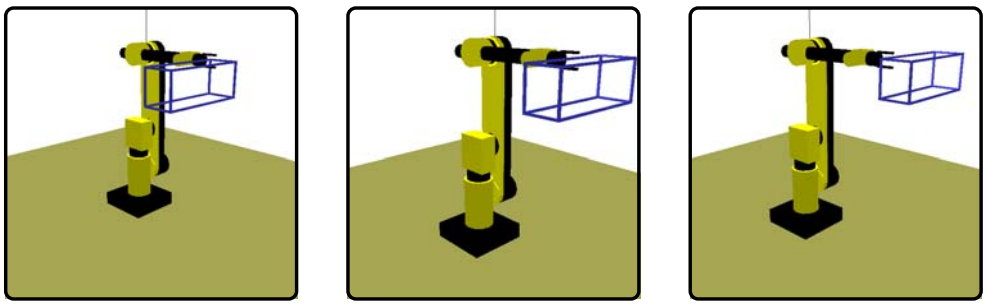
La figure 2.3 illustre un chemin calculé dans le sous-espace *PLACEMENT*. Lorsque le robot se déplace dans l'environnement, l'objet déplaçable occupe une position stable (mouvements plans sur le sol dans le cas de la figure 2.3). On peut noter qu'un chemin dans *PLACEMENT* ne correspond généralement pas à un chemin faisable du point de vue de la manipulation (l'objet se déplace sans qu'il soit saisi par le robot).

2.2.3 Contraintes de mouvement

Contrairement au robot $\mathcal{R}$, un objet déplaçable $\mathcal{M}_i$ ne peut pas se déplacer seul. Il change de configuration uniquement lorsqu'il est "*saisi*" par le robot. Les contraintes de mouvement expriment une relation, appelée **prise**, entre le mouvement de l'objet et celui du robot. Une prise est une fonction, notée $\mathcal{G}^T_{\mathcal{M}_i}$, définie de l'espace des configurations du robot $CS_{\mathcal{R}}$ vers l'espace des configurations de l'objet déplaçable $CS_{\mathcal{M}_i}$ et vérifiant $\mathcal{G}^T_{\mathcal{M}_i}(q_{\mathcal{R}}) = q_{\mathcal{M}_i}$, où $q_{\mathcal{R}} \in CS_{\mathcal{R}}$ et $q_{\mathcal{M}_i} \in CS_{\mathcal{M}_i}$ et T représente une transformation homogène entre l'organe terminal du robot et l'objet à déplacer. Elle dépend fortement du contexte de manipulation étudié.



a : Deux prises appartenant à deux classes différentes.



b : Trois prises appartenant à la même classe.

FIG. 2.4 – Différentes prises de l'objet déplaçable par le bras manipulateur.

Notons par $\mathcal{G}$ l'ensemble des classes de prises possibles défini comme donnée du problème de manipulation ($\mathcal{G} = \bigcup_{classes} \mathcal{G}_i$). Une classe de prise $\mathcal{G}_i \in \mathcal{G}$, est complètement spécifiée par la donnée de la fonction $\mathcal{G}^T_{\mathcal{M}_i}$ la décrivant. Elle peut être soit discrète soit continue. Dans le cas où la classe est discrète, elle est composée d'un nombre fini de prises (et donc de fonctions discrètes les décrivant), $\mathcal{G}_i = \{g_1, ..., g_n\}$. Dans le cas où la classe de prises est continue (elle est donc décrite par une fonction continue), $\mathcal{G}_i = \bigcup(q_{grasp})$, où q_{grasp} modélise un ou plusieurs degrés de liberté supplémentaires décrivant la prise. La figure 2.4(a) illustre deux prises appartenant à deux classes différentes d'un objet déplaçable par un bras manipulateur. La figure 2.4(b) montre trois prises d'une même classe.

Notons que le calcul des différentes classes de prises possibles permettant au robot de saisir un objet n'est pas l'objectif de ce mémoire. Nous redirigeons les lecteurs vers [Mishra 87, Pertin-Troccaz 89, Bicchi 01] pour un état de l'art détaillé sur la matière. Dans la suite, nous considérons que l'ensemble des prises possibles est défini comme entrées du problème de manipulation.

Exemple :

Afin d'illustrer la notion de prises continues, considérons l'exemple présenté dans la figure 2.5. Le robot manipulateur $\mathcal{R}$ est composé de deux corps rigides de longueurs respectives L_1 et L_2 et de deux joints rotoïdes (θ_1, θ_2). La base du robot est supposée fixe en (x_0,y_0). Une configuration de $\mathcal{R}$ est complètement spécifiée par $q_{\mathcal{R}} = (\theta_1, \theta_2)$. Dans cette figure l'objet déplaçable est une barre de longueur $L_{\mathcal{M}}$, en translation dans un environnement 2D. Le robot peut saisir l'objet sur toute sa longueur. Cependant, une prise dans ce cas est définie comme l'ensemble des configurations $q = (\theta_1, \theta_2, x_{\mathcal{M}}, y_{\mathcal{M}})$ du système global vérifiant :

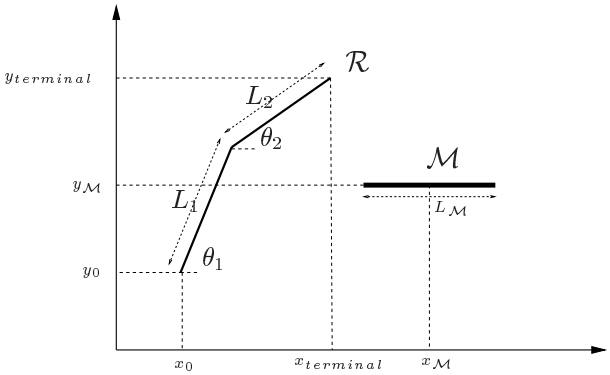


FIG. 2.5 – Un exemple 2D avec prise continue.

$$\begin{cases} x_{terminal} - x_{\mathcal{M}} = \alpha, \ avec\ \alpha \in [-\frac{L_{\mathcal{M}}}{2}, \frac{L_{\mathcal{M}}}{2}] \\ y_{terminal} - y_{\mathcal{M}} = 0 \end{cases}$$

La prise caractérisant la saisie de l'objet déplaçable par le robot est dite *"continue"* puisque le paramètre α peut prendre n'importe quelle valeur dans un intervalle défini. Ainsi à chaque $\alpha \in [-\frac{L_{\mathcal{M}}}{2}, \frac{L_{\mathcal{M}}}{2}]$ fixé, correspond une prise discrète. ■

Du point de vue géométrique les contraintes de mouvement réduisent l'espace des configurations admissibles à un sous-espace de configurations valides. Ce sous-espace est appelé *espace de prise* ou encore $GRASP$ et noté $\mathcal{CG}$.

Définition 10 *L'espace* **GRASP** *($\mathcal{CG}$) est l'ensemble des configurations de CS_{libre} dans lesquelles le robot déplace un objet $\mathcal{M}_i$ en respectant une fonction prise $\mathcal{G}^T_{\mathcal{M}}$ entrée du problème de manipulation, les autres objets déplaçables occupent des configurations statiquement stables.*

Notons par $\mathcal{F}^T_{\mathcal{G}}$ l'ensemble de toutes les fonctions décrivant les prises définies comme entrée du problème de manipulation, alors le sous-espace $\mathcal{CG}$ peut être défini comme étant :

$$\mathcal{CG} = \{q_c = (q_{\mathcal{R}}, q_{\mathcal{M}_1}, ..., q_{\mathcal{M}_r}) \in CS_{libre} \mid \exists\ f^i \in \mathcal{F}^T_{\mathcal{G}} : q_{\mathcal{M}_i} = f^i(q_{\mathcal{R}})\ et \\ \forall\ j \in \{1, ..., r\}, j \neq i,\ q_{\mathcal{M}_j} \in \mathcal{P}_j\}$$

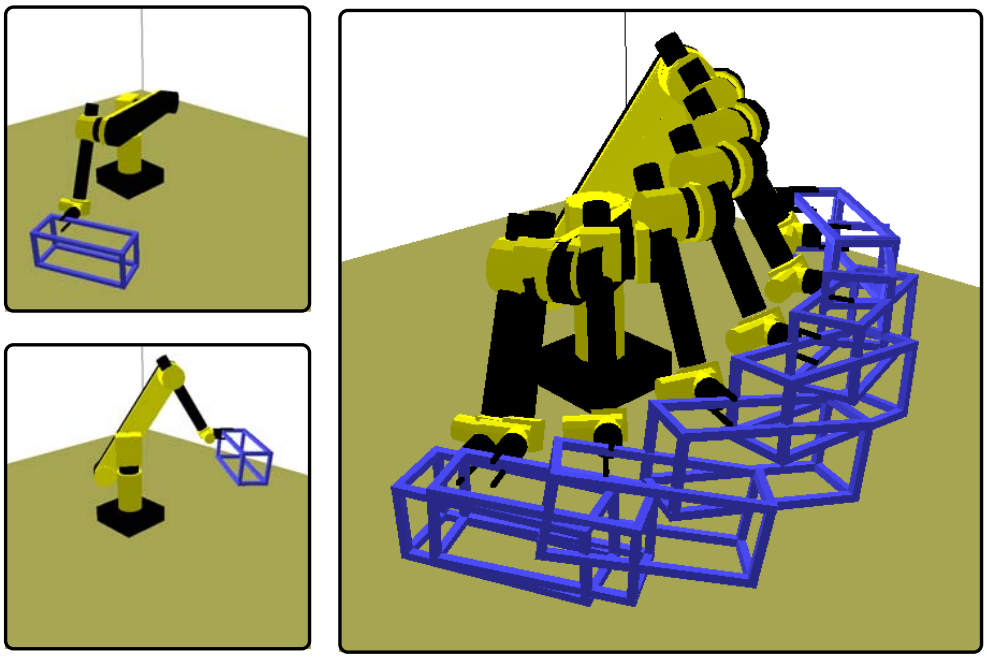


FIG. 2.6 – Un chemin dans le sous-espace *GRASP*.

La Figure 2.6 illustre un chemin calculé dans *GRASP* connectant deux configurations de ce sous-espace dans le cas d'une prise continue de l'objet. La relation $\mathcal{G}_{\mathcal{M}}^{T}(q_{\mathcal{R}}) = q_{\mathcal{M}}$, assurant l'attachement (ainsi que le contact) entre le robot et l'objet déplaçable, est vérifiée par chaque configuration du chemin. On peut noter, comme dans le cas de chemins dans *PLACEMENT*, qu'un chemin dans *GRASP* ne correspond généralement pas à un chemin faisable du point de vue de la manipulation (changement continue de prises le long du chemin).

Remarque :
Jusqu'ici, tout contact entre le robot, l'objet déplaçable ainsi que les obstacles a été interdit. Cependant, un placement stable ne peut être réalisé qu'à travers un contact entre l'objet $\mathcal{M}$ et certains obstacles définissant les zones de placements stables. Dans le cas où plusieurs objets déplaçables sont présents dans l'environnement, il est possible d'autoriser certains contacts entre les objets (plusieurs cubes superposés par exemple). Par ailleurs, la prise de l'objet par le robot nécessite elle aussi un contact entre les deux intervenants. Dans ce cas, le contact entre le robot et l'objet déplaçable est autorisé afin de garantir une saisie admissible et stable.

2.2.4 Chemin de manipulation

La solution pour un problème de manipulation est, comme nous l'avons précisé précédemment, un chemin contraint de CS_{libre}. Plus précisément c'est un chemin dans $\mathcal{CG} \cup \mathcal{CP}$, puisqu'il doit respecter les contraintes du problème de manipulation. Ce chemin (*chemin de manipulation*) est composé d'une séquence alternée de deux types de "sous-chemins" vérifiant les contraintes spécifiques du problème de

manipulation et séparés par des opérations de prise et de pose. Ces "sous-chemins" sont appelés *chemin de transit* et *chemin de transfert.*

Définition 11 *Nous appelons* **chemin de transit** *tout chemin de CS_{libre} dans lequel le robot se déplace tout seul. Les objets déplaçables demeurent stationnaires, occupant une configuration stable.*

Le long d'un chemin de transit, tous les objets déplaçables occupent une configuration stable, assurée par un contact avec les obstacles fixes de l'environnement. Un chemin de transit est une fonction continue $\tau_{ts} : [0,1] \longrightarrow \mathcal{CP}$. Ainsi, chaque configuration de ce chemin appartient au sous-espace $\mathcal{CP}$. Par ailleurs, un chemin de transit permet au robot d'atteindre une configuration dans laquelle il peut saisir un objet déplaçable, ou également d'effectuer un changement de prise. La figure 2.7 (gauche) illustre un chemin de transit permettant au bras manipulateur le changement de prise.

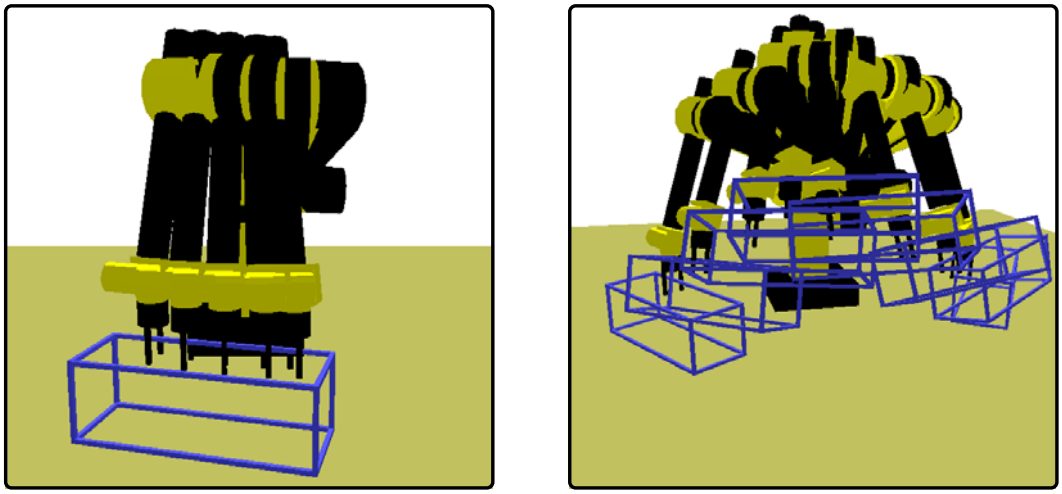


FIG. 2.7 – Chemin de transit (gauche) vs. chemin de transfert(droite).

Définition 12 *Pour une prise fixée, un* **chemin de transfert** *est défini comme étant le mouvement du robot déplaçant un objet $\mathcal{M}_i$ selon cette prise. Le long du chemin de transfert, la configuration de l'objet déplaçable $q_{\mathcal{M}_i}$ est déterminée par la fonction $\mathcal{G}^T_{\mathcal{M}_i}$, décrivant le modèle géométrique direct du robot : $q_{\mathcal{M}} = \mathcal{G}^T_{\mathcal{M}_i}(q_{\mathcal{R}})$. Les autres objets déplaçables occupent une configuration stable.*

Un chemin de transfert est inclus dans une sous-variété de $\mathcal{CG}$ définie par la prise assurant l'attachement de l'objet $\mathcal{M}_i$ au robot le long de ce chemin. Il s'agit d'une fonction continue $\tau_{tf} : [0,1] \longrightarrow \mathcal{CG}$. Le robot est supposé ne pouvoir déplacer qu'un seul objet à la fois, les autres objets demeurent statiques occupant une configuration stable. Dans un chemin de transfert, le robot déplace un objet d'un placement stable vers un autre. La figure 2.7 (droite) montre un chemin de transfert permettant le

changement du placement stable de l'objet déplaçable.

Une fois les notions de chemins de transit et de transfert spécifiées, un chemin de manipulation peut être défini comme étant une séquence finie et alternée de chemins de transit et de transfert comme l'illustre la figure 2.8.

Définition 13 *Un* **Chemin de manipulation** *est une séquence alternée finie* $(\tau_1, \tau_2, ..., \tau_{2p+1})$ *avec* $p \geq 0$ *et où :*

- $\tau_1, \tau_2, ..., \tau_{2p+1}$ *sont des chemins de transit.*
- $\tau_2, \tau_4, ..., \tau_{2p}$ *sont des chemins de transfert*
- $\forall\ l \in 1, ..., 2p,\ \tau_l(1) = \tau_{l+1}(0)$

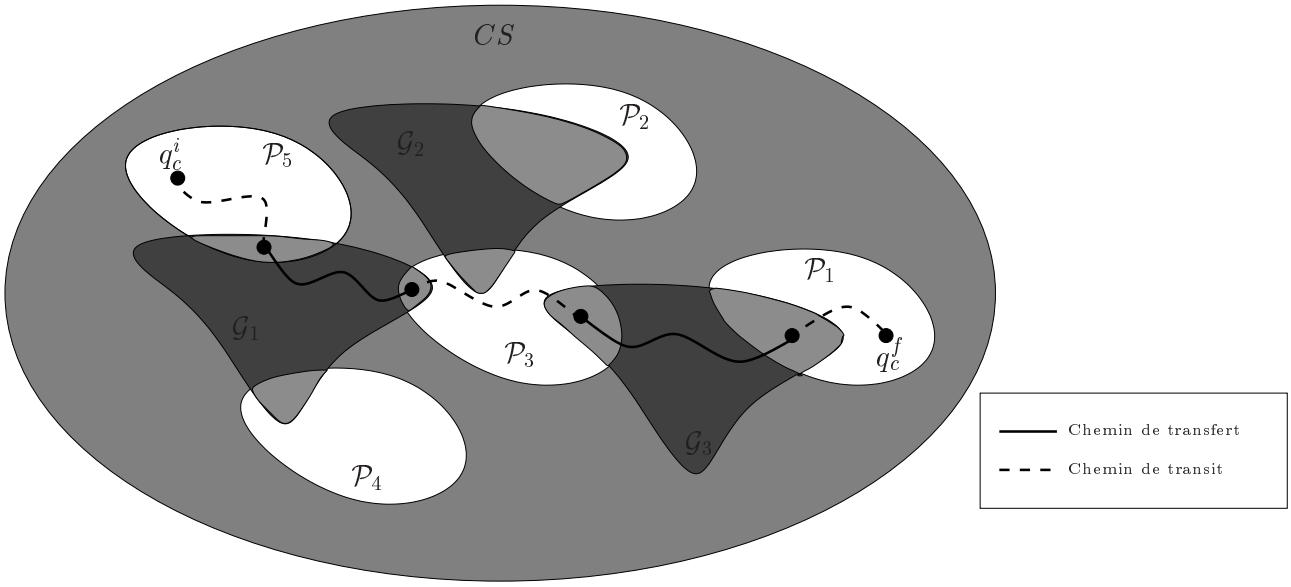


FIG. 2.8 – Structure de l'espace de recherche et chemin de manipulation.

A la fin d'un chemin de transfert, le robot place l'objet déplaçable dans une configuration stable et une opération de "*pose*" est ainsi effectuée. Alors qu'au début de ce type de chemin (à la fin d'un chemin de transit), le robot effectue une prise en exécutant une opération de "*saisie*".

2.3 Le problème de planification de tâches de manipulation

Le problème de planification de tâches de manipulation peut être maintenant formulé de la façon suivante : "*Étant donné deux ensembles (discrets ou continus)* $\mathcal{P}$ *et* $\mathcal{G}$ *définissant les placements stables et les prises possibles, le problème de planification de tâches de manipulation consiste à trouver un*

chemin de manipulation (séquence alternée et finie de chemins de transit et de transfert) connectant deux configurations du système q_c^i et q_c^f du sous-espace $\mathcal{CG} \cup \mathcal{CP}$".

En d'autres termes, le problème de planification de tâches de manipulation consiste à chercher une séquence alternée de chemins de transit et de chemins de transfert dans un sous-espace contraint de CS_{libre} caractérisé par les prises possibles et les différents placements stables. Quant aux connexions possibles entre les chemins de transfert et les chemins de transit, elles se trouvent dans une sous-variété particulière de CS_{libre}, à savoir le sous-espace $\mathcal{CG} \cap \mathcal{CP}$.

Le sous-espace $\mathcal{CG} \cap \mathcal{CP}$:

Le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est une sous-variété de CS_{libre} dans laquelle les contraintes de placements et les contraintes de mouvement sont simultanément respectées. Dans cette variété, le robot déplace l'un des objets qui garde un placement stable le long du mouvement. Les autres objets déplaçables sont supposés dans une configuration stable. D'autre part, Les chemins de transit résident dans $\mathcal{CP}$, alors que les chemins de transfert sont dans $\mathcal{CG}$. Chacun de ces deux types de chemins commence et aboutit dans le sous-espace $\mathcal{CG} \cap \mathcal{CP}$. Ce dernier représente le lieu de connexion des chemins de transfert et des chemins de transit.

Dans le cas d'un ensemble discret de prises et de placements stables, l'intersection $\mathcal{CG} \cap \mathcal{CP}$ est composée d'un ensemble fini de configurations[2]. La recherche de chemin de manipulation s'effectue en déterminant les différentes connexions entre ces configurations isolées. Dans le cas de la planification de tâches de manipulation sous contraintes continues (ensemble de prises et de placements continus), le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est composé de plusieurs domaines continus de configurations. Comme nous le verrons dans la suite, l'extension de la notion de graphe de manipulation à ce cas de contraintes passe par l'analyse de la connectivité de ce sous-espace.

Dans les deux cas, il suffit d'étudier la connexion des différentes composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$ par des chemins de transfert et des chemins de transit pour calculer le chemin de manipulation. Une structure de graphe est ainsi apparue. Les nœuds de ce graphe sont les composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$. Il comporte deux types d'arêtes : des *arêtes de transit* et des *arêtes de transfert*. Une arête de transit (resp. de transfert) entre deux nœuds indique l'existence d'un chemin de transit (resp. de transfert) connectant deux configurations de leurs composantes connexes respectives. Ce graphe est appelé **graphe de manipulation** et est noté $\mathcal{MG}$.

Construction du graphe de manipulation :

La résolution d'un problème de planification de tâches de manipulation consiste à construire dans

[2]Pour un robot redondant (cas étudié dans [Ahuactzin 98]), la cardinalité du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est infinie.

un premier temps un graphe (graphe de manipulation) capturant la connectivité des différentes sous-variétés formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$, et de chercher dans une seconde étape un chemin dans ce graphe répondant au problème posé. La construction du graphe de manipulation s'effectue en deux étapes :

- calculer les différentes composantes connexes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$;
- déterminer la connectivité de ces composantes connexes par des chemins de transfert et des chemins de transit.

Le calcul des différentes composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$ suppose l'existence d'une méthode locale permettant la planification d'un chemin entre deux nœuds d'une même composante connexe de ce sous-espace. Nous détaillerons dans la suite les différentes étapes de construction du graphe de manipulation pour deux instances particulières du problème étudiées dans [Alami 95] : le cas d'ensemble discret de prises et de poses, et le cas de contraintes continues pour des environnements polygonaux contenant un seul objet déplaçable.

2.3.1 Cas de zones de placements et de prises discrètes

Dans le cas de robot non-redondant et où les zones de placement ainsi que les prises sont discrètes, le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est réduit à un ensemble fini de configurations. En effet, si $n_1 = card(\mathcal{CP})$ et $n_2 = card(\mathcal{CG})$, alors $card(\mathcal{CG} \cap \mathcal{CP}) \leq n_1 \times n_2$ (certaines prises ne sont pas forcément possibles pour n'importe quel placement des objets). La construction du graphe de manipulation se réduit à deux étapes : calculer les configurations formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ puis déterminer leur connexion par des chemins de transit/transfert. La définition d'une méthode de planification de chemin dans $\mathcal{CG} \cap \mathcal{CP}$ n'est pas nécessaire dans ce cas particulier du problème de manipulation puisque chaque composante de ce sous-espace est réduite à une seule configuration.

Graphe de manipulation :

Les nœuds du graphe de manipulation sont toutes les configurations dans lesquelles le robot est autorisé à saisir ou poser un objet déplaçable occupant un placement donné, en tenant compte des différents placements possibles des autres objets. Notons par $p^1_{\mathcal{M}_i}, p^2_{\mathcal{M}_i}, ..., p^k_{\mathcal{M}_i}, ...$ les différents placements possibles de l'objet $\mathcal{M}_i$ et par $\mathcal{G}^{T_1}_{\mathcal{M}_i}, \mathcal{G}^{T_2}_{\mathcal{M}_i}, ..., \mathcal{G}^{T_m}_{\mathcal{M}_i}, ...$ les différentes fonctions caractérisant les prises $g^1_{\mathcal{M}_i}, g^2_{\mathcal{M}_i}, ..., g^m_{\mathcal{M}_i}, ...$ de l'objet $\mathcal{M}_i$ autorisées au robot. Le calcul de l'ensemble $\{q_{\mathcal{R}} \in CS_{\mathcal{R}} \mid \forall i \in \{1, ..., r\}, \forall m, \forall k,\ \mathcal{G}^{T_m}_{\mathcal{M}_i}(q_{\mathcal{R}}) = p^k_{\mathcal{M}_i}\}$ (ensemble de nœuds formant le graphe de manipulation) est effectué en utilisant le modèle géométrique inverse du robot et prenant en compte toutes les combinaisons possibles de placements des objets déplaçables. De plus, si nous supposons que le robot est non-redondant et que la solution n'inclut pas de configurations singulières, cet ensemble est réduit à un nombre fini de configurations. Chaque nœud du graphe de manipulation représente dans ce cas une

composante connexe de $\mathcal{CG} \cap \mathcal{CP}$ réduite à une configuration. Un nœud est caractérisé par les différents placements des objets déplaçables ainsi que la prise courante. Un nœud N_h du graphe de manipulation est donc décrit par : $N_h = (g^h_{\mathcal{M}_k}, p^h_{\mathcal{M}_1}, ..., p^h_{\mathcal{M}_r})$, où $g^h_{\mathcal{M}_k}$ est la prise courante, $p^h_{\mathcal{M}_i}$ est le placement stable occupé par l'objet déplaçable $\mathcal{M}_i$ dans le nœud N_h .

Après avoir calculé les nœuds du graphe de manipulation $\mathcal{MG}$, la deuxième phase consiste à déterminer les différentes arêtes connectant ces nœuds. Il existe deux types d'arêtes : des arêtes de transfert et des arêtes de transit. Il est inutile de tester l'existence d'un chemin de transfert (resp. transit) entre chaque paire de nœuds du graphe de manipulation. En effet, un chemin de transit connecte uniquement deux nœuds ayant le même placement des objets déplaçables, alors qu'un chemin de transfert est cherché entre deux nœuds ayant la même prise et qui diffèrent uniquement par le placement d'un seul objet. Plus formellement, nous définissons $C_{Transit}$ l'ensemble des paires de nœuds candidats à un test de connexion par un chemin de transit par : $C_{Transit} = \{(N_i, N_j) \mid N_i = (g^i_{\mathcal{M}_k}, p^i_{\mathcal{M}_1}, ..., p^i_{\mathcal{M}_r}),\ N_j = (g^j_{\mathcal{M}_l}, p^j_{\mathcal{M}_1}, ..., p^j_{\mathcal{M}_r}) : \forall \lambda \in \{1, ..., r\},\ p^i_{\mathcal{M}_\lambda} = p^j_{\mathcal{M}_\lambda}\}$. De la même façon, notons par $C_{Transfert}$ l'ensemble des paires de nœuds candidats à un test de connexion par un chemin de transfert, $C_{Transfert} = \{(N_i, N_j) \mid N_i = (g^i_{\mathcal{M}_k}, p^i_{\mathcal{M}_1}, ..., p^i_{\mathcal{M}_r}),\ N_j = (g^j_{\mathcal{M}_k}, p^j_{\mathcal{M}_1}, ..., p^j_{\mathcal{M}_r}) \mid g^i_{\mathcal{M}_k} = g^j_{\mathcal{M}_k}\ et\ \exists\, \xi \in \{1, ..., r\},\ \forall\, \lambda \in \{1, ..., r\},\ \lambda \neq \xi,\ p^i_{\mathcal{M}_\lambda} = p^j_{\mathcal{M}_\lambda}\}$. Le test de connexion par des chemins de transit est alors effectué entre les paires de nœuds de $C_{Transit}$ alors que les connexions par des chemins de transfert sont uniquement testées entre les paires de nœuds de $C_{Transfert}$.

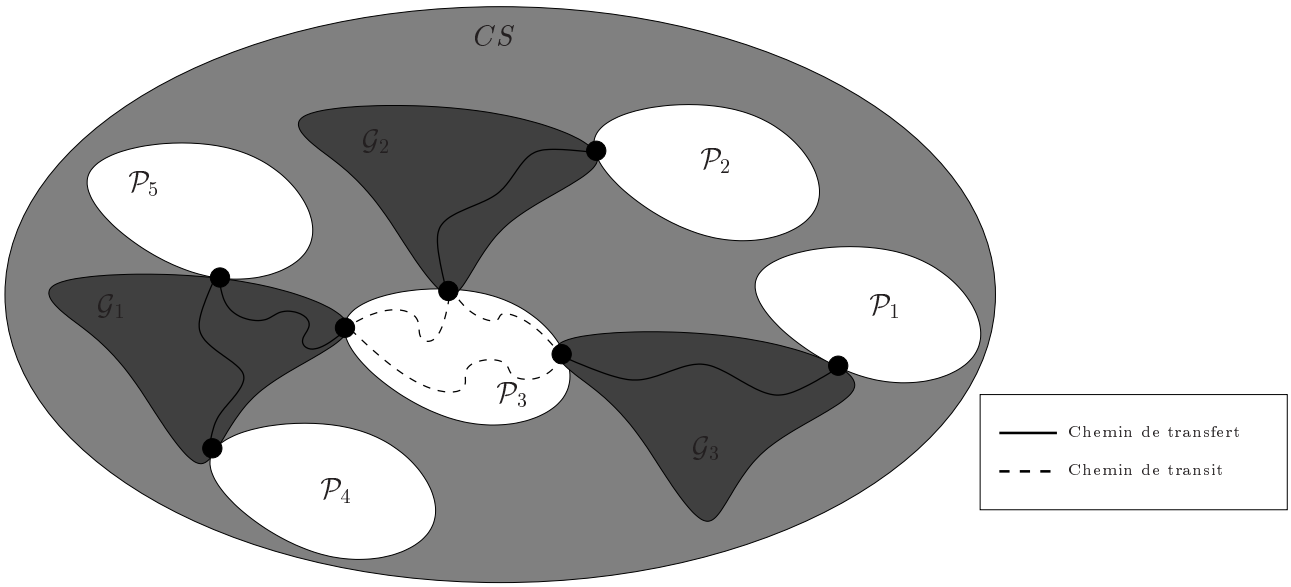
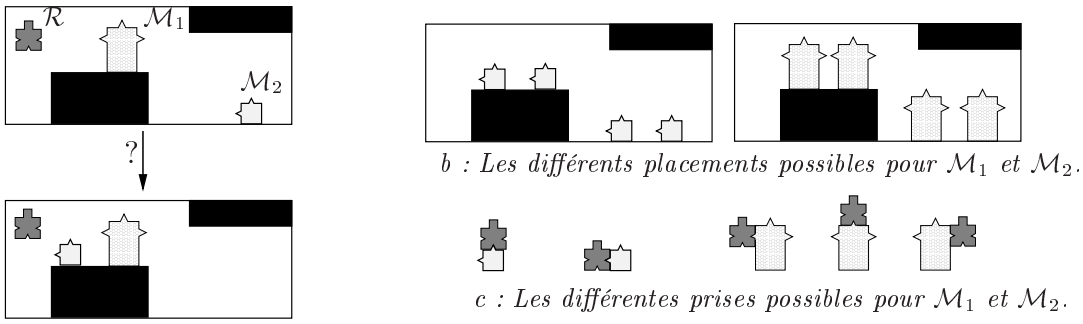


FIG. 2.9 – Graphe de manipulation dans le cas de problèmes de manipulation sous contraintes discrètes.

La recherche de chemin de manipulation s'effectue dans le graphe $\mathcal{MG}$ ainsi construit (figure 2.9). Elle peut être réalisée par un algorithme classique de type A^* couplé avec des heuristiques simples telle que la minimisation de la longueur totale du chemin de manipulation.

Exemple 2D :

Considérons l'exemple présenté dans la figure 2.10(a) (tiré de [Siméon 89]). Nous l'avons reproduit afin d'illustrer la construction de graphe de manipulation dans le cas d'un ensemble discret de prises et de placements. Il s'agit d'un environnement 2D composé de trois corps polygonaux (un robot et deux objets déplaçables). Le robot peut se déplacer uniquement en translation. Les différentes prises discrètes autorisées ainsi que les placements pour ce problème de manipulation sont présentés dans la figure 2.10 (b) et (c).



a : Configuration initiale et finale du système.

b : Les différents placements possibles pour $\mathcal{M}_1$ et $\mathcal{M}_2$.

c : Les différentes prises possibles pour $\mathcal{M}_1$ et $\mathcal{M}_2$.

FIG. 2.10 – Un exemple 2D de problème de manipulation sous contraintes discrètes [Siméon 89].

Selon l'approche décrite ci-dessus, La construction du graphe de manipulation s'effectue en deux étapes. La première étape consiste à générer les différentes configurations formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$. Il s'agit de déterminer (en utilisant un modèle géométrique inverse) les configurations dans lesquelles le robot est autorisé à saisir ou à poser un objet déplaçable occupant un placement stable. Le nombre de nœuds du graphe de manipulation $\mathcal{MG}$ croit en fonction du nombre de prises et de placements définis par l'utilisateur. Dans le cas de l'exemple simple de la figure 2.10, le nombre de nœuds de $\mathcal{MG}$ correspond au nombre *prises* $\times$ *placements*, soit $(3+2) \times (4 \times 4) = 80$ nœuds. La construction du graphe de manipulation s'effectue incrémentalement en se basant sur des heuristiques simples telles que la minimisation des distances de déplacement des objets.

Une fois les configurations de $\mathcal{CG} \cap \mathcal{CP}$ calculées, la seconde étape consiste à déterminer les différentes connexions possibles entre ces configurations par des chemins de transit et des chemins de transfert. Un nœud du graphe de manipulation est caractérisé par le triplet (prise, placement de l'objet $\mathcal{M}_1$, placement de l'objet $\mathcal{M}_2$). La connexion par des arêtes de type transfert d'un nœud N_k caractérisé par exemple par le triplet $(g^1_{\mathcal{M}_1}, p^2_{\mathcal{M}_1}, p^3_{\mathcal{M}_1})$ peut être testée avec l'ensemble des nœuds $(g^1_{\mathcal{M}_1}, -, p^3_{\mathcal{M}_1})$. De même, la connexion par des arêtes de type transit peut être établie avec l'ensemble des nœuds $(-, p^2_{\mathcal{M}_1}, p^3_{\mathcal{M}_1})$. Le trait dans les triplets caractérisant les nœuds peut être remplacé par n'importe quel(le) placement (resp. prise) défini(e) comme donnée du problème de manipulation.

Le chemin de manipulation solution au problème posé est présenté dans le chapitre précédent

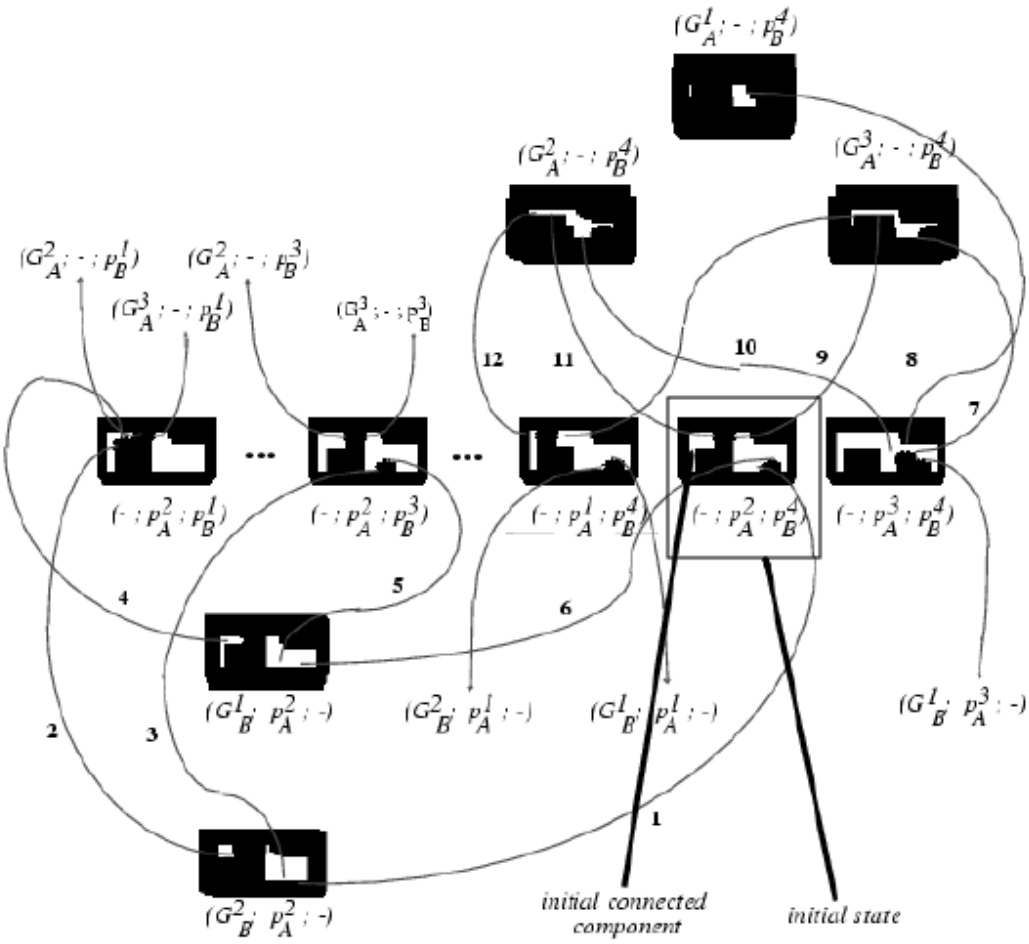


FIG. 2.11 – Connexions entre différentes tranches d'espace de configuration.

(§ 1.3.1). La figure 2.11 illustre des connexions entre plusieurs tranches d'espace de configurations traversées par le système global le long d'un chemin de manipulation. Dans cet exemple, la méthode de planification utilisée pour le calcul des chemins de transit/transfert (dans les différentes tranches d'espace de configurations) repose sur une technique classique de graphe de visibilité (après une étape de grossissement d'obstacles).

2.3.2 Cas de zones de placements et de prises continues

La structure de l'espace de recherche devient plus complexe dès qu'il s'agit d'ensembles continus de prises et de placements. En particulier, le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ n'est plus formé dans ce cas d'un ensemble fini de configurations mais plutôt de plusieurs domaines continus. L'extension de la notion de *graphe de manipulation* à ce type de domaine continu, est donc plus délicate. En effet, il faut tout d'abord répondre à la question suivante : *la connectivité du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ reflète -t- elle l'existence d'un chemin de manipulation ?*

Graphe de manipulation :

La réponse à cette question n'est pas immédiate. En effet, les chemins calculés dans $\mathcal{CG} \cap \mathcal{CP}$ ne sont pas forcément des chemins admissibles. Ces chemins correspondent à un changement continu de prise par le robot. Le long d'un chemin dans $\mathcal{CG} \cap \mathcal{CP}$, le robot peut glisser le long de l'objet déplaçable qui reste en contact avec sa zone de placement (par définition du sous-espace $\mathcal{CG} \cap \mathcal{CP}$). Ce type de chemin n'est donc pas à priori admissible au sens de la manipulation.

Les seuls travaux traitant ce cas général de problème de planification de tâches de manipulation sont ceux présentés dans [Alami 95]. L'extension au cas continu repose sur une *propriété de réduction* qui fait l'objet de la section suivante. Cette propriété montre que les chemins calculés dans le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ sont **transformables en une séquence finie de chemins de transit et de transfert**. La construction de graphe de manipulation s'effectue ainsi en deux étapes :

- une exploration du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ afin de calculer ses différentes composantes connexes,
- et déterminer la connectivité de ces composantes par des chemins de transit et de transfert.

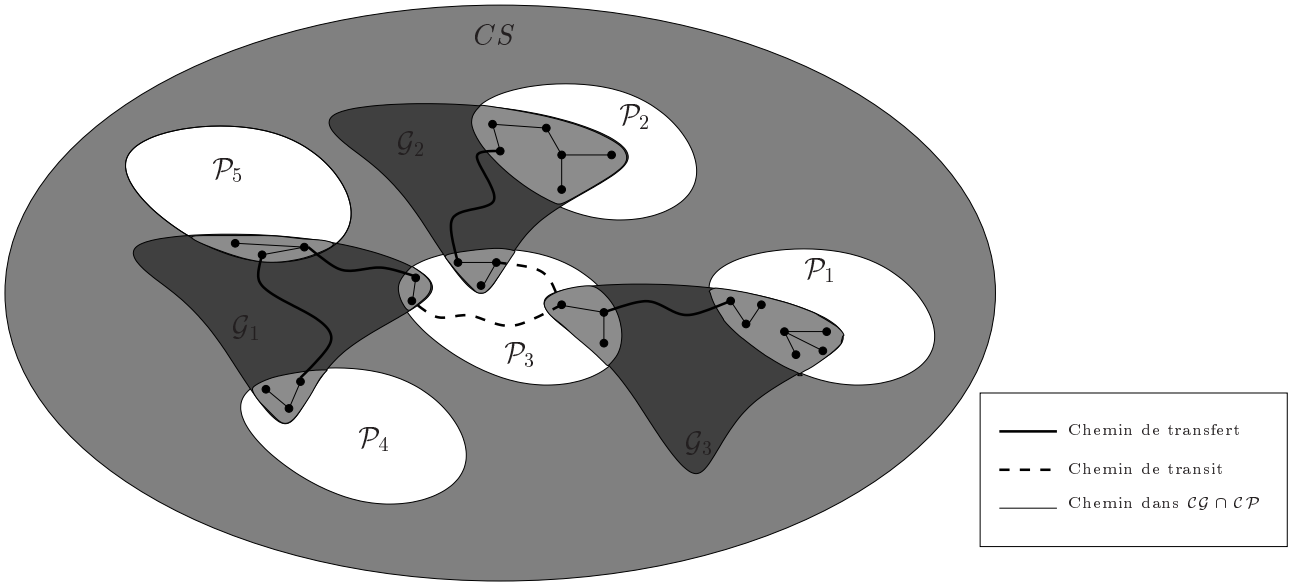


FIG. 2.12 – Graphe de manipulation dans le cas de problèmes de manipulation sous contraintes continues.

La figure 2.12 illustre la structure du graphe de manipulation dans le cas où les contraintes de placement et de prises sont continues.

Exemple 2D :

Considérons l'exemple présenté dans la figure 2.13 (tiré de [Alami 95]). Il s'agit d'un environnement 2D contenant un robot polygonal en translation et en présence d'un seul objet déplaçable. Une saisie est caractérisée par un contact entre le robot et l'objet. Le sous-espace $PLACEMENT$ s'étend dans ce cas sur

tout l'environnement (puisqu'il s'agit d'environnement 2D). En conséquence, les sous-espaces $\mathcal{CG} \cap \mathcal{CP}$ et $GRASP$ sont confondus ($\mathcal{CG} \cap \mathcal{CP} = \mathcal{CG}$). La construction de graphe de manipulation se ramène à déterminer les différentes composantes connexes du sous-espace $GRASP$ relatif à cet exemple ; puis à déterminer leurs connexions par des chemins de transit. Dans [Alami 95], la méthode proposée pour déterminer les composantes connexes du sous-espace $GRASP$ est basée sur une décomposition cellulaire de l'espace des configurations admissibles (permettant la résolution du problème de planification de mouvement coordonné). Une méthode de rétraction est ensuite appliquée sur cette décomposition afin d'obtenir une décomposition cellulaire de $\mathcal{CG}$. La connexion entre les différentes composantes connexes du sous-espace $\mathcal{CG}$ par des chemins de transit est déduite après une étude de l'espace de configuration admissible du robot. Le graphe de manipulation résultat de ce calcul est présenté dans la figure 2.14. Les traits en pointillés modélisent des chemins de transit connectant les deux composantes connexes formant $\mathcal{CG}$.

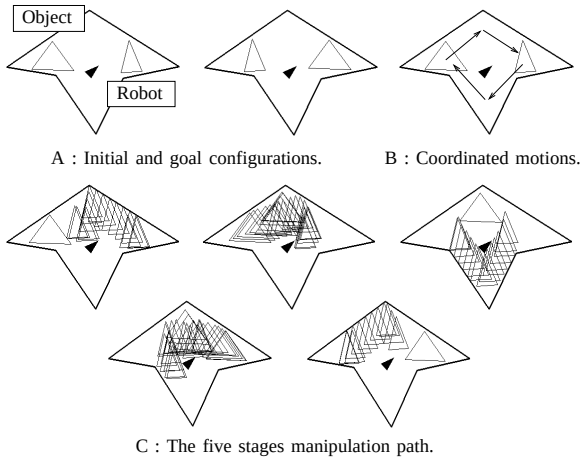


FIG. 2.13 – Un exemple 2D de problème de manipulation sous contraintes continues.

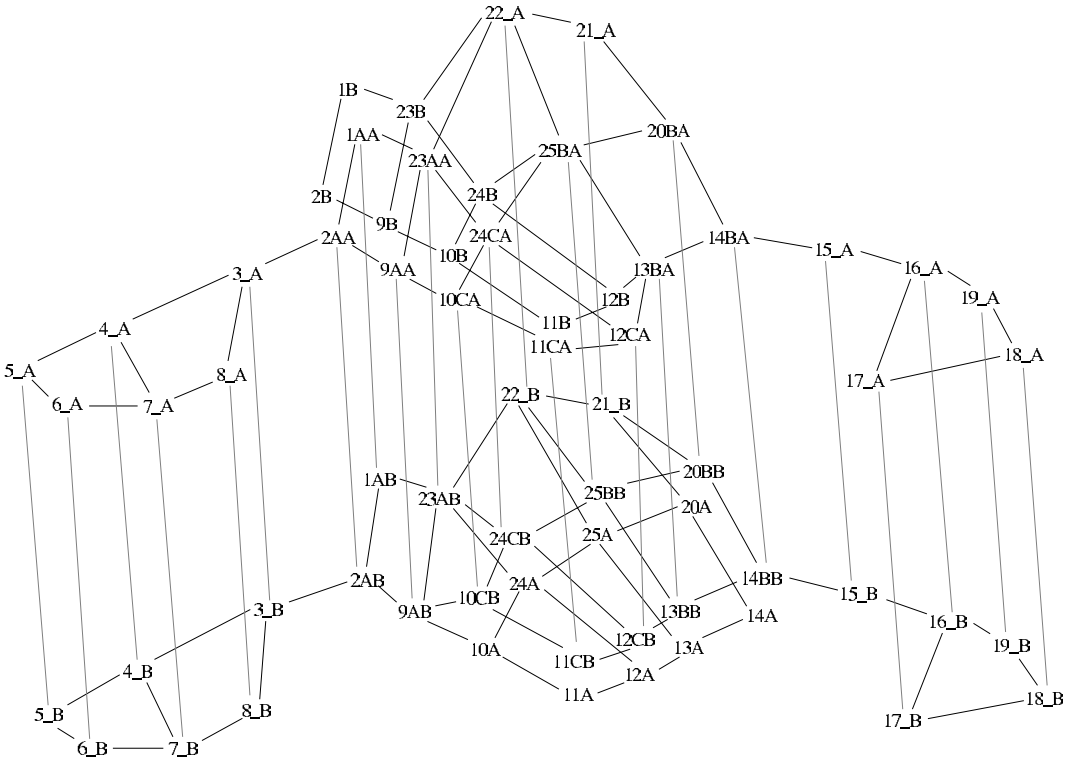


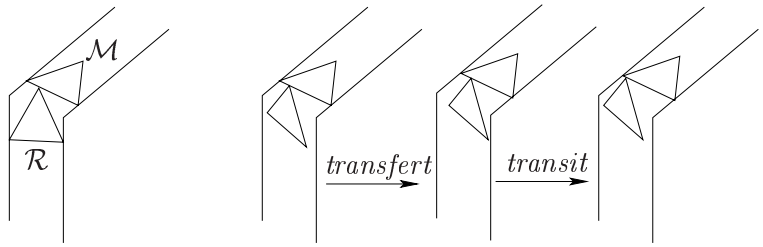
FIG. 2.14 – Le graphe de manipulation calculé.

2.4 Propriété de réduction

Dans cette section nous exposerons la propriété qui a permis l'extension de l'approche présentée au cas de contraintes continues, à savoir *la propriété de réduction* [Alami 89]. Nous avons vu que la résolution de problème de manipulation repose sur l'exploration du sous-espace $\mathcal{CG} \cap \mathcal{CP}$. Nous avons vu également que les chemins calculés dans ce sous-espace ne correspondent pas à des chemins faisables ni valides du point de vue de la manipulation. La propriété de réduction permet de palier ce problème en garantissant la transformation des chemins calculés dans cette sous-variété en un chemin de manipulation. Cette propriété est vérifiée sous une certaine hypothèse :

Hypothèse 1 : *On suppose que le robot évite tout contact avec les obstacles de l'environnement.*

L'hypothèse1 permet d'éviter des situations comme celle présentée dans la figure 2.15(a). Le robot coincé entre les obstacles statiques, ne peut pas manipuler l'objet déplaçable. La figure 2.15(b) illustre le cas où l'hypothèse1 est vérifiée et montre une séquence de mouvement possible lorsque cette hypothèse est vérifiée. Nous définissons le sous-espace $GRASP$ comme étant l'espace de "*contact*" de toutes les configurations vérifiant l'hypothèse1.



a : L'hypothèse1 non satisfaite *b : L'hypothèse1 satisfaite*

FIG. 2.15 – Illustration de l'hypothèse1.

Propriété 1 *Propriété de réduction*
Deux configurations du système appartenant à la même composante connexe de $\mathcal{CG} \cap \mathcal{CP}$ sont connectables par un chemin de manipulation.

Démonstration 1 : *[Alami 89]*
Soient q_1 et q_2 deux configurations du système appartenant à la même composante connexe de $\mathcal{CG} \cap \mathcal{CP}$. Il existe un chemin $\tau : [0,1] \rightarrow \mathcal{CG} \cap \mathcal{CP}$ liant ces deux configurations ($\tau(0) = q_1$ et $\tau(1) = q_2$). Notons par τ_r et τ_m les projections respectives de τ sur $CS_{\mathcal{R}}$ et $CS_{\mathcal{M}}$.

Soit $q_3 = \tau(t)$ une configuration sur ce chemin. Grâce à l'hypothèse1, $\tau_r(t)$ est inclus dans un ouvert de $(CS_{\mathcal{R}})_{libre}$. Nous pouvons trouver ainsi un disque D_ε de rayon $\varepsilon > 0$ et centré en $\tau_r(t)$.

Or τ est continue, il existe alors $\eta_1 > 0$ telle que : $\forall\ \lambda \in]t-\eta_1, t+\eta_1[$, $\tau_r(\lambda) \in D_{\frac{\varepsilon}{2}}$.
De même, $\tau_r - \tau_m$ est une fonction continue. il existe alors $\eta_2 > 0$ telle que : $\forall\ \lambda \in]t-\eta_2, t+\eta_2[$, $\|(\tau_r(\lambda) - \tau_m(\lambda)) - (\tau_r(t) - \tau_m(t))\|_{\mathbb{R}^2} < \frac{\varepsilon}{2}$.
Ceci se traduit par le fait qu'une configuration ne varie pas plus de $\frac{\varepsilon}{2}$ le long du chemin τ entre $\tau(t-\eta_2)$ et $\tau(t+\eta_2)$.

Soit $\eta = min\{\eta_1, \eta_2\}$, on a alors :

$$\forall\ \lambda, \quad \sigma \in]t-\eta, t+\eta[, \quad \tau_m(\sigma) + (\tau_r(\lambda) - \tau_m(\lambda)) \in D_\varepsilon. \tag{2.1}$$

Soient $q^i = \tau(\lambda_1)$ et $q^b = \tau(\lambda_2)$ deux configurations sur le chemin τ, avec λ_1 et $\lambda_2 \in]t-\eta, t+\eta[$ et $\lambda_1 < \lambda_2$. Montrons maintenant que q^i et q^b sont connectés par un chemin de transfert suivi d'un chemin de transit.

Considérons le chemin $(\tau_m(\lambda), \tau_m(\lambda) + (\tau_r(\lambda_1) - \tau_m(\lambda_1)))$, avec $\lambda \in [\lambda_1, \lambda_2]$. Ce chemin est clairement un chemin de transfert avec une prise constante $(\tau_r(\lambda_1) - \tau_m(\lambda_1))$, entre $\tau(\lambda_1)$ et $(\tau_m(\lambda_2), \tau_m(\lambda_2) + (\tau_r(\lambda_1) - \tau_m(\lambda_1)))$. Grâce à l'équation 2.1, ce chemin est admissible. De plus, le chemin $(\tau_m(\lambda_2), \tau_m(\lambda_2) + (\tau_r(\lambda) - \tau_m(\lambda)))$, avec $\lambda \in [\lambda_1, \lambda_2]$, est un chemin de transit entre $(\tau_m(\lambda_2), \tau_m(\lambda_2) + (\tau_r(\lambda_1) - \tau_m(\lambda_1)))$ et $\tau_r(\lambda_2)$. Ce chemin est aussi admissible grâce encore une fois à l'équation 2.1. La concaténation de ces deux chemins constitue un chemin de manipulation entre $\tau(\lambda_1)$ et $\tau(\lambda_2)$.

Puisque τ_r est un compact, inclus dans un ouvert $(CS_{\mathcal{R}})_{libre}$, nous pouvons réitérer cette transformation locale sur un recouvrement fini de $[0, 1]$. Nous avons ainsi un nombre fini de chemins de manipulation élémentaires formant un chemin de manipulation connectant q_1 et q_2. ■

Conséquences :

Cette propriété est très importante puisqu'elle garantit la transformation de tout chemin calculé dans le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ en une séquence alternée et finie de chemins de transit et de chemins de transfert. Ceci permet de conclure sur la validité d'un chemin de manipulation dont certaines portions sont calculées dans $\mathcal{CG} \cap \mathcal{CP}$. Pour résoudre un problème de manipulation, il suffit alors d'étudier la connectivité des différentes composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$ par des chemins de transit et des chemins de transfert.

La propriété de réduction nous permet donc d'étendre la notion de graphe de manipulation au cas où le problème de manipulation posé est décrit par un ensemble continu de prises et de placements. Les nœuds de $\mathcal{MG}$ correspondent aux différentes composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$ alors qu'une arête entre deux nœuds indique l'existence de chemins de transit/transfert connectant les deux configurations des composantes associées.

Remarque :
Le nombre de chemins élémentaires de manipulation (séquence alternée de chemins de transit et de chemins de transfert) nécessaires pour transformer un chemin τ calculé dans le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ dépend du débattement[3] du chemin τ_r dans $(CS_{\mathcal{R}})_{libre}$ (τ_r est la projection de τ sur $(CS_{\mathcal{R}})_{libre}$).

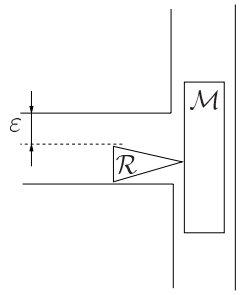


FIG. 2.16 – Le cas d'un grand nombre de chemins de transit/transfert.

La figure 2.16 illustre un cas où le nombre de chemins élémentaires pour transformer un chemin de $\mathcal{CG} \cap \mathcal{CP}$ est en $O(\frac{1}{\varepsilon})$.

2.5 Conclusion

Nous avons rappelé dans ce chapitre les différentes contraintes de manipulation, introduites dans [Alami 89], qui ont conduit à la structuration de l'espace de recherche ainsi qu'à la définition d'une approche générique pour la résolution de problèmes de manipulation. Cette dernière repose, comme nous l'avons montré dans ce chapitre, sur le calcul des différentes composantes connexes du sous-espace $GRASP \cap PLACEMENT$ et la détermination de leur connectivité. Nous avons illustré ce schéma de résolution à travers un exemple de problème de manipulation sous contraintes discrètes et avons détaillé les différentes étapes conduisant à la construction du graphe de manipulation. L'application de l'approche présentée dans ce chapitre au cas continu reste limitée à un cas particulier de problèmes de manipulation (environnement 2D, robot polygonal en translation, ...). Son extension au cas général (environnements 3D, robots articulés) nécessite la définition d'une méthode générique de planification de chemin dans $GRASP \cap PLACEMENT$ permettant d'analyser la connectivité de cet espace.

Les deux chapitres suivants présentent notre contribution à la planification de tâches de manipulation. Le chapitre 3 est consacré au cas discret du problème. Il expose une extension des méthodes probabilistes pour le calcul du graphe de manipulation basé sur la notion de *graphe dynamique*. Le cas de problème de manipulation sous contraintes continues est traité dans le chapitre 4. Nous proposons dans ce chapitre une approche plus générale dont l'efficacité repose sur une exploration *"directe"* du sous-espace $GRASP \cap PLACEMENT$.

[3]Un chemin τ a un débattement de ε, si toute configuration q appartenant à τ est à une distance d'au moins ε des obstacles.

Chapitre 3

Planification de tâches de manipulation sous contraintes discrètes

3.1 Introduction

Dans le chapitre précédent, nous avons présenté un schéma de résolution général de problèmes de planification de tâches de manipulation sous contraintes discrètes (ensemble fini de prises et de placements). Cependant, les techniques utilisées pour résoudre les différentes requêtes de transit/transfert (grossissement d'obstacles, graphe de visibilité, ...) permettant de calculer les connexions entre les différentes configurations de $\mathcal{CG} \cap \mathcal{CP}$ restent limitées à des exemples simples. La difficulté réside dans la détermination d'une méthode générique (permettant la résolution d'une grande classe de problèmes) et efficace pour gérer de nombreuses requêtes de planification dans des environnements partiellement changeants (placements des différents objets déplaçables).

Dans ce chapitre, nous proposons une approche (*approche par graphe dynamique*) capable de traiter ce type de requêtes. Cette approche repose sur l'utilisation des méthodes probabilistes pour le calcul d'un graphe initial couplées avec un mécanisme "paresseux" de mise à jour de ce graphe. Cette extension des approches probabilistes permet de gérer plus efficacement plusieurs requêtes de transit/transfert comme illustré à travers les exemples présentés à la fin de ce chapitre. Cette technique représente également une partie d'une approche plus générale permettant la résolution de problèmes de manipulation sous contraintes continues présentée dans le chapitre 4.

3.2 Graphe de manipulation et réseaux probabilistes

Nous nous plaçons pour la suite dans un espace de travail tridimensionnel ($\mathcal{W}$) contenant des obstacles statiques ($\mathcal{O}_1, ..., \mathcal{O}_k$), un robot $\mathcal{R}$ et des objets déplaçables ($\mathcal{M}_1, ..., \mathcal{M}_r$). La construction du graphe de manipulation, selon l'approche présentée dans le chapitre précédent, est composée de deux étapes : le calcul des différentes composantes connexes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ et leur connexion par des chemins de transit/transfert. Nous rappelons dans la suite la structure du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ dans le cas de contraintes de manipulation discrètes. Nous détaillerons également l'utilisation des approches probabilistes pour la construction du graphe de manipulation.

3.2.1 Le sous-espace $\mathcal{CG} \cap \mathcal{CP}$

Dans le cas d'un robot non-redondant et où les zones de placement ainsi que les prises sont discrètes, chaque composante connexe du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est réduite à une seule configuration. Le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est composé donc d'un ensemble fini de configurations. Notons par n_p le nombre de placements stables pour tous les objets déplaçables définis comme entrées du problème de manipulation et par n_g^i le nombre de prises possibles de l'objet $\mathcal{M}_i$. Le nombre de configurations N formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ vérifie alors :

$$N \leq \sum_{i=1}^{r} n_g^i \times \mathcal{A}_{n_p}^r, \textit{ où } r \textit{ est le nombre d'objets déplaçables présents dans l'environnement.}^1$$

Ou encore, si $n_1 = card(\mathcal{CP})$ et $n_2 = card(\mathcal{CG})$ alors le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est composé de N configurations avec $N \leq n_1 \times n_2$. Le calcul de ces configurations peut être effectué en utilisant un modèle géométrique inverse du robot.

La figure 3.1 (à gauche) illustre un exemple de problème de manipulation dans lequel le robot, partant de sa position initiale dans la première pièce, doit rejoindre sa position finale dans l'autre pièce. L'unique accès à la deuxième pièce est bloqué par une chaise (considérée comme étant le seul objet déplaçable dans cet exemple). Les deux configurations formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ (entrée du problème de manipulation correspondant à une seule prise et deux placements stables différents de l'objet) sont également représentées dans la figure 3.1 (droite).

3.2.2 Graphe de manipulation

Les nœuds du graphe de manipulation sont les différentes configurations formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$. Ces configurations sont calculées en utilisant le modèle géométrique inverse du robot et en prenant en considération tous les placements stables des objets ainsi que toutes les prises possibles. Il reste maintenant à déterminer les connexions entre ces configurations par des chemins de transfert et

[1] Cela suppose qu'un placement stable ne peut être occupé que par un seul objet à la fois. En particulier, un objet ne peut pas être "inséré" dans un autre. Nous supposons également que le nombre de placements stables n_n est supérieur au nombre d'objets déplaçables présents dans l'environnement r.

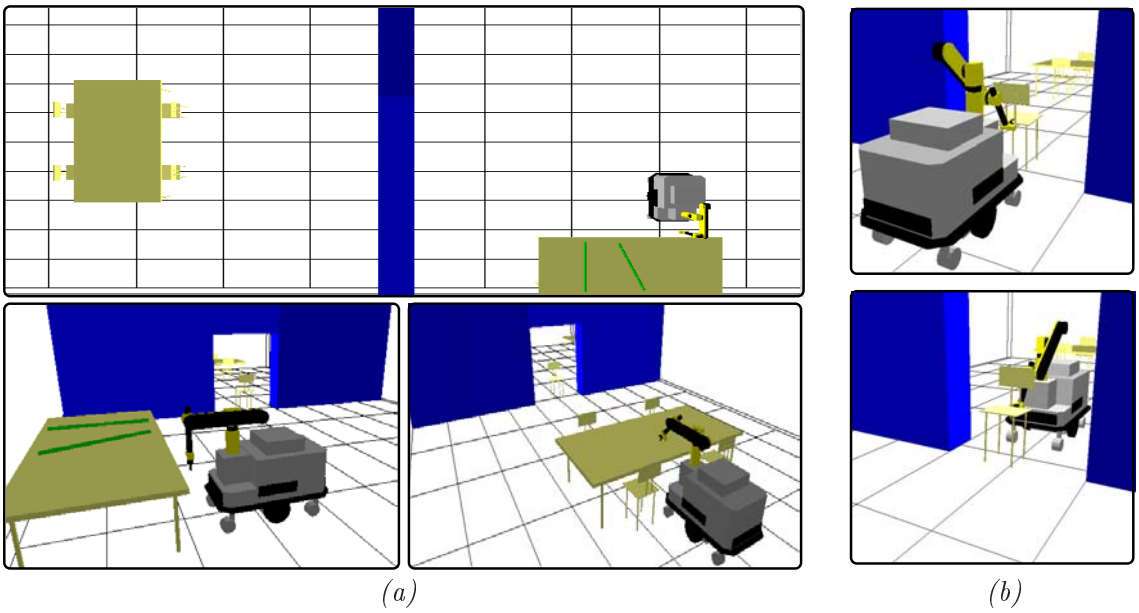


FIG. 3.1 – a) Exemple de problème de manipulation sous contraintes discrètes. b) Les deux configurations formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ entrées du problème.

des chemins de transit.

Chaque requête de transit (resp. de transfert) entre deux nœuds calculés dans $\mathcal{CG} \cap \mathcal{CP}$ est un problème de planification de mouvement à part entière. Chaque recherche de chemin de transit (resp. de transfert) est effectuée dans un sous-espace de $\mathcal{CP}$ (resp. de $\mathcal{CG}$)[2] spécifique à la situation courante des objets dans l'environnement (et à la prise courante également). Nous avons opté pour l'utilisation des approches probabilistes pour répondre à ces différentes requêtes de transit (resp. de transfert). Ce choix est motivé par la généricité de ces méthodes et par leur capacité à résoudre des problèmes de grande dimension. Elles permettent donc de traiter une classe de problèmes plus large que celle abordée dans [Alami 89]. Chaque requête de transit ou de transfert peut être traitée comme un problème indépendant de planification de mouvement. Un graphe capturant la connectivité de chaque "feuille" formant l'espace de recherche est alors calculé. L'ensemble de ces graphes constitue un *"méta-graphe* (figure 3.2) couvrant $\mathcal{CG} \cup \mathcal{CP}$.

La recherche du chemin de manipulation s'effectue dans le méta-graphe construit auquel sont ajoutées les différentes configurations formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$. Ces configurations sont ensuite connectées aux graphes correspondant par des chemins de transit et des chemins de transfert. Il est inutile de calculer les connexions de chaque configuration de $\mathcal{CG} \cap \mathcal{CP}$ avec tous les graphes construits. En effet, une configuration q_j est caractérisée par un n-uplet $(g_c^j, p_1^j, ..., p_r^j)$, où g_c^j représente la prise courante et chaque p_i^j indique le placement stable occupé par l'objet déplaçable $\mathcal{M}_i$. Un test de connexion

[2]Nous avons vu dans le chapitre 2 que les chemins de transit résident dans le sous-espace $\mathcal{CP}$, alors que les chemins de transfert sont inclus dans le sous-espace $\mathcal{CG}$.

par un chemin de transit est effectué uniquement entre q_j et les composantes du graphe calculées dans la feuille de $\mathcal{CP}$ correspondante (la feuille caractérisée par le même placement stable de tous les objets déplaçables que q_j). De la même façon, une connexion par chemin de transfert avec q_j est testée uniquement avec les composantes connexes du graphe calculées dans la feuille correspondante de $\mathcal{CG}$ (caractérisée par la même prise et les mêmes placements de tous les objets, à part celui saisi par le robot, que q_j).

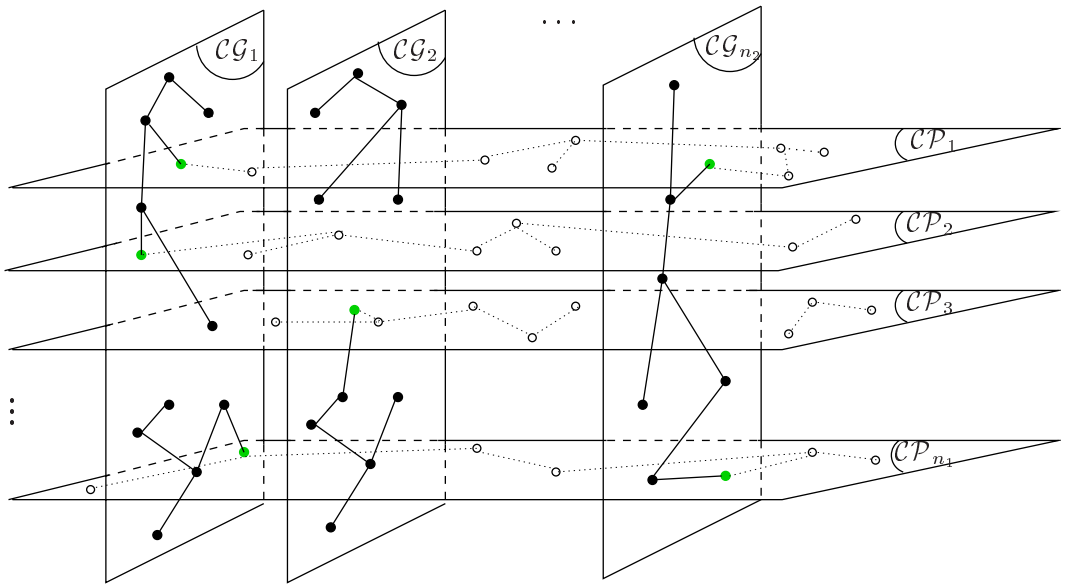


FIG. 3.2 – Structure du "méta-graphe" capturant la connectivité de l'espace de configurations valides ($\mathcal{CG} \cup \mathcal{CP}$).

La figure 3.2 présente la structure du *"méta-graphe"* construit par cette approche. Les nœuds représentés par "•" dans cette figure sont des configurations dans $\mathcal{CG}$. Ils sont connectés par des chemins de transfert (les traits continus). Les "∘" représentent des nœuds calculés dans $\mathcal{CP}$. Ils sont liés par des chemins de transit (les traits interrompus). Les nœuds représentés par "•" illustrent les différentes configurations formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$, obtenues à partir des prises et poses discrètes données comme entrées du problème.

Afin d'illustrer la construction du méta-graphe, nous reprenons le scénario présenté dans la figure 3.1. Le sous-espace $\mathcal{CG}$ est composé, pour cet exemple, d'une seule feuille relative à la seule prise définie comme entrée du problème. Le sous-espace $\mathcal{CP}$ est composé de deux feuilles correspondant chacune à l'un des deux placements stables définis. Trois graphes sont ainsi calculés en utilisant une approche probabiliste tenant compte de l'état courant de l'environnement (un graphe par feuille). La figure 3.3 illustre ces graphes calculés par une méthode probabiliste de type *"Visibilité-PRM"* [Nissoux 99a].

Les deux graphes (a) et (c) de cette figure sont calculés dans $\mathcal{CP}$ (chacun dans l'une des deux feuilles composant ce sous-espace). Les différentes arêtes de ces graphes sont des chemins de transit.

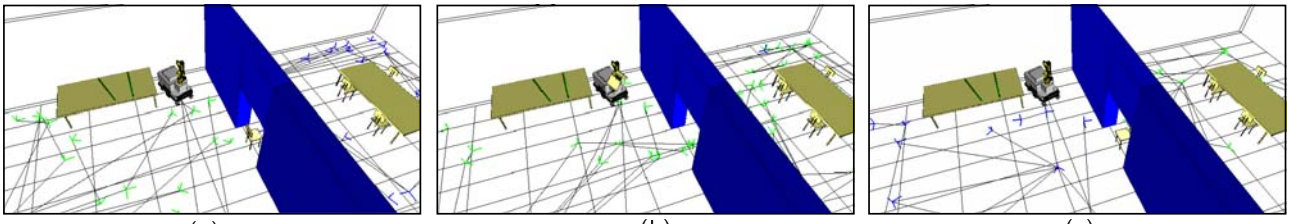


FIG. 3.3 – Les trois graphes calculés par la méthode probabiliste (*Visibility-PRM*).

Le graphe (b) est calculé dans $\mathcal{CG}$. Une arête connectant deux nœuds de ce graphe indique l'existence d'un chemin de transfert entre ces derniers. Notons que chacun des deux graphes (a) et (c) est composé de deux composantes connexes. Chaque composante couvre l'espace des configurations libres de chacune des deux pièces. Le passage connectant ces dernières étant bloqué par l'objet déplaçable (la chaise). Cependant, le graphe (b) est formé d'une seule composante connexe, puisque dans $\mathcal{CG}$ le robot se déplace solidairement avec l'objet.

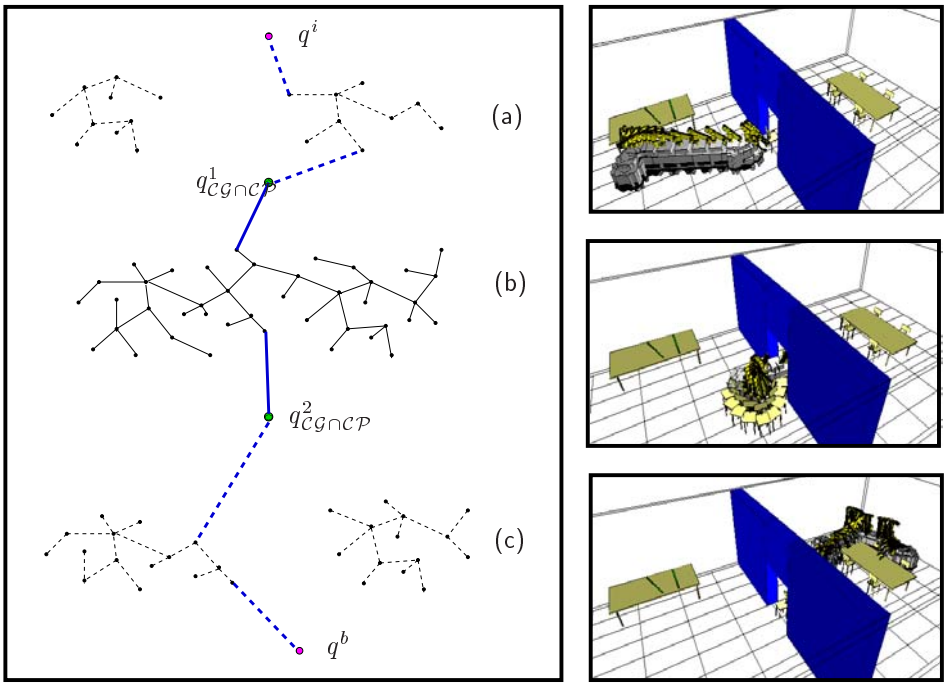


FIG. 3.4 – Le "méta-graphe" construit à partir des trois graphes calculés (gauche) permettant le calcul du chemin de manipulation solution du problème posé (droite).

Le méta-graphe calculé à partir de ces trois graphes ainsi que le chemin de manipulation solution du problème posé sont illustrés dans la figure 3.4. Les noeuds $q^1_{\mathcal{CG}\cap\mathcal{CP}}$ et $q^2_{\mathcal{CG}\cap\mathcal{CP}}$ correspondent aux deux configurations formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$.

Commentaires :

La complexité du problème de manipulation sous contraintes discrètes croît exponentiellement en fonction du nombre d'objets déplaçables présents dans l'environnement. Elle est polynomiale en fonction du nombre de prises discrètes ainsi que du nombre de placements stables définissant le problème de manipulation. Une borne supérieure du nombre de graphes nécessaires pour construire le graphe de manipulation (également pour le nombre des configurations de $\mathcal{CG} \cap \mathcal{CP}$) peut être donnée par : $[\max_i(n_p^i)]^r \times \sum_{i=0}^{r} n_g^i$, où r est le nombre d'objets déplaçables présents dans l'environnement, n_p^i et n_g^i représentent respectivement le nombre de placements stables et prises possibles définis pour l'objet $\mathcal{M}_i$. La courbe présentée dans la figure 3.5 illustre l'évolution du nombre de configurations formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ en fonction du nombre d'objets déplaçables et du nombre de placements stables (le nombre de prises étant fixé à trois pour chaque objet)[3].

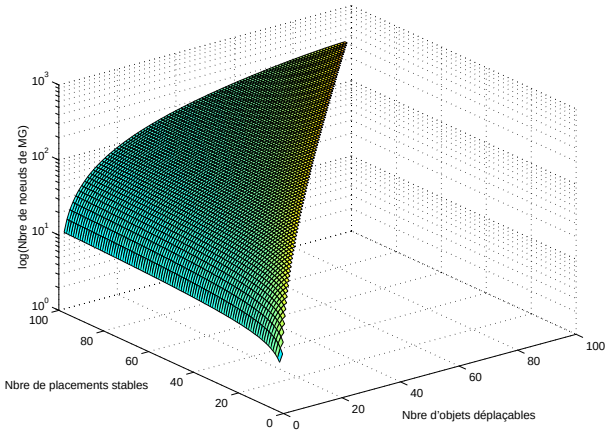


FIG. 3.5 – Evolution du nombre de configurations dans $\mathcal{CG} \cap \mathcal{CP}$ en fonction du nombre d'objets déplaçables présents dans l'environnement et de placements stables définis.

Par ailleurs, chacune des requêtes de planification de mouvement (de transit ou de transfert) est calculée dans un environnement partiellement modifié puisqu'elle correspond soit à une prise particulière soit à un placement spécifique de l'objet déplaçable. Plus précisément, seule une petite région de l'environnement est modifiée entre une requête de planification et une autre. Dans la section suivante, nous présentons une approche exploitant cette notion de changement partiel de l'environnement. Elle se base sur une mise à jour dynamique d'un seul graphe calculé pour le robot seul pour la résolution des différentes requêtes de planification. Cette approche permet également l'amortissement du coût de chaque requête de planification en réutilisant des chemins calculés dans une phase préalable.

[3]Si N_n représente le nombre de configurations formant $\mathcal{CG} \cap \mathcal{CP}$, n_p et n_g correspondent respectivement au nombre de placements stables et de prises définis pour chaque objet, et r nombre d'objets présents alors une estimation de N_n peut être donnée par : $n_g \times r \times \frac{n_p!}{(n_p - r)!}$.

3.3 Approche par graphe dynamique

Après avoir calculé les différentes configurations formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$, la seconde étape de l'approche adoptée ([Alami 89]) consiste à déterminer leur connexion par des chemins de transfert et des chemins de transit. La construction d'un graphe en utilisant une approche probabiliste pour chaque feuille de $\mathcal{CP}$ et de $\mathcal{CG}$ serait peu efficace. En effet, le nombre de graphes nécessaires pour obtenir le chemin de manipulation croît exponentiellement en fonction du nombre d'objets et de façon polynomiale en fonction du nombre de placements et de prises définis. De plus, nous avons vu dans le paragraphe précédent que les différentes requêtes de planification (requêtes de transit et de transfert) sont calculées dans des environnements partiellement modifiés. Il nous faut donc trouver une méthode capable de gérer efficacement un ensemble de requêtes de planification dans des environnements localement modifiés et qui correspondent soit à une prise spécifique soit à un placement donné de l'un des objets.

L'utilisation directe des méthodes probabilistes (de type *PRM* [Kavraki 95a], ou *Visibility-PRM* [Siméon 00], ...) pour la construction de graphe de manipulation s'avérerait peu performante à cause de la combinatoire du problème discutée dans la section précédente. Les méthodes probabilistes dites par *"diffusion"* (RPP [Barraquand 91] ou RRT [Kuffner 00d], ...), quant à elles, ne nécessitent pas une construction préalable de graphe capturant la connectivité de l'espace libre, et sont également performantes dans la résolution de problèmes de planification à requête unique. Cependant, l'utilisation directe de ces méthodes ne permet pas d'exploiter le changement partiel de l'environnement entre chaque requête de planification.

3.3.1 Approche proposée

L'idée est de combiner une méthode probabiliste classique (PRM, Visibilité-PRM, ...) avec une des méthodes dites par *"diffusion"* (RPP, RRT, ...). La première méthode nous permet de bénéficier de l'avantage d'avoir un graphe réutilisable à chaque résolution d'une requête ; alors que la deuxième méthode apporte son efficacité dans la résolution d'une requête unique. Nous proposons donc de calculer dans une étape initiale un ensemble de chemins (un graphe) en considérant uniquement les obstacles statiques, et d'utiliser un mécanisme de mise à jour paresseux permettant de contourner le problème de changement partiel de l'environnement entre deux requêtes de planification.

L'approche proposée, approche par *graphe dynamique* [Sahbani 01a, Sahbani 01b], consiste à calculer dans une première étape un graphe pour le robot seul dans l'environnement sans prendre en considération la présence des objets déplaçables. Ce graphe est calculé en utilisant une approche probabiliste classique. Avant d'exécuter une phase de recherche dans ce graphe afin de répondre à une requête de transit ou une requête de transfert donnée, le graphe est mis à jour dynamiquement de la manière suivante.

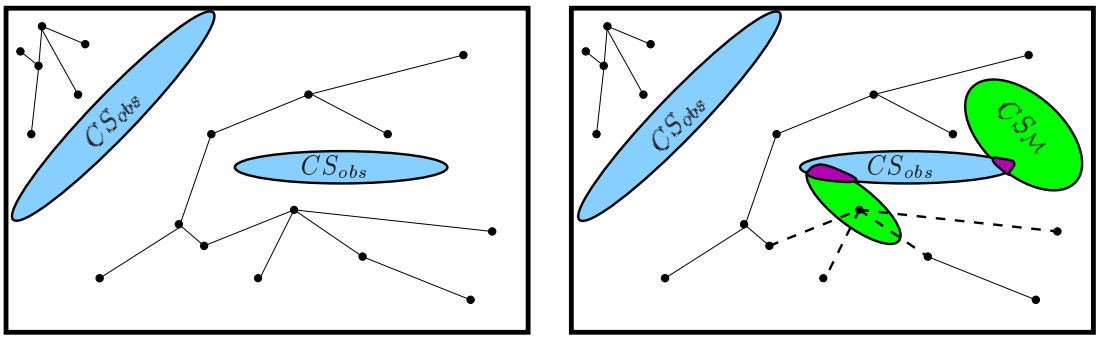


FIG. 3.6 – Le graphe marqué en fonction du placement des objets déplaçables.

Un test de collision est appliqué à chaque arête du graphe en prenant en considération, cette fois-ci, la position courante des différents objets déplaçables dans l'environnement (dans le cas d'une requête de transfert, la validité des arêtes est testée en tenant compte de l'objet déplaçable attaché au robot et de la position courante des autres objets). Dans le cas où l'un des chemins connectant deux nœuds du graphe initial est en collision, l'arête correspondante est marquée *"bloquante"* (représentée par un trait interrompu dans la figure 3.6). La recherche des chemins de transit (resp. de transfert) s'effectue alors dans le graphe ainsi marqué. Notons que le marquage des arêtes dépend de la requête de planification. Il est initialisé chaque fois qu'une phase de recherche dans le graphe est exécutée. La figure 3.6 illustre la mise à jour dynamique d'un graphe en fonction de la position courante des objets déplaçables dans l'environnement.

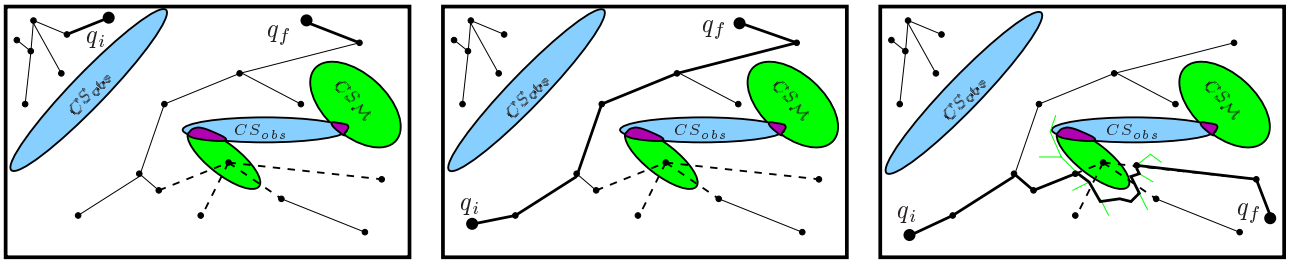


FIG. 3.7 – Recherche de chemin dans le graphe marqué : a) il n'existe pas de solution, b) un chemin est trouvé, c) une mise à jour locale des arêtes bloquantes.

Trois cas possibles peuvent se présenter lors de la phase de recherche : il n'existe pas de chemin dans le graphe répondant à la requête, ou le chemin trouvé ne contient aucune arête marquée, ou encore le chemin solution contient des arêtes marquées. Dans le premier cas, le sous-problème de planification n'admet pas de solution. En effet, dans ce cas, il n'existe pas de chemin répondant à la requête de planification même en absence d'objets déplaçables[4]. Ce cas peut être rencontré lorsque le graphe initial

[4]Nous supposons que le graphe calculé dans la phase initiale représente un couverture adéquate de l'espace libre.

est composé de plusieurs composantes connexes comme illustré dans la figure 3.7 (gauche), ou encore lorsqu'un ou plusieurs objets obstruent un passage dans l'environnement. Le deuxième cas correspond au cas où le chemin trouvé à l'issue de la phase de recherche ne contient aucune arête marquée. Ce chemin prend en considération le placement des différents objets déplaçables (ainsi que la présence d'un objet attaché au robot dans le cas d'une requête de transfert). Il représente donc une solution valide pour la requête de planification posée. Ce cas est illustré dans la figure 3.7 (figure du milieu). Le dernier cas correspond au cas intermédiaire dans lequel le chemin trouvé contient nécessairement des arêtes marquées *"bloquantes"*. Dans ce cas, le problème est traité localement par l'algorithme. Les arêtes marquées ne sont pas prises en considération lors de la phase de recherche (elles sont momentanément supprimées). Une connexion entre les deux extrémités des arêtes supprimées est alors testée en utilisant une méthode probabiliste dite par *"diffusion"* (par exemple *RRT* [Kuffner 00d]). Si cette connexion est trouvée, le chemin solution du sous-problème de planification posé est composé des arêtes non bloquantes trouvées lors de la phase de recherche et des chemins remplaçant les arêtes supprimées. Ce cas est illustré dans la figure 3.7 (figure de droite). Dans le cas contraire (cas où la connexion entre les extrémités des arêtes bloquantes a échoué), l'algorithme considère que la requête de planification posée n'admet pas de solution. Cette étape peut être vue comme une mise à jour dynamique du graphe initial.

3.3.2 Choix effectués

Choix de la méthode probabiliste utilisée pour la construction du graphe initial :
La construction du graphe initial est une étape importante pour notre approche. Ce graphe doit représenter un couverture adéquate de l'espace libre permettant ainsi, dans l'étape suivante, de répondre avec une quasi-certitude (puisque les approches probabilistes vérifient uniquement une complétude en probabilité) aux différentes requêtes de planification posées. Par ailleurs, une question se pose : faut-il choisir une approche probabiliste conduisant à un graphe dense (de type PRM par exemple) ou plutôt une approche fournissant un graphe de taille réduite mais offrant une couverture comparable de l'espace libre (Visibilité-PRM) ?

Le premier choix conduit à un graphe contenant un grand nombre d'arêtes. La phase de validation et de marquage des arêtes devient ainsi une étape coûteuse en temps de calcul (à cause du nombre d'appel au détecteur de collision nécessaire pour tester la validité de toutes les arêtes de ce graphe). Ce problème n'est pas soulevé si le graphe initial était de faible taille. D'autre part, choisir une approche probabiliste conduisant à un graphe de taille réduite présente le risque d'avoir une grande partie de ce graphe non-valide (la plupart de ces arêtes sont marquées bloquantes) pour des requêtes de planification données, ce qui fait perdre l'avantage d'avoir des chemins précalculés et réutilisables d'une requête de planification à une autre. Le choix de la méthode probabiliste utilisée pour la construction du graphe initial dépend fortement du problème de manipulation (environnement dégagé, taille des objets, ...). Nous verrons, à travers les exemples présentés dans la section suivante, les répercussions engendrées

par le choix de l'une de ces deux techniques (pour le calcul du graphe initial) sur les performances de l'approche.

Choix de la technique permettant la mise à jour du graphe initial :
Concernant la mise à jour du graphe initial, nous avons opté pour l'utilisation de la méthode *RRT* (Rapid-exploring Random Tree) [Kuffner 00d]. Cette méthode consiste à faire évoluer aléatoirement deux arbres ayant comme racines les configurations initiale et but. Les deux arbres explorent l'espace des configurations autour de leur racine respective et progressent l'un vers l'autre selon une heuristique simple. La complexité de cette méthode dépend de la longueur du chemin solution, ce qui concorde avec le principe de notre approche (avoir une méthode permettant de connecter localement un ensemble de chemins déjà calculés).

Remarque :
Du point de vue formel, la complétude en probabilité de l'approche par *graphe dynamique* est une conséquence directe de la complétude de *RRT* [P.Cheng 02] ainsi que celle des méthodes probabilistes (*PRM* [Švestka 96] ou *Visibilité-PRM* [Nissoux 99a]).

3.3.3 Exemples de problème de manipulation sous contraintes discrètes

Exemple de problème de manipulation avec deux objets déplaçables :
Considérons le problème de manipulation illustré dans la figure 3.8. Le robot $\mathcal{R}$ doit déplacer les deux objets $\mathcal{M}_1$ et $\mathcal{M}_2$ de leur position initiale (figure de droite) à leur placement final (figure de gauche). A chaque objet déplaçable sont associés trois placements stables : les positions initiale et finale et un placement intermédiaire illustré dans la figure 3.9. Une seule prise discrète est définie pour chaque objet. Le robot peut saisir un objet uniquement d'un seul côté comme illustré dans la figure 3.9 (figure de droite).

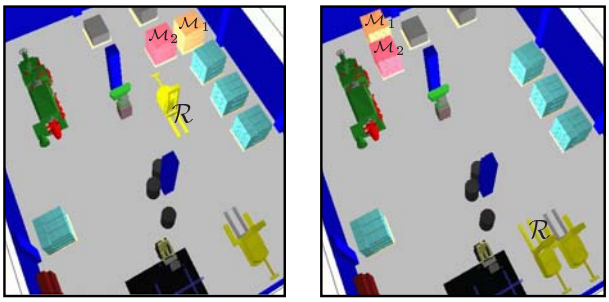


FIG. 3.8 – Exemples de problème de manipulation avec deux objets déplaçables.

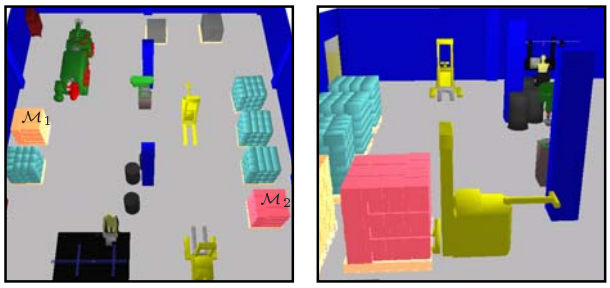


FIG. 3.9 – Les entrées du problème (placements intermédiaires et prises).

Nous avons utilisé cet exemple pour montrer les répercussions de l'utilisation de la méthode *PRM-classique* ainsi que celles de *Visisbilité-PRM* (pour la construction du graphe initial) sur le temps de résolution du problème. La figure 3.10 présente les deux graphes calculés pour le robot seul selon les

deux méthodes.

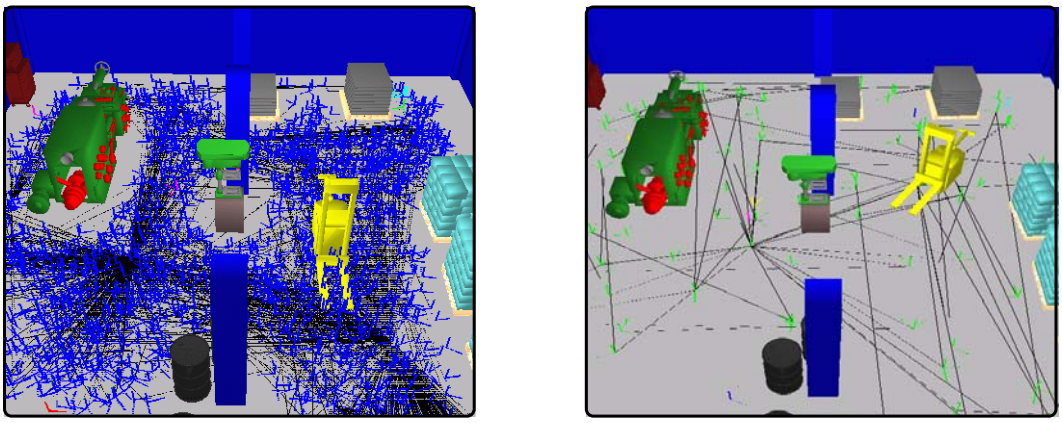


FIG. 3.10 – Les graphes calculés pour le robot en considérant uniquement les obstacles statiques et en utilisant respectivement *PRM* (gauche) et *Visibilité-PRM* (droite) .

Les caractéristiques des deux graphes sont présentés dans le tableau 3.10. Les temps de calculs donnés sont obtenus sur une station SUN ultra 10. L'objectif de ce tableau n'est pas de comparer les performances des deux méthodes dans la résolution de problème de planification de mouvement. Nous avons seulement calculé deux graphes ayant une couverture de l'espace libre comparable. L'estimation de la couverture est calculée en tirant aléatoirement n configurations libres ($n = 10^6$ pour cet exemple) et en testant la possibilité de les connecter aux nœuds du graphe. La couverture de l'espace libre à partir du graphe calculé par la méthode probabiliste de type *PRM* est estimée à 99.43%, alors que celle du graphe calculé par *Visibilité-PRM* est estimée à 98.83%. On peut noter dans ce tableau la différence significative de taille des graphes ainsi calculés. Cette taille aura-elle alors une grande influence sur le temps nécessaire pour la résolution du problème de manipulation et en particulier sur la phase de validation ?

Calcul du graphe initial	**PRM**	**Visibilité-PRM**
Nb. configurations tirées	4712	10209
Nb. configurations libres	1457	3093
Nb. appels à la méthode locale	7029	44692
Nb. appels au détecteur de collision	103128	323393
Taille du graphe	**1457**	**115**
Temps CPU	26.55 sec	122.36 sec

TAB. 3.1 – Caractéristiques des graphes calculés

Le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est composé dans cet exemple de 16 configurations. La solution obtenue pour ce problème est présentée dans la figure 3.11. Le chemin de manipulation est composé de 11

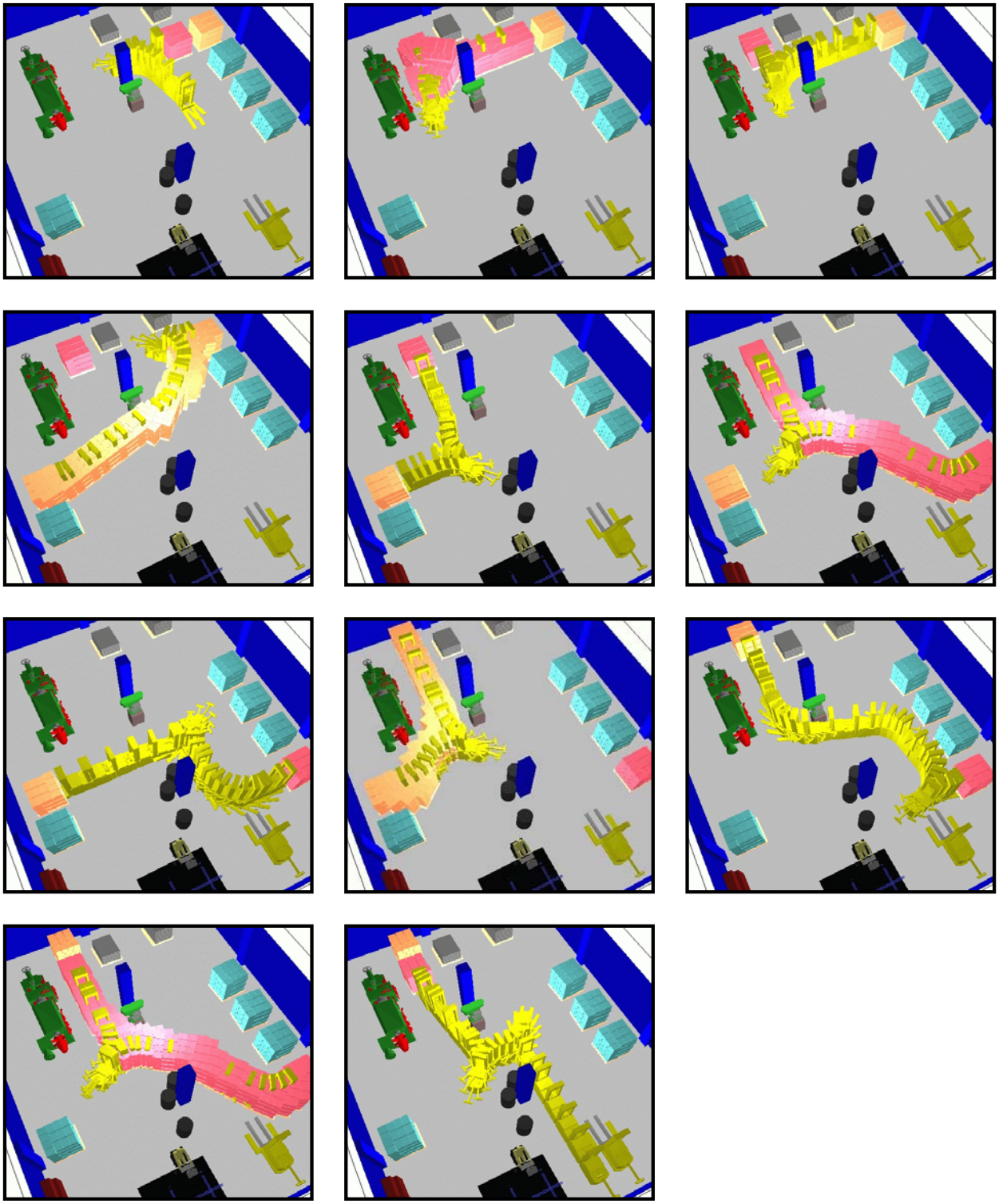


FIG. 3.11 – Le chemin de manipulation calculé.

chemins élémentaires (6 chemins de transit et 5 de transfert). Une grande partie du temps de calcul est passé dans le marquage des arêtes de graphe. Une fois cette étape effectuée, la recherche de chemin de transit (resp. de transfert) est accomplie en moins d'une seconde pour les deux méthodes utilisées dans

la construction du graphe initial. Cependant, pour le graphe initial calculé par la méthode probabiliste de type *PRM*, chaque mise à jour (marquage des arêtes) nécessite en moyenne 19.5 secondes, alors que pour celui construit par *Visibilité-PRM* la moyenne est de 1.3 secondes. Ainsi, dans le cas où le graphe initial est construit par *PRM*, le calcul des différents chemins de transit et de transfert composant le chemin de manipulation nécessite au moins $16 \times 19.5 = 312$ secondes. Dans le cas d'un graphe initial construit par *Visibilité-PRM*, ce calcul ne nécessite que $16 \times 1.3 = 20.8$ secondes. L'utilisation de *Visibilité-PRM* pour la construction du graphe initial est donc plus performante dans cet exemple que celle de *PRM*. Un graphe de taille réduite permet de limiter le coût de mise à jour dynamique tout en offrant une couverture suffisante de l'espace libre.

Par ailleurs, construire un graphe pour chaque feuille composant les sous-espaces $\mathcal{CP}$ et $\mathcal{CG}$, afin d'obtenir le chemin de manipulation solution du problème, aurait nécessité 420 secondes. L'approche par graphe dynamique est donc plus performante qu'une recherche exhaustive consistant à construire un ensemble discret de graphes.

Exemple industriel :

Le problème de manipulation présenté dans la figure 3.12 nous a été proposé par *EDF* dans le cadre du projet *MOLOG* [Siméon 01a]. Il consiste à trouver un chemin permettant au robot (pont roulant) d'évacuer le couvercle d'une turbo-pompe (l'objet déplaçable dans cet exemple) pour effectuer des opérations de maintenance.

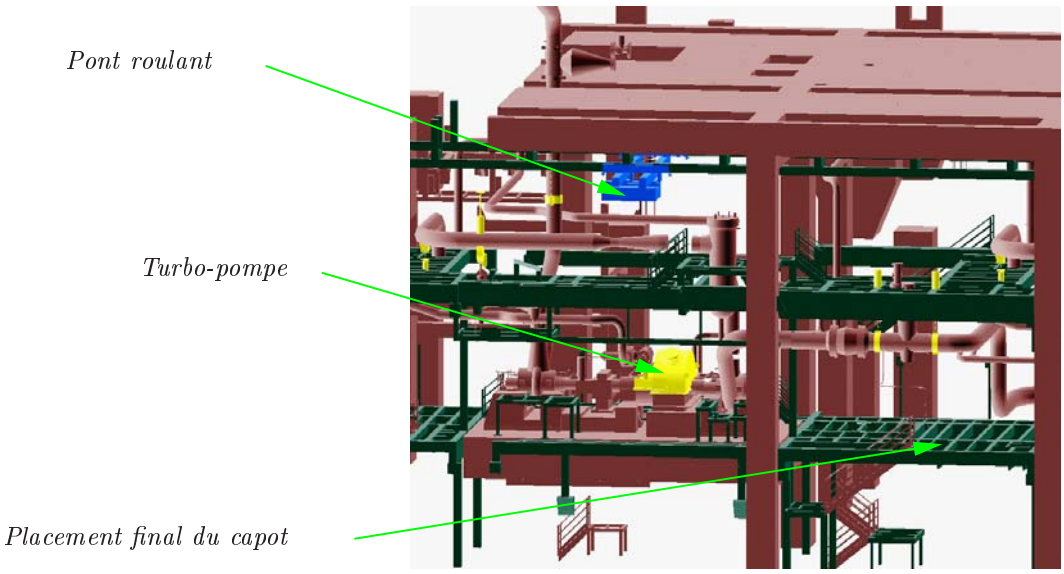


FIG. 3.12 – Un exemple industriel de problème de manipulation.

Ce problème de manipulation comporte plusieurs difficultés. Il s'agit d'un environnement hautement encombré et composé de plusieurs milliers de facettes, ce qui augmente considérablement le coût

de chaque appel au détecteur de collision. Nous avons testé également sur cet exemple l'influence de l'utilisation de la méthode *PRM* ainsi que celle de *Visibilité-PRM* (pour la construction du graphe initial) sur le temps de résolution du problème. Les deux graphes initiaux calculés en considérant uniquement les obstacles statiques sont présentés dans la figure 3.13. La couverture de l'espace libre à partir du graphe calculé par la méthode *PRM* est estimée à 99.15% alors que celle du graphe calculé par *Visibilité-PRM* est estimée à 98.78%

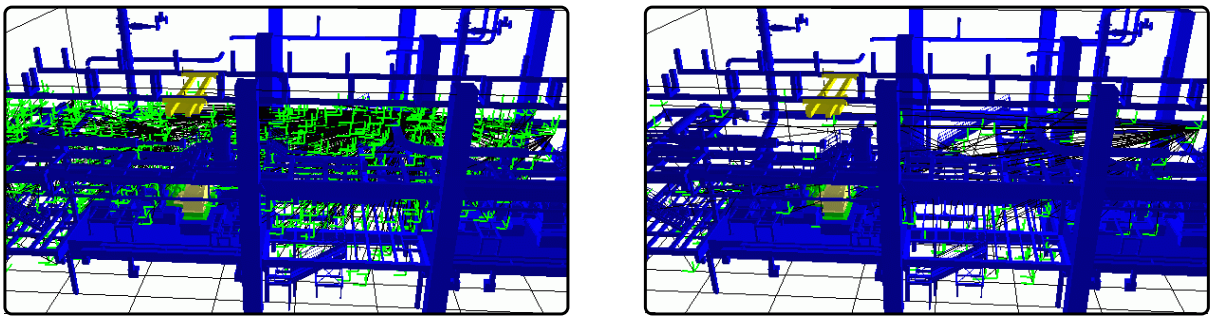


FIG. 3.13 – Le graphe initial calculé par : *PRM* (à gauche) et par *Visibilité-PRM* (à droite).

Le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ dans cet exemple est composé uniquement de deux configurations (le robot saisissant l'objet occupant respectivement sa position initiale et finale). Le chemin de manipulation solution à ce problème est constitué d'une succession de trois chemins : un chemin de transit permettant au robot de saisir l'objet, un chemin de transfert dans lequel le robot déplace l'objet de sa position initiale vers son placement final, et un chemin de transit permettant au robot d'effectuer une opération de pose et d'atteindre sa configuration finale. Le calcul du chemin de transfert représente l'étape la plus coûteuse (en temps de calcul) pour cet exemple. Ceci est dû au fait que l'objet déplaçable est l'élément le plus volumineux dans cet environnement. Le chemin de transfert a été calculé en 21 secondes en utilisant comme graphe initial celui construit par l'approche de type *PRM*; alors qu'en utilisant le graphe construit par l'approche *Visibilité-PRM*, ce chemin est trouvé en 40 secondes. Les chemins de transit ont été calculés en moins d'une seconde. L'utilisation de l'approche probabiliste de type *PRM* pour le calcul du graphe initial s'avère pour cet exemple plus efficace que celle de l'approche *Visibilité-PRM*. En effet, la densité du graphe calculé par la première approche (PRM) permet de moins solliciter l'étape de mise à jour des arêtes *"bloquantes"* en présence de l'objet déplaçable.

La figure 3.14 illustre le chemin de transfert calculé ainsi que la position initiale et finale de l'objet déplaçable. Une méthode locale de type *Manhattan* a été utilisée pour le calcul des différents chemins locaux.

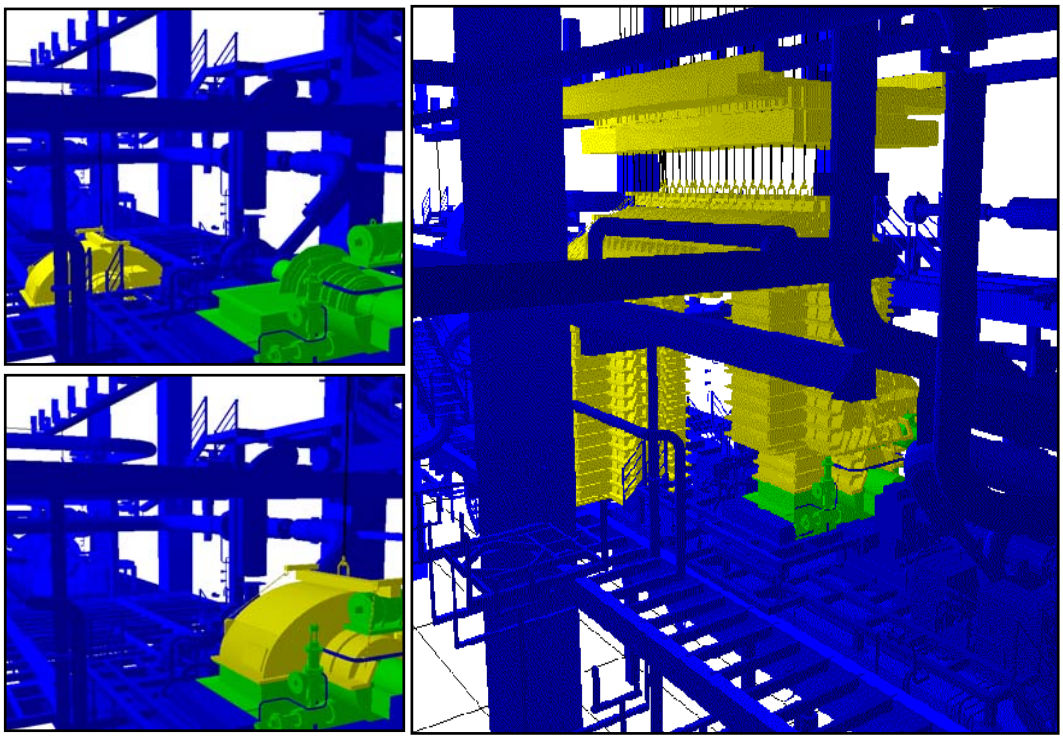


FIG. 3.14 – Pont-roulant transportant le capot d'une turbo-pompe dans une centrale nucléaire (modèle EDF).

3.4 Conclusion

Dans ce chapitre, nous avons présenté notre contribution dans le cas discret du problème de manipulation. Une application directe des méthodes probabilistes aux problèmes de manipulation a été présentée dans la première partie de ce chapitre. Elle permet de résoudre de façon exhaustive des problèmes de manipulation simples. Elle nous a permis également d'illustrer la structuration de l'espace de recherche (les différentes *"feuilles"* formant les sous-espaces $\mathcal{CP}$ et $\mathcal{CG}$) et de montrer la combinatoire de ce type de problème en fonction du nombre d'objets, prises et placements définis. L'approche par graphe dynamique, présentée dans la seconde partie de ce chapitre, est capable de traiter plus efficacement des problèmes réels de manipulation dans des environnements hautement encombrés. Nous avons montré à travers les exemples présentés les performances de cette méthode à gérer plusieurs requêtes de planification dans des environnements partiellement changeants (par le placement des objets déplaçables ou la prise courante).

La technique du graphe dynamique sera réutilisée dans l'approche "continue" présentée dans le chapitre suivant pour connecter les différentes composantes connexes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$. Cependant, la structuration de ce sous-espace dans le cas continu de problèmes de manipulation est beaucoup plus complexe que dans le cas discret. Il ne s'agit plus d'un ensemble de configurations isolées mais plutôt d'un ensemble discret de domaines continus. Dans le chapitre suivant, nous présentons une méthode effective permettant de capturer la connectivité de $\mathcal{CG} \cap \mathcal{CP}$ dans une structure de graphe. Les

connexions des différentes composantes connexes de ce graphe (connexions en dehors de $\mathcal{CG} \cap \mathcal{CP}$) sont effectuées selon l'approche par graphe dynamique.

Chapitre 4

Planification de tâches de manipulation sous contraintes continues

4.1 Introduction

Dans le cas de problèmes de planification de tâches de manipulation sous contraintes continues, le schéma de résolution présenté dans le chapitre 2 reste limité à un cas particulier de problèmes de manipulation (environnement 2D, robot polygonal en translation, ...). Cette limitation est due à la non-existence d'une méthode efficace permettant le calcul des différentes composantes connexes du sous-espace $GRASP \cap PLACEMENT$ ($\mathcal{CG} \cap \mathcal{CP}$) et donc d'accomplir la première étape de ce schéma. Dans le cas de contraintes discrètes, ce problème n'est pas soulevé puisque chaque composante connexe de ce sous-espace est réduite à une seule configuration (cas étudié dans le chapitre précédent).

Dans ce chapitre, nous présentons une approche générique de planification de tâches de manipulation capable de traiter le cas d'ensembles continus de placements stables et de prises possibles [Siméon 02a, Siméon 02b]. Cette approche est inspirée du schéma de résolution introduit dans [Alami 95] (et détaillé dans le chapitre 2). Nous présentons également une méthode permettant de capturer la connectivité du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ dans un réseau probabiliste. Nous détaillerons dans la première section de ce chapitre le principe de notre approche ainsi que les différents éléments qui la constituent. Le second volet de ce chapitre est consacré à la présentation des différents algorithmes développés selon notre approche.

4.2 Approche proposée

4.2.1 Principe

L'approche que nous proposons étend le schéma de résolution présenté dans le chapitre 2 en fournissant une méthode générique permettant la détermination des différentes composantes connexes du sous-espace $GRASP \cap PLACEMENT$. Elle est composée de trois étapes : construction du graphe de manipulation, recherche de chemin dans ce graphe, et phase d'optimisation et de lissage du chemin de manipulation trouvé.

La première phase conduit à la construction d'un graphe de manipulation couvrant l'espace des configurations admissibles $\mathcal{CG} \cup \mathcal{CP}$. Un chemin connectant les deux configurations initiale et but est ensuite cherché dans ce graphe. Ce chemin, lorsqu'il existe, est composé de trois types de sous-chemins : des chemins de transit, des chemins de transfert et des chemins dans $\mathcal{CG} \cap \mathcal{CP}$. Ces derniers ne représentent pas des chemins admissibles du point de vue de la manipulation. Néanmoins, grâce à la propriété de réduction (cf. section 2.4), ils sont transformables en une séquence finie de chemins de transit et de transfert. Le chemin de manipulation est obtenu après avoir transformé tous les chemins non-admissibles (calculés dans $\mathcal{CG} \cap \mathcal{CP}$) en séquence transit/transfert. La dernière phase de l'approche consiste à optimiser et lisser le chemin de manipulation trouvé (en minimisant par exemple le nombre de séquences de prise et de pose).

Concernant la construction du graphe de manipulation $\mathcal{MG}$, nous avons gardé le même schéma de résolution présenté dans le chapitre 2. Cette phase est ainsi composée de deux étapes :

- le calcul des différentes composantes connexes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$,
- la détermination de leur connectivité par :
  - des chemins de transfert dans $\mathcal{CG}$,
  - et des chemins de transit dans $\mathcal{CP}$.

Nous détaillons dans la suite de cette section les deux étapes composant la phase de construction du graphe de manipulation. La recherche de chemin dans le graphe ainsi que la phase d'optimisation sont inspirées des techniques utilisées par les méthodes probabilistes. Ces deux phases ainsi que leurs algorithmes respectifs sont développés dans la section suivante. Par ailleurs, la propriété de réduction a été démontrée uniquement dans le cas d'un seul robot en présence d'un seul objet déplaçable ($\mathcal{M}$). Rappelons que cette propriété nous permet de conclure sur la validité des chemins calculés dans $\mathcal{CG} \cap \mathcal{CP}$ et donc de montrer que la connectivité de ce sous-espace reflète bien l'existence d'un chemin de manipulation. Par conséquent, dans ce qui suit, nous nous plaçons dans ce cas de problème de manipulation : en présence d'un seul objet déplaçable.

4.2.2 Composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$

Rappel : Les sous-espaces $\mathcal{CG}$ et $\mathcal{CP}$

Dans ce paragraphe, nous adapterons les définitions des sous-espaces $\mathcal{CG}$ et $\mathcal{CP}$, présentées dans le chapitre 2, au cas où un seul objet déplaçable est présent dans l'environnement .

Soit $\mathcal{P}$ l'ensemble des placements stables caractérisant le problème de manipulation posé. Cet ensemble est réduit aux différents placements stables de l'unique objet déplaçable $\mathcal{M}$, ainsi $\mathcal{P} = \mathcal{P}_{\mathcal{M}}$. Le sous-espace $PLACEMENT$ ($\mathcal{CP}$) est l'ensemble des configurations du système composite dans lesquels l'objet déplaçable occupe une configuration stable. Plus formellement, ce sous-ensemble peut être défini de la façon suivante : $\mathcal{CP} = \{q_c = (q_{\mathcal{R}}, q_{\mathcal{M}}) \mid q_{\mathcal{R}} \in (CS_{\mathcal{R}})_{libre} \ et \ q_{\mathcal{M}} \in \mathcal{P}\}$.

De la même façon, soit $\mathcal{G}$ l'ensemble des classes de prises possibles définies comme entrées du problème. Chaque classe $\mathcal{G}_k$ de $\mathcal{G}$ est caractérisée par une transformation homogène $\mathcal{F}_k^T$. Le sous-espace $GRASP$ est l'ensemble des configurations du système dans lesquelles le robot déplace l'objet déplaçable. Plus formellement, $\mathcal{CG} = \{q_c = (q_{\mathcal{R}}, q_{\mathcal{M}}) \mid \ \exists \ \mathcal{F}_k^T : \mathcal{F}_k^T(q_{\mathcal{R}}) = q_{\mathcal{M}}\}$.

Le sous-espace $\mathcal{CG} \cap \mathcal{CP}$:

La première étape de la construction du graphe de manipulation consiste à déterminer les différentes composantes connexes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$. L'exploration de cette sous-variété est l'étape importante sur laquelle repose notre approche. Cependant, contrairement au cas discret dans lequel le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est composé d'un ensemble fini de configurations (cas étudié dans le chapitre 3), cette variété est constituée d'un ensemble discret de domaines continus dans le cas de problème de manipulation sous des contraintes continues.

L'espace de recherche, bien que limité à un sous-espace de dimension plus faible que CS_{libre}, reste hautement dimensionné. En effet, l'espace composite du système global : $CS = CS_{\mathcal{R}} \times CS_{\mathcal{M}}$ est de dimension $n + 6$, où n est le nombre de degrés de liberté du robot. Chaque domaine continu du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est caractérisé par une prise et un placement. Dans le cas où le placement de l'objet déplaçable est décrit par un système d'équations de degré α, la dimension du sous-espace $\mathcal{CP}$ relatif à ce placement est $n + 6 - \alpha$. De la même façon, si β est le degré du système d'équations relatif à la classe de prise $\mathcal{G}_k$, le sous-espace $\mathcal{CG}$ (correspondant à cette classe de prise) est de dimension $n+6-\beta$. Par conséquent, la sous-variété de $\mathcal{CG} \cap \mathcal{CP}$ correspondante à ce placement et à cette classe de prise est décrite par un système d'équations de degré maximal $\alpha + \beta$ (les équations décrivant les prises et les placements peuvent être dépendantes). La dimension de cette sous-variété vérifie alors :

$$n + 6 - \alpha - \beta \ \leq \ dim(\mathcal{CG} \cap \mathcal{CP}) \ \leq \ min\{n + 6 - \alpha \ , \ n + 6 - \beta\}$$

Il est clair que la dimension de la sous-variété $\mathcal{CG} \cap \mathcal{CP}$ est plus petite que celle de CS (ensemble de configurations du système composite). Néanmoins, la dimension de ce sous-espace reste élevée. Pour

cela, nous avons opté pour l'utilisation des approches probabilistes afin de capturer la connectivité de $\mathcal{CG} \cap \mathcal{CP}$ dans un graphe. Ce choix a été motivé par la généricité de ces approches ainsi que par le potentiel de ces méthodes à faire face à des espaces hautement dimensionnés.

Par ailleurs, dans le sous-espace $\mathcal{CG} \cap \mathcal{CP}$, l'objet déplaçable est saisi par le robot, tout en maintenant une configuration stable le long du mouvement. Maintenir une position stable le long du déplacement se traduit par l'apparition d'une chaîne cinématique fermée, composée par l'ensemble du système et les contraintes de manipulation.

$\mathcal{CG} \cap \mathcal{CP}$ et chaînes fermées :

Les contraintes de placement et les contraintes de mouvement réunies peuvent être vues comme des contraintes de fermeture d'une chaîne cinématique. En effet, les contraintes de fermeture sont généralement exprimées sous la forme : $f(q) = 0$ (f peut représenter une fonction distance ou encore un ensemble de fonctions dans le cas où le système est composé de plusieurs chaînes). Ainsi, l'ensemble des configurations valides pour une chaîne cinématique fermée est : $C_{fermeture} = \{q \in CS_{libre} \mid f(q) = 0\}$. D'autre part, le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est défini comme suit : $\mathcal{CG} \cap \mathcal{CP} = \{q_c = (q_{\mathcal{R}}, q_{\mathcal{M}}) \in CS_{libre} \mid \exists\ \mathcal{F}_k^T,\ \mathcal{F}_k^T(q_{\mathcal{R}}) = q_{\mathcal{M}}\ et\ q_{\mathcal{M}} \in \mathcal{P}\}$. Il peut être également présenté sous une forme plus simple : $\mathcal{CG} \cap \mathcal{CP} = \{q_c \in CS_{libre} \mid \mathcal{H}(q_c) = 0\}$, avec $\mathcal{H}$ une fonction exprimant à la fois les contraintes de placement et les contraintes de mouvement. Dans le sous-espace $\mathcal{CG} \cap \mathcal{CP}$, le système global (robot + objet) forme donc une chaîne cinématique fermée. La fermeture de la chaîne cinématique est assurée en maintenant un contact entre l'objet déplaçable et sa zone de placement stable le long d'un mouvement dans lequel le robot déplace l'objet (selon l'une des prises définies comme entrée du problème).

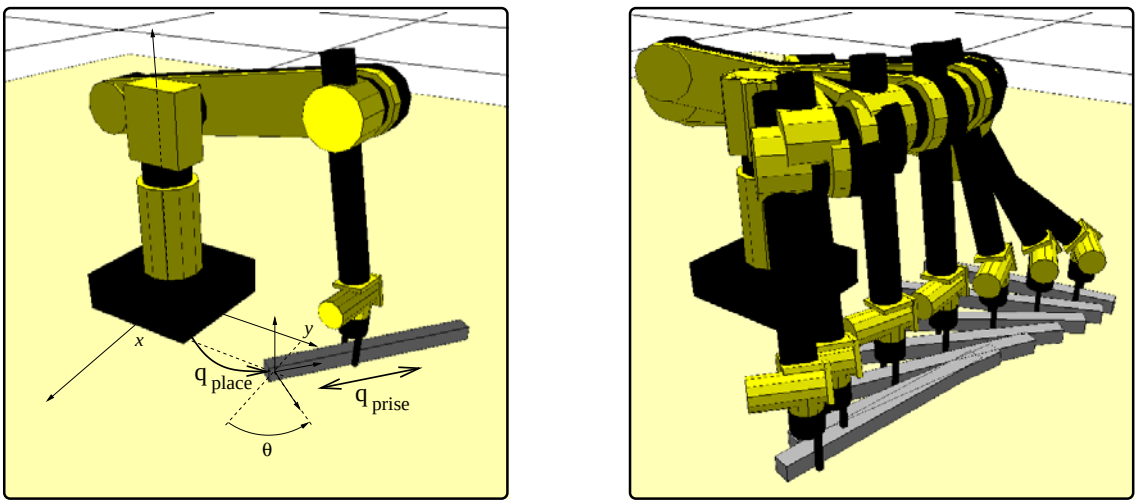


FIG. 4.1 – Exemple de chaîne fermée formée par le système global avec un chemin calculé dans $\mathcal{CG} \cap \mathcal{CP}$ (la barre se déplace sur le sol tout en glissant entre les mors de la pince).

La figure 4.1 illustre une chaîne cinématique fermée composée d'un bras manipulateur à 6 degrés

de liberté et d'une barre (objet déplaçable) posée sur le sol (zone continue de placement stable). Le chemin calculé dans $\mathcal{CG} \cap \mathcal{CP}$ montre un déplacement au contact avec le sol de l'objet déplaçable tout en glissant entre les deux corps du bras manipulateur formant la pince. Le glissement de la pince le long de l'objet est le résultat d'un degré de liberté supplémentaire, noté q_{prise}, introduit dans le système afin de caractériser la prise continue. Dans cet exemple, q_{prise} est choisi de façon à permettre une translation de la pince le long de la barre. De la même manière, la classe de placement stable $\mathcal{P}$ (correspondante à un mouvement plan dans cet exemple) est paramétrée par un vecteur $q_{place} = (x, y, \theta)$ (deux translations et une rotation), permettant le maintien du contact entre l'objet et la zone de placement stable.

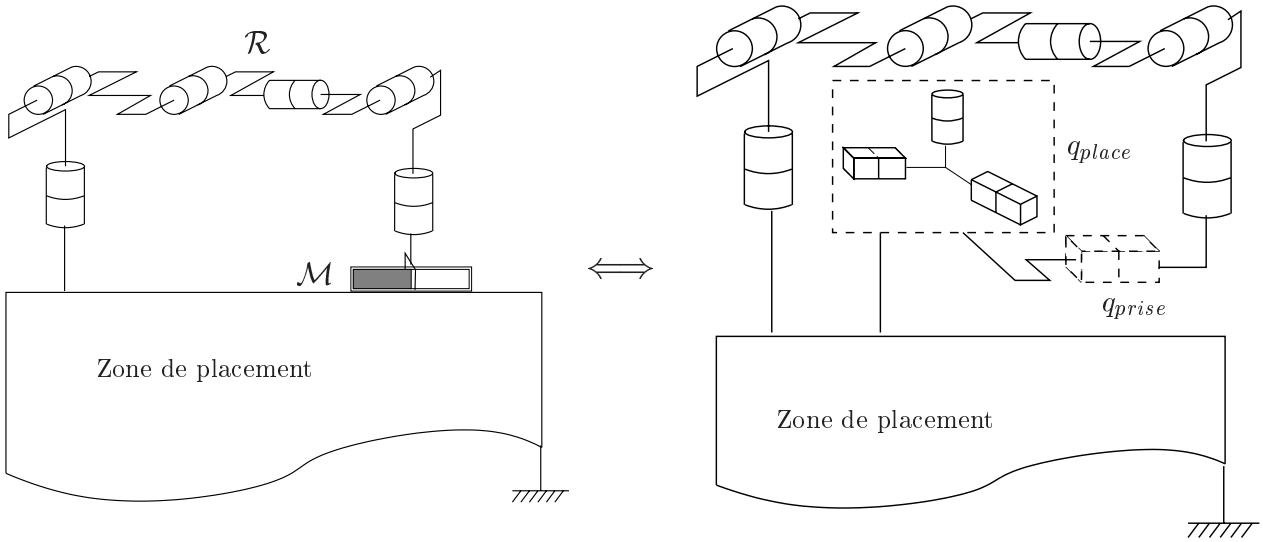


FIG. 4.2 – La chaîne cinématique fermée formée par le robot et l'objet déplaçable.

L'exemple illustré dans la figure 4.1 (robot + barre) peut être ainsi modélisé par la chaîne cinématique fermée présentée dans la figure 4.2. Prendre en considération cette chaîne cinématique virtuelle formée par l'ensemble du système (et qui satisfait les contraintes du problème de manipulation) permet de transformer l'exploration du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ en un problème de planification de mouvement. Le choix d'utiliser les méthodes probabilistes pour capturer la connectivité de ce sous-espace dans un graphe est encore confirmé par la structure complexe formée par le système global dans cette variété. Elles sont les seules capables de traiter des problèmes de planification de mouvement pour des mécanismes aussi complexes (chaînes cinématiques fermées). Cependant, la probabilité de générer une configuration dans un espace aussi contraint est presque nulle. Les méthodes probabilistes classiques ne peuvent pas ainsi être directement utilisées pour ce type de systèmes. L'extension de ces approches aux problèmes de planification de mouvement de mécanismes fermés a fait l'objet de travaux récents. Le paragraphe suivant est consacré à la présentation des approches développées pour la planification de mouvement pour ce type de mécanisme. Nous détaillerons aussi la méthode retenue.

Remarque :

Notons que la structure de la chaîne cinématique fermée formée par le système global varie selon la classe de prise. La figure 4.3 illustre deux chaînes fermées différentes formées par le même système. En changeant la classe de prise, la structure de la chaîne fermée est modifiée. L'exploration du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ se transforme ainsi en un problème de planification de mouvement non plus pour une seule chaîne fermée mais plutôt pour un ensemble de chaînes cinématiques fermées caractérisées par leur classe de prise.

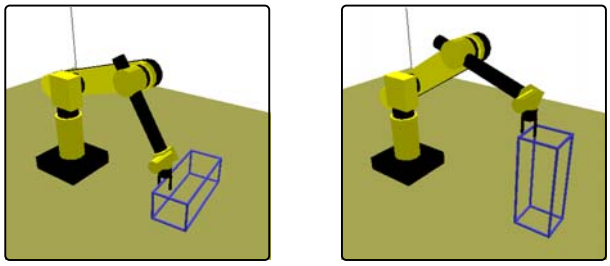


FIG. 4.3 – Deux chaînes fermées différentes selon la classe de prise

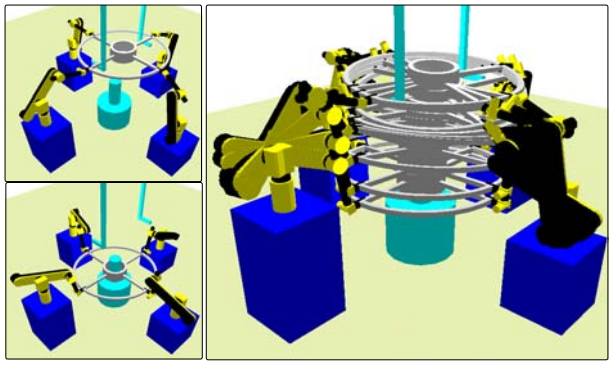
Planification pour des chaînes fermées :

L'analyse des chaînes cinématiques fermées est exprimée généralement en tant que problème de cinématique inverse et formulée comme un système d'équations non-linéaires. L'étude de ce type de mécanisme est encore aujourd'hui un sujet actif de recherche et il n'existe pas de solutions satisfaisantes pour le cas général. Plusieurs méthodes pour résoudre ces systèmes d'équations non-linéaires ont été présentées dans la littérature [Nielsen 97]. Cependant, ces solutions ne possèdent pas les caractéristiques souhaitables (généricité, robustesse, efficacité, ...) pour être intégrées dans les techniques probabilistes pour la planification de mouvement.

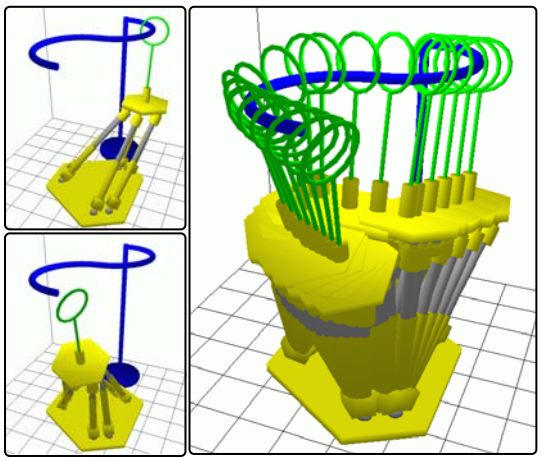
Les chaînes cinématiques fermées apparaissent dans différentes classes de problèmes étudiées en robotique : mécanismes avec boucles cinématiques (robots parallèles), manipulation coordonnée de plusieurs robots (par exemple le cas de plusieurs manipulateurs coopérant pour déplacer un objet), ou encore en modélisation moléculaire (déterminer les conformations énergétiquement stables pour une molécule). Ce type de mécanisme est composé généralement d'un nombre élevé de degrés de liberté ce qui requiert l'utilisation des techniques probabilistes vu leur performance confirmée dès qu'il s'agit de systèmes complexes. Alors que pour les chaînes cinématiques ouvertes, la génération aléatoire de configurations est une étape élémentaire (simple), pour les mécanismes avec des chaînes fermées cette étape nécessite l'utilisation de méthodes plus complexes. De plus, les contraintes de fermetures doivent être respectées le long d'une trajectoire, ce qui ajoute une difficulté dans le calcul des chemins locaux.

En dépit de l'intérêt croissant porté à ce type de mécanisme, peu de travaux concernant la planification de mouvement pour les chaînes cinématiques fermées ont été présentés dans la littérature [LaValle 99b, Yakey 99, Han 00, Cortés 02]. La courte existence des techniques probabilistes ainsi que la difficulté du problème peuvent expliquer la faible quantité de ces travaux.

Dans [LaValle 99b, Yakey 99], les auteurs combinent la technique d'échantillonnage aléatoire avec une descente de gradient permettant de réduire une fonction d'erreur qui représente les contraintes de fermeture de la chaîne cinématique. Cette approche souffre, comme la majorité des méthodes utilisant la descente de gradient, de problème de convergence dès qu'il s'agit d'un mécanisme complexe. La méthode présentée dans [Han 00] consiste à casser la boucle formée par le système en deux chaînes : *active* et *passive*. La configuration de la chaîne active est générée par un tirage aléatoire, alors que pour la chaîne passive, une cinématique inverse est utilisée afin d'assurer la fermeture de la chaîne globale. Concernant la méthode locale connectant deux configurations, les auteurs proposent de planifier uniquement pour les degrés de liberté de la partie active, et utiliser une méthode de cinématique inverse point par point pour la partie passive. Le principal inconvénient de cette méthode concerne la manière de générer les nœuds : lorsque l'organe terminal de la chaîne active (dont la configuration est générée aléatoirement) n'est pas atteignable par la chaîne passive, une configuration fermée du système ne peut pas être obtenue. Cet inconvénient se répercute sur les performances de la méthode. Dans [Cortés 02], les auteurs reprennent la même approche de [Han 00], en modifiant la façon du tirage aléatoire de la chaîne active. Ils proposent un algorithme, appelé "*Random Loop Generator*" (*RLG*), permettant de guider l'organe terminal de la chaîne active vers une approximation conservative de l'espace atteignable par la chaîne passive afin d'augmenter la probabilité de générer des configurations valides (fermées).



a : Exemple de coopération de bras manipulateurs pour déplacer un objet.



b : Plate-forme stewart : exemple de robots parallèles.

FIG. 4.4 – Exemples de problèmes de planification de mouvement pour des mécanismes fermés.

La résolution de problèmes de planification de tâches de manipulation repose sur l'exploration d'une sous-variété de l'espace de configurations, à savoir $\mathcal{CG} \cap \mathcal{CP}$. Dans cette sous-variété, le système global (robot + objet) forme une chaîne cinématique fermée. Pour déterminer les différentes composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$, notre choix s'est porté sur l'utilisation de l'algorithme *RLG*, présenté dans [Cortés 02]. Ce choix a été motivé par les performances montrées par cet algorithme dans la résolution de problème de planification de mouvement pour des chaînes cinématiques fermées complexes ayant un nombre de degrés de liberté élevé (> 10). La figure 4.4 (tirée de [Cortés 03]) illustre des problèmes

de planification de mouvement, invoquant des mécanismes fermés, résolus en se basant sur le même principe de l'approche présentée dans [Cortés 02].

$\mathcal{CG} \cap \mathcal{CP}$ et roadmap probabiliste :

Le principe des approches probabilistes, utilisées pour capturer la connectivité du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ dans un graphe, consiste à échantillonner de manière aléatoire l'espace des configurations du système mécanique et à connecter les configurations générées par des chemins locaux (le principe de ces techniques est détaillé dans le chapitre 1). Ces méthodes conduisent à la construction d'un graphe capturant la connectivité de l'espace libre. Par ailleurs, nous avons montré que le système global dans le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ forme une chaîne cinématique fermée. La génération d'une configuration aléatoire d'une chaîne cinématique ouverte est une étape triviale pour la majorité des méthodes probabilistes. Cependant, dans le cas d'un mécanisme comportant une chaîne fermée, la probabilité de générer une configuration respectant les contraintes de fermeture de la chaîne, est quasiment nulle. Comme nous l'avons précisé dans le paragraphe précédent, nous avons opté pour l'utilisation de l'algorithme *RLG* ([Cortés 02]) pour la génération des nœuds dans $\mathcal{CG} \cap \mathcal{CP}$. La connexion de ces nœuds dans ce sous-espace est effectuée par une approche similaire à celle présentée dans [Han 00]. Nous avons fait également le choix d'utiliser la technique de *Visibilité* ([Siméon 00]) pour la construction du graphe. L'approche "*Visibilité-PRM*" présente l'avantage de capturer la connectivité de l'espace dans une structure bien plus compacte que celle fournie par les méthodes probabilistes classiques. La figure 4.5 illustre les différentes composantes connexes, formant le sous-espace $\mathcal{CG} \cap \mathcal{CP}$, capturées dans une structure de graphe. Nous détaillerons dans la suite l'application de ces différentes approches pour le calcul de ces composantes connexes.

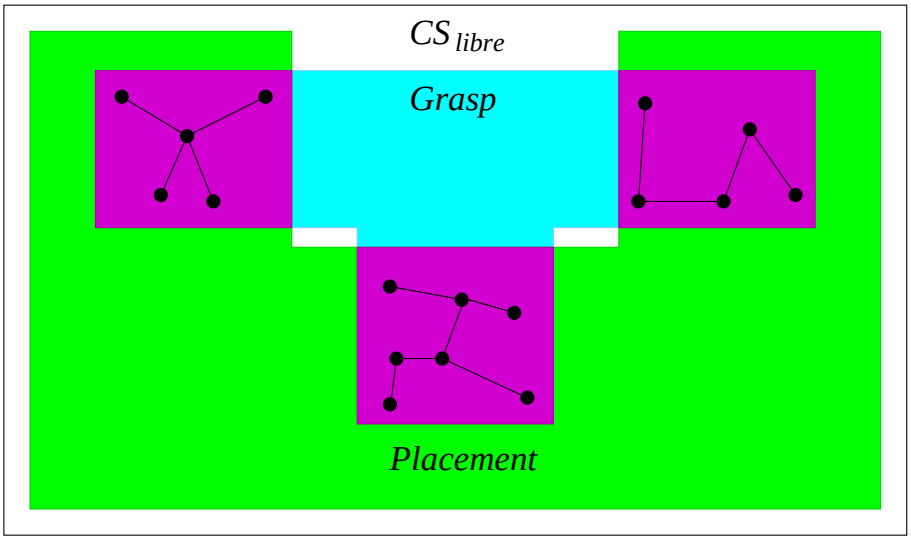


FIG. 4.5 – Capturer la connectivité de $\mathcal{CG} \cap \mathcal{CP}$ dans un réseau probabiliste.

Pour chaque génération d'un nœud dans $\mathcal{CG} \cap \mathcal{CP}$, la chaîne cinématique formée par le système

global est "cassée" en deux sous-chaînes : active et passive (selon la méthode proposée dans [Han 00] et [Cortés 02]). La configuration de la chaîne active est générée par un tirage aléatoire guidé. Ce type d'échantillonnage permet de générer une configuration dans laquelle l'organe terminal de la partie active est situé dans une région atteignable par la chaîne passive. Il permet donc d'augmenter la probabilité[1] d'avoir une configuration valide (fermée) à chaque tirage effectué. Un modèle cinématique inverse est ensuite utilisé pour la génération de la chaîne passive. Par ailleurs, tout les nœuds générés ne sont pas forcément conservés. Seuls les gardiens et les connecteurs sont ajoutés au graphe (selon la technique de visibilité décrite dans le chapitre 1). Les gardiens sont des nœuds qui ne se voient pas mutuellement. Les connecteurs sont des nœuds visibles par au moins deux gardiens. Pour le calcul des arêtes, le planificateur local est limité à agir sur les degrés de liberté de la chaîne active. La partie passive suit le mouvement de la chaîne active grâce à l'utilisation d'une méthode cinématique inverse point par point. Notons que les connexions par des chemins dans $\mathcal{CG} \cap \mathcal{CP}$ sont calculées uniquement pour les nœuds partageant la même classe de prise ainsi que la même classe de placement.

L'avantage d'utiliser la technique de visibilité est également de minimiser le nombre de nœuds générés et de réduire ainsi la taille du graphe capturant la connectivité du sous-espace $\mathcal{CG} \cap \mathcal{CP}$. Ceci permet aussi de limiter le nombre de requêtes de planification de mouvement, coûteuse en terme de temps de calcul, nécessaire à la détermination des connexions par les chemins de transit et les chemins de transfert des différentes composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$.

La figure 4.6 illustre un graphe capturant la connectivité du sous-espace $\mathcal{CG} \cap \mathcal{CP}$, calculé en utilisant l'approche décrite ci-dessus. Ce graphe est composé de quatre composantes connexes : deux composantes principales séparées par l'obstacle divisant la zone de placement stable en deux parties, et deux autres composantes réduites chacune à un seul nœud (entourées par le cercle en pointillés dans la figure de gauche). Ces deux dernières correspondent aux placements de l'objet à l'intérieur de la cage lorsqu'il est saisi par le robot à travers l'ouverture au milieu de cet obstacle. La configuration du système global dans l'un de ces nœuds est représentée dans la figure 4.6 (image en haut à droite). L'autre nœud correspond au même placement de l'objet tourné de 180 degrés. L'image à droite en bas de la figure 4.6 montre un nœud appartenant à l'une des composantes principales du graphe construit et qui correspond au même placement de l'objet mais avec une prise différente. La connexion de ces deux nœuds (correspondant aux configurations présentées dans la figure 4.6) par un chemin dans $\mathcal{CG} \cap \mathcal{CP}$ nécessite un changement continu de la prise, ce qui n'est pas possible à cause de la présence de l'obstacle (la cage). Ce changement de prises nécessite le calcul d'un chemin sans collision en dehors du sous-espace $\mathcal{CG} \cap \mathcal{CP}$. Ce type de connexions fait l'objet de la section suivante.

[1] La probabilité d'avoir une configuration fermée avec ce type de tirage n'est pas égale à 1, puisque la région atteignable de l'organe terminal n'est pas explicitement calculée. L'algorithme RLG détermine uniquement une approximation conservative de cet espace.

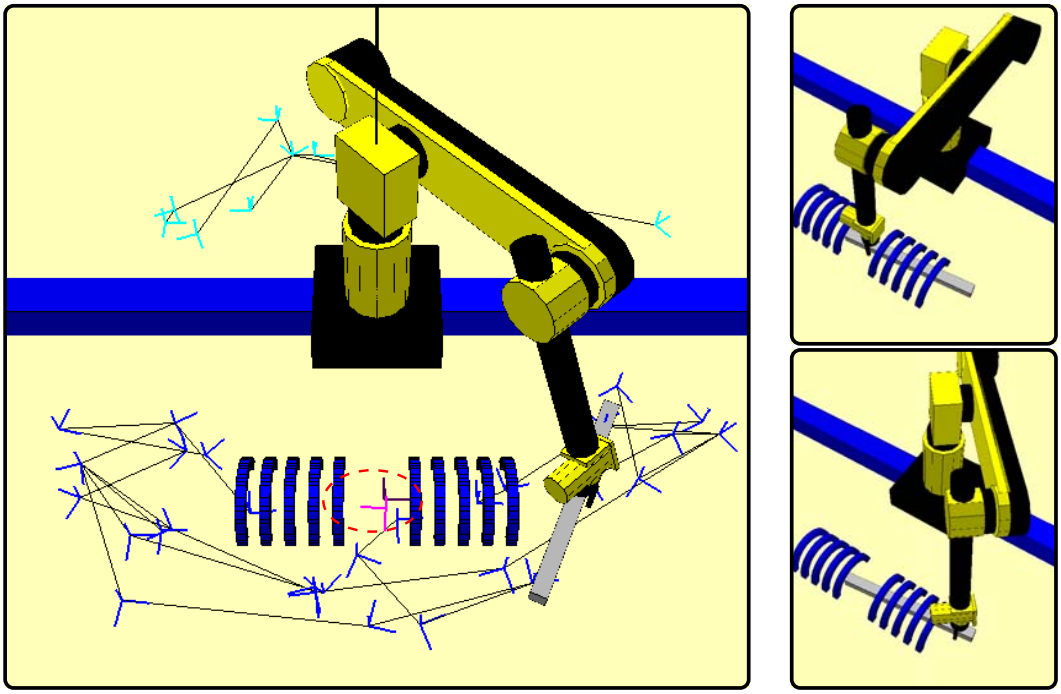


FIG. 4.6 – Un graphe calculé dans $\mathcal{CG} \cap \mathcal{CP}$ en utilisant la technique de visibilité (à gauche) et deux configurations du système appartenant à deux composantes connexes différentes de $\mathcal{CG} \cap \mathcal{CP}$ pour un même placement de l'objet (à droite).

4.2.3 Connexions par des chemins de transit et de transfert

Considérons maintenant deux nœuds N_1 et N_2 générés lors de la phase précédente et qui ne sont pas connectés par un chemin dans $\mathcal{CG} \cap \mathcal{CP}$. Ceci se traduit par le fait que N_1 et N_2 n'appartiennent pas à la même composante connexe. Chacun de ces deux nœuds correspond à une prise fixe et un placement déterminé, ainsi ils sont caractérisés par un doublet de type $(g_i, p_i)_{i=1,2}$. Le problème maintenant est de déterminer la connexion de ces deux nœuds par des chemins transit/transfert dans les sous-espaces $\mathcal{CG}$ et $\mathcal{CP}$. Une telle connexion est possible s'il existe :

- un chemin de transfert entre (g_1, p_1) et (g_1, p_2) suivi par un chemin de transit entre (g_1, p_2) et (g_2, p_2).
- ou un chemin de transit entre (g_1, p_1) et (g_2, p_1) suivi par un chemin de transfert entre (g_2, p_1) et (g_2, p_2).

La figure 4.7 illustre la structure du graphe de manipulation $(\mathcal{MG})$. Il est structuré en plusieurs composantes connexes de taille réduite (composantes connexes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$) connectées par des chemins de transit ou des chemins de transfert. Le chemin de manipulation est cherché dans le graphe $\mathcal{MG}$ ainsi construit. Le chemin solution est composé d'une alternance de chemins de transit/transfert avec des chemins élémentaires calculés dans $\mathcal{CG} \cap \mathcal{CP}$. Ces derniers ne correspondent pas

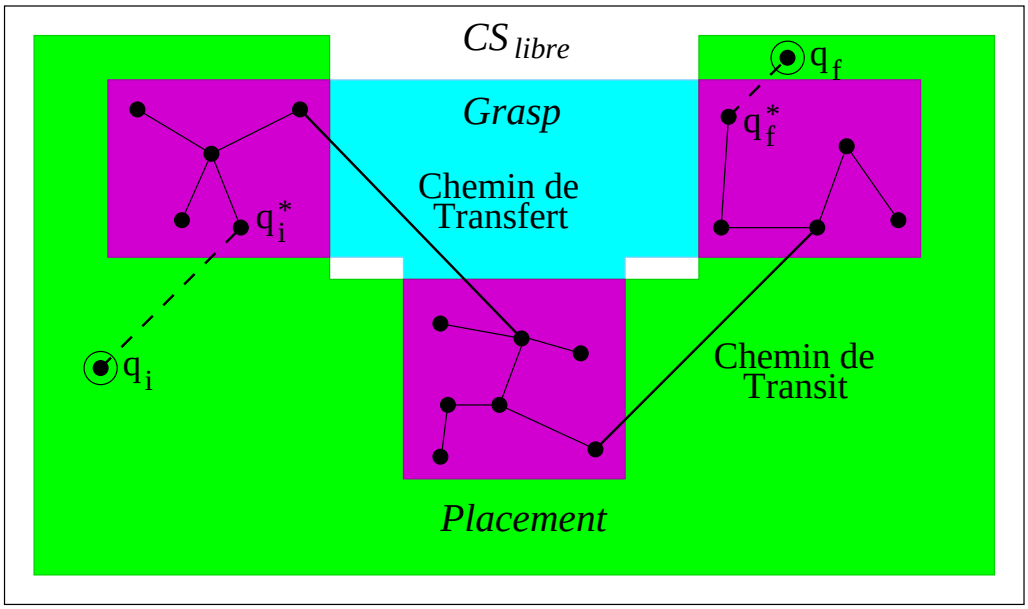


FIG. 4.7 – Structure du graphe de manipulation.

à des chemins faisables de point de vue de la manipulation. Leur transformation en séquence transit/transfert (grâce à la propriété de réduction présentée dans le chapitre 2) est effectuée dans une étape ultérieure. Par ailleurs, chaque requête de transit (resp. de transfert) représente un problème de planification de mouvement à part entière. Notons que chacune de ces requêtes de planification est calculée dans un environnement partiellement modifié puisqu'elle correspond soit à une prise particulière soit à un placement spécifique de l'objet déplaçable. La résolution de ces différentes requêtes est assurée par la technique de planification par *graphe dynamique* ([Sahbani 01a, Sahbani 01b]) présentée dans le chapitre précédent. L'utilisation de cette approche permet d'exploiter cette notion de changement partiel de l'environnement et d'amortir en conséquence le coût de chaque requête de planification en réutilisant des chemins calculés dans une phase préalable sans prendre en compte la présence de l'objet déplaçable.

Connexions par des chemins de transit :
Un chemin de transit est un chemin de CS_{libre} dans lequel le robot se déplace seul, l'objet déplaçable demeure stationnaire (occupant un placement stable). Par conséquent, ce type de chemin connecte uniquement deux nœuds partageant le même placement de l'objet déplaçable. Ils permettent au robot d'effectuer un changement de prise. La figure 4.8 illustre un chemin de transit connectant deux nœuds appartenant à deux composantes connexes différentes de $\mathcal{CG} \cap \mathcal{CP}$.

Le calcul des chemins de transit est effectué selon l'approche par *graphe dynamique*. Nous calculons dans un premier temps un graphe capturant la connectivité de $(CS_{\mathcal{R}})_{libre}$ (pour le robot avec les obstacles statiques sans prendre en considération la présence de l'objet déplaçable dans l'environnement).

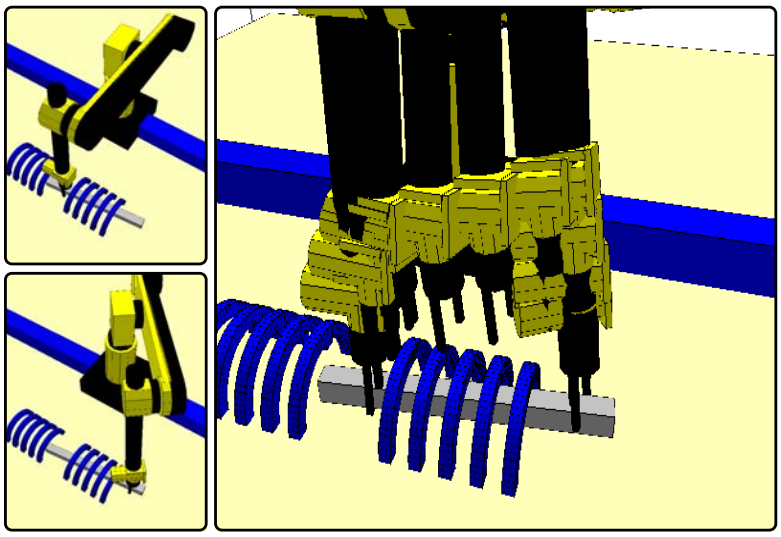


FIG. 4.8 – Un chemin de transit connectant deux nœuds partageant le même placement de l'objet déplaçable.

Avant la recherche d'un chemin de transit, le graphe est instancié selon l'état courant de l'environnement. Chaque arête est validée (par un test de collision) selon la position courante de l'objet déplaçable. Si une arête ne correspond pas à un chemin sans collision, elle est marquée *bloquante*. La recherche du chemin de transit se poursuit dans le graphe ainsi marqué. S'il n'y a pas de chemin répondant à la requête, ceci se traduit par le fait qu'il n'existe pas de chemin même en l'absence d'objets déplaçables. Le sous-problème de planification de mouvement n'admet pas alors de solution. S'il existe un chemin ne contenant aucune arête marquée, la solution du sous-problème est donc trouvée. Dans le cas intermédiaire où le chemin trouvé contient au moins une arête marquée, la mise à jour est effectuée localement en utilisant un algorithme de type RRT [Kuffner 00d] (Rapidly Random Trees) afin de connecter les deux extrémités de l'arête marquée (ou l'ensemble des arêtes marquées).

Connexions par des chemins de transfert :

Un chemin de transfert est un chemin de CS_{libre} dans lequel le robot déplace l'objet déplaçable selon une prise *fixée*. Le long du chemin de transfert, la configuration de l'objet déplaçable $q_{\mathcal{M}}$ est déterminée par la fonction $\mathcal{G}^T_{\mathcal{M}}$, décrivant le modèle géométrique direct du robot : $q_{\mathcal{M}} = \mathcal{G}^T_{\mathcal{M}}(q_{\mathcal{R}})$. Ainsi, un chemin de transfert connecte uniquement les paires de nœuds partageant la même prise, et qui diffèrent par le placement de l'objet. Ce type de chemin permet au robot de déplacer l'objet d'un placement stable vers un autre. La figure 4.9 illustre un chemin de transfert connectant deux configurations du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ ayant la même prise.

Le calcul des chemins de transfert est effectué en utilisant aussi l'approche par *graphe dynamique*. Le graphe initial est dynamiquement mis à jour en considérant la présence de l'objet déplaçable attaché au robot selon la prise courante. Notons que dans le cas de problèmes de manipulation sous contraintes discrètes, les paires de nœuds candidats à une connexion par un chemin de transfert (resp. de transit)

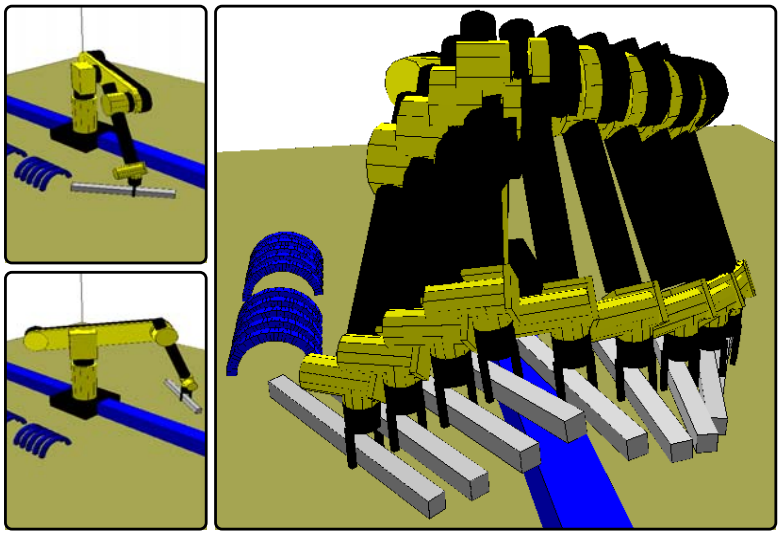


FIG. 4.9 – Un chemin de transfert connectant deux nœuds partageant la même prise.

peuvent être triées, parmi les nœuds générés dans $\mathcal{CG} \cap \mathcal{CP}$, selon leur prise discrète (resp. selon leur placement discret)[2]. Cependant, dans le cas de problème de manipulation sous contraintes continues, ce tri n'est plus possible puisque les valeurs de prises (resp. de placements) sont aussi tirées aléatoirement lors de la génération d'un nœud dans $\mathcal{CG} \cap \mathcal{CP}$. La probabilité d'avoir deux nœuds partageant le même placement (resp. la même prise) est très faible. Ainsi, deux nœuds N_1 et N_2 appartenant à deux composantes connexes différentes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ sont connectés par un chemin de transit suivi d'un chemin de transfert ou inversement. Le chemin de transit permet au robot de changer de prise (i.e. passer de la prise définie en N_1 à celle définie en N_2), alors que le chemin de transfert permet au robot de déplacer l'objet de son placement caractérisant le nœud N_1 au placement caractérisant le nœud N_2. Ce processus est itéré entre les paires de composantes connexes formant $\mathcal{CG} \cap \mathcal{CP}$ jusqu'à ce qu'une connexion soit trouvée ou que la liste des nœuds de ces composantes soit épuisée.

Remarque :

La résolution d'un problème de manipulation s'effectue en plusieurs étapes élémentaires (calcul des composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$, connexions par des chemins transit/transfert, ...). Dans le schéma présenté, nous avons proposé pour chaque étape élémentaire la méthode de résolution la plus adaptée (*Visibilité-PRM, RLG, ...*). Il est bien évident qu'en se basant sur ce même schéma générique, nous aurions pu résoudre le problème de manipulation en utilisant d'autres méthodes. Les techniques proposées ne remettent en aucun cas en cause la généricité de notre approche.

[2] $C_{Transit}$ (resp. $C_{Transfert}$) représente l'ensemble des paires de nœuds candidats à un test de connexion par un chemin de transit (resp. de transfert) dans le cas de contraintes discrètes (cf. section 2.3.1).

4.3 Algorithme de manipulation

Cette section détaille l'algorithme développé pour implémenter l'approche présentée ci-dessus. L'algorithme prend en entrée les différentes classes continues de prises possibles $\mathcal{G}_i$ et un ensemble de domaines continus de placements stables $\mathcal{P}_j$. Chaque classe de prise continue ($\mathcal{G}_i$) est définie par la donnée d'une transformation homogène T_{g_i}, et d'un ensemble de paramètres noté q_{prise}, variant dans un intervalle fixé. De la même façon, chaque région continue de placements stables $\mathcal{P}_j$ est caractérisée par une transformation homogène $T_{\mathcal{P}_j}$ et par la donnée d'un vecteur $q_{place} = (x, y, \theta)$ représentant deux translations dans un domaine rectangulaire et une rotation autour d'un axe perpendiculaire à ce plan. Pour chaque couple d'ensemble $(\mathcal{G}_i, \mathcal{P}_i)$, la chaîne cinématique virtuelle formée par le système global est considérée lors de l'analyse de la connectivité des sous-espaces composant la sous-variété $\mathcal{CG} \cap \mathcal{CP}$ et correspondant à ce couple.

4.3.1 Construction du graphe de manipulation

L'algorithme proposé construit incrémentalement le graphe de manipulation $\mathcal{MG}$ en alternant les deux étapes de notre approche : le calcul des composantes connexes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ et la détermination de leur connexion par des chemins de transfert/transit. Il commence par explorer le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ en capturant sa connectivité à travers un graphe et dès que la taille de ce dernier atteint un certain seuil, les connexions des composantes calculées par des chemins de transfert/transit sont favorisées. L'arrêt de l'algorithme est contrôlé par un compteur $ntry$ (inspiré de la technique utilisée par la méthode *"Visibilité-PRM"* [Siméon 00]), qui comptabilise le nombre d'échecs dans l'insertion d'un nouveau nœud. L'étape de construction du graphe est achevée lorsque ce compteur atteint un seuil $Ntry_{max}$, paramètre important de l'algorithme. Ce paramètre permet d'estimer la couverture fournie par le graphe de manipulation $\mathcal{MG}$ de l'espace des configurations composite du système.

La fonction *Expand_ Graphe*, décrite ci-dessous, permet la construction du graphe de manipulation $\mathcal{MG}$ en alternant les deux étapes grâce à la fonction *Choisir_ Stratégie*. Cette fonction effectue une sélection (un tirage aléatoire) biaisée afin de déterminer la stratégie à utiliser. En effet, il est inutile de commencer par une étape de connexion alors qu'aucune composante connexe du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ n'est encore calculée. Durant les premières phases du construction du graphe de manipulation, la probabilité de choisir *Expand_ Dans_* $\mathcal{CG} \cap \mathcal{CP}$ est grande. Au fur et à mesure que la taille du graphe augmente, les connexions par des chemins de transit/transfert entre les différentes composantes connexes sont progressivement testées. La probabilité de sélectionner la fonction *Connect_ En_ Dehors_ De_* $\mathcal{CG} \cap \mathcal{CP}$ augmente proportionnellement au pourcentage de couverture de CS_{libre} qui est estimée à $(1 - \frac{1}{ntry})$.

Algorithme 4 *Expand_ Graphe($\mathcal{MG}$)*

1: $ntry \leftarrow 0$
2: $expand \leftarrow$ Faux
3: **Tant Que** $((ntry < Ntry_{max}(PM))$ et $(\neg expand))$ **Faire**
4: $S \leftarrow$ Choisir_Stratégie(PM)
5: **Si** (S) **Alors**
6: $expand \leftarrow$ Expand_Dans_$\mathcal{CG} \cap \mathcal{CP}(\mathcal{MG})$
7: **Si** (expand) **Alors**
8: $ntry \leftarrow 0$
9: **Sinon**
10: $ntry = ntry + 1$
11: **Fin Si**
12: **Sinon**
13: $expand \leftarrow$ Connect_En_Dehors_De_$\mathcal{CG} \cap \mathcal{CP}(\mathcal{MG})$
14: **Fin Si**
15: **Fin Tant Que**

Expansion dans $\mathcal{CG} \cap \mathcal{CP}$:

La sous-variété $\mathcal{CG} \cap \mathcal{CP}$ est composée de plusieurs sous-espaces caractérisés chacun par un couple $(\mathcal{G}_i, \mathcal{P}_i)$. L'analyse de la connectivité de chaque sous-espace est effectuée en considérant la chaîne cinématique fermée relative au couple $(\mathcal{G}_i, \mathcal{P}_i)$ le caractérisant. La fonction *Expand_ Dans_* $\mathcal{CG} \cap \mathcal{CP}$ exécute une étape d'expansion du graphe de manipulation $\mathcal{MG}$ dans le sous-espace $\mathcal{CG} \cap \mathcal{CP}$. A chaque appel à cette fonction, un nouveau nœud est généré aléatoirement dans $\mathcal{CG} \cap \mathcal{CP}$ selon les entrées du problème de manipulation MP (i.e : selon les placements et les prises continues définis par l'utilisateur). Ce nœud est ajouté au graphe si et seulement si il est *"gardien"* ou *"connecteur"* (selon le principe de la visibilité [Siméon 00]).

Un nœud candidat N est généré aléatoirement par la fonction *Nouveau_ Nœud*. Cette fonction commence par sélectionner une classe de placement continu $\mathcal{P}_j$ parmi celles définies comme entrées du problème PM. Un placement stable $p \in \mathcal{P}_j$ pour l'objet déplaçable est ensuite généré en tirant aléatoirement q_{place}. De la même manière, une classe de prise continue $\mathcal{G}_i$ est sélectionnée et ses paramètres q_{prise} sont tirés aléatoirement. Dans le cas où la prise et le placement générés correspondent à une configuration sans-collision et faisable pour la chaîne cinématique fermée, le nœud N est généré.

La fonction *Connect_ Dans_* $\mathcal{CG} \cap \mathcal{CP}$ teste les connexions possibles du nouveau nœud aux différentes composantes connexes du graphe $\mathcal{MG}$ par des chemins dans $\mathcal{CG} \cap \mathcal{CP}$. Ces connexions sont calculées uniquement avec les nœuds caractérisés par le même couple $(\mathcal{G}_i, \mathcal{P}_j)$ utilisé pour la génération du nœud N. Pour chaque composante connexe C_i de $\mathcal{MG}$, un chemin dans $\mathcal{CG} \cap \mathcal{CP}$ est testé entre les nœuds candidats de C_i et le nouveau nœud N. Ce test est arrêté si un chemin est trouvé ou si la

liste de tous les nœuds candidats de C_i est complètement explorée. Le calcul du chemin dans $\mathcal{CG} \cap \mathcal{CP}$ est effectué en utilisant une technique de cinématique inverse point à point (selon l'approche présentée dans [Cortés 02]).

Algorithme 5 *Expand_Dans_$\mathcal{CG} \cap \mathcal{CP}(\mathcal{MG})$*

1: $N \leftarrow$ Nouveau_Nœud(MP)
2: $n_{linked\ comp.} \leftarrow 0$; $n_c \leftarrow$ Nbre_comp($\mathcal{MG}$)
3: **Pour** ($i = 1$) **à** n_c **Faire**
4: **Si** (Connect_Dans_$\mathcal{CG} \cap \mathcal{CP}(N\ ,\ C_i)$) **Alors**
5: $n_{linked\ comp.} = n_{linked\ comp.} + 1$
6: **Fin Si**
7: **Fin Pour**
8: **Si** ($n_{linked\ comp.} = 0$) **Alors**
9: Ajouter_Nœud (Isolé, $N, \mathcal{MG}$) ; Mettre_à_Jour_Graphe($\mathcal{MG}$)
10: **Retourne** Vrai
11: **Sinon**
12: **Si** ($n_{linked\ comp.} > 1$) **Alors**
13: Ajouter_Nœud (Connecteur, $N, \mathcal{MG}$) ; Mettre_à_Jour_Graphe($\mathcal{MG}$)
14: **Retourne** Vrai
15: **Sinon**
16: **Retourne** Faux
17: **Fin Si**
18: **Fin Si**

Remarque : Nous avons également implémenté une version basée sur la méthode *PRM* pour le calcul des différentes composantes connexes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$. Ainsi, tous les nœuds tirés sont gardés. Une comparaison entre les performances de ces deux techniques dans l'exploration du $\mathcal{CG} \cap \mathcal{CP}$ est présentée dans le chapitre 5, § 5.3.2. ■

Notons par N_{k,C_i} un nœud de la composante connexe C_i. L'échec d'une connexion dans $\mathcal{CG} \cap \mathcal{CP}$ entre le nouveau N et N_{k,C_i} peut nous informer sur l'existence d'une opération de *"re-grasping"* possible (un changement de prise permettant d'établir la connexion entre les deux nœuds). Cette possibilité est également considérée par la fonction *Connect_Dans_$\mathcal{CG} \cap \mathcal{CP}$*. En effet, si le chemin connectant les nœuds N et N_{k,C_i} dans $\mathcal{CG} \cap \mathcal{CP}$ est uniquement valide pour l'objet (le chemin pour le robot étant en collision), cette fonction teste la possibilité de connecter ces deux nœuds en effectuant un changement de prise. Un parcours de ce chemin est alors effectué afin d'identifier un placement intermédiaire p_{int} associé à deux prises intermédiaires g_{int} et g'_{int} (issues des deux configurations du système global sur le chemin juste avant et juste après la collision, si elles existent). L'identification de ce placement intermé-

diaire est effectuée en exploitant les informations fournies par le détecteur de collision lors du parcours de ce chemin. Un chemin de transit entre les nœuds $N_{int} = (g_{int}, p_{int})$ et $N'_{int} = (g'_{int}, p_{int})$ est ensuite calculé. Si ce chemin est valide pour le robot et si le chemin dans $\mathcal{CG} \cap \mathcal{CP}$ connectant N'_{int} et N_{k,C_i} est sans collision, alors N et N_{k,C_i} sont connectés. Le chemin connectant ces deux nœuds est composé du chemin de transit calculé entre N_{int} et N'_{int}, précédé par le chemin dans $\mathcal{CG} \cap \mathcal{CP}$ entre N et N_{int} et suivi par le chemin dans $\mathcal{CG} \cap \mathcal{CP}$ connectant N'_{int} et N_{k,C_i}. Finalement, N est ajouté au graphe s'il est connecté à autre composante connexe du graphe $\mathcal{MG}$ (différente de C_i). Dans le premier cas, une nouvelle composante connexe composée uniquement du nœud N est créée et le graphe est mis à jour. Dans le second cas, les composantes connexes connectées à N sont fusionnées. Le nœud N est alors ajouté à cette fusion ainsi que les différentes arêtes permettant la connexion de ce nœud aux composantes unies.

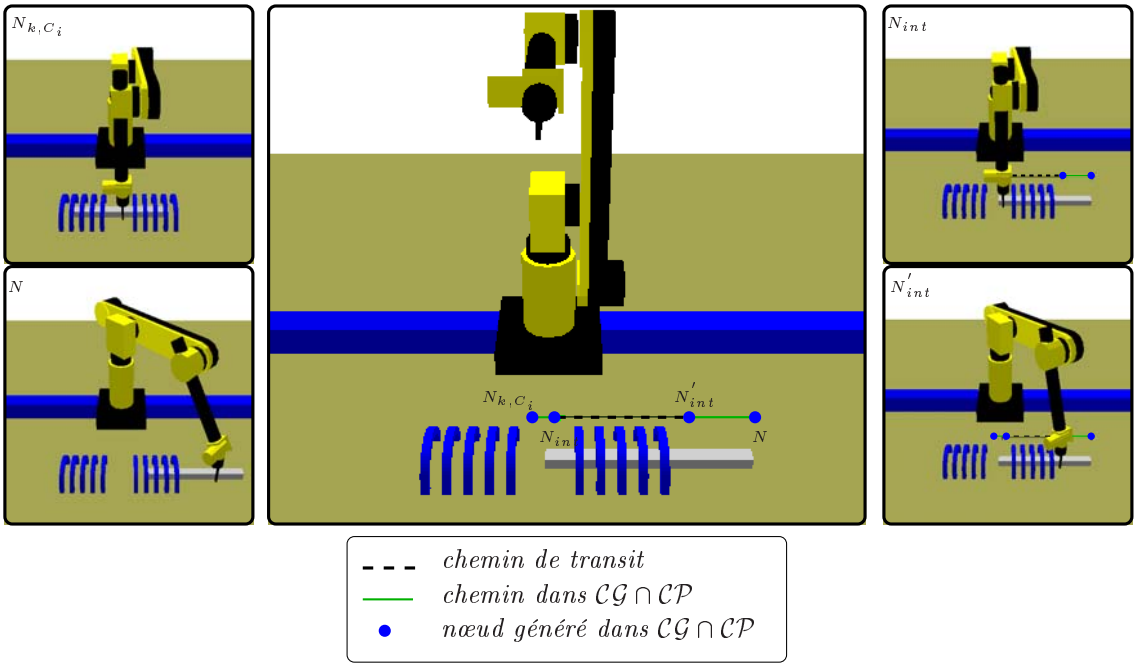


FIG. 4.10 – Connexion par *"re-grasping"* entre deux nœuds N et N_{k,C_i} de $\mathcal{CG} \cap \mathcal{CP}$.

La figure 4.10 illustre une opération de *"re-grasping"* déterminée suite à un échec de connexion dans $\mathcal{CG} \cap \mathcal{CP}$ entre les nœuds N et N_{k,C_i} (à gauche). Le chemin connectant ces deux nœuds dans $\mathcal{CG} \cap \mathcal{CP}$ est discrétisé en un ensemble de paires (placement, prise). Un test de collision est effectué pour chaque configuration de ce chemin (et donc pour chaque paire (placement, prise)). Le nœud N_{int} correspond à la paire (p_{int}, g_{int}) juste avant collision (collision entre le robot et la cage). Le placement de l'objet déplaçable est alors fixé à p_{int} (placement intermédiaire de l'objet). Le nœud N'_{int} est déterminé en incrémentant le paramètre de la prise (le robot glisse le long de l'objet figé à son placement intermédiaire p_{int}) jusqu'à ce que la configuration du robot soit valide (sans collision).

Connexions par des chemins de transit et de transfert :

Le rôle de la fonction *Connect_ En_ Dehors_ De_* $\mathcal{CG} \cap \mathcal{CP}$ consiste à connecter les différentes composantes connexes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ par une séquence de deux chemins, à savoir un chemin de transit suivi d'un chemin de transfert ou inversement. Chaque nœud N_i du graphe de manipulation $\mathcal{MG}$ est caractérisé par la paire (g_i, p_i). Deux nœuds N_i et N_j sont connectés par un chemin en dehors de $\mathcal{CG} \cap \mathcal{CP}$ si et seulement si il existe :

- un chemin de transfert connectant (g_i, p_i) et (g_i, p_j) suivi par un chemin de transit entre (g_i, p_j) et (g_j, p_j).
- ou un chemin de transit entre (g_i, p_i) et (g_j, p_i) suivi par un chemin de transfert connectant (g_j, p_i) et (g_j, p_j).

Algorithme 6 *Connect_ En_ Dehors_ De_* $\mathcal{CG} \cap \mathcal{CP}$

1: $N \leftarrow$ Sélectionner_Nœud($\mathcal{MG}$)
2: $n_{linked\ comp.} \leftarrow 0$
3: **Pour** (i=1) **à** (N_comp($\mathcal{MG}$) | $C_i \neq (C \Leftarrow N)$) **Faire**
4: **Si** (Connect_Avec_Transit_Transfert(N,C_i)) **Alors**
5: $n_{linked\ comp.} = n_{linked\ comp.} + 1$
6: **Fin Si**
7: **Fin Pour**
8: **Si** ($n_{linked\ comp.} \neq 0$) **Alors**
9: Mettre_à_Jour_Graphe($\mathcal{MG}$)
10: **Retourne** Vrai
11: **Sinon**
12: **Retourne** Faux
13: **Fin Si**

La fonction *Connect_ En_ Dehors_ De_* $\mathcal{CG} \cap \mathcal{CP}$ commence par sélectionner aléatoirement un nœud N du graphe $\mathcal{MG}$. Les connexions entre ce nœud et les différentes composantes connexes de $\mathcal{MG}$ sont ensuite testées par la fonction *Connect_ Avec_ Transit_ Transfert*. Un test de connexions entre le nœud N sélectionné et les nœuds N_{k,C_i} formant la composante connexe C_i est effectué. Ce test est arrêté si un chemin entre N et N_{k,C_i} est trouvé ou si toute la liste des nœuds de C_i a été visitée. Afin de connecter N à N_{k,C_i}, deux nœuds intermédiaires N_{int1} et N_{int2} sont calculés. Ces nœuds correspondent respectivement aux doublets $(g_N, p_{N_{k,C_i}})$ et à $(g_{N_{k,C_i}}, p_N)$. Ces deux nœuds permettent de tester les deux connexions possibles entre N et N_{k,C_i} : un chemin de transfert entre N et N_{int1} suivi par un chemin de transit entre N_{int1} et N_{k,C_i}, ou un chemin de transit entre N et N_{int2} suivi par un chemin de transfert entre N_{int2} et N_{k,C_i}. Dans le cas où l'une de ces deux possibilités est valide, les composantes connectées sont fusionnées.

Le calcul des chemins de transfert et de transit est effectué en utilisant la méthode par *"graphe dynamique"* décrite dans le chapitre précédent.

4.3.2 Recherche du chemin de manipulation

Une fois le graphe de manipulation $\mathcal{MG}$ construit, il peut être utilisé pour la résolution des différentes requêtes de planification de tâches de manipulation. La recherche du chemin de manipulation est composée de deux étapes : une recherche de chemin dans $\mathcal{MG}$ et une transformation de la solution, si elle existe, en chemin admissible. La première étape consiste à relier dans un premier temps les configurations initiale et finale du système global aux nœuds du graphe $\mathcal{MG}$ en utilisant les deux fonctions *Connect_ Dans_* $\mathcal{CG} \cap \mathcal{CP}$ et *Connect_ Avec_ Transit_ Transfert* décrites précédemment. En cas de succès, une recherche de chemin dans le graphe $\mathcal{MG}$ est effectuée. La seconde étape consiste à transformer tous les sous-chemins calculés dans $\mathcal{CG} \cap \mathcal{CP}$ de la solution trouvée en une séquence de transit/transfert.

Recherche du chemin :

Notons par q^i_{sys} et par q^f_{sys} les configurations initiale et finale du système global (robot+objet). Généralement, ces deux configurations n'appartiennent pas au sous-espace $\mathcal{CG} \cap \mathcal{CP}$, elles sont respectivement caractérisées par les doublets $(-, p^i)$ et $(-, p^f)$ (le trait indique qu'aucune prise n'est effectuée dans cette configuration du système). Afin de connecter ces configurations au graphe de manipulation, deux nœuds $\tilde{N}_i$ et $\tilde{N}_f$ sont calculés dans $\mathcal{CG} \cap \mathcal{CP}$ correspondant aux configurations $\tilde{q}^i_{sys} = (\tilde{g}_i, p^i)$ et $\tilde{q}^f_{sys} = (\tilde{g}_f, p^f)$, où $\tilde{g}_i$ et $\tilde{g}_f$ sont deux prises sélectionnées aléatoirement parmi les classes de prises définies comme entrées du problème de manipulation (leurs paramètres respectifs sont tirés aléatoirement). Les deux chemins de transit connectant respectivement q^i_{sys} à $\tilde{q}^i_{sys}$ et q^f_{sys} à $\tilde{q}^f_{sys}$ sont calculés et des tests de connexion des deux nœuds $\tilde{N}_i$ et $\tilde{N}_f$ à $\mathcal{MG}$ sont effectués[3]. Ces deux nœuds sont connectés dans premier temps par des chemins dans $\mathcal{CG} \cap \mathcal{CP}$ aux différentes composantes connexes du graphe de manipulation $\mathcal{MG}$ (en utilisant la fonction *Connect_ Dans_* $\mathcal{CG} \cap \mathcal{CP}$). En cas d'échec, une connexion en dehors du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est alors testée en utilisant la fonction *Connect_ Avec_ Transit_ Transfert.* Le chemin obtenu n'est pas admissible (à cause des sous-chemins calculés dans $\mathcal{CG} \cap \mathcal{CP}$) ; une phase de transformation est alors nécessaire.

Transformation des chemins dans $\mathcal{CG} \cap \mathcal{CP}$:

Grâce à la propriété de réduction (présentée dans le chapitre 2, §2.4), les chemins calculés dans le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ sont transformables en une séquence alternée et finie de chemins de transit et de chemins de transfert (comme illustré dans la figure 4.11). Afin d'effectuer cette transformation, nous proposons un algorithme récursif de type dichotomie (*"divide-and-conquer"*).

[3]Dans le cas où l'une des prises sélectionnées ou l'un des paramètres tirés ne correspond pas à une configuration valide du système (configuration en collision pour le robot, objet non atteignable, ...), un autre tirage et/ou une autre sélection sont effectués.

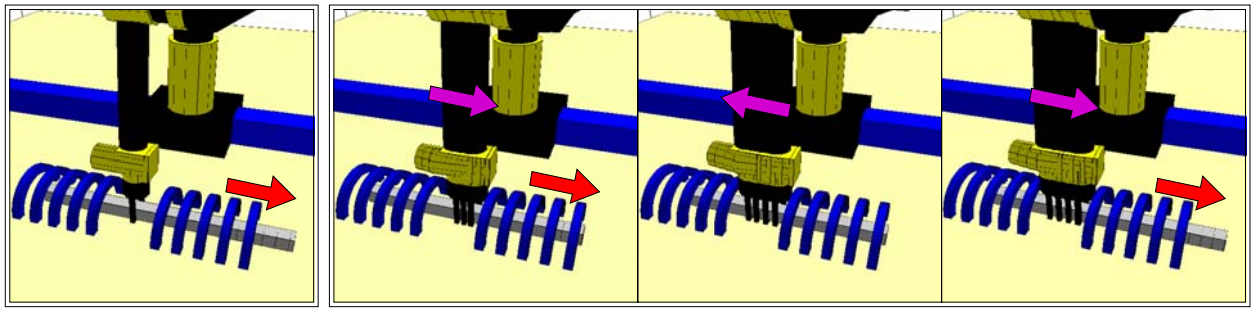


FIG. 4.11 – Un chemin calculé dans $\mathcal{CG} \cap \mathcal{CP}$ (à gauche) transformé en une séquence de trois chemins admissibles *transfert/transit/transfert* (droite).

Algorithme 7 *Transform_ Chemin* (q_1, q_2)

1: **Si** (Connect_Transit_Transfert(q_1, q_2)) **Alors**
2: Stocker_chemin(τ)
3: **Retourne**
4: **Sinon**
5: **Si** (Dist_placement$(q_1, q_2) \leq \varepsilon$) **Alors**
6: $q_{int.} \leftarrow$ Calculer_Configuration_Intermédiaire (q_1, q_2)
7: Transform_Chemin $(q_1, q_{int.})$
8: Transform_Chemin $(q_{int.}, q_2)$
9: **Sinon**
10: **Retourne**
11: **Fin Si**
12: **Fin Si**

Soient N_1 et N_2 deux nœuds du graphe de manipulation $\mathcal{MG}$ caractérisés par les doublets (g_1, p_1), (g_2, p_2), et connectés par un chemin dans $\mathcal{CG} \cap \mathcal{CP}$. Ces deux nœuds partagent donc la même classe de prise mais avec un paramètre q_{prise} différent (ils partagent également la même classe de placement mais avec q_{place} différent puisqu'ils sont connectés par un chemin dans $\mathcal{CG} \cap \mathcal{CP}$). La fonction *Connect_ Transit_ Transfert* commence par tester l'existence d'une séquence simple transit/transfert connectant N_1 à N_2 : un chemin de transit entre (g_1, p_1) et (g_2, p_1) suivi par un chemin de transfert entre (g_2, p_1) et (g_2, p_2). En cas de succès, cette séquence de chemins transit/transfert remplace le chemin initial calculé dans $\mathcal{CG} \cap \mathcal{CP}$. En cas d'échec, une séquence simple transfert/transit est alors testée. Dans le cas où ces deux types de connexions ne sont pas possibles, un nœud intermédiaire N_{int} est calculé (N_{int} correspond à une configuration intermédiaire appartenant au chemin connectant $N1$ et $N2$ calculé dans $\mathcal{CG} \cap \mathcal{CP}$) et le processus est réitéré entre $N1$ et N_{int} d'une part et entre N_{int}, et $N2$ d'autre part.

4.3.3 Optimisation et lissage du chemin de manipulation

Une fois l'étape de transformation des chemins calculés dans $\mathcal{CG} \cap \mathcal{CP}$ achevée, le chemin de manipulation est composé uniquement d'une séquence alternée de chemins de transit/transfert. Ce chemin est admissible et constitue une solution au problème de manipulation posé. Cependant, il est généralement formé par une longue succession de prises et de poses dont la plupart ne sont pas vraiment utiles pour la résolution du problème. Cette longue séquence de prises et de poses est essentiellement due au caractère probabiliste de l'approche. L'algorithme présenté ci-dessous permet de remplacer (si une solution existe) une séquence de plusieurs chemins de transit/transfert par un seul chemin de transit suivi d'un chemin de transfert. Toutes les connexions possibles entre les nœuds *"non-successive"* formant le chemin de manipulation sont testées et seules celles minimisant le nombre de séquences prise/pose sont gardées.

Algorithme 8 *Minimiser_Séquences (τ)*

1: $L \leftarrow \{N_1, ..., N_k\}$ (*Liste des nœuds formant τ*)
2: $i \leftarrow 1$
3: **Tant Que** $(i < k - 1)$ **Faire**
4: $j \leftarrow k$; $simplifier \leftarrow$ Faux
5: **Tant Que** $(j > i + 2)$ **Faire**
6: **Si** (Connect_Transit_Transfert (N_i, N_j)) **Alors**
7: $simplifier \leftarrow$ Vrai
8: $L \leftarrow \{N_1, ..., N_i, N_j, ..., N_k\}$; $i \leftarrow j$; $j \leftarrow j + 2$
9: **Sinon**
10: $j \leftarrow j - 1$
11: **Fin Si**
12: **Fin Tant Que**
13: **Si** $(\neg\, simplifier)$ **Alors**
14: $i \leftarrow i + 1$
15: **Fin Si**
16: **Fin Tant Que**

Après cette étape de minimisation des séquences de prises et de poses, une phase de lissage des chemins de transit ainsi que des chemins de transfert est effectuée par des techniques similaires à celles présentées dans [Laumond 94]. Elle consiste à découper aléatoirement le chemin de transit (resp. de transfert) en trois parties et à relier chacune des portions par le chemin de transit (resp. transfert) reliant leurs extrémités. Dans le cas où le coût d'une portion (la longueur du chemin dans ce cas) est plus faible que la séquence initiale du chemin, elle la remplace. Cette méthode est appliquée à tous les chemins de transit et de transfert constituant le chemin de manipulation. L'algorithme général de cette technique de lissage a été présenté dans le chapitre 1 (§1.2.1).

Remarque :
Les méthodes probabilistes capturent généralement la connectivité de l'espace libre dans une structure d'arbre (au lieu d'un graphe) pour des raisons de performance. Dans ce cas, elles ne permettent donc pas de capturer les différentes classes d'homotopies de CS_{libre}. Les étapes de minimisation et de lissage ne garantissent donc pas l'obtention du chemin optimal (au sens d'une optimisation globale prenant en compte toutes les classes d'homotopies).

4.4 Conclusion

Nous avons présenté dans ce chapitre une nouvelle approche pour la planification de tâches de manipulation. Le point fort de cette approche réside dans sa capacité à traiter des problèmes de manipulation sous contraintes de prises et de placements continues. Elle est basée sur une structuration de l'espace de recherche permettant de capturer directement la connectivité d'une sous-variété particulière de CS_{libre} dans un réseau probabiliste, calculé en considérant une chaîne cinématique virtuelle définie par le robot et les contraintes de prise et de pose de l'objet. Cette sous-variété correspond au lieu de connexions des chemins de transfert et des chemins de transit. Cette structuration nous a permis également de concevoir un planificateur [Sahbani 02] de tâches de manipulation permettant de générer automatiquement, à l'intérieur de domaines continus, la séquence de prises/poses composant le chemin de manipulation solution du problème.

Ce planificateur de tâches de manipulation a été implémenté et testé avec succès sur divers exemples. Les résultats obtenus ainsi que les différents exemples développés font l'objet du prochain chapitre. Nous présentons également une étude sur les performances ainsi que la sensibilité de notre planificateur au changement de ses paramètres de contrôle.

Chapitre 5

Résultats et Analyses

5.1 Introduction

Les algorithmes de manipulation présentés dans le chapitre précédent ont été implantés au sein de la plate-forme logicielle *Move3D* développée depuis plusieurs années dans le groupe Robotique et Intelligence Artificielle du LAAS. Après une brève présentation de l'architecture de cette plate-forme, nous décrivons l'extension effectuée sur cette dernière afin de supporter des problèmes de planification de tâches de manipulation.

La deuxième partie de ce chapitre présente les différents environnements développés pour étudier les performances de notre planificateur. Nous présentons également les résultats de simulation obtenus dans la résolution de ces problèmes. Une analyse de la sensibilité de notre planificateur à une variation des paramètres de contrôle est également développée à la fin de ce chapitre.

5.2 Présentation de la plate-forme logicielle Move3D

5.2.1 Architecture de Move3D

Move3D est une plate-forme logicielle développée dans l'objectif de mettre en œuvre un outil générique dédié à la planification de mouvement dans des environnements tridimensionnels [Nissoux 99a, Siméon 01a, Siméon 01b]. Elle est structurée selon trois couches génériques (Figure 5.1) :

- *La géométrie,*
- *La cinématique,*

– *L'algorithmique.*

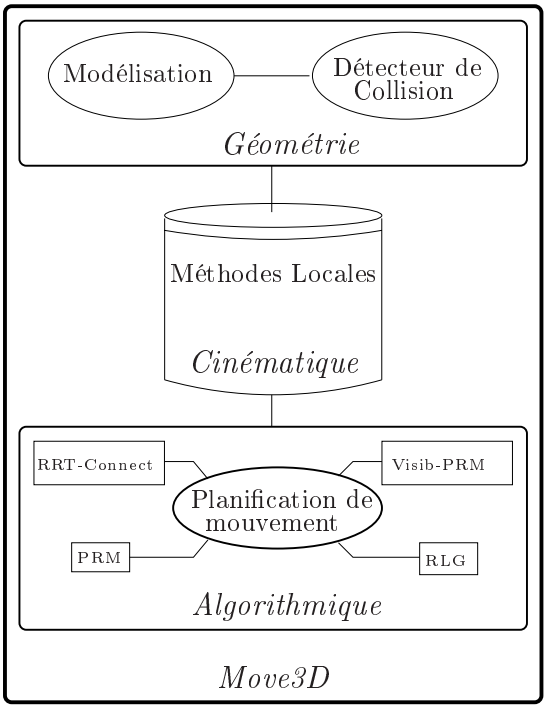


FIG. 5.1 – Architecture de la plate-forme logicielle *Move3D*.

La première couche (la géométrie) englobe deux modules importants de *Move3D* : la modélisation des problèmes posés et le détecteur de collision. Le module de modélisation permet à l'utilisateur la description des différents systèmes mécaniques ainsi que les environnements permettant la définition des problèmes de planification de mouvement. Il stocke également les différentes données géométriques décrivant le problème posé dans les structures de données de *Move3D*. Ces données peuvent être utilisées dans la suite par le module de détection de collision. Elles sont dans un premier temps filtrées. En effet, une requête de planification concerne uniquement une partie localisée de l'environnement. Le filtrage permet d'effectuer un pré-traitement des différentes données géométriques, décrivant le problème, en déterminant les objets qui peuvent entrer en collision lors de la recherche d'un chemin. Ce pré-traitement permet la réduction du nombre d'appels au détecteur de collision lors de la recherche d'un chemin. Après cette phase de filtrage, le détecteur de collision utilisé détermine si un chemin donné est en collision ou non. Il est composé de deux niveaux : un mécanisme de tri des paires de polyèdres et un algorithme de détection exacte. Le premier niveau gère dynamiquement une liste de paires de polyèdres (issu du filtrage). Elle est mise à jour dès que la position d'un polyèdre est modifiée. Le second niveau examine chaque paire de polyèdre et détermine s'ils sont en collision ou non en se basant sur des techniques spécifiques (régions de Voronoï des polyèdres, OBBtrees, ...). Trois algorithmes de détection de collision ont été intégrés dans *Move3D* : I-Collide ([Cohen 95]), V-Collide ([Hudson 97]) et KCD ([Geem 01]).

La deuxième couche correspond à la bibliothèque de méthodes locales permettant la construction de chemins locaux respectant les contraintes cinématiques et mécaniques du système mobile (déplacement d'une seule articulation à la fois, roulement sans glissement pour les systèmes à roues, ...etc). Un chemin local décrit l'évolution des paramètres de configurations du système entre deux situations initiale et finale données tout en respectant ces contraintes cinématiques. Une méthode locale construit un chemin local reliant un couple donné de configurations. Plusieurs méthodes locales ont été implantées dans *Move3D* : linear, manhattan, non-holonome, ... Une combinaison de ces méthodes peut être envisagée afin de définir de nouvelles méthodes locales plus complexes pour des systèmes mécaniques soumis à différentes contraintes de mouvement (pour un manipulateur mobile, une méthode locale de type non-holonome est utilisée dans la planification de mouvement de la base du robot alors qu'une méthode de type linear est utilisée pour le bras manipulateur fixé sur cette base).

La dernière couche est composée des différents algorithmes développés pour la planification de mouvement. Elle intègre plusieurs techniques probabilistes de planification de mouvement proposées dans la littérature. La méthode *PRM* (*Probabilistic Roadmap Method*) [Kavraki 96] ainsi que deux de ces variantes (*Visibilité-PRM* [Siméon 00] et *Gaussian sampling* [Boor 99]) font partie de ces techniques probabilistes implantées dans *Move3D*. Le principe de ces méthodes est de construire un graphe connectant des configurations libres tirées aléatoirement, puis d'utiliser ce graphe pour répondre rapidement à plusieurs requêtes de planification de mouvement. *Move3D* intègre également une méthode probabiliste par diffusion (*RRT-Connect* [Kuffner 00d]). Cette dernière est utilisée pour répondre à une requête de planification sans nécessiter la construction préalable d'un graphe.

Move3D est muni également d'une interface graphique (elle n'est pas présentée dans la figure 5.1) permettant à l'utilisateur la définition de problème de planification de mouvement, ainsi que l'appel aux algorithmes et la visualisation des trajectoires obtenues. Elle permet également à l'utilisateur de choisir la méthode de planification désirée, et de régler ces paramètres selon le problème posé.

5.2.2 Extension de l'architecture au problème de manipulation

L'introduction de la manipulation dans *Move3D* augmente l'architecture existante et a nécessité la modification de certains modules. La figure 5.2 présente l'extension de l'architecture au problème de manipulation. Cette modification touche essentiellement la première et dernière couche de la plate-forme logicielle (les parties grisées dans la figure).

Dans le cas de problèmes de planification de mouvement, l'environnement est composé d'objets statiques formant les obstacles et du robot évoluant parmi ces derniers. Pour résoudre des problèmes de planification de tâches de manipulation, les deux modules formant la première couche (modélisation et détection de collision) ont évolué afin de tolérer la présence d'une nouvelle entité dans l'environnement, à savoir les objets déplaçables. Le module modélisation a été étendu afin de supporter la description des

différents éléments définissant un problème de manipulation (objets déplaçables, placements stables, différentes classes de prises, ...etc). Le module de détection de collision a lui aussi été modifié afin de s'adapter aux exigences des problèmes de manipulation. Rappelons qu'une opération de prise (resp. de pose) est accomplie à travers un contact entre l'objet déplaçable et le robot (resp. l'obstacle définissant la zone de pose stable), ce qui n'est pas permis par le détecteur de collision. Ce problème peut être résolu en modifiant les phases de filtrage et de sélection afin que la paire objet-robot (resp. objet-zone de placement) ne soit prise en considération le long d'un chemin de transfert (resp. chemin de transit).

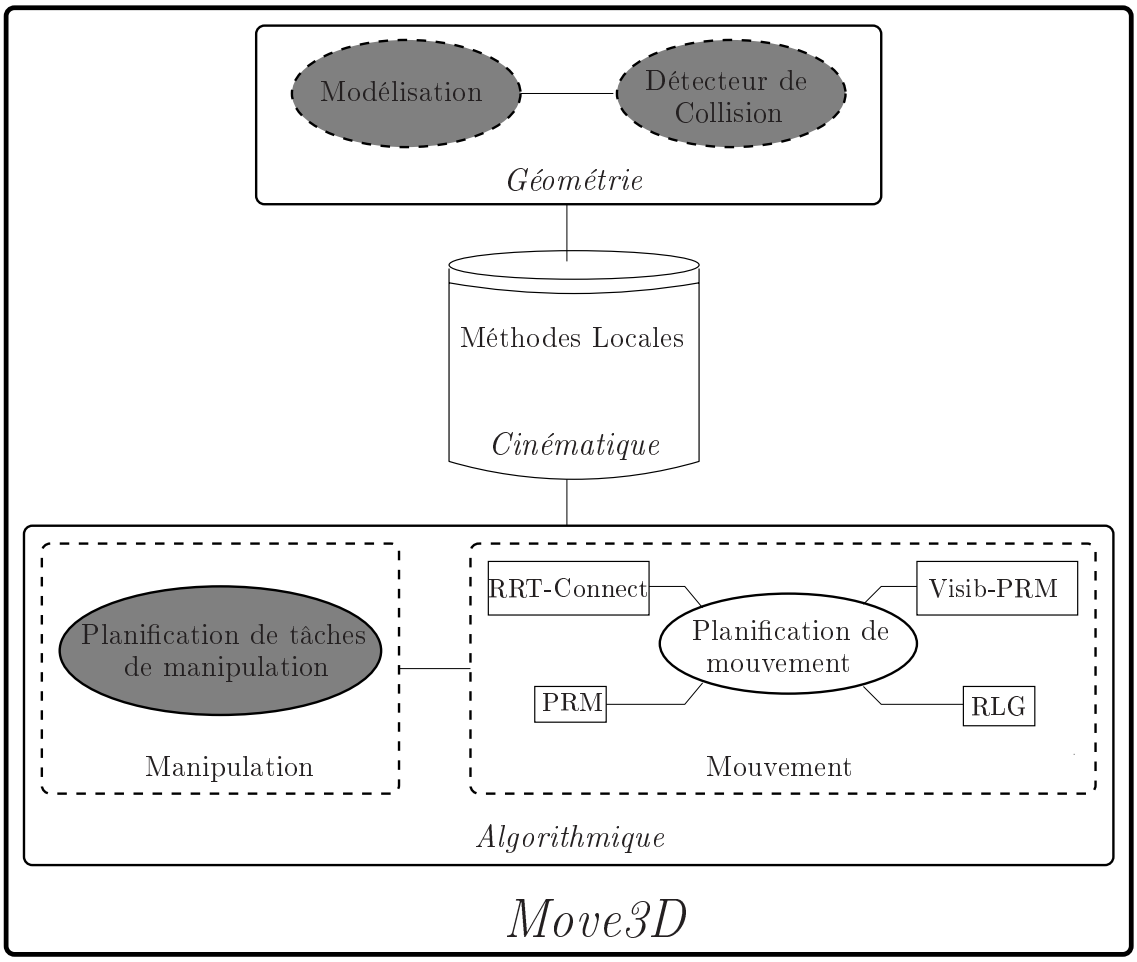


FIG. 5.2 – Extension aux problèmes de planification de tâches de manipulation.

Par ailleurs, nous avons intégré dans *Move3D* les différents algorithmes de planification de tâches de manipulation décrits dans les chapitres 3 et 4. La section suivante présente divers problèmes de planification de tâches de manipulation résolus en utilisant ces algorithmes.

5.3 Exemples de problèmes de manipulation

5.3.1 Description d'un problème de manipulation

Un problème de manipulation est décrit par la donnée des configurations initiale et finale du système global (robot + objet) et par la spécification de l'ensemble de prises et de poses. Nous nous sommes limités dans notre implantation à des domaines de placements rectangulaires (de type plan). Un placement continu est donc défini par la donnée du quadruplet $(x_{min}, x_{max}, y_{min}, y_{max})$ décrivant les limites de ce plan et par un vecteur précisant l'orientation de placement stable sur cette zone de placement. L'ensemble des facettes de l'objet déplaçable permettant le contact avec la zone de placement afin de réaliser une pose stable sont aussi précisées. Les prises continues sont également limitées, dans notre implémentation, à une translation de l'organe terminal du robot le long de deux facettes parallèles de l'objet. Une prise continue est définie par la donnée d'une transformation homogène et d'un paramètre variant dans un intervalle fixé.

Plusieurs environnements ont été développés afin de tester les performances de notre planificateur de tâches de manipulation. Les résultats obtenus font l'objet des sections suivantes. Les temps de calcul donnés ont été obtenus sur une station SUN ultra 10 avec un processeur cadencé à 330Mhz. Ils comprennent l'étape de calcul des différentes composantes connexes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ ainsi que leur connexion par des chemins de transfert/transit (uniquement la construction du graphe de manipulation). La transformation des chemins calculés dans $\mathcal{CG} \cap \mathcal{CP}$ et la recherche du chemin solution sont effectuées une fois le graphe de manipulation construit.

5.3.2 Le bras manipulateur et la cage

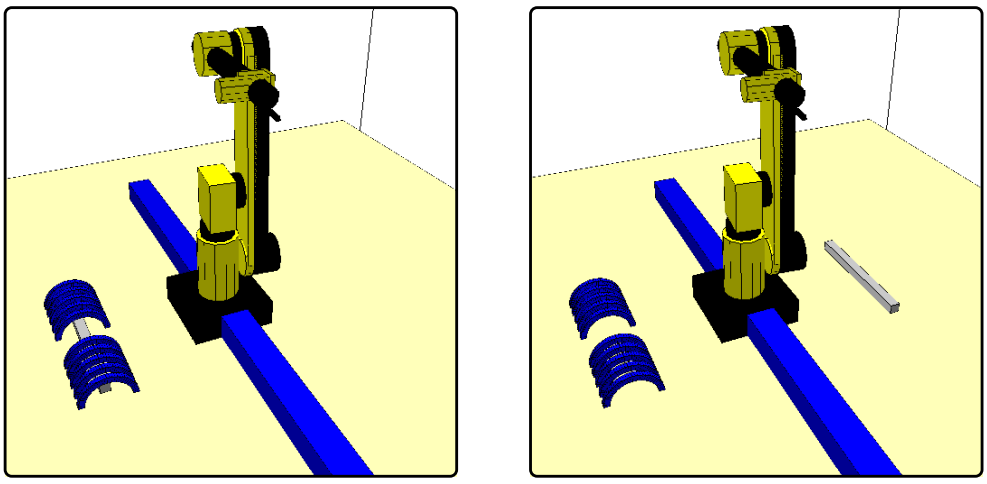


FIG. 5.3 – Configurations initiale (à gauche) et finale (à droite) du système global.

Un bras manipulateur à 6 degrés de liberté doit trouver la séquence de mouvement lui permettant de sortir l'objet déplaçable (la barre) de sa position initiale à l'intérieur de la cage (obstacle) et de la

déplacer vers un placement final. Les configurations initiale et finale du système global sont illustrées dans la figure 5.3. Une prise et un placement continus sont définis comme entrées du problème de manipulation (le robot peut effectuer une saisie le long de l'objet et peut le placer n'importe où sur le sol).

Nous avons choisi ce problème pour comparer deux techniques différentes utilisées pour capturer la connectivité du sous-espace $\mathcal{CG} \cap \mathcal{CP}$: *PRM* [Kavraki 95a] et *Visibilité-PRM* [Nissoux 99a]. Un ensemble de 20 tests différents (correspondant chacun à une série de nombres aléatoires différente) a été effectué pour chacune de ces deux méthodes. Nous avons réglé le paramètre contrôlant le choix de la stratégie d'expansion à 99% (cf. chapitre 4, § 4.3.1). Ce qui se traduit par une probabilité de choisir une expansion du graphe de manipulation dans $\mathcal{CG} \cap \mathcal{CP}$ de 0.99 (vs. une probabilité de choisir une connexion par transit/transfert de 0.01). La figure 5.4 illustre deux graphes de manipulation construits par notre planificateur en utilisant respectivement *PRM* (graphe de gauche) et *Visibilité-PRM* (graphe de droite) pour l'exploration de $\mathcal{CG} \cap \mathcal{CP}$.

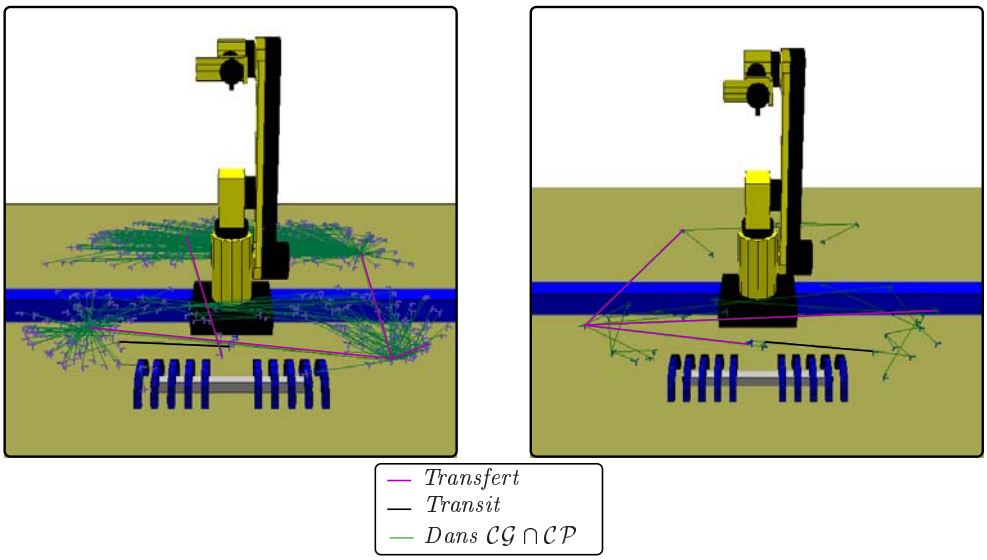


FIG. 5.4 – Le graphe de manipulation calculé avec : a) *PRM* (gauche). b) *Visibilité-PRM* (droite).

Les caractéristiques de ces deux graphes sont présentées dans le tableau 5.1. La différence entre les temps de calcul est directement liée au nombre de nœuds générés par les deux méthodes. En effet, l'algorithme *PRM* garde tous les nœuds tirés, ce qui nécessite un grand nombre d'appels à la méthode locale pour tester les connexions dans $\mathcal{CG} \cap \mathcal{CP}$. Moins de tests sont nécessaires avec l'algorithme *Visibilité-PRM* qui ne conserve qu'un nombre réduit de nœuds (les gardiens et les connecteurs). Pour les connexions par des chemins de transit/transfert, nous avons utilisé (pour les deux graphes de manipulation) un même graphe initial calculé avec *Visibilité-PRM* (selon l'approche par graphe dynamique présenté dans le chapitre 3). Ceci explique les temps relativement faibles mis par le planificateur pour

connecter les composantes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ par des chemins de transit/transfert. Le nombre de chemins de manipulation indiqué dans ce tableau englobe les chemins de transit, de transfert et ceux calculés dans $\mathcal{CG} \cap \mathcal{CP}$.

	PRM	**Visibilité-PRM**
Temps global	68.97 sec	45.56 sec
Calcul des composantes de $\mathcal{CG} \cap \mathcal{CP}$	45.90 sec	28.42 sec
Connexions par transit/transfert	21.78 sec	11.46 sec
Nbre. de nœuds	583	32
Nbre. de chemins de manip.	577	29

TAB. 5.1 – Caractéristiques des graphes de manipulation construits.

Un résumé des tests effectués pour évaluer les performances de notre planificateur, selon l'utilisation de l'une de ces méthodes, est présenté dans le tableau 5.2. Ces résultats montrent que l'utilisation de la technique *Visibilité-PRM* pour capturer la connectivité du sous-espace $\mathcal{CG} \cap \mathcal{CP}$ est plus efficace pour la résolution de ce problème. Nous avons présenté également dans ce tableau le temps minimal et maximal mis par le planificateur pour construire le graphe de manipulation selon la technique utilisée pour explorer $\mathcal{CG} \cap \mathcal{CP}$.

	PRM	**Visibilité-PRM**
Temps moyen de résolution	61.59 sec	45.82 sec
Temps max. de résolution	385.04 sec	183.58 sec
Temps min. de résolution	4.18 sec	13.13 sec

TAB. 5.2 – Comparaison entre les temps de résolution du problème selon la technique utilisée pour capturer la connectivité du sous-espace $\mathcal{CG} \cap \mathcal{CP}$.

Nous avons également testé l'influence de la méthode utilisée pour la construction du graphe initial (graphe utilisé pour répondre aux requêtes de transit/transfert) sur les performances du planificateur. Un exemple de deux graphes calculés pour le robot seul dans l'environnement est présenté dans la figure 5.5. Ces graphes sont calculés en utilisant respectivement une méthode probabiliste de type *PRM* (image de gauche de cette figure) et *Visibilité-PRM* (image de droite). Nous avons testé les différentes com-

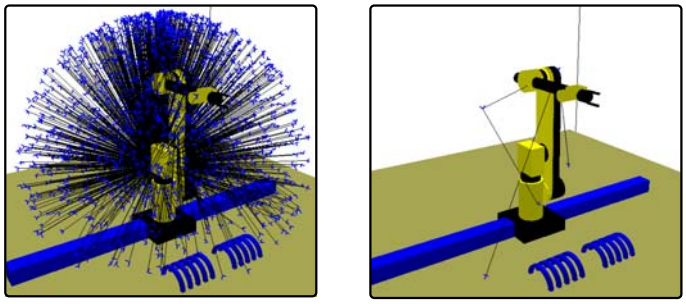


FIG. 5.5 – Graphe initial pour le robot calculé avec : a) PRM. b) Visibilité-PRM.

binaisons possibles d'utiliser l'une de ces méthodes soit dans la construction du graphe initial, soit dans le calcul des composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$. Le tableau 5.3 illustre les temps de calcul du graphe de manipulation selon les techniques utilisées pour le calcul du graphe initial et l'exploration de $\mathcal{CG} \cap \mathcal{CP}$. Les temps de construction du graphe de manipulation, en utilisant un graphe initial calculé par *Visibilité-PRM* (présentés dans ce tableau), correspondent aux temps moyens obtenus selon l'utilisation de l'une de ces deux approches dans l'exploration de $\mathcal{CG} \cap \mathcal{CP}$. Nous avons réutilisé la même série de tests (donc les mêmes séquences de nombres aléatoires qui ont conduit à ces temps moyens) en initialisant le problème par un graphe (pour le robot seul) calculé par *PRM*. Ces résultats obtenues (tableau 5.3) montrent que l'utilisation d'un graphe initial calculé avec la méthode *Visibilité-PRM* est la plus efficace. En effet le graphe calculé avec la méthode *PRM* comporte un grand nombre d'arêtes et de nœuds et demande donc un grand nombre de tests de collision pour valider les différentes composantes de ce graphe à chaque requête de transit et de transfert. La taille réduite de l'objet déplaçable contribue également dans l'efficacité de l'utilisation d'un graphe initial calculé avec *Visibilité-PRM*.

Graphe initial / **Exploration de $\mathcal{CG} \cap \mathcal{CP}$**	**PRM**	**visibilité-PRM**
PRM	82.88 sec	61.59 sec
visibilité-PRM	79.12 sec	45.82 sec

TAB. 5.3 – Influence de la technique utilisée pour le calcul du graphe initial sur les temps de résolution.

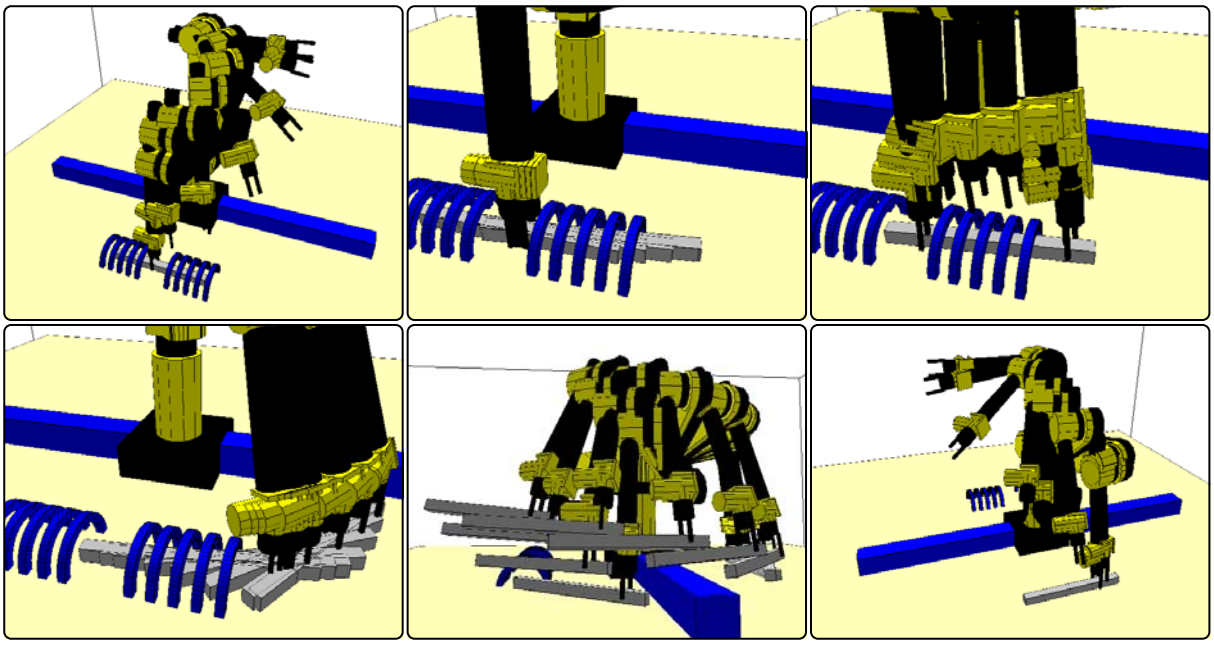


FIG. 5.6 – La séquence de chemins transit/transfert composant le chemin de manipulation solution au problème.

Le chemin de manipulation solution du problème est présenté dans la figure 5.6. Il est composé

de 20 chemins élémentaires avec 8 opérations de changement de prise. La transformation des chemins calculés dans $\mathcal{CG} \cap \mathcal{CP}$ en une séquence de chemin de transit/transfert est accomplie en 18 secondes. Après cette transformation, une étape de minimisation des séquences de prises et de poses est effectuée. Elle nécessite 180 secondes de temps de calcul. La dernière étape consiste à lisser le chemin obtenu. Elle est effectuée en 3 secondes.

5.3.3 Le bras manipulateur avec multiples orientations de prise/pose de l'objet

Considérons le problème illustré dans la figure 5.7. Le bras manipulateur doit trouver l'enchaînement de mouvements nécessaire pour déplacer l'objet déplaçable de sa position initiale (image en haut à droite) à son placement final (image en bas à droite). Trois placements stables continus sont définis comme entré du problème (caractérisé chacun par une orientation de pose de l'objet sur le dessus de la table en face du robot). Huit prises continues sont également définies correspondant chacune à une saisie le long d'une des côtés de l'objet.

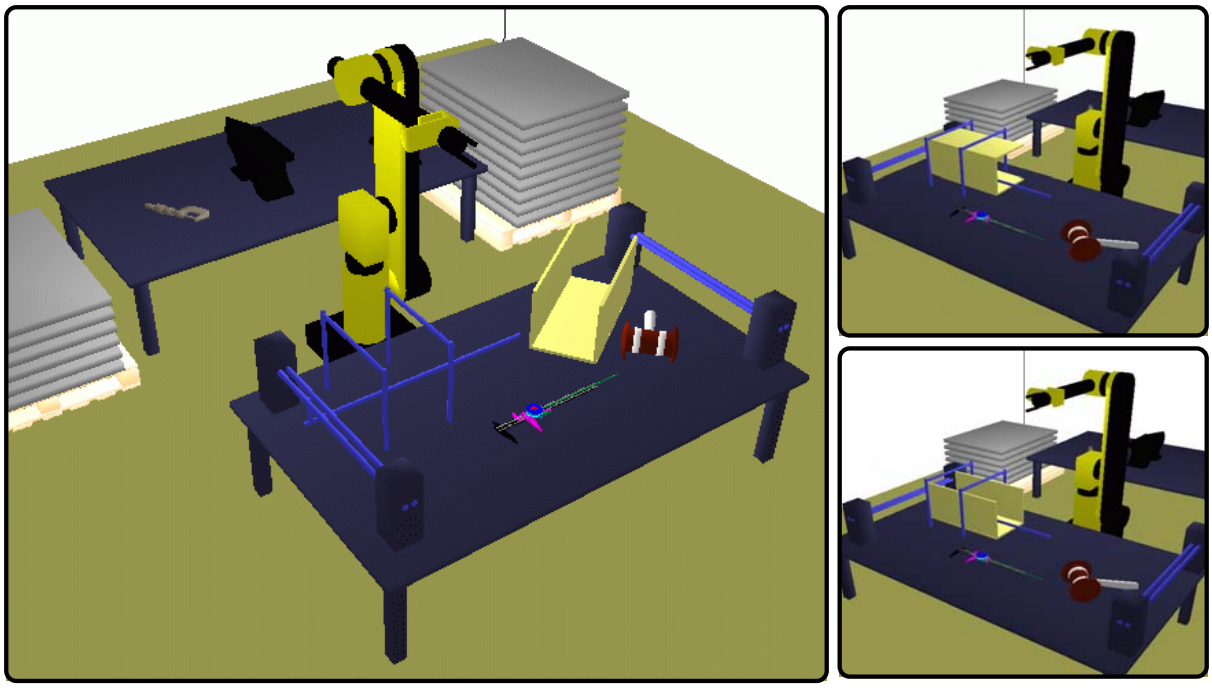


FIG. 5.7 – Scène et configurations initiale et finale du système global.

La difficulté de ce problème réside dans le grand nombre de classes de prise défini comme entrées du problème de manipulation. Une autre difficulté consiste à trouver la séquence de prise et pose adéquate permettant au robot de dégager l'objet de son placement (contraint) initial et de le remettre dans ce placement avec une autre orientation de pose. Le graphe de manipulation construit par le planificateur est illustré dans la figure 5.8. Ces caractéristiques sont présentées dans le tableau 5.4.

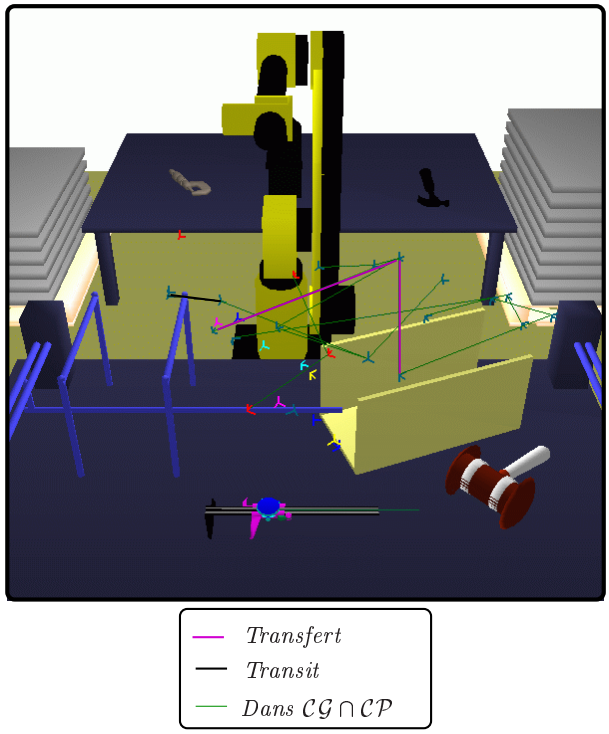


FIG. 5.8 – Le graphe de manipulation construit pour ce problème.

Graphe de manipulation	**Caractéristiques**
Temps global	330.23 sec
Calcul des composantes de $\mathcal{CG} \cap \mathcal{CP}$	10.14 sec
Connexions par transit/transfert	317.94 sec
Nbre. de nœuds	32
Nbre. de chemins de manip.	28

TAB. 5.4 – Caractéristiques du graphe de manipulation.

Le nombre de nœuds réduit de ce graphe est dû à l'utilisation de la méthode de *Visibilité-PRM* pour capturer les différentes composantes connexes du sous-espace $\mathcal{CG} \cap \mathcal{CP}$. Notons qu'une grande partie du temps de calcul est passée dans le test de connexions par des chemins de transfert/transit. Ceci montre l'intérêt de l'approche proposée qui limite le nombre de ce type de connexion en commençant par calculer les différentes composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$.

Le chemin de manipulation solution de problème est présenté dans la figure 5.9. Il est composé de 39 chemins élémentaires avec 14 opérations de changement de prise. La transformation des chemins calculés dans le sous-espace $\mathcal{CG} \cap \mathcal{CP}$ en une séquence de chemins de transit/transfert a été effectuée en 65 secondes. La minimisation des séquences de prises et de poses a nécessité 85 secondes de temps de calcul. Finalement, le lissage du chemin solution a été effectué en 17 secondes.

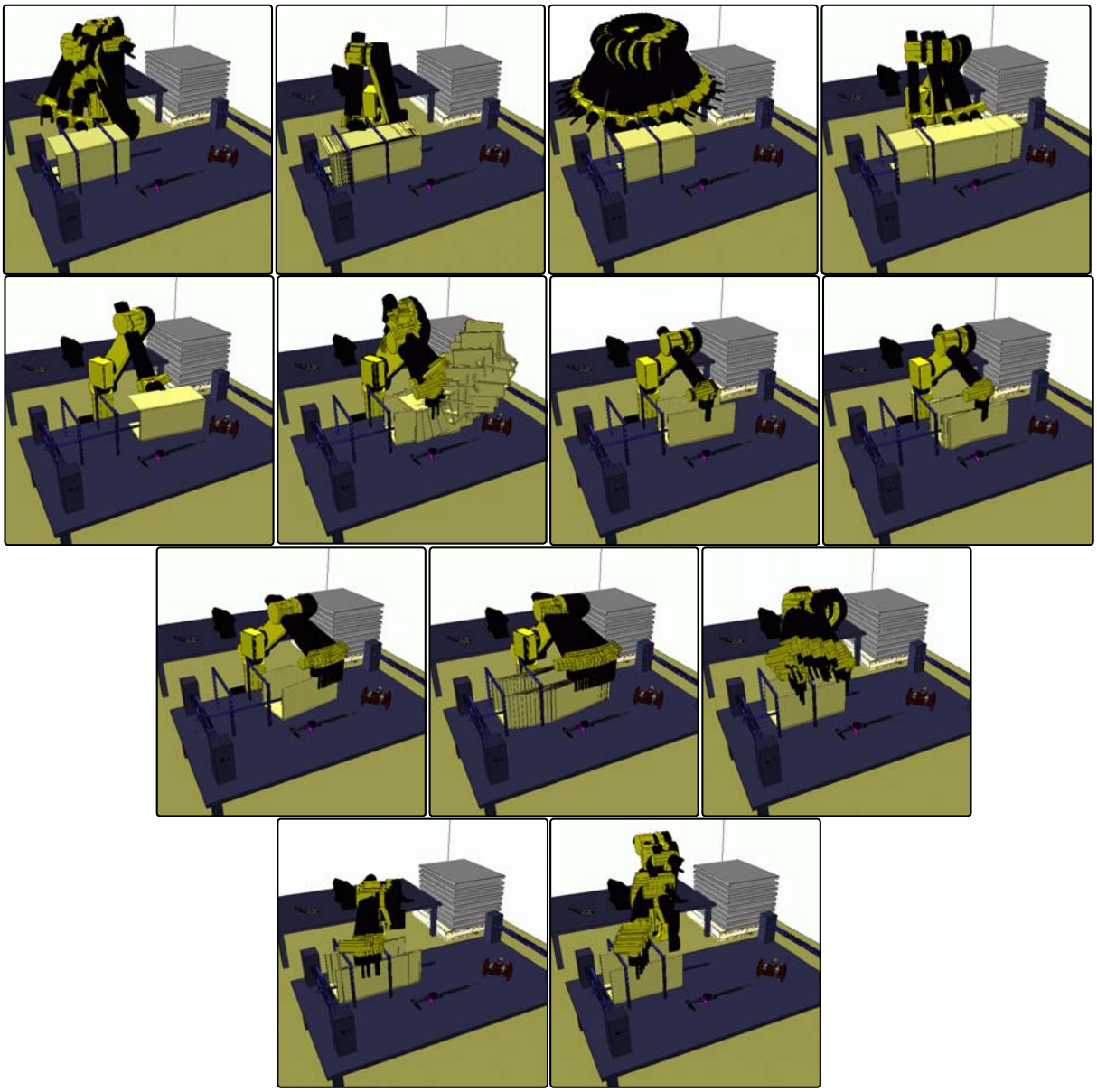


FIG. 5.9 – La séquence de chemins transit/transfert composant le chemin de manipulation solution au problème.

5.3.4 Manipulateur holonome mobile

Dans le problème de manipulation présenté dans la figure 5.10, le manipulateur mobile (9 degrés de liberté) peut passer d'un côté de la scène à un autre uniquement à travers le passage sous le grand obstacle en forme de X. Cependant, ce passage est trop étroit pour l'objet déplaçable (objet en forme de carré). Un ensemble de prises continues est défini tout autour de l'objet. Ce dernier peut être placé sur le long de l'obstacle au milieu de la scène. Les configurations initiale et finale du système global sont présentées dans la figure 5.10 (resp. image en haut et en bas à droite).

La difficulté dans cet exemple réside dans la redondance présentée par le système mécanique. Elle

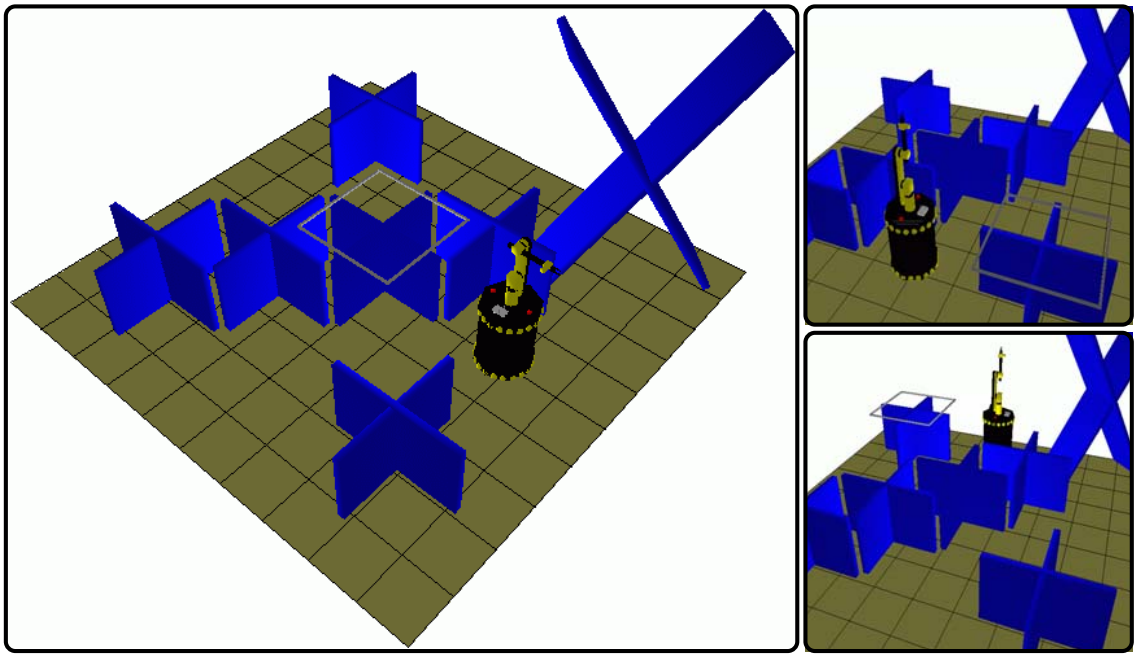


FIG. 5.10 – Exemple de problème de manipulation avec un robot redondant.

se traduit par un ensemble infini de configurations du robot réalisant la même prise. La redondance est un problème difficile dans la planification de mouvement de chaînes cinématiques fermées. Cet exemple nous permet de montrer l'efficacité de l'algorithme *RLG* [Cortés 03] dans la planification de mouvement de ce type de système.

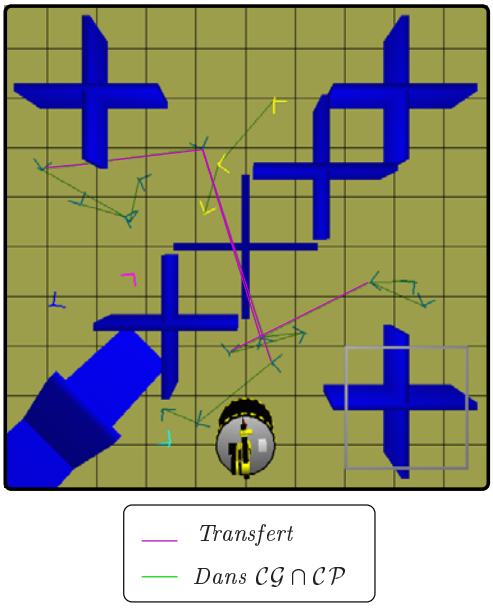


FIG. 5.11 – Le graphe de manipulation calculé pour ce problème.

Graphe de manipulation	Caractéristiques
Temps global	288.88 sec
Calcul des composantes de $\mathcal{CG} \cap \mathcal{CP}$	7.81 sec
Connexions par transit/transfert	279.54 sec
Nbre. de nœuds	21
Nbre. de chemins de manip.	21

FIG. 5.12 – Caractéristiques du graphe de manipulation.

Le graphe de manipulation calculé pour ce problème est illustré dans la figure 5.11. Ces caractéristiques, présentées dans le tableau 5.12, montrent qu'une grande partie du temps de construction du graphe de manipulation est passée dans les tests de connexions par des chemins de transit/transfert des composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$. Ceci est dû à la nature du problème dont la solution est composée essentiellement de ce type de chemins.

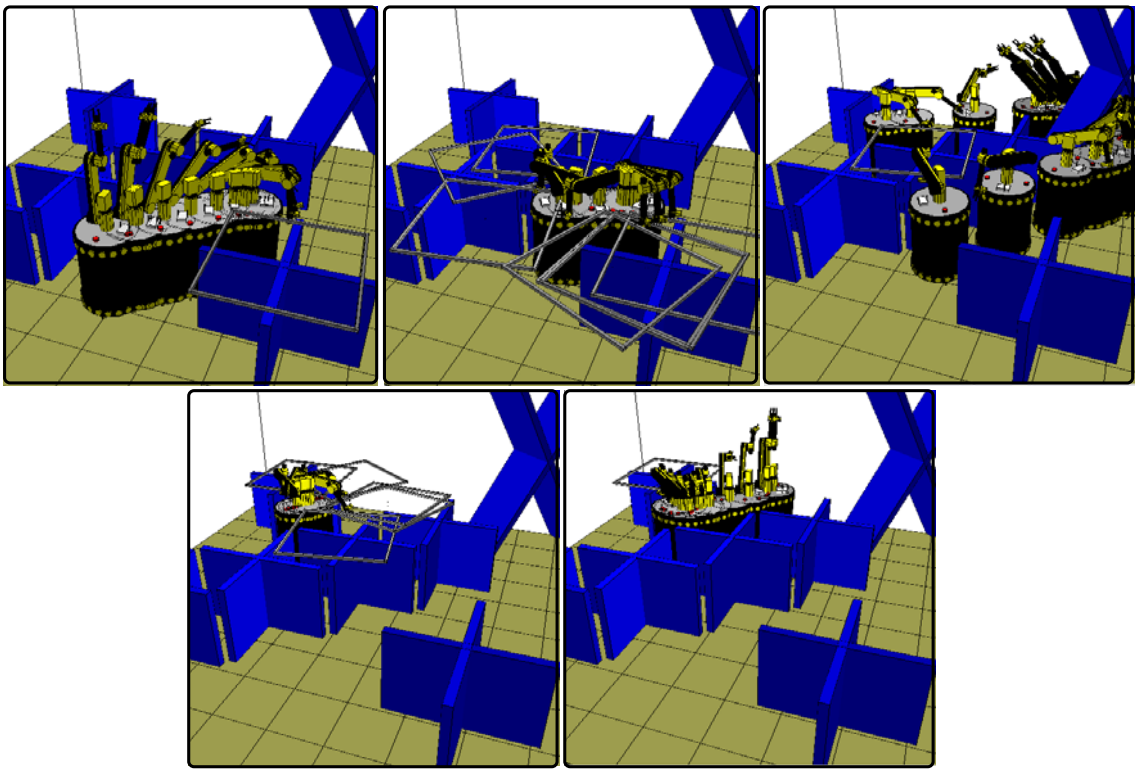


FIG. 5.13 – Le chemin de manipulation solution de ce problème.

Le chemin de manipulation solution du problème est présenté dans la figure 5.13. Il est composé de 5 chemins élémentaires et une seule opération de changement de prise. La transformation des chemins calculés dans $\mathcal{CG} \cap \mathcal{CP}$ est effectuée en 2 secondes, alors que les étapes de transformation et de lissage de ce chemin ont été accomplies en 5 secondes.

5.3.5 Pont Roulant

Le problème de manipulation illustré dans la figure 5.14 présente un exemple industriel traité dans le cadre du projet MOLOG [Siméon 01a]. Contrairement aux autres exemples, la structure mécanique du robot dans ce problème est simple. Il s'agit d'un pont roulant à 4 degrés de liberté. Deux prises discrètes sont définies comme entrées du problème, correspondant chacune à une saisie de l'objet par l'un des deux anneaux. Le robot peut placer l'objet déplaçable le long des obstacles statiques formant le sol.

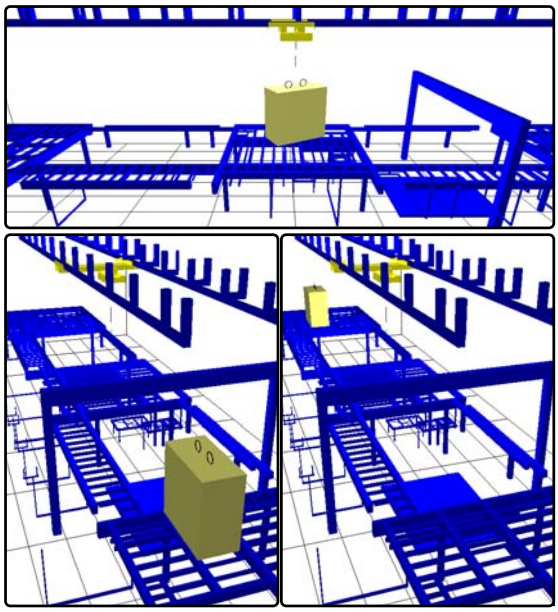


FIG. 5.14 – Exemple de problème de logistique industrielle (projet MOLOG).

La difficulté dans ce problème réside dans la taille importante de l'objet déplaçable par rapport aux autres corps constituant la scène. Les requêtes de transfert sont les plus coûteuses en temps de calcul dans ce type d'environnement. Une grande partie du graphe initial ne peut être utilisable pour répondre à ce type de requête (plusieurs arêtes du graphe initial sont marquées bloquantes). Le graphe de manipulation construit pour ce problème est présenté dans la figure 5.15. Le tableau 5.5 montre les caractéristiques de ce graphe. Nous remarquons que le temps nécessaire pour la construction de ce graphe est passé presque entièrement dans les tests de connexions par des chemins de transfert/transit.

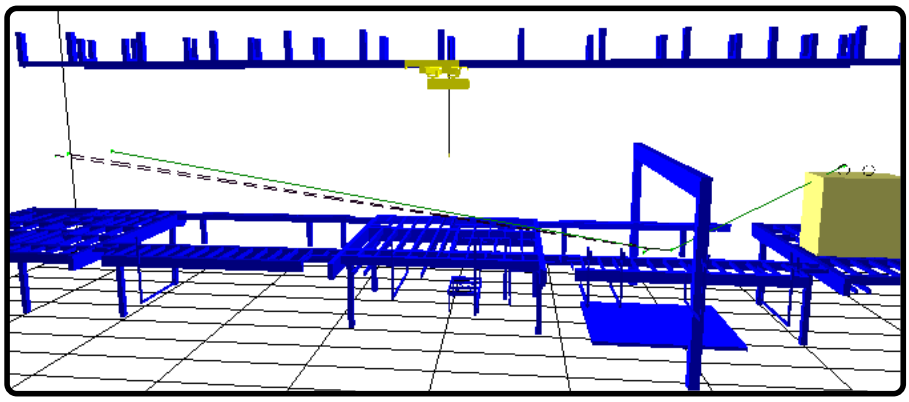


FIG. 5.15 – Le graphe de manipulation construit par le planificateur.

Le planificateur détermine automatiquement le placement intermédiaire de l'objet sous l'arceau permettant après un changement de prise de déposer l'objet à la position but. Le chemin de manipu-

Graphe de manipulation	**Caractéristiques**
Temps global	146 sec
Calcul des composantes de $\mathcal{CG} \cap \mathcal{CP}$	1 sec
Connexions par transit/transfert	143 sec
Nbre. de nœuds	10
Nbre. de chemins de manip.	6

TAB. 5.5 – Caractéristiques du graphe de manipulation.

lation solution à ce problème est présenté dans la figure 5.15. Il est composé de 5 chemins élémentaires avec une seule opération de changement de prise. La solution de ce problème comporte un nombre réduit de chemins calculés dans $\mathcal{CG} \cap \mathcal{CP}$, ce qui rend l'étape de leur transformation assez rapide. Elle est effectuée en moins d'une seconde. La phase de minimisation des séquences de prises et de poses est accomplie dans 14 secondes. Finalement, l'étape de lissage du chemin solution a nécessité 17 secondes.

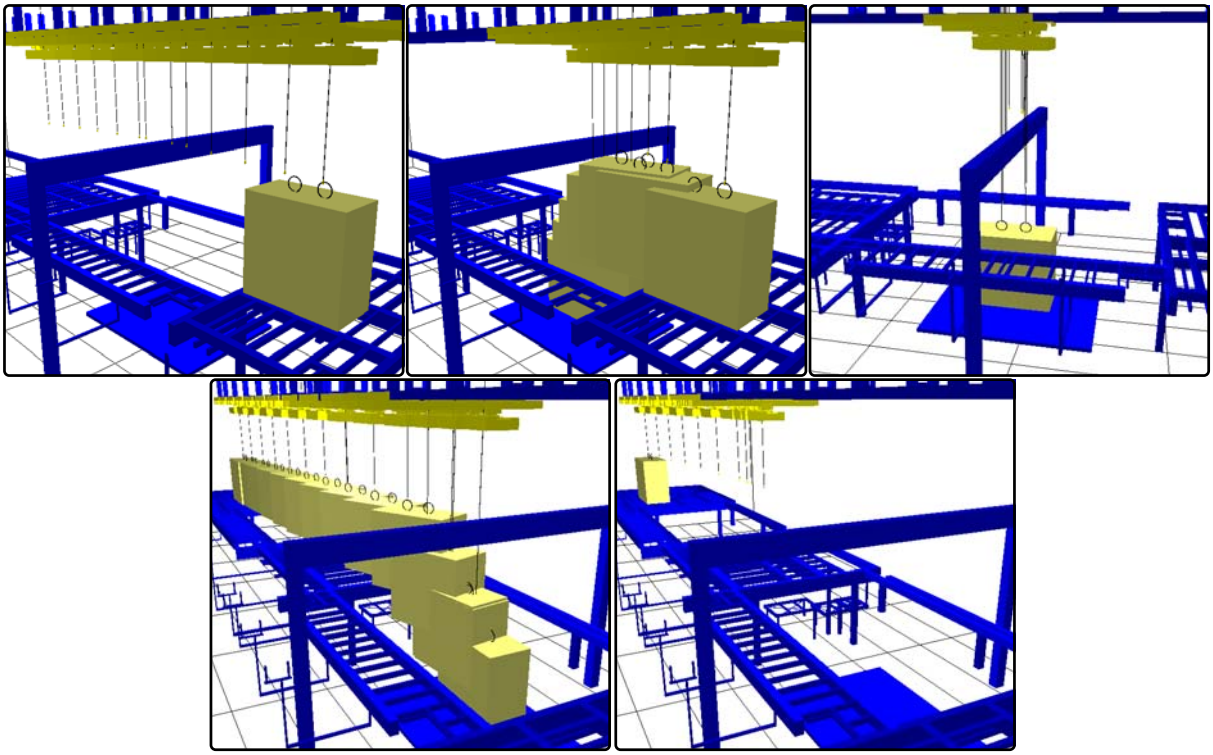
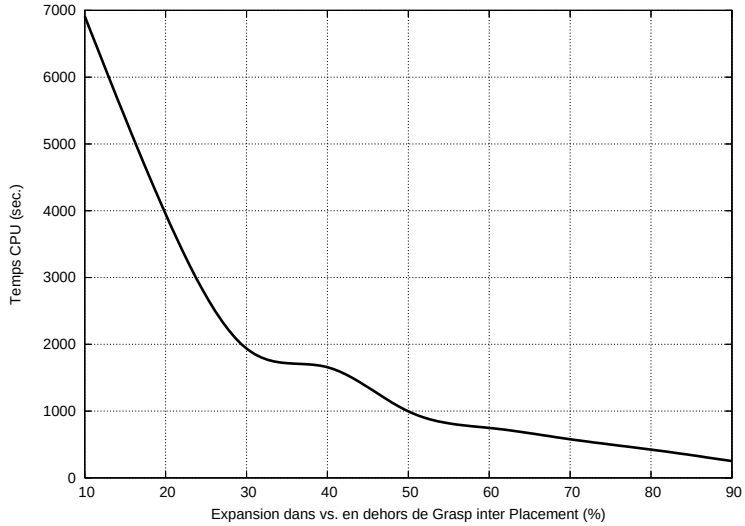


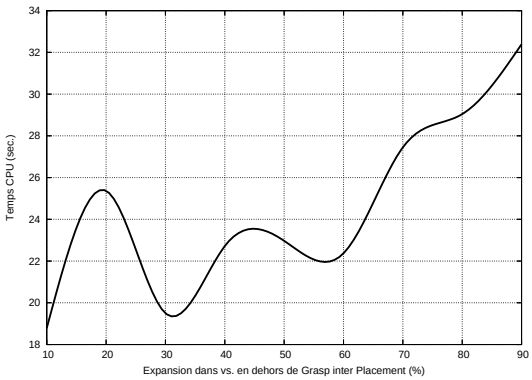
FIG. 5.16 – Le chemin de manipulation calculé pour ce problème.

5.4 Analyse de la sensibilité de l'algorithme en fonction d'une variation des paramètres

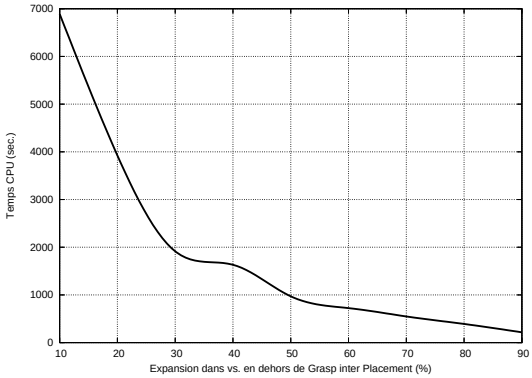
La construction du graphe de manipulation s'effectue en deux étapes : le calcul des composantes connexes de $\mathcal{CG} \cap \mathcal{CP}$ et la détermination de leur connectivité par des chemins de transit/transfert. Nous avons vu que notre planificateur construit incrémentalement le graphe de manipulation en alternant ces deux étapes. Cette alternance est assurée par un tirage aléatoire biaisé en fonction de la taille du graphe. Une question se pose donc : Quelle est l'influence du choix de la stratégie d'expansion (ou encore du réglage de l'alternance entre les deux étapes de construction du graphe) sur les performances du planificateur ?



Temps de calcul global pour la construction du graphe de manipulation.



Temps de calcul des chemins dans $\mathcal{CG} \cap \mathcal{CP}$.



Temps de calcul des chemins de transit/transfert.

FIG. 5.17 – Évolution du temps de construction du graphe de manipulation en fonction d'une variation de la probabilité de choisir l'une des stratégies d'expansion (expansion dans $\mathcal{CG} \cap \mathcal{CP}$ vs. connexions par transit/transfert).

Nous avons étudié ce problème sur l'exemple du *"bras manipulateur et la cage"* (premier exemple

présenté dans la section précédente). Un ensemble de tests expérimentaux a été effectué consistant à faire varier un paramètre représentant la probabilité de choisir l'une des stratégies et de voir l'influence sur les performances du planificateur. Ce paramètre est noté P_{strag}. Les résultats obtenus sont présentés dans la figure 5.17. La première courbe représente l'évolution du temps mis par le planificateur pour résoudre ce problème (noté T_{global}) en fonction d'une augmentation de la probabilité de choisir la stratégie d'expansion du graphe dans $\mathcal{CG} \cap \mathcal{CP}$. Les deux autres courbes représentent la variation du temps de calcul passé respectivement dans la détermination des composantes de $\mathcal{CG} \cap \mathcal{CP}$ (noté T_{comp}) et le calcul de leur connexion par des chemins de transit/transfert ($T_{connect}$).

Ces résultats obtenus conduisent à deux conclusions importantes. La première est tirée de la courbe indiquant l'évolution de T_{comp} en fonction de P_{strag} (courbe en bas à gauche). En effet, cette courbe montre que l'exploration de $\mathcal{CG} \cap \mathcal{CP}$ est peu sensible à une variation de P_{strag} (T_{comp} est comprise entre 19 et 32 secondes). Ces résultats confirment donc les performances de l'algorithme RLG dans l'exploration d'espaces hautement contraints (mécanismes contenant des chaînes cinématiques fermées). La deuxième conclusion concerne l'évolution de T_{global} en fonction P_{strag}. En effet, quand la probabilité de choisir une expansion dans $\mathcal{CG} \cap \mathcal{CP}$ (P_{strag}) tend vers zéro, le problème peut être vu comme une discrétisation de l'ensemble continu de prises et de placements. Ainsi, plus la recherche tend à capturer la connectivité de $\mathcal{CG} \cap \mathcal{CP}$ (au lieu de capturer indépendamment la connectivité de $\mathcal{CG}$ et $\mathcal{CP}$) plus rapide est la résolution du problème. Ceci montre bien l'efficacité de notre approche qui vise à capturer directement la connectivité de $\mathcal{CG} \cap \mathcal{CP}$ plutôt que de tenter de connecter des configurations de cette sous-variété par des chemins de transit et de transfert.

5.5 Conclusion

Plusieurs résultats et simulations ont été présentés dans ce chapitre montrant l'efficacité et la généricité de notre approche. Cependant, ces résultats nécessitent quelques commentaires.

Tout d'abord, il faut noter que la complexité géométrique des environnements est très variable, ce qui explique les écarts entre les différents temps de calcul. Cette complexité varie de quelques dizaines de polyèdres (environnement composé de 35 polyèdres dans le cas du bras manipulateur avec la cage avec 1714 facettes) à plus de 1000 polyèdres (1767 polyèdres pour l'exemple du pont roulant avec 10198 facettes). Ceci a une grande influence sur le temps de calcul nécessaire pour un test de collision et donc sur le temps de calcul des connexions des différentes composantes du graphe, ce qui explique le temps mis par le planificateur pour résoudre le problème du pont roulant, pourtant plus simple que les autres exemples de test.

Par ailleurs, la complexité des environnements n'est pas le seul facteur expliquant cette disparité dans les temps de calcul. La complexité de la tâche de manipulation est également variable d'un problème à un autre. En effet, la séquence de mouvements nécessaire pour résoudre le problème "bras +

cage" est de loin plus difficile que celle nécessaire pour résoudre l'exemple du manipulateur mobile.

Ces exemples montrent la capacité de notre planificateur à traiter une large classe de problème de manipulation. Les performances actuelles permettent d'envisager d'attaquer des problèmes de manipulation en présence de plusieurs objets déplaçables et de plusieurs robots.

Conclusion générale

La ligne forte des travaux présentés dans ce manuscrit est de favoriser des méthodes effectives de planification. En effet, face au manque de robustesse et à la forte complexité des algorithmes exacts, nous avons choisi la perte de complétude et privilégié les notions de complétudes approchée ou probabiliste, ainsi que la performance et la robustesse. Ces méthodes nous ont permis d'étendre dans un premier temps l'approche introduite dans [Alami 95] afin d'attaquer des problèmes de manipulation plus complexes dans des environnements hautement encombrés.

Nous avons également présenté dans ce mémoire une approche plus générique capable de traiter le cas de problèmes de manipulation sous contraintes continues. Elle est basée sur une structuration géométrique de l'espace de recherche permettant de capturer directement la connectivité d'un sous-espace spécifique dans une structure de graphe. Ce graphe est calculé en considérant une chaîne cinématique virtuelle (fermée) définie par le robot et les contraintes de prise et de pose. Par ailleurs, la structuration de l'espace de recherche ainsi que la considération de cette chaîne virtuelle nous ont permis de concevoir un planificateur de tâches de manipulation capable de générer automatiquement, à l'intérieur de domaines continus, les prises ainsi que les placements particuliers permettant la résolution du problème. L'analyse des performances de ce planificateur montre clairement que plus les recherches sont guidées dans la sous-variété définie par les contraintes de manipulation ($\mathcal{CG} \cap \mathcal{CP}$), plus le temps de résolution du problème est court. D'autre part, notre approche a montré son efficacité sur des problèmes complexes de manipulation, mais la résolution de certains problèmes rencontrés pourraient encore l'accroître. Les résultats obtenus sur les performances de notre planificateur ouvrent des perspectives quant à son amélioration et à son extension.

Améliorations :
Ce sont les fonctions de bases des techniques de planification de mouvement, utilisées par notre planificateur, qui peuvent être améliorées : tests de collision sur des ensembles de données complexes (plusieurs milliers de polyèdres), élaboration des nouvelles méthodes locales plus performantes du point de vue calculatoire, ... Le développement d'une bibliothèque de modèles géométriques inverses (pour une grande variété de systèmes mécaniques) permet également d'aborder de nouveaux problèmes de manipulation.

Sur le plan conceptuel, plusieurs améliorations possibles peuvent être apportées à notre approche, en particulier sur l'alternance entre les deux étapes de construction du graphe de manipulation. Nous avons montré à travers l'analyse de la sensibilité de notre planificateur que ces performances dépendent fortement de cette alternance. La taille du graphe de manipulation ne suffit pas seule à décider du choix de la stratégie d'expansion. En effet, ce choix dépend de la nature du problème de manipulation posé.

Extensions :

Notre planificateur est aujourd'hui limité à un seul objet déplaçable manipulé par un seul robot. Les extensions futures de notre travail porteront, à notre sens, sur des planifications de tâches de manipulation pour des problèmes comportant plusieurs objets déplaçables et plusieurs robots. Une première étape dans ces investigations consistera à étudier le cas de plusieurs objets déplaçables manipulés par un seul robot. Ce cas exige en premier lieu une étude des conditions sous lesquelles la propriété de réduction peut être étendue à de telles situations.

Par ailleurs, des travaux portant sur l'extension de notre approche au problème de manipulation de plusieurs robots et plusieurs objets sont en cours d'étude [Gravot 02]. Ils consistent à développer une approche alternative combinant un niveau symbolique de planification de tâches avec notre planificateur pour résoudre des problèmes de manipulation ayant un niveau de complexité plus élevé et en présence de multiples objets et robots. Le planificateur de tâches calcule dans un premier temps un plan dont la transition entre deux états est validée par notre planificateur en montrant l'existence d'un chemin de manipulation faisable. En cas d'échec, le plan initial est modifié et le processus est réitéré.

FIG. 5.18 – Exemple d'une simulation d'interactions entre homme-humanoïde (tiré de [Chatila 02]).

Ces extensions ouvrent des champs d'applications multiples tels que la manipulation coordonnée, l'animation graphique et en particulier la simulation via la manipulation des interactions homme-machine. Un exemple de ces interactions est présenté dans la figure 5.18 (tirée de [Chatila 02]).

Références bibliographiques

[Ahuactzin 92] J.M. Ahuactzin, E.-G. Talbi, P. Bessière & E. Mazer. *Using Genetic Algorithms for Robot Motion Planning.* In 10*th* Europ. Conf. Artific. Intelligence, pages 671–675, London, England, 1992.

[Ahuactzin 94] J.-M. Ahuactzin. *Le Fil d'Araine : Une Méthode de Planification Générale. Application à la Planification Automatique de Trajectoires.* PhD thesis, L'Institut National Polythechnique de Grenoble, Septembre 1994.

[Ahuactzin 95] J.M. Ahuactzin & K. Gupta. *Using Manhattan Paths for Manipulation Planning.* Third IASTED International Conference on Robotics and Manufacturing, Cancùn, México, 1995.

[Ahuactzin 98] J.-M. Ahuactzin, K. Gupta & E. Mazer. *Manipulation Task Planning for Redundant Robots : A Practical Approach.* In K. Gubta & A.P. Del Pobil, editeurs, Practical Motion Planning in Robotics : Current Approches and Future Directions, pages 79–113. J. Wiley, 1998.

[Alami 89] R. Alami, T. Siméon & J-P. Laumond. *A Geometrical Approach to Planning Manipulation Tasks. The case of discrete placements and grasps.* In Robotics Research : The Fifth International Symposium,Tokyo, August 1989.

[Alami 95] R. Alami, J.-P. Laumond & T. Siméon. *Two Manipulation Planning Algorithms.* In Algorithmic Foundations of Robotics (WAFR'94), K. Goldberg et al. (Eds.), pages 109–125. A K Peters, Wellesley (MA), 1995.

[Amato 98] N.M. Amato, O. Bayazit, L. Dale, C. Jones & D. Vallejo. *OBPRM : an Obstacle-Based PRM for 3D Workspaces.* In Robotics : The Algorithmic Perspective (WAFR'98), P.K. Agarwal et al. (Eds.), pages 155–168. AK Peters, 1998.

[Amato 02] N.M. Amato, K.A. Dill & G. Song. *Using Motion Planning to Map Protein Folding Landscapes and Analyze Folding Kinetics of Known Native Struc-*

tures. Proceedings the 6th International Conference on Computational Molecular Biology (RECOMB), pages 2–11, April 2002.

[Anshelevich 00] E. Anshelevich, S. Owens, F. Lamiraux & L. Kavraki. *Deformable Volumes in Path Planning Applications.* IEEE International Conference on Robotics and Automation, San Francisco (USA), 2000.

[Apaydin 02] M.S. Apaydin, D.L. Brutlag, C. Guestrin, D. Hsu. & J.C. Latombe. *Stochastic Roadmap Simulation : An Efficient Representation and Algorithm for Analyzing Molecular Motion.* Proc. RECOMB'02, Washington D.C., pages 12–21, 2002.

[Barraquand 91] J. Barraquand & J.C. Latombe. *Robot Motion Planning : a distributed representation approach.* International Journal of Robotics Research, 10, (6), pages 628–649, 1991.

[Barraquand 94] J. Barraquand & P. Ferbach. *A Penalty Function Method for Constrained Motion Planning.* In IEEE Int. Conf. on Robotics and Automation, pages 1341–1348, 1994.

[Barraquand 96] J. Barraquand, L. Kavraki, J.-C. Latombe & T.-Y. Li. *A Random Sampling Scheme for Path Planning.* In G. Giralt & G. Hizinger, editeurs, Proc. of the 7th International Symposium on Robotics Research, Springer, pages 249–264, New York, 1996.

[Bessière 93] P. Bessière, J.M. Ahuactzin, E.-G. Talbi & E. Mazer. *The "Ariadne's Clew" Algorithm : Global Planning with Local Methods.* IEEE Internationnal Conference on Intelligent Robots and Systems (IROS), vol. 2, pages 1373–1380, 1993.

[Bicchi 01] A. Bicchi & V. Kumar. *Robotic Grasping and Manipulation.* In Siciliano, Bicchi, Nicosia & Valigi, editeurs, Ramsete : Articulated and Mobile Robots for Services and Technology, volume 270 of *Lecture Notes in Control and Information Series (LNCIS).* Springer Verlag, 2001.

[Bohlin 00] R. Bohlin & L.E. Kavraki. *Path Planning using Lazy PRM.* Proc. IEEE International Conference on Robotics and Automation, pages 521–528, 2000.

[Boneschanscher 88] N. Boneschanscher, H. Van der Drift, S.J. Buckley & R.H. Taylor. *Subassembly stability.* In Proceedings of the Seventh National Conference on Artificial Intelligence, pages 780–785, 1988.

[Boor 99] V. Boor, M.H. Overmars & A.F. van der Stappen. *The Gaussian Sampling Strategy for Probabibilistic Roadmap Planners.* Proc. IEEE International Conference on Robotics and Automation, pages 1018–1023, 1999.

[Branicky 01] M.S. Branicky, S.M. LaValle, K. Olson & L. Yang. *Quasi-randomized path planning.* IEEE International Conference on Robotics and Automation, pages 1481–1487, 2001.

[Canny 87] J.F. Canny. *The Complexity of Robot Motion Planning.* PhD thesis, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Mai 1987.

[Chatila 02] R. Chatila, R. Alami, J. Pettré & L. Jaillet. *Safe, Reliable and Friendly Interaction Between Humans and Humanoids.* The Third International Workshop on Humanoid and Humain Friendly Robotics, December 2002. Japan.

[Cohen 95] J. Cohen, M. Lin, D. Manocha & K. Ponamgi. *I-COLLIDE : An Interactive and Exact Collision Detection System for Large-Scale Environments.* Proc. of ACM Interactive 3D Graphics Conference, pages 189–196, 1995.

[Cortés 02] J. Cortés, T. Siméon & J.-P. Laumond. *A Random Loop Generator for Planning the Motions of Closed Kinematic Chains using PRM Methods.* IEEE International Conference on Robotics and Automation, May 2002.

[Cortés 03] J. Cortés & T. Siméon. *Probabilistic Motion Planning for Parallel Mechanisms.* Soumis à IEEE International Conference on Robotics and Automation, 2003.

[Dubins 57] L.E. Dubins. *On Curves of Minimal Length with a Constraint on Qvergge Curvature and with Prescribed Initial and terminal positions and tangents.* American Journal of Mathematics, no. 79, pages 497–516, 1957.

[Ferbach 94] P. Ferbach & J. Barraquand. *Path Planning through Variational Dynamic Programming.* Rapport technique 33, Paris Research Laboratory, Digital Equipement Coorporation, 1994.

[Ferbach 96] P. Ferbach. *Contribution à la Planification de Trajectoires.* PhD thesis, Ecole Polytechnique en collaboration avec Electricité De France (EDF), France, 1996.

[Geem 01] C. Van Geem & T. Siméon. *KCD : a collision detector for path planning in factory models.* Rapport technique, LAAS-CNRS N°01073, Mars 2001.

[Gravot 02] F. Gravot & R. Alami. *A method for handling multiple roadmaps and its use for complex manipulation planning.* IEEE International Conference on Intelligent Robots and Systems, October 2002. Switzerland.

[Gupta 98] K. Gupta & A.P. Del Pobil (eds). Practical motion planning in robotics : Current approches and future directions. J. Wiley and Sons, 1998.

[Han 00] L. Han & N.M. Amato. *A Kinematics-based Probabilistic Roadmap Method for Closed Chain Systems.* In In Algorithmic and Computational Robotics – New Directions (WAFR'00), B.R. Donald et al. (Eds.), pages 233–246. AK Peters, 2000.

[Holleman 98] C. Holleman, L. Kavraki & J. Warren. *Planning Paths for a Flexible Surface Patch.* IEEE International Conference on Robotics and Automation, pages 21–26, 1998.

[Hsu 97] D. Hsu, J.-C. Latombe & R. Motwani. *Path Planning in Expansive Configuration Spaces.* IEEE International Conference on Robotics and Automation, vol. 3, pages 2719–2726, 1997.

[Hsu 98] D. Hsu, L.Kavraki, J.-C. Latombe, R. Motwani & S. Sorkin. *On Finding Narrow Passage with Probabilistic Roadmap Planner.* In Robotics : The Algorithmic Perspective (WAFR'98), P.K. Agarwal et al. (Eds). AK Peters, 1998.

[Hudson 97] T.C. Hudson, M. Lin, J. Cohen, S. Gottschalk & D. Manocha. *V-COLLIDE : Accelerated Collision Detection for VRML.* Proc. 2 nd Annu. Sympos. on the Virtual Reality Modeling Language, pages 119–125, 1997.

[Kavraki 95a] L. Kavraki. *Random Networks in Configuration Space for Fast Path Planning.* PhD thesis, Standford University, 1995.

[Kavraki 95b] L. Kavraki, J.-C. Latombe, R. Motwani & P. Raghavan. *Randomized Query Processing in Robot Motion Planning.* ACM Symposium on Theory of Computing, pages 353–362, 1995.

[Kavraki 96] L. Kavraki, P. Švestka, J.-C. Latombe & M. Overmars. *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces.* IEEE Transactions on Robotics and Automation, vol. 12(4), pages 566–580, 1996.

[Kavraki 98] L. Kavraki, J.-C. Latombe, R. Motwani & P. Raghavan. *Randomized Query*

Processing in Robot Motion Planning. Journal of Computer and System Sciences, vol. 57(1), pages 50–60, 1998.

[Khatib 86] O. Khatib. *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots.* International Journal of Robotics Research, vol. 5(1), pages 90–98, 1986.

[Koga 94a] Y. Koga, K. Kondo, J.J. Kuffner & J.-C. Latombe. *Planning Motions with Intentions.* Proc. SIGGRAPH'94, pages 395–408, 1994.

[Koga 94b] Y. Koga & J.-C. Latombe. *On Multi-Arm Manipulation Planning.* IEEE International Conference on Robotics and Automation, vol. 2, pages 945–952, 1994.

[Koga 95] Y. Koga. *On Computing Multi-Arm Manipulation Trajectories.* PhD thesis, Mechanical Engineering Dept, Stanford University, February 1995.

[Kriegman 97] D. Kriegman. *Let Them Fall Where They May : Capture Regions of Curved Objects and Polyhedra.* International Journal of Robotics Research, vol. 16 :4, pages 448–472, August 1997.

[Kuffner 99] J.J. Kuffner. *Autonomous Agents for Real-time Animation.* PhD thesis, Stanford University, Stanford, CA, December 1999.

[Kuffner 00a] J. Kuffner, S. Kagami, M. Inaba & H. Inoue. *Graphical Simulation and High-Level Control of Humanoid Robots.* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'00), vol. 3, pages 1943–1948, Takamatsu, Japan, November 2000.

[Kuffner 00b] J.J. Kuffner, M. Inaba & H. Inoue. *Automating Object Manipulation Tasks for Humanoid Robots.* In Proc. 2000 JSME Conf. on Robotics and Mechatronics (ROBOMEC'00), vol. 2P2, pages 79–103, May 2000. (short paper).

[Kuffner 00c] J.J. Kuffner & J.-C. Latombe. *Interactive Manipulation Planning for Animated Characters.* In Pacific Graphics'00, October 2000. (short paper).

[Kuffner 00d] J.J. Kuffner & S.M. LaValle. *RRT-Connect : An Efficient Approach to Single-Query Path Planning.* IEEE International Conference on Robotics and Automation (ICRA'00), San Francisco, 2000.

[Lamiraux 00] F. Lamiraux & J.-P. Laumond. *Flatness and Small-Time Controllability of Multi-Body Mobile Robots : Application to Motion Planning.* IEEE Transac-

tion on Automatic Control, vol. 45 (10), 2000.

[Lamiraux 01] F. Lamiraux & L.E. Kavraki. *Planning Paths for Elastic objects under manipulation constraints.* International Journal of Robotics Research 20, pages 188–208, 2001.

[Latombe 91a] J.-C. Latombe. *A Fast Path Planner for a Car-like Indoor Mobile Robot.* AAAI, Ninth National Conference on Artificial Intelligence, pages 659–665, 1991.

[Latombe 91b] J.-C. Latombe. Robot motion planning. Kluwer Academic Publishers, Boston, MA, 1991.

[Latombe 99] J.C. Latombe. *Motion Planning : A Journey of Robots, Molecules, Digital Actors, and Other Artifacts.* International Journal of Robotics Research, Special Issue on Robotics at the Millennium – Part I, vol. 18(11), pages 1119–1128, November 1999.

[Laugier 88] C. Laugier. *Planning Robot Motions in The SHARP system.* In CAD Based Programming for Sensory Robots, vol. 50, pages 151–187, 1988.

[Laumond 89] J.-P. Laumond & R. Alami. *A Geometric Approach to Planning Manipulation Tasks : the case of a circular robot and a movable circular object admist polygonal obstacles.* Rapport technique Rep. No. 88314, LAAS-CNRS, 1989.

[Laumond 90] J.-P. Laumond, M. Taix & P. Jacobs. *A Motion Planner for Car-like Robots Based on a Mixed Global/Local Approach.* IEEE International Workshop on Intelligent Robots and Systems' 90 (IROS'90), Tsuchira (Japon), pages 765–773, 3-6 Juillet 1990.

[Laumond 94] J.-P. Laumond, P.E. Jacobs, M. Taix & R.M. Murray. *A Motion Planner for Nonholonomic Mobile Robots.* IEEE International Conference on Robotics and Automation, vol. 10(5), pages 577–593, 1994.

[Laumond 98] J.-P. Laumond, editeur. Robot motion planning, lectures notes in control and information sciences, volume 229. Springer, 1998.

[Laumond 00] J.-P. Laumond & T. Siméon. *Notes on Visibility Roadmaps and Path Planning.* In In Algorithmic and Computational Robotics – New Directions (WAFR'00), B.R. Donald et al. (Eds.), pages 317–328. AK Peters, 2000.

[LaValle 98] S.M. LaValle. *Rapidly-Exploring Random Trees : A New Tool for Path Plan-*

ning. Technical Report No. 98-11, Dept. of Computer Science, Iowa State University, 1998.

[LaValle 99a] S.M. LaValle & J.J. Kuffner. *Randomized Kinodynamic Planning.* Proc. IEEE International Conference on Robotics and Automation, 1999.

[LaValle 99b] S.M. LaValle, J. Yakey & L.E. Kavraki. *A Probabilistic Roadmap Approach for Systems with Closed Kinematic Chains.* Proc. IEEE International Conference on Robotics and Automation, pages 1671–1676, 1999.

[LaValle 00a] S.M. LaValle, P.W. Finn, L.E. Kavraki & J.C. Latombe. *A Randomized Kinematics-Based Approach to Pharmacophore-Constrained Conformational Search and Database Screening.* J. of Computational Chemistry, vol. 21(9), pages 731–747, July 2000.

[LaValle 00b] S.M. LaValle & J.J. Kuffner. *Rapidly-Exploring Random Trees : Progress and Prospects.* In In Algorithmic and Computational Robotics – New Directions (WAFR'00), B.R. Donald et al. (Eds.), pages 293–308. AK Peters, 2000.

[LaValle 01] S.M. LaValle & J.J. Kuffner. *Randomized Kinodynamic Planning.* International Journal of Robotics Research, vol. 20, no. 5, pages 378–400, 2001.

[LaValle 02] S.M. LaValle & M.S. Branicky. *On the relationship between classical grid search and probabilistic roadmaps.* Workshop on the Algorithmic Foundations of Robotics, December 2002.

[Lozano-Pérez 83] T. Lozano-Pérez. *Spatial Planning : A Configuration Space Approach.* IEEE Transaction on computer, vol. 32(2), pages 108–120, 1983.

[Lozano-Pérez 87] T. Lozano-Pérez, J.L. Jones, E. Mazer, P.A. O'Donnell, W.E.L Grimson, P. Tournassoud & A. Lanusse. *HANDEY : A Robot System that Recognizes, Plans, and Manipulates.* Proceedings of 1987 IEEE International Conference on Robotics and Automation, pages 843–849, March 1987.

[Matousek 99] J. Matousek. *Geometric Discrepancy : An Illustrated Guide.* Springer, Berlin, 1999.

[Mazer 87] E. Mazer. *Handey : Un Modèle de Planificateur pour la Programmation Automatique des Robots.* PhD thesis, Doctorat d'Etat, Institut National Polythechnique de Grenoble, Décembre 1987.

[Mazer 98] E. Mazer, J.M. Ahuactzin & P. Bessière. *The Ariadne's Clew Algorithm.*

Journal of Artificial Intelligence Research, pages 295–316, 1998.

[Mazon 90] I. Mazon, R. Alami & P. Violero. *Automatic Planning of Pick and Place Operations in Presence of Uncertainties.* IEEE International Workshop on Intelligent Robots and Systems (IROS'90), pages 33–40, Juillet 1990.

[Mishra 87] B. Mishra, J.T. Schwartz & M. Sharir. *On the Existence and Synthesis of Multifinger Positive Grips.* Special Issue : Robotics, Algorithmica, pages 541–558, 1987.

[Motwani 95] R. Motwani & P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[Nielsen 97] J. Nielsen & B. Roth. *Formulation and Solution of the Direct and Inverse Kinematics Problem for Mechanism and Mechatronic Systems.* In Proceedings of the NATO Advanced Study Institute on Computational Methods in Mechanisms, vol. 1, pages 233–252, 1997. Bulgaria.

[Nielsen 00] C.L. Nielsen & L.E. Kavraki. *A Two Level Fuzzy PRM for Manipulation Planning.* IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000.

[Nilsson 69] N.J. Nilsson. *A Mobile Automation : An Application of Artificial Intelligence Techniques.* Proceedings of the 1st International Joint Conference on Artificial Intelligence, pages 509–520, 1969.

[Nissoux 99a] C. Nissoux. *Visibilité et Méthodes Probabilistes pour la Planification de Mouvement en Robotique.* PhD thesis, LAAS-CNRS, Universté Paul Sabatier de Toulouse, Décembre 1999.

[Nissoux 99b] C. Nissoux, T. Siméon & J.-P. Laumond. *Visibility Based Probabilistic Roadmaps.* IEEE International Conference on Intelligent Robots and Systems, pages 1316–1321, 1999.

[O'Dùnlaing 82] C. O'Dùnlaing & C.K. Yap. *A Retraction Method for Planning The Motion of a Disc.* Journal of Algorithms, vol. 6, pages 104–111, 1982.

[P.Cheng 02] P.Cheng & S.M. LaValle. *Resolution Complete Rapidly-Exploring Random Trees.* IEEE International Conference on Robotics and Automation, 2002.

[Pertin-Troccaz 89] J. Pertin-Troccaz. *Grasping : A State of The Art.* The Robotics Review 1. Edited by O. Khatib, J.J. Craig and T. Lozano-Pérez, MIT Press, 1989.

[Pettré 02] J. Pettré, T. Siméon & J.-P. Laumond. *Planning Human Walk in Virtual Environments.* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02), October 2002.

[Reeds 90] J.A Reeds & L.A. Shepp. *Optimal Paths for a Car that goes Both Forwards and Backwards.* Pacific Journal Mathematics, vol. 145, no. 2, pages 367–393, 1990.

[Reif 79] J.H. Reif. *Complexity of the Mover's Problem and Generalisations.* In Proc. 20th IEEE Symposium on Foundations of Computer Science (FOCS), pages 421–427, 1979.

[Sahbani 01a] A. Sahbani. *Approches Probabilistes pour la Planification de Tâches de Manipulation.* 14 Jounées des Jeunes Chercheurs en Robotique, May 2001. Evry.

[Sahbani 01b] A. Sahbani. *Planification de Tâches de Manipulation par des Méthodes Probabilistes.* 2ème Congrès des Doctorants. Ecole Doctorale Systèmes, 2001. Toulouse.

[Sahbani 02] A. Sahbani, J. Cortès & T. Siméon. *A Probabilistic Algorithm for Manipulation Planning under Continuous Grasps and Placements.* IEEE International Conference on Intelligent Robots and Systems, pages 1560–1565, October 2002. Switzerland.

[Schmitzberger 02] E. Schmitzberger, J.-L. Bouchet, M. Dufaut, D. Wolf & R. Husson. *Capture of Homotopy Classes with Probabilistic Road map.* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02), October 2002.

[Schwartz 83] J.T. Schwartz & M. Sharir. *On The Piano Mover's Problem : I. The Case of a Two-Dimensional Rigid Polygonal Body Moving Admist Polygonal Barriers.* Communications on Pure and Applied Mathematics, vol. 36, pages 345–398, 1983.

[Sekhavat 98] S. Sekhavat, P. Švestka, J.-P. Laumond & M.H Overmars. *Multilevel Path Planning for Nonholonomic Robots using semiholonomic subsystems.* International Journal of Robotics Research, 17, pages 840–857, 1998.

[Sent 01] D. Sent & M.H. Overmars. *Motion Planning in an Environment with danger-zones.* IEEE International Conference on Robotics and Automation, pages 1488–1493, 2001.

[Siméon 89] T. Siméon. *Génération Automatique de Trajectoires sans Collision et Plani-*

fication de Tâches de Manipulation en Robotique. PhD thesis, LAAS-CNRS, Universté Paul Sabatier de Toulouse, 1989.

[Siméon 00] T. Siméon, J.-P. Laumond & C. Nissoux. *Visibility Based Probabilistic Roadmaps for Motion Planning.* Advanced Robotics Journal, vol. 14(6), 2000.

[Siméon 01a] T. Siméon, J.-P. Laumond, C. Van Geem & J. Cortés. *Computer Aided Motion : Move3D within MOLOG.* IEEE International Conference on Robotics and Automation, 2001.

[Siméon 01b] T. Siméon, J.-P. Laumond & F. Lamiraux. *Move3D : a Generic Platform for Path Planning.* 4th International Symposium on Assembly and Task Planning, 2001.

[Siméon 02a] T. Siméon, J. Cortés, A. Sahbani & J.-P. Laumond. *A General Manipulation Task Planner.* Fifth International Workshop on Algorithmic Foundations of Robotics (WAFR'02), pages 293–308, December 2002. Nice, France.

[Siméon 02b] T. Siméon, J. Cortés, A. Sahbani & J.-P. Laumond. *A Manipulation Planner for Pick and Place Operations under Continuous Grasps and Placements.* IEEE International Conference on Robotics and Automation, pages 2022–2027, May 2002. Washington D.C.

[Teodoro 01] M.L Teodoro, G.N. Phillips Jr & L.E. Kavraki. *Molecular Docking : A Problem With Thousands Of Degrees Of Freedom.* IEEE International Conference on Robotics and Automation, pages 960–966, May 2001.

[Violero 91] P. Violero. *SPARA : Un Système de Programmation Automatique de Tâche de Manipulation Robotisée.* PhD thesis, LAAS-CNRS, Universté Paul Sabatier de Toulouse, Novembre 1991.

[Švestka 96] P. Švestka. *On Probabilistic Completeness and Expected Complexity of Probabilistic Path Planning.* Rapport technique, Departement of Computer Science, Utrecht University, 1996.

[Švestka 97a] P. Švestka. *Robot Motion Planning using Probabilistic Roadmap.* PhD thesis, Utrecht University, 1997.

[Švestka 97b] P. Švestka & M.H. Overmars. *Motion Planning for Car-like Robots, a Probabilistic learning approach.* International Journal of Robotics Research 16, pages 119–143, 1997.

[Wilfong 88] G. Wilfong. *Motion Planning in The Presence of Movable Obstacles.* Proceedings of the 4th ACM Symposium on Computational Geometry, pages 279–288, 1988.

[Wilmarth 99a] S.A. Wilmarth, N.M. Amato & P.F. Stiller. *MAPRM : A Probabilistic Roadmap planner with sampling on the Medial Axes of the Free Space.* IEEE International Conference on Robotics and Automation, pages 1024–1031, 1999.

[Wilmarth 99b] S.A. Wilmarth, N.M. Amato & P.F. Stiller. *Motion Planning for a Rigid Body Using Random Networks on the Medial Axes of the Free Space.* ACM Symposium on Computational Geometry, pages 173–180, 1999.

[Yakey 99] J. Yakey. *Randomized Path Planning for Linkages with Closed Kinematic Chains.* PhD thesis, Computer Science Departement, Iowa State University, 1999.

Planification de tâches de manipulation en robotique par des approches probabilistes

Résumé :

Les travaux présentés dans cette thèse portent sur la planification de tâches de manipulation pour des systèmes robotiques. Dans ce contexte, la planification ne se limite pas au calcul automatique des chemins sans collision pour un robot mais intègre également le séquencement des différentes sous-tâches de prise et de pose d'objets. De part leur complexité, ces problèmes ont longtemps résisté aux méthodes déterministes initialement proposées dans la littérature. Le développement récent des techniques probabilistes permet aujourd'hui de les revisiter. Après avoir présenté un état de l'art détaillé du domaine, le deuxième chapitre reprend et étend la formulation géométrique adoptée. Nous détaillons les notions d'espace de prise et de pose sur lesquelles reposent la structuration de l'espace de recherche. Nous distinguons ensuite deux instances du problème de manipulation : le cas discret et le cas continu d'ensemble de prises et de poses. Le troisième chapitre présente une extension des méthodes probabilistes, appelée approche par graphe dynamique, permettant de traiter dans le cas discret une large classe de problème de manipulation. Cette technique est également utilisée dans l'approche générique, détaillée dans le chapitre 4, qui permet la résolution de problèmes de manipulation sous contraintes continues. Cette approche consiste à capturer la connectivité d'une sous-variété particulière de l'espace des configurations composite dans une structure de graphe, appelé "graphe de manipulation". Le dernier chapitre est dédié à la présentation de la plate-forme logicielle Move3D dans laquelle notre planificateur a été implanté. Une analyse des performances de l'algorithme est également présentée à travers différents exemples de problèmes de manipulation résolus par le planificateur.

Mots-clés : Planification de tâches de manipulation, méthodes probabilistes, chaînes fermées.

Manipulation planning for robotic systems using probabilistic approches

Abstract :

The work presented in this thesis addresses the manipulation planning problem for robotic systems. Manipulation planning is not limited to the automatic computation of the collision-free paths for a robot ; it also integrates the sequencing of the various sub-tasks of grasp and placement of objects. Because of their complexity, these problems have resisted for a long time to the deterministic methods initially proposed in the literature. The recent development of the probabilistic techniques has led us to revisit such problems. After having presented a detailed state of the art, the second chapter shows and extends the adopted geometrical formulation. We detail the concept of grasp and placement spaces allowing the structuring of the research space. We distinguish, following this structuring, two cases of the manipulation problem : the discrete and the continuous sets of grasp and placement. The third chapter presents an extension of the probabilistic methods, called approach by dynamic graph, capable of dealing, in the discrete case, with a large class of manipulation problems. This technique represents part of a more general approach which is detailed in chapter 4. The latter allows the resolution of manipulation problems under continuous constraints. This approach consists in capturing the connectivity of a particular sub-manifold of the composite configuration space in a roadmap structure, called "manipulation graph". The last chapter is dedicated to the presentation of the software platform Move3D in which our planner was implemented. An analysis of the performances of the algorithm is also presented through various examples of manipulation problems solved by the planner.

Keywords : Manipulation planning, probabilistic methods, closed chain systems.