

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Systèmes Informatiques

Présentée et soutenue par :
Mokhtar Gharbi

le : lundi 8 novembre 2010

Titre :

Planification de mouvements et manipulation d'objets par des torses humanoïdes

---

JURY

Rachid Alami - Directeur de Recherche CNRS, Président du jury
Lluis Ros - Chargé de Recherche, Membre
Juan Cortés - Chargé de Recherche CNRS, Membre
Thierry Siméon - Directeur de Recherche CNRS, Membre

Ecole doctorale :
Systèmes (EDSYS)

Unité de recherche :
UPR 8001 - LAAS-CNRS

Directeur(s) de Thèse :

Thierry Siméon - Directeur de Recherche CNRS

Rapporteurs :
Veronique Perdereau - Professeur des Universités
Philippe Fraisse - Professeur des Universités



THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Systèmes Informatiques

Présentée et soutenue par :
Mokhtar Gharbi

le : lundi 8 novembre 2010

Titre :

Planification de mouvements et manipulation d'objets par des torses humanoïdes

---

JURY

Rachid Alami - Directeur de Recherche CNRS, Président du jury
Lluis Ros - Chargé de Recherche, Membre
Juan Cortés - Chargé de Recherche CNRS, Membre
Thierry Siméon - Directeur de Recherche CNRS, Membre

Ecole doctorale :
Systèmes (EDSYS)

Unité de recherche :
UPR 8001 - LAAS-CNRS

Directeur(s) de Thèse :

Thierry Siméon - Directeur de Recherche CNRS

Rapporteurs :
Veronique Perdereau - Professeur des Universités
Philippe Fraisse - Professeur des Universités

À la mémoire de mon grand-père

Remerciements

Je tiens à remercier Raja Chatila, directeur du LAAS-CNRS, pour m'avoir accueilli dans ce laboratoire. Je remercie également Rachid Alami, responsable du groupe de Robotique et InteractionS, de m'avoir permis de travailler dans ce groupe et d'avoir accepté de présider mon jury de thèse.

Je tiens à exprimer ma sincère reconnaissance à mes deux rapporteurs Philippe Fraisse de l'Université Montpellier 2 et Véronique Perdereau de l'Université Pierre et Marie Curie, pour avoir accepté d'être rapporteurs de mes travaux de thèse ainsi que pour leur lecture minutieuse du manuscrit. Merci à Lluis Ros, pour avoir accepté d'être membre de mon jury de thèse.

Je voudrais également adresser de sincères remerciements à mes deux directeurs de thèse Thierry Siméon et Juan Cortés, qui ont su me guider durant ces 4 années de travail ainsi que pour leur rigueur scientifique. Merci pour votre patience et vos conseils avisés.

Un grand merci va également à tous les permanents du pôle de Robotique et Intelligence Artificielle, spécialement Daniel Sidobre, Matthieu Herrb, Marc Renaud et Natacha Ouwanssi, qui m'ont soutenu et m'ont aidé à progresser. Je tiens également à remercier tous mes collègues du LAAS particulièrement Xavier qui m'a supporté pendant 3 ans et avec qui on a passé des moments mémorables. Akin, Romain, Jim, Jean-Philippe, Oussama, Mathieu, Alireza, Minh, Sébastien, Assia, Samir, Yi, Manish, Raquel, Luis, Thierry, Julien, Benjamin, Mathias ... en somme tous les doctorants et post doctorants, merci pour votre aide et pour les bons moments passés ensemble. J'aimerai également remercier Manu, Eric, Émilie, Steph, Matthieu, Laetitia, Hervé, Nicolas, Clémentine, Sabrina, Amine, Cédric, Jeremy, David, Stelly, Fabien, Aurélie.

Finalement je remercie Mélanie pour sa joie et sa bonne humeur ainsi que pour sa patience sans fin, sa douceur et son soutien dans les moments difficiles. Je dédie cette thèse à toute ma famille qui m'a soutenu pendant toutes ces années et qui a une place incommensurable dans mon coeur. Merci Maman, Papa, Mamoun je vous aime tous.

Table des matières

1 Introduction **1**

1.1 Contributions de la thèse . . . 2
1.2 Organisation du manuscrit . . . 2
1.3 Publications associées à cette thèse . . . 2
1.4 Contexte de la thèse . . . 3

2 État de l'art **5**

2.1 Planification de mouvement . . . 5
2.1.1 Notions de base . . . 6
2.1.2 Formulation du problème . . . 7
2.1.3 Differentes approches . . . 7
2.2 Algorithmes probabilistes . . . 8
2.2.1 Les réseaux probabilistes . . . 9
2.2.2 Les méthodes de diffusion . . . 11
2.3 Résolution de tâches complexes . . . 13
2.3.1 Planification multi-robots . . . 13
2.3.2 Planification de chaînes cinématiques fermées . . . 14
2.3.3 Planification de tâches de manipulation . . . 17

3 Fusion de réseaux probabilistes **21**

3.1 Décomposition du système . . . 22
3.2 Vue d'ensemble de l'approche . . . 23
3.3 Construction des réseaux indépendants . . . 25
3.4 Fusion des réseaux indépendants . . . 28
3.4.1 Construction des Nœuds de Super Graphe . . . 29
3.4.2 Connexion des Nœuds de Super Graphe . . . 31
3.5 Résolution des requêtes de planification . . . 33
3.6 Résultats . . . 34
3.7 Conclusion . . . 38

4 Planificateur pour la manipulation multi-bras **39**

4.1 Formulation et paramétrisation . . . 39

4.1.1 Structure de l'espace des configurations 39
4.1.2 Cinématique inverse 40
4.1.3 Configurations singulières 41
4.1.4 Décomposition du système 42
4.2 Approche basée sur les réseaux probabilistes 43
4.2.1 Construction des couches de réseaux probabilistes 44
4.2.2 Connexion des différentes couches 46
4.2.3 Résolution des requêtes de planification 48
4.3 Approche basée sur les méthodes de diffusion 48
4.3.1 Construction des couches 48
4.3.2 Extension de l'arbre vers des configurations singulières 49
4.4 Lissage des chemins solution 50
4.5 Résultats 51
4.5.1 Implémentation 51
4.5.2 Exemple académique 52
4.5.3 Applications sur le robot Justin du DLR 53
4.6 Conclusion 56

5 Tâche de prise et pose avec ressaisie **59**
5.1 Vue d'ensemble de l'approche 60
5.2 Formulation du problème 61
5.3 Planification de prise 62
5.3.1 Planification de simples prises 62
5.3.2 Planification de double prise 65
5.4 Planification de mouvement 67
5.4.1 Précalcul du réseau probabiliste 67
5.4.2 Configurations de saisie et de pose 68
5.4.3 Configuration d'échange 68
5.4.4 Résolution des requêtes de planification 71
5.4.5 Validation des chemins solution pour l'objet 71
5.5 Résultats 72
5.6 Conclusion 76

6 Expérimentations **77**
6.1 Le planificateur de mouvement Move3D 77
6.2 Planificateur intégré sur le robot Justin 78
6.2.1 Vue globale du planificateur 79
6.2.2 Générateur des configurations de saisie et pose 80
6.2.3 Planification de mouvement de la plate-forme mobile 82
6.2.4 Planification de mouvement du manipulateur 82
6.2.5 Mouvements d'approche de saisie et de pose 83

6.2.6 Expérimentations sur le robot Justin Mobile 85
6.2.7 Conclusion 87
6.3 Planificateur intégré sur le robot Jido 88
6.3.1 Prise en compte de l'environnement changeant 88
6.3.2 Intégration sur le robot Jido 89
6.3.3 Conclusion 92

7 Conclusion et perspectives **95**
7.1 Conclusion 95
7.2 Améliorations et extensions 96

Références **99**

1

Introduction

Le terme *robot* a été introduit en 1920 par *Karel Čapek* dans une de ses pièces de théâtre en faisant référence à des humains artificiels. Ce terme définit un système mécatronique (où se combinent mécanique, électronique et informatique) capable d'effectuer une ou plusieurs tâches, dans un environnement donné, de manière autonome. La *robotique* est la discipline s'intéressant au fonctionnement et à l'utilisation des robots pour effectuer des tâches de manière automatique. À ses débuts, les connaissances et les moyens techniques ne permettaient d'envisager que des tâches extrêmement simples, dans des environnement maîtrisés et connus auxquels le robot est dédié. Cependant de nos jours, l'objectif de la robotique est beaucoup plus ambitieux. Il consiste à construire des robots capables d'évoluer dans des environnements variés, connus ou non. Les robots doivent également réaliser des tâches de plus en plus complexes, de la manière la plus autonome possible. Les applications vont de l'exploration spatiale à la chirurgie, en passant par l'aide à la personne.

Pour réaliser les tâches dans un monde physique, un système robotique agit sur son environnement par le mouvement. Sa capacité à planifier ses mouvements est donc une composante essentielle de son autonomie.

Le problème de la planification de mouvement en robotique [Lozano-Pérez 83] peut être sommairement défini de la manière 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 tout en évitant les obstacles. La planification de mouvement est un domaine de recherche qui a largement été étudié durant ces dernières décennies [Latombe 91, LaValle 06]. L'objectif de ces travaux est de mettre au point des méthodes algorithmiques performantes permettant le calcul automatique de trajectoires pour des systèmes robotiques complexes. Parmi les différentes

types de méthodes proposées, les algorithmes probabilistes [Kavraki 96, Švestka 97] sont aujourd'hui les plus étudiés et utilisés, grâce à leur efficacité pour traiter des problèmes complexes. Ces méthodes ont permis d'élargir le champ d'application de la planification de mouvement à de nouveaux domaines : conception assistée par ordinateur, bioinformatique structurale, etc.

1.1 Contributions de la thèse

Le travail présenté dans ce manuscrit porte sur le problème de planification de mouvement pour les torses humanoïdes dans le but de réaliser des tâches de manipulation.

Nous proposons, en premier lieu, un planificateur probabiliste capable de traiter efficacement la complexité des robots munis de plusieurs bras. Le système est décomposé en différentes parties élémentaires, puis un réseau probabiliste est créé pour chaque partie. Ces réseaux élémentaires sont ensuite fusionnés avant d'être utilisés pour résoudre les requêtes de planification. Ce planificateur est destiné à générer les mouvements des torses d'humanoïdes pour se saisir des objets.

La seconde contribution est une méthode traitant des problèmes de transport d'un objet par au moins deux manipulateurs indépendants ou par un torse humanoïde. On se trouve donc face à un système comportant des chaînes cinématiques fermées. Nous proposons un traitement explicite des configurations singulières lors de la génération des réseaux probabilistes, permettant ainsi une meilleure maniabilité de l'objet.

Enfin, un algorithme de planification de tâche de prise et pose qui traite de problèmes nécessitant une ressaisie est proposé. Ce planificateur est utilisé lorsque le système robotique a pour tâche de prendre un objet qui n'est accessible que par l'un des bras manipulateurs et de le déposer à un autre endroit, uniquement accessible par un autre bras. Le planificateur détermine la double prise, ainsi que la position d'échange de l'objet.

1.2 Organisation du manuscrit

Après un état de l'art des grands principes et des bases de la planification de mouvement (Chapitre 2), nous détaillerons une méthode de planification probabiliste conçue pour des systèmes multi-bras (Chapitre 3). Nous présenterons ensuite un algorithme de planification de mouvement pour des systèmes possédant des chaînes cinématiques fermées, traitant explicitement le besoin de reconfiguration des manipulateurs (Chapitre 4). Puis, nous décrirons un planificateur de tâche de prise et pose nécessitant une reprise des deux mains (Chapitre 5). Enfin, nous montrerons deux expérimentations réalisées avec des plates-formes robotiques différentes (Chapitre 6).

1.3 Publications associées à cette thèse

Les publications suivantes sont associées à la thèse :

1. Mokhtar Gharbi , Juan Cortés, Thierry Siméon, *Sampling-based planner for dual-arm manipulation*, IEEE/ASME International Conference on Advanced Mechatronics, Xi'an, China, July 2008.
2. Mokhtar Gharbi , Juan Cortés, Thierry Siméon, *Roadmap composition for multi-arm systems path planning*, IEEE/RSJ International Conference on Intelligent Robots and Systems, Saint Louis, USA, October 2009.
3. Jean-Philippe Saut, Mokhtar Gharbi , Juan Cortés, Daniel Sidobre, Thierry Siméon, *Planning Pick and Place tasks with two-hand regrasping*, IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, October 2010.

1.4 Contexte de la thèse

Cette thèse a été réalisée au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS), dans le cadre du projet européen Phriends (Physical Human-Robot Interaction : depENDability and Safety). Le but du projet est de développer des robots pouvant co-exister et coopérer avec les humains, autorisant ainsi une interaction physique homme-robot sûre. Le projet Phriends concerne le développement des composants clés de la prochaine génération de robots, devant respecter les standards stricts de sécurité, tout en améliorant les performances. Ceci pose de nouveaux défis dans la conception du système, incluant la mécanique, le contrôle, les algorithmes de planification et les systèmes de supervision. Le robot Justin du DLR, respectant ces standards, a été utilisé comme démonstrateur du projet posant ainsi le problème de planification de mouvement et de tâches de manipulation pour les torses humanoïdes.

2
État de l'art

Un système robotique ayant une tâche de manipulation à accomplir, doit être apte à se mouvoir et à transporter les objets manipulés dans un environnement contraint. La planification de mouvement est une brique essentielle dans la résolution d'un problème de manipulation.

L'algorithmique du mouvement robotique est un thème de recherche qui a suscité de nombreuses études depuis les travaux pionniers de Lozano [Lozano-Pérez 83]. De nombreuses approches ont été proposées [Latombe 91, Choset 05, LaValle 06], mais peu d'entre elles sont capables de résoudre dans un temps raisonnable les problèmes complexes comme ceux imposés par les systèmes multi-bras. La première partie de ce chapitre présente un bref état de l'art des approches existantes ainsi qu'un rappel de la formulation standard des problèmes de planification de mouvement. Dans la seconde partie, nous détaillerons les approches basées sur des techniques probabilistes de planification de mouvement. Enfin, nous verrons dans la troisième section plusieurs extensions de problèmes liées à la résolution des tâches de manipulation, sur lesquelles reposent certaines contributions de la thèse.

2.1 Planification de mouvement

Dans sa formulation de base, connue sous le nom de "*déménageur de piano*" [Schwartz 83]. Le problème de planification de mouvement s'énonce de la manière suivante : "Existe t-il un chemin libre de toute collision dans l'espace Euclidien (ou espace de travail) permettant de déplacer un ensemble de corps rigides d'une position et une orientation initiales, vers une position et une orientation finales, connaissant exactement l'environnement et la position des obstacles ?". Cette section introduit des notions de base du problème (Section 2.1.1) fondamentales pour sa formulation (Section 2.1.2). Finalement, différentes approches de planification de mouvement

seront présentées dans la Section 2.1.3.

2.1.1 Notions de base

Les obstacles et objets à manipuler

L'espace de travail est une scène 2D ou 3D comportant diverses entités géométriques qui peuvent être mobiles (*les objets à manipuler*) ou fixes (*les obstacles*), chacune étant composée d'un ou de plusieurs corps rigides. La position et l'orientation de ces corps sont définies par une transformation qui permet de passer d'un repère de référence de l'espace de travail $\mathcal{F}_{\mathcal{W}}$ à un repère $\mathcal{F}_{\mathcal{B}}$ associé au corps $\mathcal{B}$. Dans le cas où l'environnement est composé d'*obstacles statiques*, les transformations caractérisant les corps rigides de ces entités sont invariantes. Dans le cadre de cette thèse nous allons également considérer des *obstacles mobiles* qui vont soit être manipulés par le robot, soit être des obstacles qui peuvent apparaître et disparaître de l'espace de travail et modifier ainsi la structure de l'environnement.

Le robot

Un robot est un ensemble de corps rigides reliés par des liaisons cinématiques qui contraignent leurs mouvements relatifs. L'ensemble de ces liaisons définissent *la chaîne cinématique* du robot. Dans le cas précis d'un manipulateur, une des extrémités de la chaîne est reliée à une base (fixe ou mobile), tandis que l'autre est généralement dotée d'un *organe terminal* (*e.g.* un outil ou une pince) permettant la manipulation d'objets dans l'espace. Lorsque cette chaîne comprend des boucles cinématiques, des contraintes rendant certaines liaisons interdépendantes apparaissent. Le mouvement d'un robot peut être vu comme une transformation d'une posture du robot à une autre. La posture d'un robot dans $\mathcal{W}$ peut être définie de façon unique grâce à un ensemble de paramètres nommés *degrés de liberté* définis par la chaîne cinématique du robot. Les fonctions permettant d'exprimer la position et l'orientation de l'organe terminal en fonction de ces paramètres sont appelées fonctions de *cinématique directe*.

Espace des configurations $\mathcal{C}$

L'espace de la représentation paramétrique des postures du robot est appelé *espace de configuration* $\mathcal{C}$ du robot. L'avantage de cette représentation est que le problème de planification de mouvement se réduit à trouver un chemin sans collision pour un point représentant la configuration du robot. Cette simplification induit une dimension de l'espace des configurations égale au nombre de degrés de liberté du robot. Une *configuration* q, définie par un vecteur multidimensionnel, représente une posture du robot, chaque paramètre du vecteur correspondant à la valeur d'un de ses *degrés de liberté*. Cette notion d'espace de configuration originaire de la mécanique a été introduite en robotique par Lozano-Pérez [Lozano-Pérez 83]. La portion de $\mathcal{C}$ décrivant l'ensemble des configurations sans collisions est notée $\mathcal{C}_{free}$. De même l'ensemble des configurations en collision est noté $\mathcal{C}_{obst}$.

2.1.2 Formulation du problème

Le problème de planification de mouvement peut être sommairement défini de la façon suivante : étant donné un robot $\mathcal{A}$ évoluant dans un environnement statique parsemé d'obstacles, que l'on nommera $\mathcal{W}$, trouver s'il existe un chemin τ amenant le robot $\mathcal{A}$ d'une posture initiale à une posture désirée tout en respectant les contraintes cinématiques du robot ainsi qu'en évitant les obstacles $\mathcal{B}_i$.

En utilisant la notion d'espace des configurations [Lozano-Pérez 83], la représentation de $\mathcal{A}$ se trouvant à une position donnée devient un point $q \in \mathcal{C}$. Le chemin τ peut donc être représenté comme une fonction continue $\tau : [0,1] \rightarrow \mathcal{C}$ tel que $\tau(0) = q_{init}$ et $\tau(1) = q_{final}$, avec q_{init} la configuration initiale et q_{final} la configuration finale. Le chemin solution τ est dit *faisable*, si il est *libre* de collision et qu'il est *admissible* ; *i.e.* qu'il respecte les différentes contraintes internes du robot (*e.g.* contraintes de chaîne fermée, contraintes différentielles,...).

Formulé de cette façon, le problème de planification de mouvement consiste en l'exploration de la connectivité du sous-espace $\mathcal{C}_{free}$. Une solution existe si et seulement si q_{init} et q_{final} appartiennent à la même composante connexe de ce sous-espace. La principale difficulté de cette formulation est la représentation du sous-espace $\mathcal{C}_{obst}$. Nous allons voir dans la Section 2.1.3 et la Section 2.2 des méthodes qui s'affranchissent de cette représentation en utilisant les modèles 3D du robot et des obstacles combinés à des algorithmes de détection de collision.

Les documents de référence dans le domaine [Canny 88, Latombe 91, Choset 05, LaValle 06] donnent plus de détails sur cette formulation du problème de planification de mouvement.

2.1.3 Differentes approches

Trois grandes familles d'approches ont été développées durant ces dernières décennies pour résoudre le problème de planification de mouvement : les méthodes de *décomposition cellulaire* [Schwartz 83], les *champs de potentiel* [Khatib 86] et la *construction de graphes* [Nilsson 69]. Une des propriétés importantes de ces méthodes de planification est la notion de *complétude*. On distingue des planificateurs *complets* (méthodes exactes), *complets en résolution* ou *en probabilité*. Une description plus complète de ces approches est présente dans [Latombe 91, Choset 05, LaValle 06]

Traditionnellement, l'algorithmique du mouvement conduisait à des méthodes exactes (*e.g.* [Schwartz 83, Canny 88] pour la résolution du problème du déménageur de piano) qui garantissaient de trouver un chemin solution s'il existe, ou de signaler qu'aucune solution n'est possible. Cependant ces méthodes n'ont qu'un intérêt théorique car elles ne permettent pas de résoudre des problèmes de planification pratiques dans des temps raisonnables. En effet, il a été montré que le problème est intrinsèquement difficile [Reif 79] et que sa complexité croît exponentiellement avec le nombre de degrés de liberté du robot [Canny 88]. De nouvelles approches basées sur la discrétisation de $\mathcal{C}$ ont été proposées [Faverjon 84, Lozano-Pérez 87]. Ces planificateurs garantissent de trouver un chemin solution s'il existe, mais uniquement à une résolution donnée : ils sont donc complets en résolution. Dans les années 90, sont apparus des planificateurs plus

généraux ne reposant plus sur une construction préalable d'une représentation exacte ou approchée de l'espace de configuration [Barraquand 91, Kavraki 96, Švestka 97, LaValle 98]. Ces méthodes de planification dites *méthodes probabilistes*, permettent de faire face à la complexité inhérente au problème. Le terme *probabiliste* caractérise la complétude probabiliste de ces algorithmes qui garantissent de trouver une solution en un temps fini si elle existe. Ces méthodes, très efficaces en pratique, sont aujourd'hui largement utilisées. La section suivante fait la synthèse de travaux récents sur ces techniques.

2.2 Algorithmes probabilistes

Les algorithmes probabilistes consistent en la construction d'un graphe capturant la connexité de l'espace libre $\mathcal{C}_{free}$ en utilisant l'aléa, dans le but de s'affranchir d'une représentation explicite de l'espace de configuration $\mathcal{C}$. Deux grandes familles de méthodes probabilistes sont distinguables : les méthodes de construction de *réseaux probabilistes* aussi appelées PRM [Kavraki 96, Švestka 97], et les méthodes de *diffusion* ou *incrémentales*(*e.g.* RRT [LaValle 98], EST [Hsu 99], ...) (Voir Figure 2.1). Le choix de l'utilisation d'une méthode ou de l'autre dépend fortement de l'application.

Les réseaux probabilistes se caractérisent par une phase de précalcul visant à capturer la connexité de l'espace libre dans un graphe. Les requêtes de planification sont ensuite résolues rapidement en se connectant au réseau précalculé. Également appelées méthodes à *requêtes multiples*, les réseaux probabilistes conviennent aux applications où plusieurs requêtes de planification d'un même système dans un même environnement doivent être résolues.

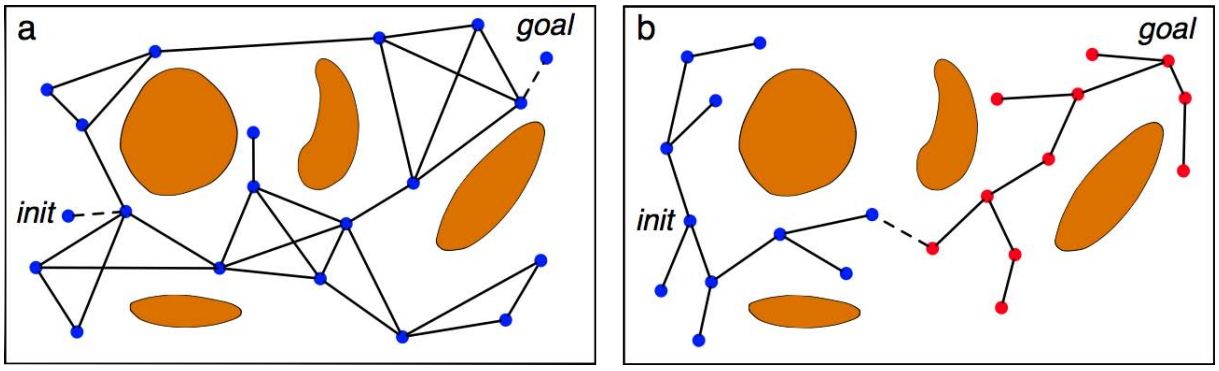

FIGURE 2.1 – On distingue deux grandes familles de méthodes probabilistes : les méthodes de construction de réseaux probabilistes (a) et les méthodes de diffusion (b).

Les méthodes de recherche incrémentale sont, quant à elles, utilisées pour résoudre des *requêtes uniques* comme par exemple des problèmes de désassemblage contraints. Ces méthodes construisent de manière incrémentale des arbres de recherche (*i.e.* des graphes ne contenant aucun cycle) à partir des configurations initiale et finale mais ne cherchent pas, contrairement aux précédentes, à calculer toute la connexité de $\mathcal{C}_{free}$. Leur performance vient du fait qu'elles ne nécessitent pas de phase de précalcul. Ces méthodes sont adaptées pour la résolution de

problèmes de désassemblage [Chang 95], où les configurations initiale et/ou finale sont dans des zones très contraintes.

Dû au caractère probabiliste de ces deux familles de méthodes, une phase d'optimisation est exécutée de manière à produire des chemins lisses et réguliers.

2.2.1 Les réseaux probabilistes

Ces méthodes ont été introduites en parallèle par Kavraki et Latombe sous le nom de PRM (Probabilistic Roadmap Method) [Kavraki 95, Kavraki 96] et Švestka et Overmars sous le nom de PPP (Probabilistic Path Planner) [Švestka 97]. Le principe de ces méthodes est d'échantillonner aléatoirement des configurations dans l'espace et de conserver celles appartenant à $\mathcal{C}_{free}$ en tant que nœuds du réseau. Ces nœuds sont reliés entre eux par des arêtes qui correspondent à des *chemins locaux* libres de collision et réalisables par le système. Une fois ce réseau construit, une requête de planification peut être résolue en connectant les configurations initiale et finale au graphe et en cherchant un chemin à travers celui-ci à l'aide d'algorithmes comme Dijkstra ou A^*.

Génération des configurations

La difficulté de l'échantillonnage aléatoire de $\mathcal{C}$ est de trouver des connexions à travers des régions "critiques" de $\mathcal{C}_{free}$ aussi connues sous le nom de *passages étroits*. Pour palier cette difficulté, plusieurs travaux ont proposé des variantes afin d'augmenter la densité d'échantillonnage dans ces zones. Une approche possible [Hsu 98, Saha 05, Hsu 06], est d'autoriser dans un premier temps une certaine pénétration dans $\mathcal{C}_{obst}$. Ces configurations seront ensuite "poussées" vers $\mathcal{C}_{free}$. Une approche similaire "Obstacle Based PRM" a également été proposée dans [Amato 98], où les configurations sont tirées dans $\mathcal{C}_{obst}$ puis "poussées" dans $\mathcal{C}_{free}$. [Boor 99] propose une approche basée sur un "tirage gaussien", dans laquelle on génère directement des configurations proches de $\mathcal{C}_{obst}$. En s'inspirant de ces travaux, une méthode plus sélective appelée "bridge test" privilégie les configurations libres entourées de deux configurations en collision [Hsu 03, Ding 07]. Un autre type d'approche consiste à s'éloigner le plus possible de $\mathcal{C}_{obst}$. [Wilmarth 98, Lien 03, Yang 04] proposent des méthodes basées sur l'échantillonnage autour de l'axe médian de $\mathcal{C}_{free}$. Ces méthodes sont plus coûteuses en temps que les précédentes car elles utilisent en plus de la détection de collision, un calcul de distance aux obstacles.

D'autres formes d'échantillonnage dites *pseudo-aléatoires* existent [LaValle 04]. L'intérêt de ces méthodes est que les échantillons générés soient répartis d'une façon plus uniforme dans $\mathcal{C}$ qu'avec un échantillonnage aléatoire simple. Ces techniques sont basées sur une génération de points à l'intérieur de structures comme les grilles ou les lattices, ou encore sur un tirage spécifique pour chaque coordonnée du vecteur de configuration comme les méthodes de Halton [Halton 64] et de Hammersley [Hammersley 56].

Connexion des échantillons

Plusieurs stratégies de connexion peuvent être utilisées pour relier les échantillons générés. La stratégie la plus simple est de tenter de connecter tous les nœuds existants du graphe. Cette solution aboutit cependant à un réseau extrêmement redondant et coûteux à construire. En pratique, les connexions sont limitées aux k nœuds les plus proches [Kavraki 96], ou encore aux nœuds situés à une distance inférieure à un seuil donné [Bohlin 00]. Une autre stratégie Visib-PRM décrite par [Nissoux 99] limite les tests de connexion à chaque composante connexe dans le but de réduire la taille du graphe. D'autres méthodes visent à conserver les cycles utiles dans les graphes générés. Dans [Nieuwenhuisen 04], la connexion entre deux nœuds est testée seulement si la distance qui les relie dans le graphe est k fois plus grande que la distance les séparant réellement. Une méthode, proposée dans [Jaillet 08b], s'affranchit des paramètres de contrôle, créant des réseaux cycliques de faible taille permettant de capturer les différents types de chemin libre de l'espace. Une comparaison des graphes calculés avec les méthodes PRM [Kavraki 96], Visib-PRM et PDR est illustrée Figure 2.2.

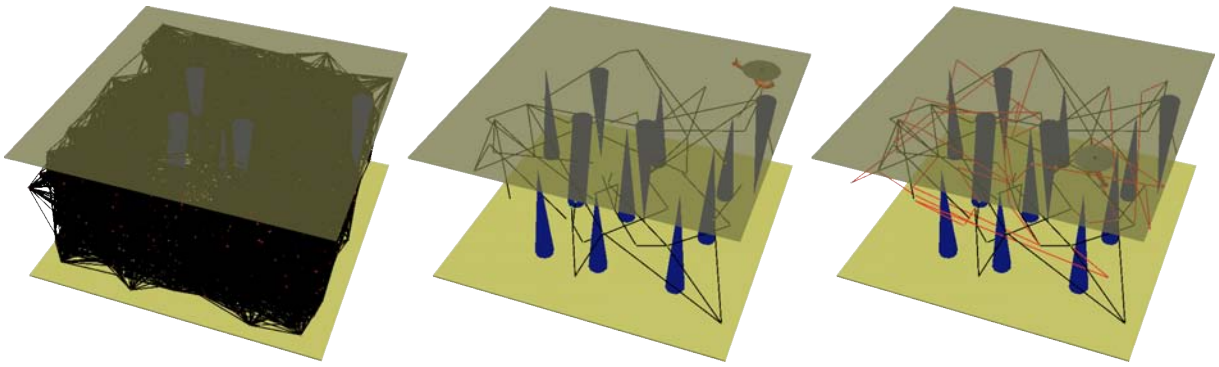

FIGURE 2.2 – Comparaison du nombre de nœuds du réseau entre une méthode PRM (gauche), une méthode Visib-PRM (milieu) et la méthode PDR (droite) créant des cycles (en rouge) (d'après [Jaillet 05a]).

Métrique dans $\mathcal{C}$

La plupart de ces méthodes nécessitent d'évaluer la distance entre deux configurations. Une bonne métrique est critique pour la performance des méthodes probabilistes en général, vu que cette information est utilisée pour décider quelles sont les connexions à tester ou dans quelle direction se fera l'expansion des arbres aléatoires (voir la Section 2.2.2). La métrique idéale serait de considérer les contraintes appliquées au système (*e.g.* les limites articulaires, les obstacles, les contraintes différentielles, ...). Malheureusement, établir une métrique de la sorte est un problème extrêmement complexe. Généralement, une simple métrique Euclidienne normée dans $\mathcal{C}$ permet d'obtenir de bons résultats pour la planification holonome [Amato 00]. Cependant, quand le système est soumis à des contraintes différentielles, la mise en œuvre d'une métrique appropriée est plus difficile [Laumond 98, LaValle 02].

Détection de collision

La phase la plus coûteuse (en temps de calcul) lors de la construction des réseaux probabilistes est la détection de collision des configurations et des chemins locaux les reliant. L'exécution de ces algorithmes représente en moyenne 90% du temps total de calcul pour les algorithmes PRM standards [Geem 01]. De nombreux algorithmes pour la détection de collision reposent sur une décomposition hiérarchique de chaque objet de l'environnement ainsi que du système robotique en formes géométriques simples, par exemple des sphères ou des boîtes ou des enveloppes convexes [Jiménez 98, Larsen 00, Lin 03]. Par ailleurs, des méthodes comme *Lazy-PRM* [Bohlin 00] et *Fuzzy-PRM* [Nielsen 00], retardent l'appel au détecteur de collision au maximum de façon à ne vérifier que les nœuds et les arêtes qui vont potentiellement être utilisés dans le chemin final calculé.

Contrôle de l'algorithme

Une autre difficulté liée à la construction de réseaux probabilistes est le critère d'arrêt. La plupart des algorithmes considèrent simplement un nombre maximal de nœuds à générer ou un temps de précalcul maximal. Peu de méthodes proposent un critère d'arrêt pertinent. L'algorithme *Visib-PRM* [Nissoux 99, Siméon 00] propose un critère d'arrêt en fonction d'une estimation de la proportion de $\mathcal{C}_{free}$ couverte par le réseau. Le principe de l'algorithme est de n'ajouter un nouveau nœud dans le graphe que si sa configuration associée relie au moins deux composantes connexes différentes ou si elle n'est reliable à aucun nœud du graphe. Le critère d'arrêt de cette méthode est basé sur le nombre d'échecs consécutifs d'ajout de nœuds dans le graphe qui permet d'estimer la couverture de $\mathcal{C}_{free}$. La méthode *PDR* proposée dans [Jaillet 08b], utilise un critère similaire pour arrêter la création des cycles utiles.

Lissage des chemins solutions

Dus à leur caractère probabiliste, les chemins solutions générés par les méthodes présentées ne sont pas directement exploitables. Ces chemins, irréguliers et souvent trop longs, doivent être d'abord lissés. La méthode "Shortcut" [Hsu 00] consiste à couper aléatoirement le chemin à lisser en trois parties et essayer de relier leur extrémités directement avec un chemin local unique. Une autre méthode qui optimise la longueur du chemin en le suréchantillonant et en utilisant une méthode de recherche pour relier les configurations éloignées [Guernane 09]. Un autre algorithme d'optimisation basé sur la méthode VFM (Voronoï Fast Marching Method) est proposé dans [Garrido 08].

2.2.2 Les méthodes de diffusion

Les méthodes de diffusion, contrairement aux réseaux probabilistes, cherchent uniquement à résoudre une requête de planification entre une configuration initiale et une configuration finale, sans toutefois chercher à explorer tout l'espace. On distingue généralement des versions monodirectionnelles qui ne développent qu'un seul arbre et des versions bidirectionnelles où les

deux arbres générés ont respectivement pour racine la configuration initiale et la configuration finale du problème. Ces méthodes sont particulièrement efficaces pour traiter des problèmes contraints au départ et/ou à l'arrivée.

Le planificateur RPP (Randomized Path Planner) [Barraquand 91] est basé sur un champ de potentiel qui attire le robot vers sa configuration finale. Les mouvements aléatoires du robot lui permettent de sortir des minima locaux. Cependant ce planificateur nécessite le réglage de nombreux paramètres qui agissent directement sur la performance. La méthode du "Fil d'Ariane" (ACA) [Bessiere 93, Mazer 98] est basée sur les algorithmes génétiques pour résoudre les problèmes de planification. Contrairement aux autres planificateurs probabilistes, les calculs ne sont pas fait dans l'espace des configurations mais dans l'espace des commandes. La technique proposée dans [Hsu 97] utilise une stratégie de diffusion basée sur la densité d'échantillonnage. Le nœud à étendre est choisi aléatoirement avec une probabilité inversement proportionnelle à la densité du graphe (nombre de nœuds situés dans un voisinage). Une extension de la méthode a été proposée dans [Sánchez 03] en reprenant l'idée des tests de collisions paresseux utilisée dans les "Lazy-PRM" [Bohlin 00].

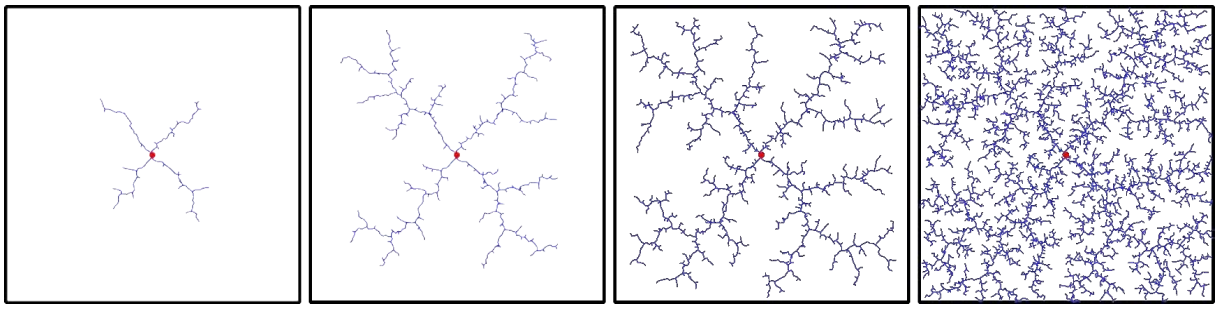

FIGURE 2.3 – Évolution d'un arbre couvrant l'espace libre par la méthode RRT.

La méthode d'exploration RRT (Rapidly-exploring Random Tree) [LaValle 98], est devenue la méthode de diffusion la plus populaire ces dernières années. Cet algorithme a été initialement développé pour résoudre des problèmes de planification sous contraintes kinodynamiques. Il a également été adapté au cas plus simple des systèmes holonomes qui peuvent être directement formulés dans $\mathcal{C}$ [Kuffner 00, LaValle 01]. Comme pour les méthodes citées dans le paragraphe précédent, les algorithmes RRT se composent d'une phase de construction et d'une phase de connexion. Une configuration est aléatoirement échantillonnée. La droite passant par cette configuration et le nœud de l'arbre le plus proche déterminant le nœud et la direction d'extension de l'arbre (voir Figure 2.3). Dans la méthode RRT basique (plus communément nommée "RRT-Extend"), l'extension se fait avec un pas régulier. Ceci signifie que les configurations générées sont au plus à une distance d du nœud de l'arbre le plus proche. La variante "RRT-Connect" proposée dans [Kuffner 00] réitère la phase d'extension de RRT-Extend tant que les contraintes du système (*e.g.* absence de collisions) sont respectées. L'intérêt de ces algorithmes vient d'un guidage efficace de l'exploration : la probabilité pour un nœud d'être sélectionné pour l'extension est proportionnelle à sa région de Voronoï.

De nombreuses méthodes basées sur la méthode RRT ont été développées pour résoudre différents types de problème ou pour améliorer certains aspects de l'algorithme. Des travaux ont été menés afin d'essayer de résoudre plus efficacement des requêtes de planification dans des espaces des configurations contenant des passages étroits. Une méthode "Obstacle Based RRT" proposée par [Rodriguez 06] utilise différentes méthodes d'extension de l'arbre suivant la distance des échantillons aux obstacles. La méthode "Density Avoided Sampling RRT" [Khanmohammadi 08] introduit un biais dans l'échantillonnage des configurations directrices en s'éloignant des zones denses de $\mathcal{C}$. La méthode proposée dans [Jaillet 05b] se base sur un calcul de la région de visibilité des nœuds existant dans l'arbre de diffusion pour générer les nouveaux échantillons. D'autres méthodes visent à biaiser l'étape d'extension de la méthode en s'appuyant sur des grilles [Ranganathan 04, Guitton 09], sur une heuristique [Urmson 03], ou sur des cartes de coût [Jaillet 08a]. Finalement des méthodes ont été proposées pour résoudre des problèmes de désassemblage entre objets articulés [Cortés 08, Le 09].

2.3 Résolution de tâches complexes

La performance actuelle des méthodes probabilistes, en particulier pour la résolution de problèmes de grande dimension, permet d'aborder de nouveaux problèmes plus complexes dont la résolution était très coûteuse, voire pas envisageable. Trois extensions en rapport avec les contributions de cette thèse sont présentées dans cette section. Nous allons premièrement nous intéresser à la planification de systèmes mécaniques composés de plusieurs robots évoluant simultanément dans le même espace de travail. Nous présentons ensuite des méthodes de planification pour des systèmes comportant une ou plusieurs chaînes cinématiques fermées. Enfin, nous abordons l'extension des méthodes à la planification de tâches de manipulation.

2.3.1 Planification multi-robots

Trois grandes familles d'approches existent pour planifier le mouvement simultané de plusieurs robots. Les approches *centralisées* [Schwartz 83] considèrent un seul système formé de l'ensemble des robots. Ces approches se placent dans un espace des configurations composé de l'espace des configurations de chacun des robots. Les approches *découplées* [Kant 86], quant à elles, planifient plus ou moins indépendamment les trajectoires pour chacun des robots et cherchent ensuite à les coordonner. Finalement les approches de *coordination de réseaux* [Švestka 95], s'inspirent des méthodes découplées et génèrent des réseaux pour chacun des robots puis les coordonnent afin de rechercher une solution pour le système composé comme le font les approches centralisées.

Les approches centralisées ont pour avantage d'être des méthodes complètes en résolution, garantissant de trouver une solution si elle existe, même si en pratique, leur complexité les limite à des problèmes impliquant peu de robots. À l'inverse, les méthodes découplées sont incomplètes mais permettent d'obtenir des solutions de manière efficace quand le problème n'est pas trop contraint. Les approches de coordination de réseaux se situent entre les deux : elles garantissent

une meilleure complétude que les méthodes découplées tout en conservant l'efficacité de ces méthodes. En effet, le fait de calculer un chemin pour chacun des robots contraint la coordination de l'ensemble. Or avec les méthodes de coordination de réseaux, ces contraintes sont relaxées grâce à l'utilisation de réseaux au lieu de chemins.

Les méthodes centralisées ont été décrites pour la première fois dans [Schwartz 83], où une décomposition cellulaire est utilisée pour planifier le mouvement de plusieurs disques autour d'obstacles polygonaux. D'autres méthodes ont utilisé différentes techniques de *champs de potentiel*. Tournassoud [Tournassoud 86] propose une méthode où la coordination est exprimée comme un problème de minima locaux. Dans [Barraquand 90, Barraquand 91], les auteurs s'appuient sur les RPP [Barraquand 91] en y ajoutant des champs de potentiels pour s'échapper des minima locaux.

Parmi les méthodes découplées, les méthodes prioritaires [Erdmann 86] définissent un ordre de priorité entre les robots et calculent de manière séquentielle des chemins valides pour les robots en commençant par ceux ayant la plus grande priorité. La notion de diagramme de coordination [O'Donnell 89], ne considérant que deux robots, est utilisée pour représenter leurs positions respectives le long des chemins menant à une collision mutuelle. L'extension proposée dans [Siméon 02] permet d'utiliser le diagramme de coordination pour résoudre des problèmes impliquant plus d'une centaine de robots. Dans [Alami 95], les auteurs proposent une méthode décentralisée où les robots fusionnent incrémentalement leurs plans en prenant en compte leur état courant ainsi que les actions qu'ils vont réaliser. Enfin [Gil-Pinto 07] propose l'utilisation d'information localement générées afin que chaque robot (non holonome) calcule sa trajectoire optimale.

L'idée de coordination de réseaux a été développée sous différentes formes (*e.g.* [Shibata 93, LaValle 96, Švestka 97]). Pour trouver un chemin libre de collision pour l'ensemble du système, le planificateur calcule un graphe pour chacun des robots. Ces graphes sont ensuite fusionnés dans une structure qui représente leur produit cartésien. Plus récemment, une méthode se positionnant entre les méthodes centralisées et les méthodes découplées a été proposée dans [Saha 06]. Les auteurs suggèrent de mélanger les deux étapes majeures des méthodes décentralisées en fusionnant au fur et à mesure les chemins obtenus pour chacun des robots. Nos travaux présentés dans le Chapitre 3 s'inspirent des méthodes de coordination de réseaux pour traiter des problèmes où le système robotique est composé de plusieurs bras.

2.3.2 Planification de chaînes cinématiques fermées

Cette section porte sur l'extension des méthodes de planification de mouvement à des mécanismes comportant des chaînes cinématiques fermées (Figure 2.4). Les paramètres de configuration d'un système robotique contenant une ou plusieurs boucles cinématiques sont liées par des équations polynomiales non linéaires.

Des solutions exactes capables, en théorie, de résoudre ce problème ont été proposées (*e.g.* [Schwartz 83, Canny 88]). Les équations définissant les contraintes de fermeture cinématique sont directement traitées comme des contraintes holonomes au même titre que celles imposées par

les obstacles. Cependant, ces méthodes ne sont pas utilisables en pratique à cause de leur complexité et au temps de calcul. D'autres méthodes exactes ont été proposées pour des mécanismes particuliers, généralement simples. Une méthode proposée dans [Trinkle 02] ne traite que le cas de mécanismes comportant des articulations sphériques et ne prend pas en compte les obstacles. Récemment, une méthode complète en résolution a été proposée pour planifier les mouvements avec une cinématique arbitraire (*i.e.* avec plusieurs boucles cinématiques) [Porta 07]. Cette méthode est cependant limitée à des systèmes relativement simples.

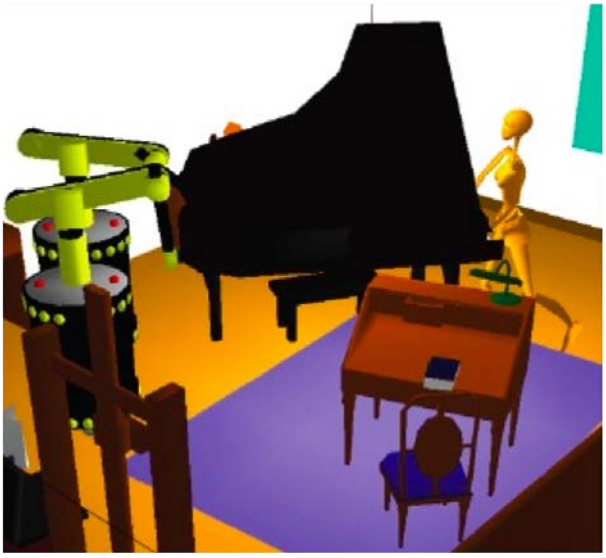

FIGURE 2.4 – Un exemple du problème du déménageur de piano. Un personnage virtuel et deux robots transportent un piano d'un côté à l'autre de la pièce en évitant les obstacles. Le système créé contient trois chaînes cinématiques fermées [Esteves 06].

La première extension des méthodes probabilistes pour la résolution de problèmes sous des contraintes de fermeture cinématique a été proposée par Koga et Latombe [Koga 94]. Ce planificateur génère des trajectoires pour de multiples bras robotiques coopérant dans la manipulation d'un même objet dans un environnement encombré. La trajectoire de l'objet est d'abord calculée à l'aide de l'algorithme RPP [Barraquand 91], puis un test, destiné à vérifier si les manipulateurs peuvent saisir l'objet, est effectué en utilisant une des solutions de la cinématique inverse de chaque bras.

Deux stratégies basées sur les méthodes probabilistes développées dans [LaValle 99, Han 01], proposent des extensions des méthodes PRM prenant en compte des contraintes de fermeture cinématique, mais abordent le problème différemment. Dans [LaValle 99], des méthodes d'optimisation sont utilisées tandis que dans [Han 01], des outils de résolution cinématique sont mis en œuvre. L'idée dans [Han 01] est de diviser chaque chaîne cinématique en deux sous-chaînes. La configuration liée à l'une des sous-chaînes (nommée *chaîne active*) est générée par échantillonnage aléatoire puis la résolution de la cinématique inverse de l'autre chaîne (nommée *chaîne passive*) calcule le reste des paramètres de la configuration afin de forcer la fermeture

cinématique. La difficulté avec cette méthode est de générer des configurations de la chaîne active dans l'espace de travail de la chaîne passive.

La méthode RLG (Random Loop Generator) introduite dans [Cortés 03, Cortés 04a] permet d'échantillonner plus efficacement les configurations. En effet, dans les méthodes présentées, la probabilité de générer une configuration d'une longue chaîne active permettant de trouver une configuration pour la chaîne passive fermant la boucle est faible. L'algorithme RLG permet de "guider" cet échantillonnage aléatoire de la chaîne active afin d'accroître la probabilité de fermer la boucle une fois le calcul de cinématique inverse résolu. Une des contributions de cette thèse propose une extension de cet algorithme. Il est donc détaillé ci-dessous.

Description de l'algorithme RLG

RLG est un algorithme d'échantillonage basé sur la décomposition du mécanisme en sous-chaînes ouvertes, actives et passives [Han 01]. Lorsque le système est composé de plusieurs boucles cinématiques, elles sont ordonnées puis chacune est traitée séparément. Les articulations communes à plusieurs boucles, une fois leurs valeurs calculées, sont considérées comme des liaisons rigides pour le calcul des configurations des boucles suivantes.

Le principe de l'algorithme est de réduire de façon itérative la complexité de la chaîne cinématique traitée jusqu'à ne plus avoir qu'une sous-chaîne non-redondante (la partie passive), avec une cinématique inverse simple à résoudre. La solution idéale est l'échantillonnage aléatoire des paramètres actifs dans un sous-ensemble de valeurs satisfaisant la fermeture cinématique. Cependant le calcul d'un tel sous-ensemble est aussi complexe que la résolution de la cinématique inverse du système entier. Une approximation couvrant le maximum de régions de $\mathcal{C}$ est utilisée. Pour une configuration q^a des chaînes actives, la configuration q^p de la partie passive est calculée en résolvant le problème de cinématique inverse de la chaîne passive. Généralement, un nombre fini solutions de cinématique inverse sont possibles. On obtient finalement n_{sol} configurations du système en combinant q^a et les différentes q^p.

Ces différentes étapes sont illustrées Figure 2.5, où l'algorithme est appliqué à un système six axes plan décomposé en deux parties actives ($\theta 1$, $\theta 2$ et $\theta 6$), et une passive ($\theta 3$, $\theta 4$ et $\theta 5$). Les Figures 2.5-a,b,c montrent comment les valeurs $\theta 1$, $\theta 2$ et $\theta 6$ sont itérativement obtenues. Chaque variable d'articulation est échantillonnée en approximant l'intervalle de fermeture cinématique. La Figure 2.5-d montre les deux solutions de cinématique inverse du mécanisme à trois axes plan correspondant à la sous-chaîne passive.

La méthode RLG est générale et capable de traîter des systèmes complexes ayant plusieurs boucles cinématiques. L'approche a été étendue à des robots parallèles [Cortés 03] ainsi qu'à un ensemble de bras robotisés manipulant un même objet [Cortés 04a] ou encore à des applications en biologie [Cortés 04b]. La contribution proposée Chapitre 4 porte sur l'extension de cette méthode par un traitement explicite des configurations singulières, permettant de trouver des chemins solutions nécessitant la reconfiguration du robot.

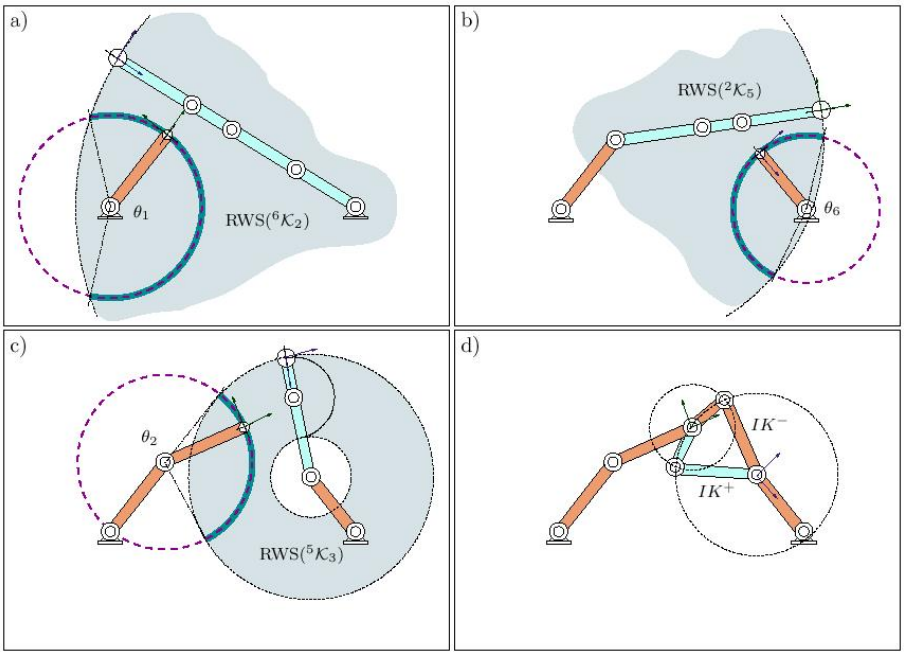

FIGURE 2.5 – Étapes d'exécution de l'algorithme RLG sur un système 6 axes plan [Cortés 04a].

2.3.3 Planification de tâches de manipulation

Le premier article traitant d'un problème de tâche de manipulation est celui de Wilfong [Wilfong 88] qui s'intéresse à la planification de mouvement d'un robot dans un environnement à deux dimensions contenant des objets déplaçables. Le robot peut éventuellement déplacer les objets pour atteindre son but. Alami et al. ont proposé une approche originale basée sur une formulation géométrique du problème [Alami 90, Alami 95]. Les auteurs introduisent une notion de sous-espaces particuliers nommés "GRASP" et "PLACEMENTS" en partant du fait qu'en saisissant un objet ou en le déposant à un autre endroit de l'environnement, le robot change son propre espace des configurations. Ces méthodes ont été améliorées en particulier dans [Chen 91, Latombe 91] avant l'introduction des planificateurs de mouvement aléatoires où différents espaces de configuration sont traités en fonction des positions relatives du robot et des objets mobiles.

Plusieurs approches se sont ensuite basées sur cette formulation (*e.g.* [Ahuactzin 98, Nielsen 00, Siméon 04]). La Figure 2.6 montre un exemple de tâche de manipulation mettant en scène un robot manipulateur mobile devant déplacer un objet encombrant dans un environnement contraint, résolu grâce à l'approche [Siméon 04]. Les travaux proposés dans [Stilman 04, Stilman 07] ainsi que dans [Okada 04] s'intéressent au problème de la navigation en présence d'obstacles mobiles. L'approche est basée sur la construction d'un graphe représentant la structure de l'environnement (voir Figure 2.7) mis à jour dynamiquement. À chaque déplacement d'un obstacle mobile par le robot, le graphe est mis à jour afin de déterminer la prochaine action à réaliser pour atteindre la destination finale du robot, en se frayant un chemin parmi les obstacles.

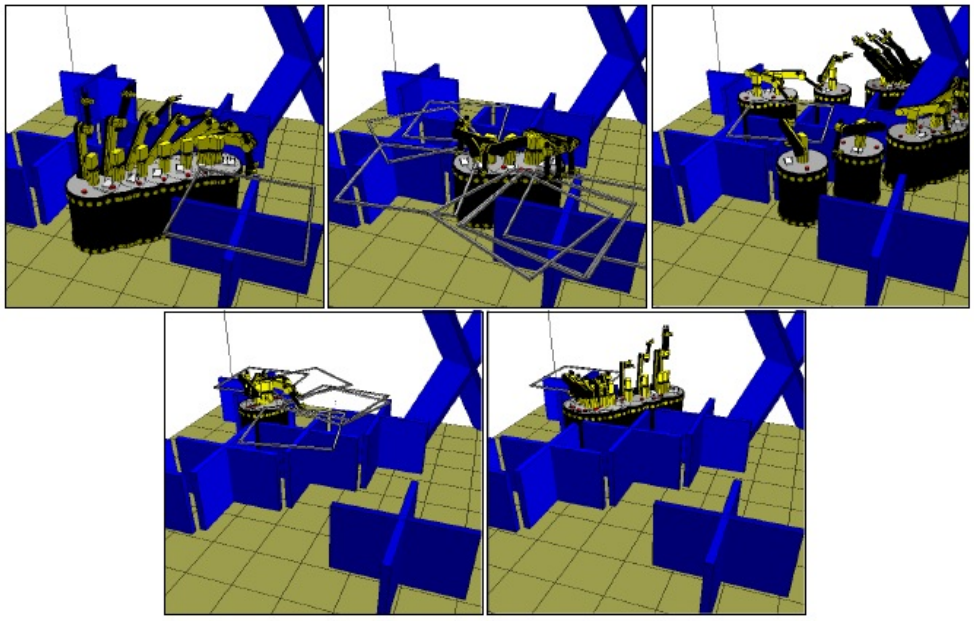

FIGURE 2.6 – Exemple de planification de manipulation avec relâche et ressaisie de l'objet. Le robot mobile doit transporter le cadre carré depuis sa position initiale (première vignette) à sa position finale (dernière vignette). Les obstacles forcent le robot à faire une pose (seconde vignette) et une ressaisie (quatrième vignette) pour achever la tâche [Siméon 04].

Ces travaux ont également été étendus à des problèmes de coopération multi-bras pour la manipulation [Koga 94]. Les auteurs proposent une approche incrémentale pour la construction du graphe de manipulation. Pour réduire la complexité du problème, ils supposent que l'ensemble des prises et l'ensemble des poses sont finis et connus. Cette approche planifie, dans un premier temps, un chemin de l'objet sans collision, en utilisant la méthode RPP [Barraquand 91], puis les configurations du système sont générées en calculant la cinématique inverse de chacun des bras à partir de la trace des points de prises déduite du chemin de l'objet.

Plus récemment, l'algorithme *BiSpace* [Diankov 08] propose la génération de trajectoires pour la saisie d'un objet. L'idée de cette approche est de calculer un ensemble de configurations de prise pour le torse humanoïde, indépendamment des obstacles et du robot. Une fois qu'une ou plusieurs configurations libres de collision sont trouvées, des arbres de diffusion sont calculés en partant de ces configurations de prise. Simultanément un autre arbre de diffusion est construit à partir de la position de départ du robot pour explorer son espace des configurations.

Un autre algorithme proposé dans [Berenson 08], décompose le problème de prise et pose en deux parties. Partant d'un ensemble de prises de l'objet déjà calculé, une première phase d'optimisation détermine la meilleure prise de l'objet ainsi que la configuration du robot permettant cette saisie de l'objet dans ses positions initiales et finales. Ensuite ces deux configurations sont reliées en utilisant un arbre de diffusion bidirectionnel.

Dans le cadre de tâches de saisie et ressaisie bi-bras, [Vahrenkamp 09] présente une méthode de calcul de cinématique inverse des manipulateurs partant de l'espace d'accessibilité du ro-

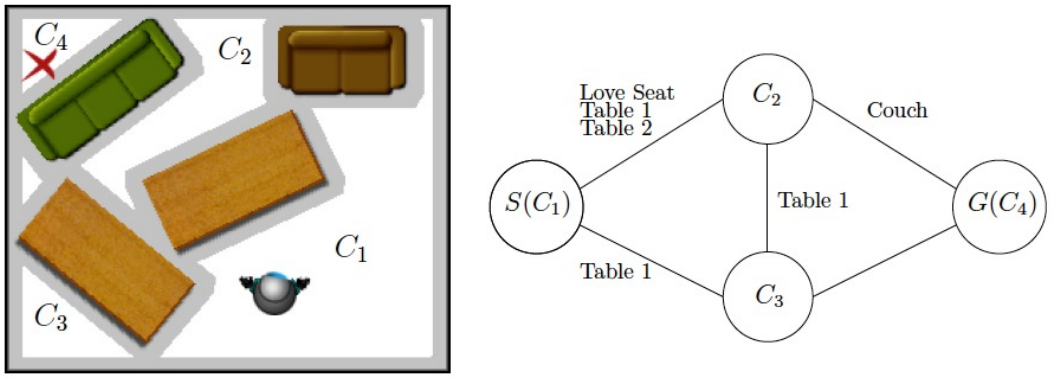

FIGURE 2.7 – Représentation dynamique de l'environnement sous forme de graphe. Les noeuds représentent les différentes composantes connexes de $\mathcal{C}_{free}$ et les arêtes les objets mobiles par lesquelles elles sont liées [Stilman 04].

bot à l'aide d'une descente de gradient. L'espace d'accessibilité est calculé par échantillonnage aléatoire. Deux algorithmes, combinés à la phase de recherche de fermeture cinématique, sont utilisés pour générer un chemin de manipulation sans collision : IK-RRT et J^+ RRT. Grâce au précalcul de l'espace d'accessibilité des deux bras manipulateurs du robot, il est possible de rapidement déterminer la probabilité pour que l'objet, à une position donnée, puisse être atteignable par les deux bras. L'algorithme IK-RRT se base sur cette méthode pour ne retenir que les configurations directrices atteignable par les deux bras lors de l'expansion de l'arbre de diffusion. La méthode J^+ RRT s'affranchit du calcul explicite des solutions de cinématique inverse en dirigeant l'expansion de l'arbre RRT vers la position finale de l'objet. La pseudo-inverse d'une jacobiène est utilisée pour diriger l'exploration de l'arbre dans l'espace des configurations.

Ces dernières méthodes sont clairement les plus génériques et abouties vu leur capacité à résoudre des tâches de prise et pose. Cependant, elle ne considèrent que des objets simples ou supposent qu'un ensemble de prises est fournit.

Des travaux s'adressant à des problèmes de manipulation dextre ont également été menés. Yashima propose dans [Yashima 02, Yashima 03] une méthode permettant de changer le mode de contact (glissement, rotation, ... au point de contact) de la prise. Ce changement de mode permet d'augmenter la taille du domaine atteignable des doigts ainsi que les mouvements autorisés. La méthode développée dans [Saut 07b], calcule la trajectoire des doigts et celle de l'objet ainsi que la séquence de repositionnement des doigts. Ces trajectoires sont obtenues par l'exploration de sous-espaces de prises, chacun étant le sous-espace de prise à n doigts. L'approche présentée dans [Caurin 09] calcule les trajectoires de manipulation grâce à une discrétisation de la cinématique de la main et utilise un système de transition discret pour passer d'un type de prise à un autre. Cette méthode permet de générer des trajectoires comportant plusieurs modes de contacts.

Dans le Chapitre 5, nous présentons une nouvelle approche pour réaliser des opérations de prise et de pose de l'objet qui requièrent une double prise pour changer l'objet de main.

3

Fusion de réseaux probabilistes

Ce chapitre présente une nouvelle méthode de planification de mouvement pour des systèmes multi-bras dans des environnements encombrés. Malgré l'efficacité avérée des planificateurs du type PRM, la construction de réseaux probabilistes capables de résoudre des problèmes contraints pour des systèmes multi-bras est complexe étant donné la grande dimension de l'espace des configurations. En effet, un système de type torse d'humanoïde (un torse et deux bras), est composé d'une vingtaine de degrés de liberté. Les mouvements de ces systèmes, en plus des contraintes induites par les obstacles de l'environnement, sont également limités par les contraintes d'auto-collision. Considérons l'exemple illustré Figure 3.1. La tâche que le robot Justin [Ott 06] doit réaliser est de retourner le tabouret pour mettre l'assise en haut. La figure montre la configuration finale de la requête de planification où le robot croise les bras pour retourner l'objet. Ce mouvement de saisie est difficile à réaliser à cause de la coordination des deux bras dans une région contrainte de l'espace de travail.

L'idée de la méthode présentée est de décomposer le système multi-bras en un ensemble de sous-systèmes (Section 3.1). Ces sous-systèmes sont traités comme des robots différents. Des réseaux probabilistes compacts sont calculés pour chacun d'entre eux (Section 3.3), puis un super-graphe est construit en fusionnant les réseaux élémentaires (Section 3.4). Notre méthode est inspirée des méthodes multi-robots de coordination de réseaux introduites dans [Švestka 95] pour des robots mobiles. La Section 3.6 décrit l'application de la technique ainsi qu'une analyse empirique des performances pour la planification de mouvement en environnement contraint de deux systèmes : un ensemble de 3 bras, et le robot Justin. Les résultats obtenus pour ces systèmes montrent l'efficacité de l'approche de fusion de réseaux probabilistes en comparaison des approches planifiant directement les mouvements du système complet.

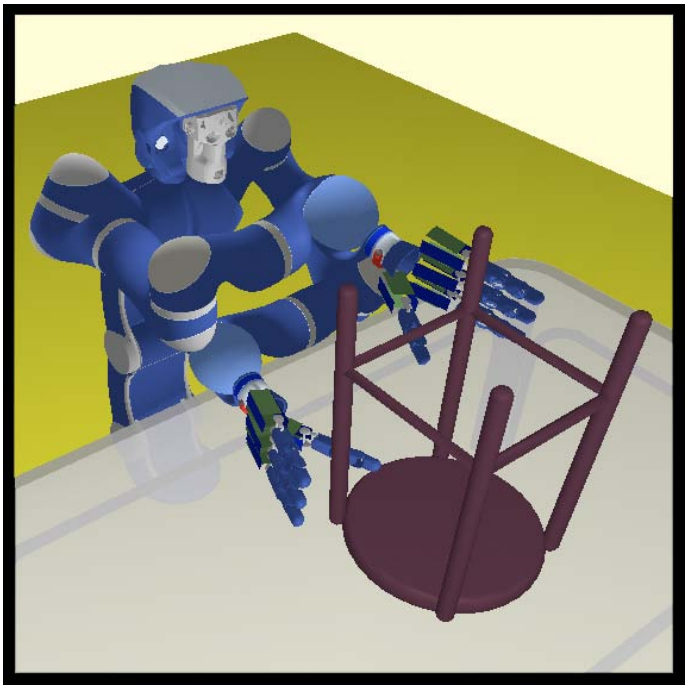

FIGURE 3.1 – Exemple de planification de mouvement dans un environnement contraint. Le robot doit croiser les bras pour retourner le tabouret.

3.1 Décomposition du système

La première étape de l'approche présentée est la décomposition du système multi-bras en parties élémentaires $\mathcal{P}_1 \dots \mathcal{P}_n$. Chaque partie correspond à une chaîne cinématique du robot. Une partie $\mathcal{P}_i$ est dite *indépendante* si le changement d'une valeur d'une de ses articulations n'affecte pas la position spatiale d'une autre partie du système : on la note $\mathcal{P}_i^I$. Dans le cas contraire, elle est appelée partie commune et notée $\mathcal{P}_i^C$. L'identification des parties indépendantes et communes est facilement réalisable à partir d'une analyse du diagramme cinématique du système multi-bras.

La Figure 3.2 illustre les différentes parties du robot Justin. Les bras sont indépendants l'un de l'autre. Si la valeur d'une articulation d'un bras est modifiée, le changement n'affecte la position d'aucune autre partie du système. Cependant, le changement d'une des articulations du torse change la position des bras ainsi que celle de la tête. En considérant les définitions ci-dessus, la tête est également une partie indépendante. Ce système est donc composé de trois parties indépendantes : le bras droit, le bras gauche (respectivement $\mathcal{P}_r^I$ et $\mathcal{P}_l^I$) et la tête ($\mathcal{P}_h^I$) ; et d'une seule partie commune : le torse ($\mathcal{P}_t^C$). La tête étant peu mobile, elle peut être considérée comme faisant partie du torse afin de faciliter la décomposition du système.

Mis à part les torses humanoïdes, la méthode proposée peut être appliquée à des systèmes multi-bras plus complexes. Prenons par exemple les robots humanoïdes (munis de jambes). Ces systèmes sont également séparables en quatre parties distinctes : les deux bras, la tête en parties indépendantes et le torse en partie commune. Les jambes de l'humanoïde sont également

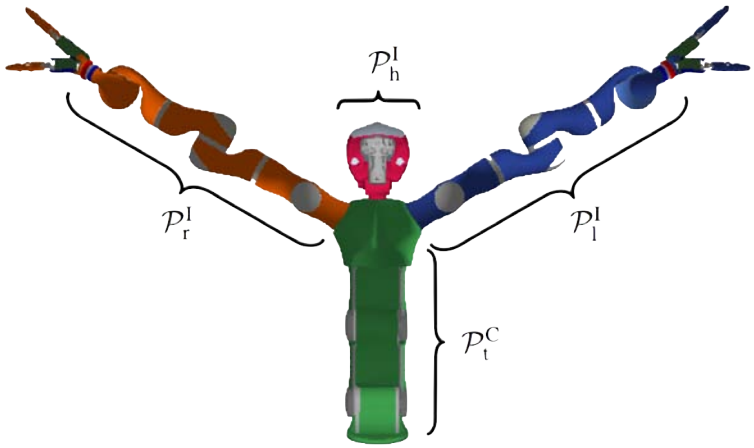

FIGURE 3.2 – Décomposition d'un système du type torse d'humanoïde en parties élémentaires. Les deux bras et la tête sont des parties "indépendantes" et le torse la partie "commune".

considérées comme partie commune. En effet, pour préserver la stabilité du robot, le mouvement d'une jambe influe directement sur la position spatiale du reste de la chaîne cinématique du robot (par exemple lors de la marche).

Les systèmes "arborescents" constituent un autre exemple de systèmes composés de torses et de bras, chaque bras étant composé d'un torse racine de plusieurs bras. Prenons l'exemple des systèmes présentés Figure 3.3. Ces systèmes sont techniquement concevables. Ils peuvent être décomposés de la façon suivante (respectivement de gauche à droite) :

- Le premier système en quatre parties : trois indépendantes et une commune.
- Le second en cinq parties : trois indépendantes et deux communes.
- Le troisième en huit parties : cinq indépendantes et trois communes.
- Le quatrième système en dix-sept parties : dix indépendantes et sept communes.

Les parties indépendantes sont représentées en vert tandis que les parties communes sont déclinées en plusieurs nuances de bleu.

Dans le cas général, les parties indépendantes sont celles se trouvant à l'extrémité de la chaîne cinématique. Le nombre de parties communes est déterminé par le nombre d'articulations directement reliées à des parties indépendantes. Ces articulations sont marquées en rouge dans la Figure 3.3. Les parties communes sont donc les chaînes cinématiques entre ces articulations et la base du système.

3.2 Vue d'ensemble de l'approche

La méthode proposée pour la planification de mouvement de robots multi-bras est basée sur la décomposition du système en parties élémentaires présentée ci-dessus. Cette décomposition permet de diviser la construction des réseaux probabilistes en deux étapes. La première étape consiste en la génération d'un réseau sans collisions $\mathcal{R}_i$ pour chaque sous-système composé d'une

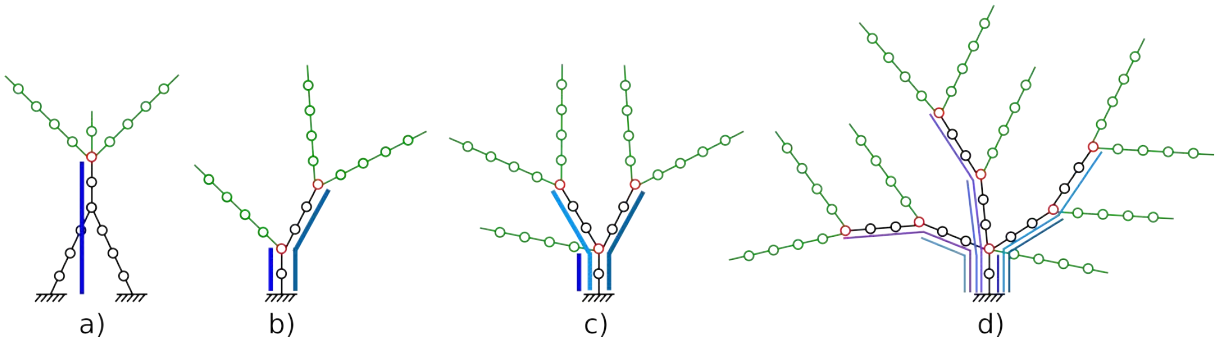

FIGURE 3.3 – Exemples de décomposition de systèmes robotiques arborescents. Les parties indépendantes sont illustrées en vert et les parties communes en nuances de bleu. Les articulations en rouge sont les articulations déterminant le nombre de parties communes.

partie indépendante $\mathcal{P}_i^I$ et de la partie commune correspondante [1] $\mathcal{P}_j^C$. Lors de la construction de ces graphes, seules les auto-collisions du sous-système et ses collisions avec les obstacles de l'espace de travail sont prises en compte. Toutes les méthodes de type PRM peuvent être utilisées pour générer ces réseaux. Cependant, afin de limiter la taille du graphe après fusion, on préfèrera les méthodes générant des graphes compacts comme Visib-PRM [Siméon 00] ou PDR [Jaillet 08a]. Ce graphe est défini comme le produit cartésien des réseaux de tous les sous-systèmes, dont la taille peut être très importante si des méthodes PRM standards sont utilisées. L'algorithme PDR est probablement le meilleur choix car il génère des cycles utiles requis lors de la phase de fusion, tout en gardant la taille du graphe suffisamment petite.

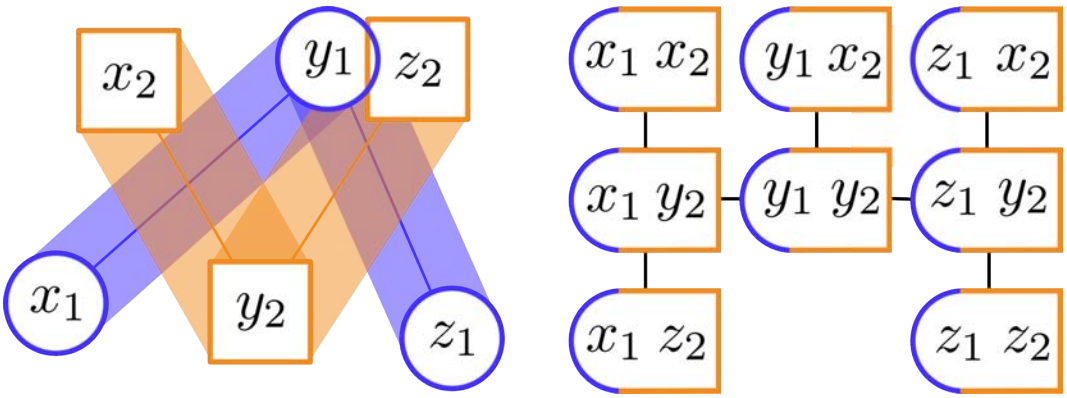

FIGURE 3.4 – À gauche, les réseaux probabilistes élémentaires pour un système simple composé de deux parties indépendantes (cercle et carré). À droite, le *Super Graphe* issu de la fusion.

Les réseaux probabilistes ainsi construits sont fusionnés en un réseau composite appelé *Super Graphe* ($\mathcal{SG}$), étendant l'idée initialement proposée dans [Švestka 97] pour le cas spécifique d'un ensemble de robots de type voiture. La Figure 3.4 illustre le principe de construction d'un *Super*

1. La partie commune correspondante à $\mathcal{P}_i^I$ est la partie dont la variation d'une de ses articulation influe sur la position spatiale de $\mathcal{P}_i^I$

Graphe pour un système composé de deux parties indépendantes (le cercle et le carré). Un réseau probabiliste est construit pour chaque partie. Le $\mathcal{SG}$ est ensuite construit par fusion des réseaux élémentaires. Chaque nœud du $\mathcal{SG}$ correspond à une configuration faisable et sans collisions de deux parties. Chaque arête correspond à un mouvement possible d'une ou des deux parties. L'Algorithme 3.1 montre les différentes phases de l'algorithme.

Algorithm 3.1: SUPERGRAPHCONSTRUCTION

Input: the robot A, the environment E
Output: the Super Graph $\mathcal{SG}$

for $i \leftarrow 1$ **to** n **do** // n = number of $\mathcal{P}^I$
  $\mathcal{R}_i \leftarrow$ EMPTYROADMAP;
  $\mathcal{R}_i \leftarrow$ COMPUTEELEMENTARYROADMAP(A, E);
$\mathcal{SG} \leftarrow$ EMPTYSUPERGRAPH;
$\mathcal{SG}$_NodeList $\leftarrow$ SUPERGRAPHNODEMERGE(A, E, $\mathcal{R}_{1 \ldots n}$);
for $i \leftarrow 1$ **to** k **do** // k = number of $\mathcal{SG}$ nodes
  for $j \leftarrow i$ **to** k **do**
    $\mathcal{SG} \leftarrow$ CONNECTTWOSGNODES(A, E, $\mathcal{SG}$_NodeList[i], $\mathcal{SG}$_NodeList[j]);

Les sections suivantes détaillent les deux phases de construction du *Super Graphe*. La Section 3.3 explique la construction des réseaux élémentaires pour chacun des sous-systèmes indépendants. La Section 3.4 décrit la fusion de ces réseaux élémentaires en un *Super Graphe*. Finalement, les requêtes de planification sont résolues en cherchant un chemin dans le *Super Graphe* $\mathcal{SG}$ (Section 3.5).

3.3 Construction des réseaux indépendants

L'algorithme PDR [Jaillet 08a] (pour "Path Deformation Roadmap") est particulièrement indiqué pour générer les réseaux probabilistes pour chaque sous-système, composé d'une partie indépendante $\mathcal{P}_i^I$ et de la partie commune correspondante $\mathcal{P}_j^C$. Il constitue une approche récente de planification de mouvement probabiliste basée sur les méthodes PRM. Le principe est de calculer des réseaux probabilistes de bonne qualité, capturant la connexité de l'espace des configurations dans des graphes compacts mais néanmoins représentatifs. De plus l'algorithme capture les différentes variétés de chemins libres de collisions. L'approche repose sur la notion de déformation de chemins indiquant si un chemin donné peut ou ne peut pas être déformé de façon continue en un autre chemin existant. L'algorithme PDR étend la méthode Visib-PRM [Siméon 00], construisant des réseaux en forme d'arbres compacts capturant la connexité de $\mathcal{C}_{free}$. La Figure 3.5-a illustre un arbre de visibilité généré par la méthode Visib-PRM et la Figure 3.5-b, les cycles additionnels ajoutés par la méthode PDR.

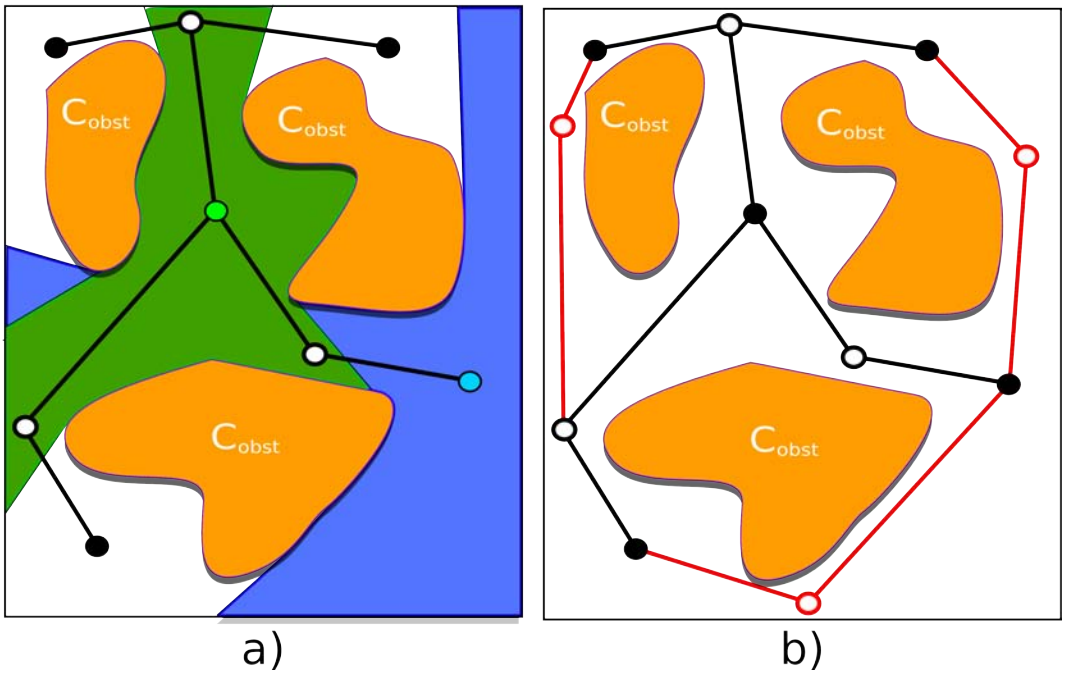

FIGURE 3.5 – a) Exemple d'un arbre de visibilité construit à l'aide d'une méthode locale linéaire. Les nœuds *gardiens* sont en couleur et les nœuds *connecteurs* en blanc. Les zones de couleur sont les domaines atteignables par les gardiens. b) Cycles générés par la méthode PDR partant de l'arbre de visibilité ci-contre.

La méthode Visib-PRM

L'idée principale de cette méthode, comme pour l'approche PRM basique, est de capturer la topologie de $\mathcal{C}_{free}$ dans un graphe. La principale différence est que seuls les nœuds possédant une certaine *visibilité* ou caractéristiques de connexion sont ajoutés au graphe. La *visibilité* (ou domaine *atteignable*) d'une configuration q aléatoirement échantillonnée pour une méthode locale $\mathcal{L}$ donnée, est définie comme suit :

$$Vis_{\mathcal{L}}(q) = \{q' \in \mathcal{C}_{free} \; telque \; \mathcal{L}(q, q') \subset \mathcal{C}_{free}\}$$

La configuration q est appelée *gardien* de $Vis_{\mathcal{L}}(q)$. Le principe de la méthode Visib-PRM est de générer aléatoirement des configurations dans $\mathcal{C}_{free}$. Ces configurations ne sont ajoutées dans le graphe que si elles ne "voient" aucun nœud du graphe (cas d'un *gardien*), ou si elles voient au minimum deux nœuds du graphe appartenant à des composantes connexes distinctes (cas d'un *connecteur*). La Figure 3.5-a montre un graphe de visibilité construit avec une méthode locale linéaire.

La méthode PDR

Une déformation du premier ordre entre deux chemins existe si et seulement s'il est possible de parcourir simultanément les deux chemins tout en maintenant une contrainte de visibilité entre

les points de chaque chemin (voir Figure 3.6). Cette formulation fournit un moyen algorithmique pour tester l'existence d'une déformation du premier ordre, également appelée *déformation de visibilité* entre deux chemins. Soit $\mathcal{L}_{lin}$ un segment de droite reliant deux configurations. La fonction paramétrée de visibilité Vis de deux chemins (τ, τ') est définie de la façon suivante :

$$Vis : \begin{cases} [0,1] \times [0,1] & \rightarrow & \{0,1\} \\ Vis(t,t') & = & 1 \text{ si } \mathcal{L}_{lin}(\tau(t), \tau'(t')) \in \mathcal{C}_{free} \\ Vis(t,t') & = & 0 \text{ sinon} \end{cases}$$

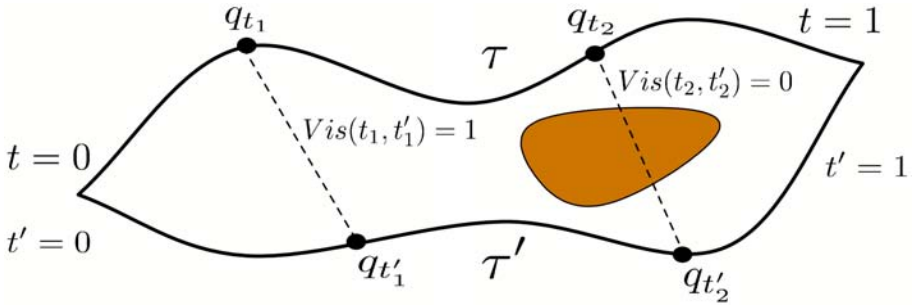

FIGURE 3.6 – La fonction paramétrée de visibilité de deux chemins évalue la visibilité entre les points de chaque chemin.

Le diagramme de visibilité des chemins (τ, τ') est défini comme le diagramme bidimensionnel de la fonction Vis. Deux exemples de diagrammes de visibilité et leurs chemins correspondant sont montrés Figure 3.7.

La déformation de visibilité (i.e. de premier ordre) entre deux chemins peut donc être définie de la façon suivante : deux chemins (τ, τ') ayant les mêmes points terminaux sont *visiblement déformables* de l'un sur l'autre si et seulement s'il existe un chemin dans leur diagramme de visibilité reliant les points de paramètres $(0,0)$ et $(1,1)$. Il est donc possible de tester la déformation de visibilité entre deux chemins en calculant leur diagramme de visibilité et en y cherchant un chemin reliant les points $(0,0)$ et $(1,1)$. Dans la Figure 3.7, une telle déformation n'est possible que dans l'exemple (b).

Le réseau probabiliste est initialisé avec une structure d'arbre calculée à l'aide de la méthode Visib-PRM. Cette initialisation assure une couverture de l'espace libre avec un nombre limité de nœuds et d'arêtes (i.e. sans cycles). À chaque itération, une configuration libre de collisions q_v est échantillonnée aléatoirement et la connectivité du sous-réseau visible est calculée. Un ensemble de nœuds visibles depuis q_v est donc défini en vérifiant si le segment droit reliant q_v à chaque nœud du réseau est libre. La connectivité de ces nœuds du point de vue de q_v est ensuite testée en deux phases consécutives. La première considère que toutes les arêtes du réseau sont potentiellement visibles. Deux nœuds sont donc détectés comme déconnectés si tous les chemins les reliant dans le réseau passent par au moins un nœud invisible. Si ce test rapide n'est pas suffisant pour déterminer la connectivité du sous-réseau visible, un test de visibilité des arêtes reliant les nœuds visibles est effectué. Ce test équivaut au test de validité d'une facette dans

l'espace des configurations définie par q_v et les configurations terminales de l'arête.

Le test de redondance est uniquement effectué quand le sous-réseau visible est déconnecté. Pour ce test, deux composantes connexes déconnectées du sous-réseau sont aléatoirement choisies et les plus proches gardiens [2] n_1, n_2 de q_v sont sélectionnés. Une déformation de visibilité entre le chemin $\tau = n_1 - q_v - n_2$ et un chemin du réseau est testée. Si une telle déformation existe, la configuration q_v est inutile au sein du réseau existant et est donc rejetée. L'algorithme mémorise le nombre d'échecs consécutifs depuis la dernière insertion de cycle utile. Cette information est utilisée pour arrêter l'algorithme quand l'insertion d'un nouveau cycle devient difficile, signifiant que la plupart des cycles utiles ont déjà été capturés dans le réseau.

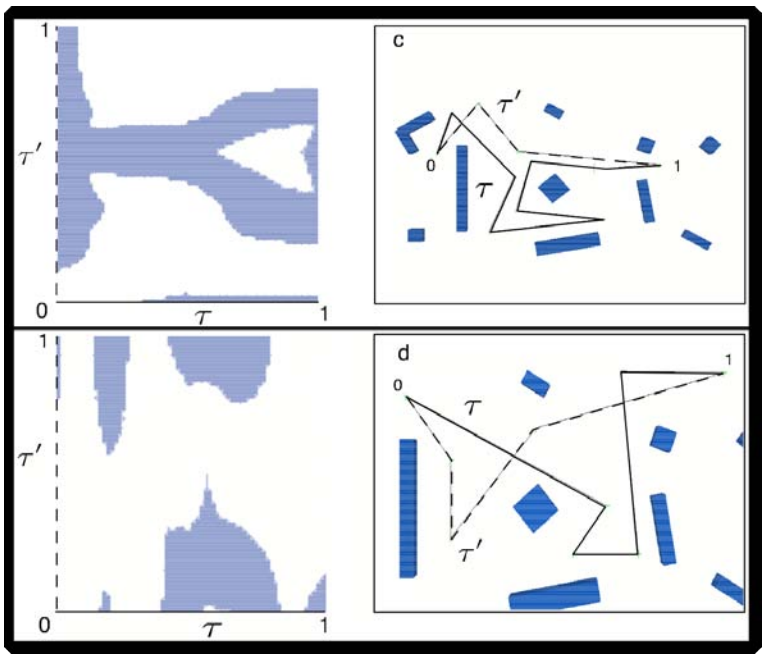

FIGURE 3.7 – Diagrammes de visibilité (à gauche) pour une paire de chemins ayant les même points terminaux (à droite). les zones blanches représentent les régions où $Vis(t, t') = 1$. Une déformation de visibilité est uniquement possible dans l'exemple (b), où un chemin valide reliant les points $(0, 0)$ et $(1, 1)$ peut être trouvé dans le diagramme de visibilité.

3.4 Fusion des réseaux indépendants

La construction du *Super Graphe* est réalisée grâce à la fusion des réseaux élémentaires. La fusion d'un nœud x_i de chaque réseau $\mathcal{R}_i$ produit un *Nœud* de *Super Graphe* X. Deux Nœuds de *Super Graphe* X et Y peuvent être connectés via une *Arête* de *Super Graphe*. La création et la connexion des Nœuds de Super Graphe sont détaillées dans les deux sous sections suivantes.

2. Voir [Siméon 00] pour une description précise de l'algorithme Visib-PRM

3.4.1 Construction des Nœuds de Super Graphe

Les Nœuds de $\mathcal{SG}$ sont créés en fusionnant les nœuds élémentaires x_i de chaque réseau $\mathcal{R}_i$. La Figure 3.8 illustre ce procédé pour un système possédant deux parties indépendantes et une seule partie commune. On peut distinguer trois cas différents. Le plus simple est lorsqu'un système est composé de n parties indépendantes et aucune partie commune (Figure 3.8 à gauche, "Cas A"). Dans ce cas, chaque nœud x_i est indépendant des autres. Les différents nœuds des réseaux élémentaires sont donc tout simplement concaténés. Quand le système est composé en plus des parties indépendantes d'une partie commune, deux cas de figure sont distinguables. Dans le premier cas, la configuration de la partie commune est identique pour tous les nœuds sélectionnés des réseaux élémentaires (Figure 3.8 au centre, "Cas B"). La configuration du système résultant de la fusion est composée de la concaténation de la configuration de la partie commune ainsi que celles des parties indépendantes. Dans le second cas, la configuration de la partie commune n'est pas identique pour les nœuds élémentaires sélectionnés (Figure 3.8 à droite "Cas C"). Dans ce cas, la fusion consiste en la création de k Nœuds de $\mathcal{SG}$, un pour chaque configuration différente de la partie commune. À ces configurations sont concaténées celles des parties indépendantes.

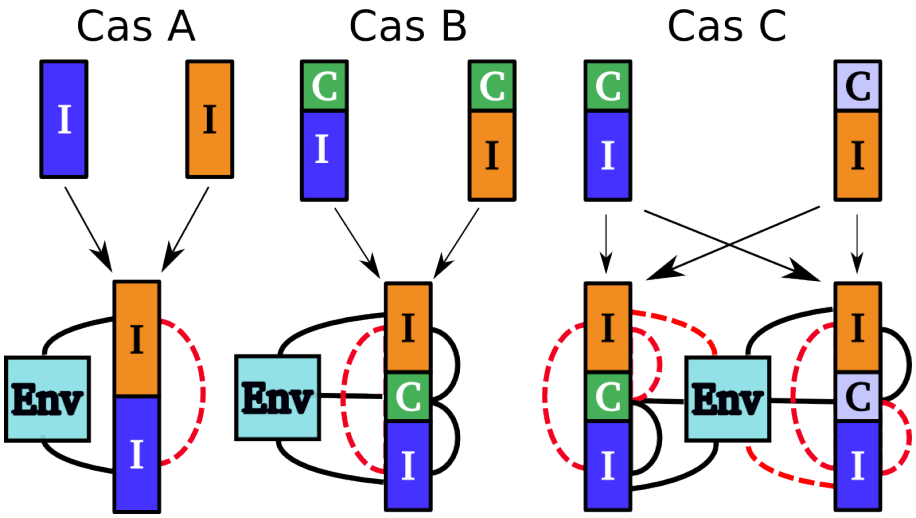

FIGURE 3.8 – Trois cas de fusion des nœuds de réseaux élémentaires. Les rectangles marqués "I" représentent les configurations des parties indépendantes et les rectangles marqués "C" les configurations de la partie commune. Les tests de collisions ne sont réalisés qu'entre les paires connectées avec un arc en pointillés. Les arcs pleins indiquent les tests de collision déjà réalisés entre deux paires durant la construction des réseaux élémentaires.

L'Algorithme 3.2 montre le pseudo-code de la phase de génération des Nœuds de $\mathcal{SG}$ par fusion des nœuds élémentaires. L'algorithme présenté est dédié à des systèmes possédant deux parties indépendantes et une partie commune.

Dans le cas de système possédant plusieurs parties communes (systèmes arborescents), la fusion des nœuds élémentaires est réalisée séquentiellement. Comme présenté dans la Section 3.1, ces systèmes sont composés de "bras" et de "torses" étant eux mêmes des "bras". Cette

Algorithm 3.2: SUPERGRAPHNODEMERGE

Input: the robot A, the environment E, the elementary roadmaps $\mathcal{R}_1$ & $\mathcal{R}_2$
Output: $\mathcal{N}_{vect}$ (Set of $\mathcal{SG}$ Nodes)

```
for All x1 ∈ R1 do
    for All x2 ∈ R2 do
        SGN ← EmptySGNodeSet() ;          // SGN is a tmp set of SG nodes
        // Node Merge
        if HasSameOrNoCommunePart(x1, x2) then
            c ← GetCommunePart(x1);
            SGN ← MergeNodes(c, x1, x2);
        else
            c ← GetCommunePart(x1);
            SGN ←MergeNodes(c, x1, x2);
            c ← GetCommunePart(x2);
            SGN ←MergeNodes(x2, x1, x2);
        // SG node collision detection
        for All n ∈ SGN do
            if IsCollisionFree(n, A, E) then
                N_vect ← n;
```

décomposition se retrouve donc dans la fusion des nœuds des réseaux élémentaires calculés. Tout d'abord, les configurations des parties indépendantes ayant la même partie commune sont fusionnées comme dans le "Cas B ou C" expliqués ci-dessus. Les configurations obtenues lors de cette fusion, sont elles mêmes fusionnées avec les autres configurations possédant les mêmes parties communes. Ce processus est réitéré jusqu'à l'obtention de la configuration du système entier. Prenons comme exemple le système "b)" de la Figure 3.3. Ce système contient deux parties communes $\mathcal{P}_1^C$ et $\mathcal{P}_2^C$ ainsi que trois parties indépendantes $\mathcal{P}_{1...3}^I$ (Voir Figure 3.9). Les configurations q_1, q_2 et q_4, sont les configurations appartenant aux réseaux élémentaires. Afin d'obtenir la configuration q_3, la fusion des configurations q_1 et q_2 est réalisée comme expliqué

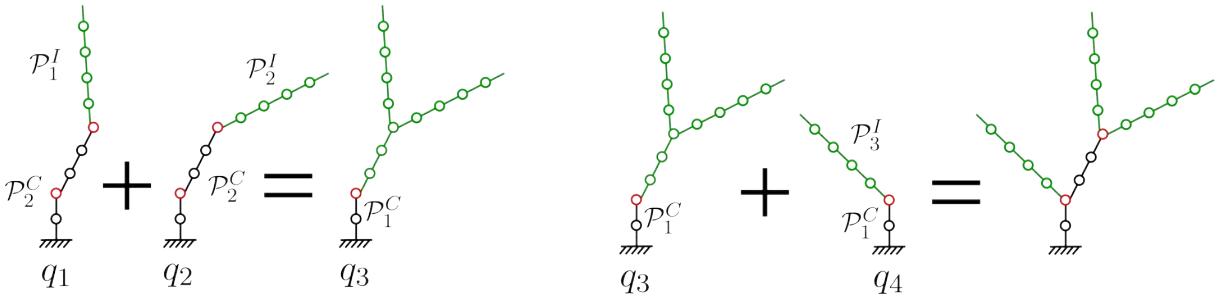

FIGURE 3.9 – Étapes de fusion d'un système arborescent composé de deux parties communes et de trois parties indépendantes. La fusion des configurations q_1 et q_2 est d'abord réalisée. La configuration q_3 obtenue est à son tour fusionnée avec q_4 pour obtenir la configuration finale du système.

dans le paragraphe ci-dessus. Les configurations q_3 et q_4 sont ensuite fusionnées. En effet, ces deux configurations possèdent la même partie commune $\mathcal{P}_1^C$ et peuvent être fusionnés de la même manière que précédemment pour d'obtenir la configuration du système tout entier.

Chaque Nœud de $\mathcal{SG}$ est testé en collision afin d'assurer la compatibilité des configurations des différentes parties. Ce test de collision est partiellement réalisé, étant donné que certaines collisions ont déjà été testées lors de la création des réseaux élémentaires. Si le système ne contient aucune partie commune ou que la configuration des parties communes sont identiques dans tous les nœuds élémentaires ("Cas A et B"), seules les collisions entre les parties indépendantes doivent être testées. Cependant, dans le cas général ("Cas C"), chaque partie indépendante doit être testée avec les autres parties indépendantes, la partie commune, si ce test n'est pas déjà effectué, et dans certains cas les obstacles de l'environnement. En effet, dans ce cas, seule une partie indépendante a été testée en collision avec la partie commune. Toutes les autres parties indépendantes doivent être testées avec elle pour vérifier leur compatibilité. Étant donné la définition d'une partie "commune", une modification de la configuration de cette partie entraine un changement de position des parties indépendantes reliées. Ce changement de position impose d'effectuer un nouveau test de collision de ces parties indépendantes avec les obstacles de l'environnement.

3.4.2 Connexion des Nœuds de Super Graphe

Une fois qu'un nœud X est créé puis ajouté dans le $\mathcal{SG}$, sa connexion avec les autres Nœuds de $\mathcal{SG}$ est testée. Afin de préserver l'efficacité de la construction du réseau, et en raison du coût élevé de l'étape de validation des arêtes, un filtre est utilisé pour sélectionner les nœuds à tester pour les connecter. Ce filtre utilise tout d'abord les informations présentes dans les réseaux élémentaires précédemment calculés. En effet, deux Nœuds de $\mathcal{SG}$ (X et Y) peuvent être connectés si et seulement si les nœuds élémentaires, respectivement x_i et y_i, qui les composent sont eux même connectés dans les réseaux élémentaires. Considérons en exemple un système bi-bras. Le nœud X construit à partir de deux nœuds élémentaires x_1 et x_2 et Y à partir de y_1 et y_2 sont connectables si :

- $x_1 = y_1$ et x_2 et y_2 sont connectés dans $\mathcal{R}_2$
- x_1 et y_1 sont connectés dans $\mathcal{R}_1$ et $x_2 = y_2$
- x_1 et y_1 sont connectés dans $\mathcal{R}_1$ et x_2 et y_2 sont connectés dans $\mathcal{R}_2$

Une autre stratégie applicable pour améliorer la performance de l'algorithme est de construire un $\mathcal{SG}$ en forme d'arbre au lieu d'un graphe. Dans ce cas, les tests de connexion sont seulement effectuées entre les Nœuds de $\mathcal{SG}$ appartenant à différentes composantes connexes de $\mathcal{SG}$. La Figure 3.10 illustre ces deux filtres. Le graphe de la vignette "a" est obtenu à la sortie du premier filtre. Seuls les Nœuds de $\mathcal{SG}$ reliés par des arêtes noires sont connectables. Dans la vignette "b", seules les arêtes en vert sont testées en collision. Les arêtes rouges ne sont pas testées car elle connectent des nœuds faisant partie de la même composante connexe. Le $\mathcal{SG}$ obtenu après l'exécution des deux filtres et du test de collision des arêtes est illustré dans la vignette "c".

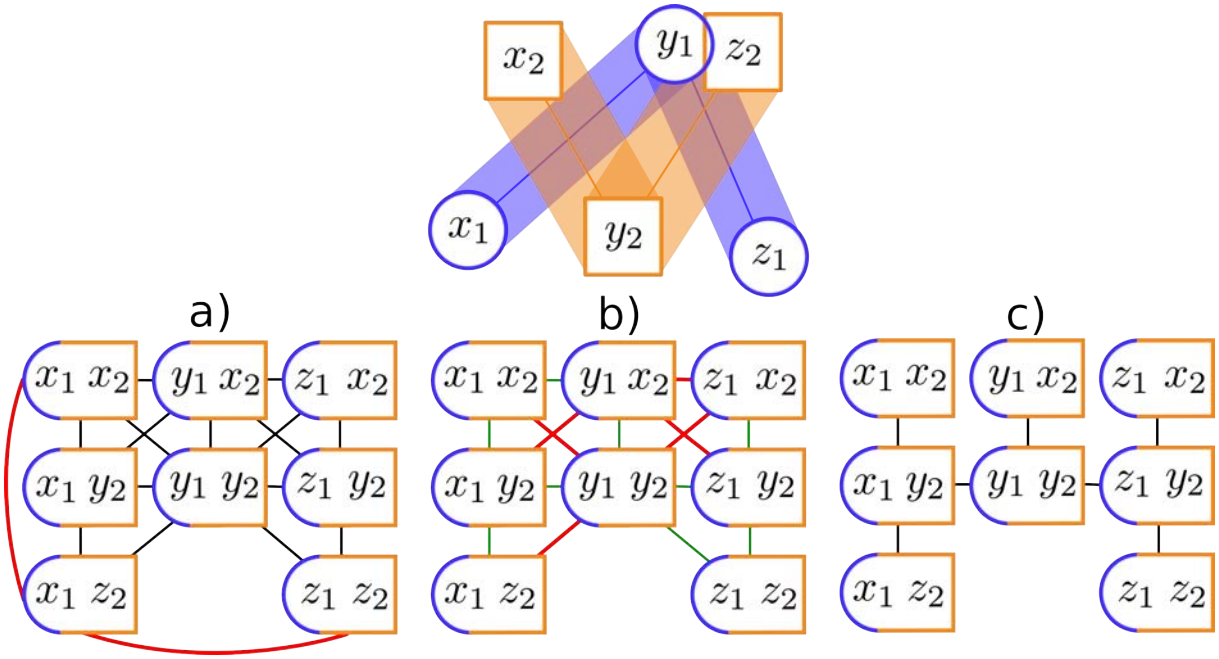

FIGURE 3.10 – Exemple d'exécution, pour un système ayant deux parties indépendantes, des deux filtres utilisés pour améliorer les performances de l'approche présentée. Les réseaux élémentaires sont représentés en haut de la figure. La vignette "a" montre en noir les arêtes après le premier filtrage. Les arêtes rouges ne sont pas testées. La vignette "b" illustre en vert les arêtes testées en collision et en rouge celles rejetée par le second filtre. La vignette "c" montre le $\mathcal{SG}$ résultant de la fusion des deux réseaux élémentaires

Comme pour les Nœuds de $\mathcal{SG}$, la validation des Arêtes de $\mathcal{SG}$ requiert uniquement de tester les collisions entre les différentes parties du système, et entre ces parties et les obstacles de l'environnement qui n'ont pas été testées lors de la génération des arêtes des réseaux élémentaires. Prenons en exemple un système bi-bras. La Figure 3.11 illustre les différents cas possibles. Si le système ne contient pas de partie commune, seule la collision entre les deux parties indépendantes est testée lors de la validation des arêtes ("Cas A"). Si le système possède une partie commune et que les valeurs des articulations de cette partie dans les deux Nœuds de $\mathcal{SG}$ sont identiques, ou que les paires des nœuds élémentaires fusionnés ont les mêmes valeurs pour la partie commune, le test de collision est également uniquement effectué entre les deux parties indépendantes ("Cas B"). Cependant dans le cas général, les tests de collision doivent être effectués intégralement : les parties indépendantes entre elles, ainsi qu'avec la partie commune, en plus du test de toutes les parties du robot avec l'environnement ("Cas C").

Une Arête de $\mathcal{SG}$ n'est ajoutée au *Super Graphe* que si elle est libre de collisions, autrement, les nœuds élémentaires à partir desquelles les extrémités de l'Arête de $\mathcal{SG}$ sont créées, sont marqués pour une utilisation future en cas d'enrichissement du Super Graphe (Voir la Section 3.5). L'algorithme 3.3 montre le pseudo code de la phase de connexion des Nœuds de $\mathcal{SG}$. Les deux filtres utilisés sont facilement identifiables (Lignes 1 et 4). Une fois ces deux étapes passées, une Arête de $\mathcal{SG}$ est créée puis testée en collision.

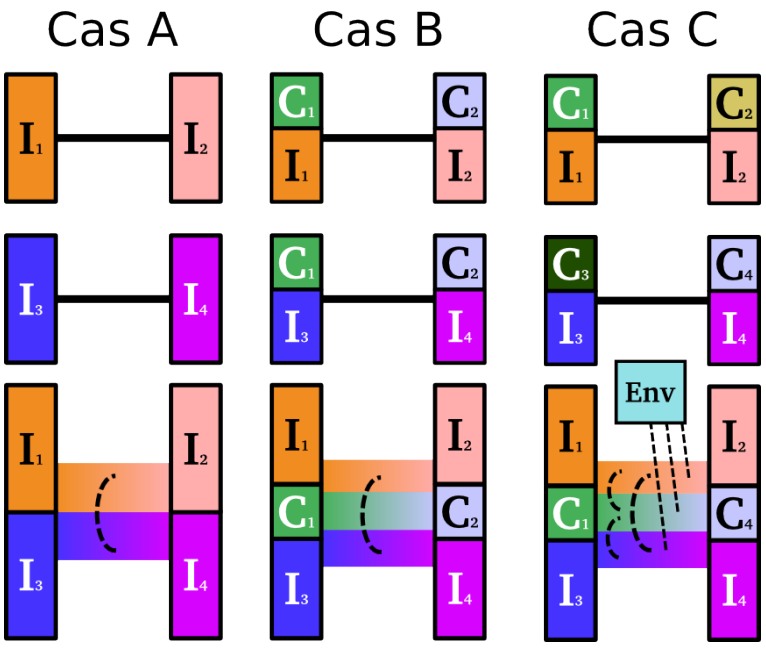

FIGURE 3.11 – Validation des Arêtes de $\mathcal{SG}$. Les tests de collisions sont effectués entre les parties reliées avec les arcs pointillées. Dans le "Cas A et B" seules les collisions entre les parties indépendantes ont besoin d'être testées étant donné que le reste des tests ont déjà été effectués lors de la création des arêtes dans les réseaux élémentaires. Dans le "Cas C" toutes les parties doivent être testées entre elles ainsi qu'avec l'environnement.

3.5 Résolution des requêtes de planification

La résolution des requêtes de planification consiste en l'ajout d'un Nœud de $\mathcal{SG}$ initial S et un autre final G dans le $\mathcal{SG}$ et de rechercher par la suite un chemin dans le réseau créé. Si S et G sont dans la même composante connexe, la requête est directement résolue. Autrement, le *Super Graphe* précalculé peut être enrichi incrémentalement lors de la résolution des requêtes de planification.

Plusieurs étapes sont nécessaires pour enrichir le $\mathcal{SG}$. La première consiste en la décomposition des configurations initiales et finales en configurations élémentaires. Les nœuds résultants de cette décomposition s_i et g_i sont ensuite insérés dans les réseaux élémentaires $\mathcal{R}_i$ précédemment calculés et leur connexion avec les autres nœuds des réseaux élémentaires est testée. Si s_i et g_i n'appartiennent pas à la même composante connexe de $\mathcal{R}_i$, le réseau est enrichi en réitérant la phase de création de nœuds élémentaires présentée Section 3.3. Notons que la résolution de la requête de planification de mouvement peut échouer à ce stade de l'algorithme si une des conditions d'arrêt (*e.g.* nombre limite de nœuds d'un des réseaux élémentaires ou du $\mathcal{SG}$) est atteinte avant que les s_i et g_i ne soient connectés.

Les nœuds insérés, sont ensuite fusionnés avec les autres nœuds des différents réseaux élémentaires et sont ajoutés au *Super Graphe* comme expliqué dans la Section 3.4.1. S et G peuvent néanmoins être déconnectés bien que tous les nœuds élémentaires s_i et g_i soient connectés dans

Algorithm 3.3: ConnectTwoSGNodes

Input: the robot A, the environment E, the Super Graph $\mathcal{SG}$, the $\mathcal{SG}$ Nodes X & Y
Output: *true* if X & Y are connected, $false$ otherwise

for $i \leftarrow 1$ **to** n **do** `// n = number of` $\mathcal{P}^I$
  if $(x_i \neq y_i)$ **or** (**not** AreConnected$(x_i, y_i, \mathcal{R}_i)$) **then**
    return $false$; `// Elementary nodes are not connected. Abort`
if PathExist$(\mathcal{SG}, X, Y)$ **then**
  return $false$; `// There is already a path between` X, Y `in` $\mathcal{SG}$`. Abort`
$Edge \leftarrow$ CreateSGEdge(X, Y);
if IsCollisionFree$(\mathcal{SG}, Edge)$ **then**
  AddEdgeToSG$(\mathcal{SG}, Edge)$;
  return *true*;
return $false$;

leurs réseaux respectifs $\mathcal{R}_i$. Dans ce cas, plusieurs stratégies peuvent être adoptées pour enrichir le $\mathcal{SG}$. La plus basique consiste à réitérer les phases d'enrichissement des réseaux élémentaires et de fusion. Une autre stratégie plus judicieuse consiste à générer des nœuds x_i et à les connecter avec les nœuds de $\mathcal{R}_i$ précédemment marqués lors de la phase de validation des Arêtes de $\mathcal{SG}$. Le but de cette stratégie est de construire des cycles permettant d'éviter les collisions apparues lors de la phase de fusion. L'utilisation des deux stratégies en même temps permet d'explorer l'espace des configurations libres en plus de la création de cycles pour contourner les arêtes invalides à cause des auto-collisions. Dans notre implémentation, cette combinaison est utilisée et réitérée jusqu'à ce qu'un chemin solution soit trouvé, ou qu'une condition d'arrêt soit atteinte (*e.g.* le nombre de nœuds limite du réseau est atteinte).

3.6 Résultats

Cette section présente une analyse de la performance de la méthode proposée sur deux systèmes multi-bras. Le premier système (Figure 3.12) est composé par trois manipulateurs cinématiquement indépendants et le second correspond au modèle du robot Justin (Figure 3.13) qui possède une partie commune (le torse). Le premier objectif de cette analyse est de comparer la performance de l'approche de la décomposition des réseaux proposée avec des méthodes centralisées construisant directement des réseaux probabilistes pour le système tout entier. Le second est d'analyser l'influence de la méthode utilisée pour la construction des réseaux élémentaires sur la performance globale du planificateur.

Pour chaque système, des requêtes de planification de difficulté variable ont été définies. La difficulté des requêtes de planification de mouvement croît en fonction des contraintes imposées par les obstacles de l'environnement et de la difficulté à coordonner les mouvements des manipulateurs. Six variantes d'algorithmes ont été testées pour chaque problème de planification. Trois d'entre elles - PDR, Visib-PRM et PRM - construisent des réseaux pour le système tout entier. Les trois autres - $\mathcal{SG}$ PDR, $\mathcal{SG}$ Visib-PRM et $\mathcal{SG}$ PRM - appliquent la méthode

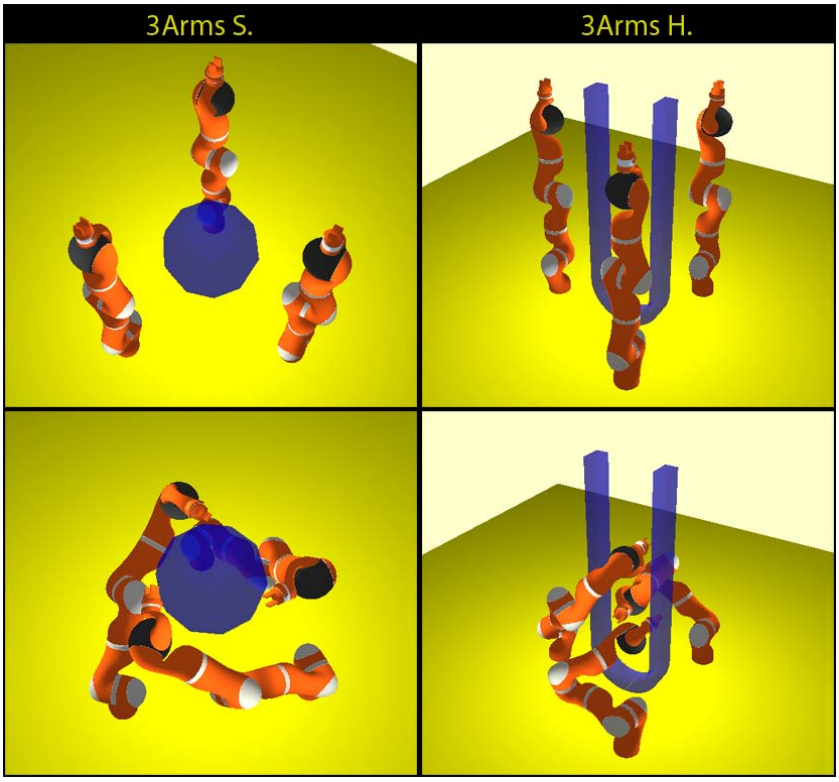

FIGURE 3.12 – Deux problèmes de planification de mouvement pour un système multi-bras composé de trois bras Light Weight Robot III. "3Arms S." représente des problèmes relativement simples tandis que "3Arms H." des problèmes plus complexes. Les configurations initiales sont en haut et finales en bas.

de fusion proposée en utilisant respectivement PDR, Visib-PRM et PRM pour construire les réseaux élémentaires. Les tests consistent en la construction incrémentale des réseaux jusqu'à ce qu'une solution soit trouvée pour une requête donnée. Il est évident que les réseaux capables de résoudre des requêtes d'une difficulté donnée, peuvent également résoudre des requêtes d'un niveau moindre. Tous les algorithmes ont été implantés dans le logiciel de planification de mouvement *Move3D* [Siméon 01]. Les résultats de simulation reportés dans cette section ont été moyennés sur 20 exécutions du planificateur. Les temps de calculs correspondent à un processeur AMD Opteron Dual-Core 2222 cadencé à 3.0 GHz.

La Figure 3.12 illustre un système composé de trois bras manipulateurs "Light Weight Robot III" [Hirzinger 02, Albu-Schäffer 07]. Ce système se décompose en trois parties cinématiquement indépendantes, chacune correspondant à un manipulateur à sept degrés de liberté. Deux problèmes de planification de mouvement sont représentés dans la figure. Dans le problème "3Arms S.", les trois manipulateurs doivent placer leur effecteur terminal sur différents points de la surface de la sphère en partant d'une position de repos où les bras sont droits. Ce mouvement nécessite une simple coordination des bras. Cependant dans le problème "3Arms H.", les trois manipulateurs doivent se coordonner afin d'éviter les auto-collisions lors de leurs mouvements à l'intérieur de l'obstacle en forme de "U".

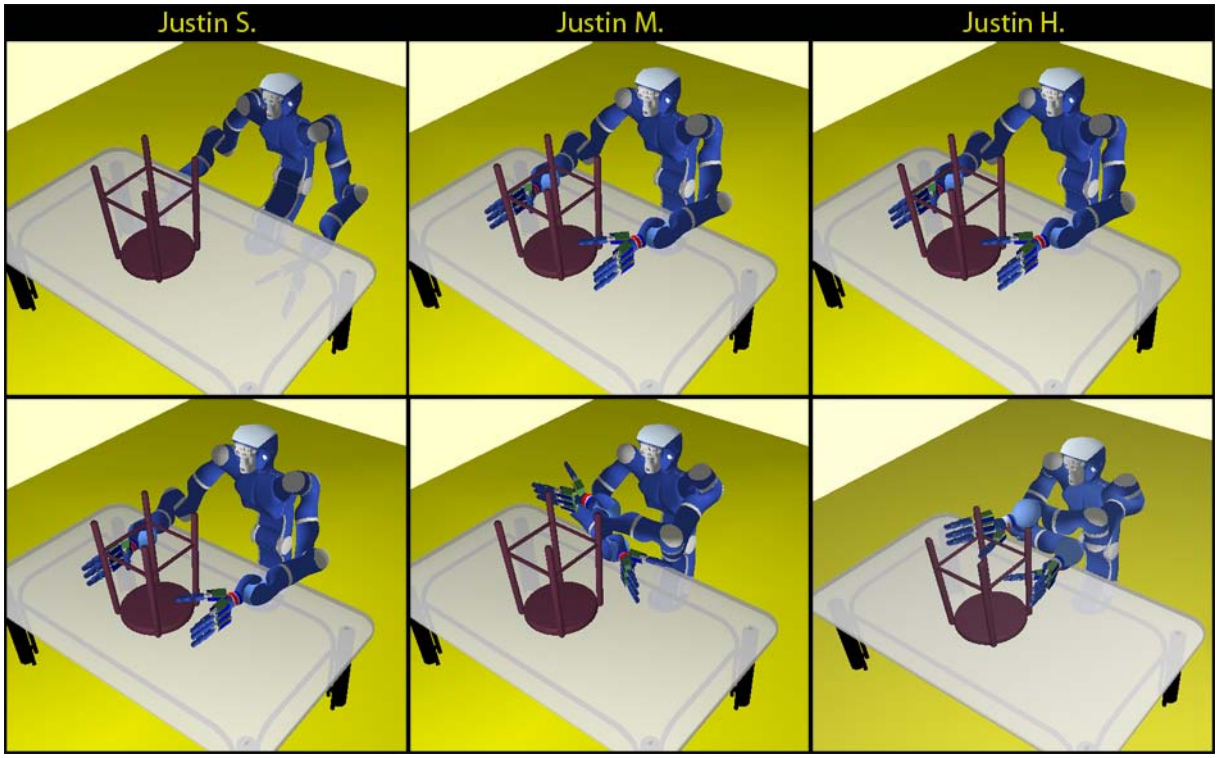

FIGURE 3.13 – Configurations Initiales (haut) et finales (bas) caractérisant trois requêtes de planification de différentes difficultés.

Dans la Figure 3.13 trois requêtes de planification de mouvement du robot Justin [Ott 06] dans le même environnement sont représentées. Le robot est composé de deux bras "DLR-Lightweight-Robot-III" [Hirzinger 02] à sept degrés de liberté (DdL) montés sur un torse à trois DdL. Justin possède également deux mains "DLRHand-II" et une tête montée sur un cou à deux DdL. Dans nos expérimentations, les articulations des mains et du cou sont considérées fixes. Les degrés de liberté de Justin pris en compte par le planificateur sont donc au nombre de 17. Comme expliqué dans la Section 3.1, ce robot peut être décomposé en trois parties : les bras (munis des mains fixes) sont les parties indépendantes et le torse (avec la tête fixe) la partie commune. Dans la requête "Justin S.", le robot doit déplacer ses bras de sous la table à au dessus. Cette requête simple ne requière guère la coordination des deux bras. À l'opposé, la coordination des bras est nécessaire pour résoudre la requête "Justin H." qui correspond à une tâche difficile. En effet, le robot doit atteindre une configuration très contrainte par les obstacles de l'environnement en croisant totalement les bras afin de saisir le tabouret. La requête "Justin M." est un exemple de tâche de complexité moyenne.

Le Tableau 3.1 montre les temps de calcul et le nombre de nœuds requis pour résoudre les cinq requêtes de planification de mouvement avec les six algorithmes testés. Nous pouvons noter que les résultats pour la requête "Justin H.", lors de l'utilisation des algorithmes PDR, Visib-PRM, PRM et $\mathcal{SG}$ PRM, ne sont pas mentionnés étant donné que ces planificateurs sont incapables de trouver un chemin solution en un temps de calcul raisonnable ($<$ 2 heures). Comme on

TABLE 3.1 – Résultats numériques

Problème	PDR		$\mathcal{SG}$ PDR		Vis-PRM		$\mathcal{SG}$ Vis-PRM	
	$n_{\text{nœuds}}$	T (sec)	$n_{\text{nœuds}}$	T (sec)	$n_{\text{nœuds}}$	T (sec)	$n_{\text{nœuds}}$	T (sec)
3Arms S.	31	13.3	44	2.6	30	3.2	24	0.4
3Arms H.	247	165.8	174.8	11.4	229	130.7	63	2.4
Justin S.	82	23.6	118	2.6	72	18.4	99	1.0
Justin M.	161	112.14	258	4.3	158	90.9	223	1.9
Justin H.	–	$>$ 2h	7269	173.8	–	$>$ 2h	6494	101.1

Problème	PRM		$\mathcal{SG}$ PRM	
	$n_{\text{nœuds}}$	T (sec)	$n_{\text{nœuds}}$	T (sec)
3Arms S.	180	2.6	44	0.5
3Arms H.	20512	701	2415	17.8
Justin S.	1624	14.9	1719	10.7
Justin M.	8673	202.4	32277	2082
Justin H.	–	$>$ 2h	–	$>$ 2h

peut noter dans la Figure 3.14, représentant le gain de performance, les planificateurs basés sur la fusion des réseaux sont jusqu'à soixante-dix fois plus rapides que les méthodes centralisées. Ce gain est moindre pour les exemples simples que pour les exemples complexes nécessitant une coordination des manipulateurs. La tendance est uniquement inversée lors de la résolution de la requête "Justin M." en utilisant l'algorithme basique PRM pour construire les réseaux indépendants. La raison de cette contre-performance est due au temps conséquent passé par la méthode $\mathcal{SG}$ PRM à fusionner les réseaux élémentaires contenant un grand nombre de nœuds. Ceci peut également être remarqué dans le Tableau 3.2, qui présente une décomposition du temps de calcul global en temps nécessaire pour chaque étape de l'algorithme présenté : la construction des réseaux indépendants et leur fusion.

TABLE 3.2 – Répartition du temps de calcul pour la construction du *Super Graphe* (sec)

Problème	$\mathcal{SG}$ PDR		$\mathcal{SG}$ Vis-PRM		$\mathcal{SG}$ PRM	
	Élem	Fusion	Élem	Fusion	Élem	Fusion
3Arms S.	2.3	0.25	0.4	0.01	0.08	0.4
3Arms H.	10.3	1.1	2.3	0.02	0.7	17.1
Justin S.	2.2	0.4	0.6	0.4	0.3	10.4
Justin M.	3.4	0.9	1.0	0.9	2.3	2080
Justin H.	94.8	79	44.8	56.3	–	–

Finalement, on note que la performance de $\mathcal{SG}$ PDR et $\mathcal{SG}$ Visib-PRM est similaire. L'algorithme $\mathcal{SG}$ Visib-PRM est un peu plus performant du point de vue du temps de calcul. Les résultats du Tableau 3.2 montrent que le temps passé à calculer les réseaux élémentaires à l'aide de la méthode PDR est la raison expliquant la performance moindre de $\mathcal{SG}$ PDR. En contre

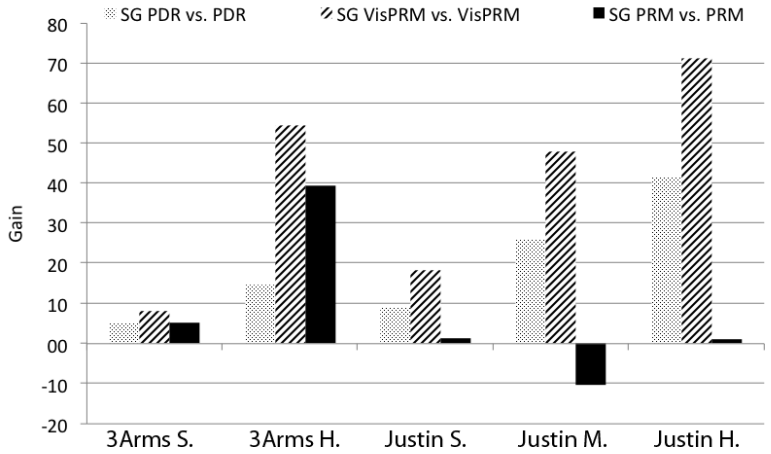

FIGURE 3.14 – Gain de performance des méthodes de fusion de réseau ($\mathcal{SG}$) comparé aux méthodes centralisées.

partie l'utilisation de PDR permet de calculer des réseaux probabilistes de meilleure qualité contenant des cycles utiles pour la phase de fusion. De plus, les chemins solutions générés par $\mathcal{SG}$ PDR sont généralement plus courts.

3.7 Conclusion

Nous avons présenté dans ce chapitre une méthode de planification de mouvement pour des systèmes multi-bras évoluant dans des environnements contraints. Elle est basée sur la décomposition du système en parties cinématiquement indépendantes, traitées comme des robots individuels dans une approche multi-robots de fusion de réseaux. Cette approche est particulièrement efficace pour des tâches de manipulation à effectuer à l'aide de plusieurs manipulateurs, ou d'un système multi-bras avec une partie commune ayant un faible nombre de degrés de liberté. Dans le cas où la partie commune du système possède un grand nombre de degrés de libertés, le bénéfice de la construction du réseau en deux temps se trouve fortement réduit vu que l'étape de fusion des réseaux sera plus coûteuse en temps de calcul. Cependant, la méthode reste générale et peut directement être appliquée à des systèmes multi-bras complexes. Les résultats présentés montrent le gain calculatoire de la méthode.

Une des améliorations possible de l'algorithme est de valider les Arêtes de $\mathcal{SG}$ le plus tard possible, cette validation étant la phase la plus coûteuse en temps de calcul lors de la fusion des réseaux élémentaires. Le *Super Graphe* peut être construit en suivant la méthode proposée, mais en différant la phase de validation des arêtes, comme dans l'algorithme Lazy-PRM [Bohlin 00].

4

Planificateur pour la manipulation multi-bras

Ce chapitre traite d'une extension de la méthode de planification de mouvement proposée par Cortés et al. [Cortés 04a] (voir Chapitre 2) pour les systèmes comportant des chaînes cinématiques fermées en y ajoutant le traitement explicite des configurations singulières. L'intérêt de la méthode est de permettre la génération de chemins nécessitant la reconfiguration du robot. La Figure 4.1, illustre deux problèmes de planification de mouvement, pour deux manipulateurs plans à trois axes tenant un même objet. Le "Problème A" peut être résolu sans la reconfiguration d'aucun des deux bras. Cependant, la présence de l'obstacle en bas à gauche du "Problème B" force la reconfiguration des deux bras en faisant passer la seconde articulation (coude) d'une configuration "basse" à une configuration "haute". Nous nous proposons dans ce chapitre de résoudre ce type de problèmes à l'aide d'une méthode qui étend les algorithmes de type PRM et RRT. Les résultats présentés en fin de chapitre montrent la capacité de la méthode à résoudre des problèmes complexes de manipulation coordonnée.

4.1 Formulation et paramétrisation

4.1.1 Structure de l'espace des configurations

La manipulation coordonnée d'objet nécessite le calcul du mouvement simultané de plusieurs robots. Ces prises multiples induisent des contraintes modélisables comme des contraintes de fermeture cinématique. Lorsqu'un mécanisme articulé contient des chaînes cinématiques fermées, les valeurs des articulations constituant ces boucles sont reliées par des *équations de fermeture cinématique* ayant la forme $f(\mathbf{q}) = 0$ ($\mathbf{q}$ représentant l'ensemble des valeurs des articulations formant la boucle cinématique). Les contraintes de fermeture cinématique sont un cas particulier

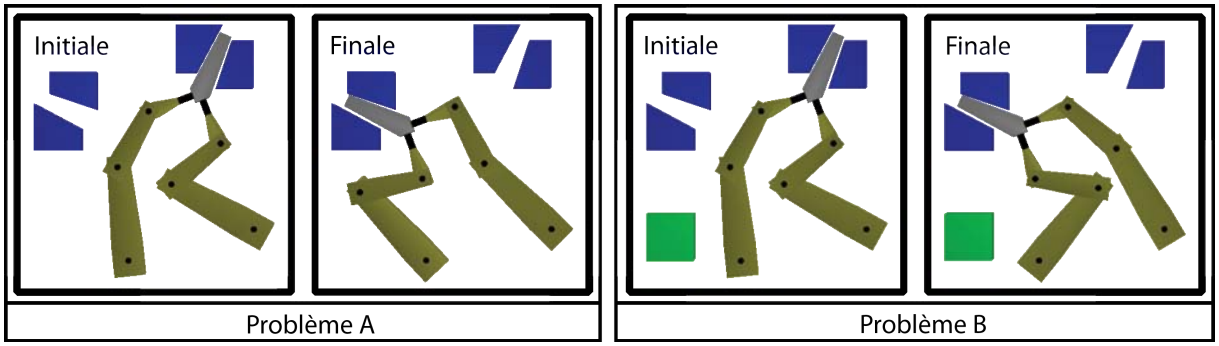

FIGURE 4.1 – Exemple académique d'un problème de planification de mouvement de manipulation coordonné. La solution du problème B nécessite la reconfiguration des deux manipulateurs.

de contraintes holonomes rendant le problème de planification de mouvement plus complexe. Dans le cas général, $\mathcal{C}$ est composé de sous-variétés $\mathcal{M}_i$ disjointes ayant une structure complexe. De plus, ces sous variétés peuvent être connectées par des ensembles $\mathcal{S}_k$ de dimension plus petite [Cortés 04a].

Considérons l'exemple fictif avec trois variables articulaires $\{\theta_1, \theta_2, \theta_3\}$ présenté Figure 4.2. Considérons également une équation de la forme de $f(\theta_1, \theta_2, \theta_3) = 0$, représentant une contrainte de fermeture cinématique. Cette fonction induit des sous-variétés $\mathcal{M}_i$, dans l'espace articulaire $\mathcal{Q}$. Dans cet exemple, $\mathcal{M}_1$ et $\mathcal{M}_2$ sont connectées par un ensemble de dimension plus petite $\mathcal{S}$. Dans cette figure, les zones $\mathcal{Q}_{obst}$ représentent des obstacles dans l'espace articulaire du système. $\mathcal{C}_{free}$ est donc l'intersection de $\mathcal{Q}_{free}$ avec les différentes sous variétés $\mathcal{M}_i$.

Dans cet exemple, plusieurs situations peuvent être rencontrées lors de l'exécution des requêtes de planification. La première est une requête entre la configuration q_1 et q_2. Ces deux configurations sont sur la même sous-variété $\mathcal{M}_2$ et dans la même composante connexe de $\mathcal{C}_{free}$: il y a donc un chemin entre elles. De même, un chemin existe aussi entre q_1 et q_3 et entre q_1 et q_4 même si ce chemin contient des configurations singulières. Au contraire les configurations q_4 et q_5 ne peuvent pas être reliées à cause de la présence d'obstacles. Un dernier cas, n'apparaissant pas pour des chaînes cinématiques ouvertes, peut apparaître quand le système est soumis à des contraintes de fermeture cinématique. Par exemple, q_1 et q_4 appartiennent à la même composante connexe de $\mathcal{Q}_{free}$ mais à différentes composantes de $\mathcal{C}$!

4.1.2 Cinématique inverse

Les équations de cinématique directe établissent la relation entre les paramètres articulaires et la position et l'orientation de l'organe terminal. Le problème de *"cinématique inverse"* ($\mathcal{IK}$) consiste en la résolution d'équations qui en fonction de la position et l'orientation de l'organe terminal, déterminent les paramètres articulaires du manipulateur. La solution à ce problème, pour un manipulateur série (*i.e.* un bras), est fondamentale si l'on veut transformer le mouvement de l'organe terminal dans l'espace de travail du robot $\mathcal{W}$ en valeurs articulaires pour exécuter ce mouvement. Étant donnée une valeur articulaire pour chacune des articulations constituant la

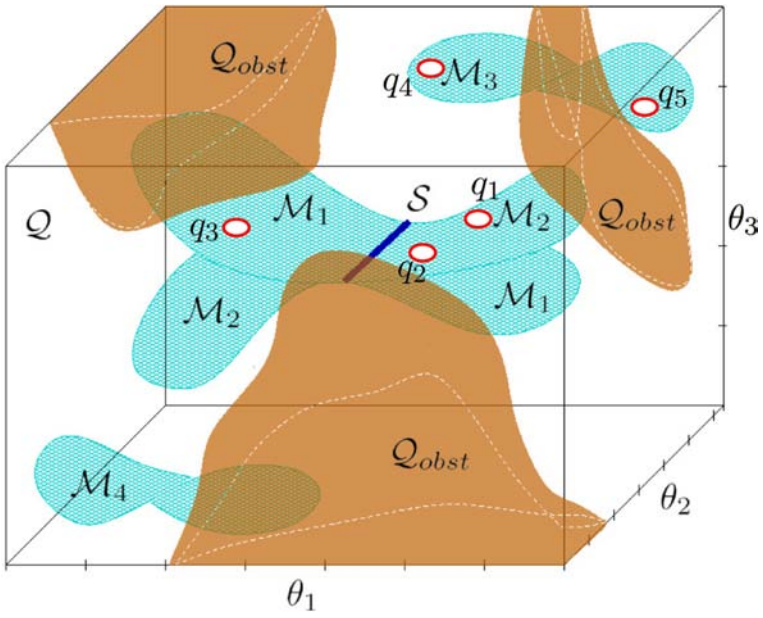

FIGURE 4.2 – Illustration de l'espace des configurations d'un mécanisme ayant trois valeurs articulaires soumises à des contraintes de fermeture cinématique. Les configurations valides par rapport à ces contraintes se trouvent sur les surfaces $\mathcal{M}_i$ représentant les sous variétés de l'espace. L'intersection, $\mathcal{S}$, de $\mathcal{M}_1$ et $\mathcal{M}_2$ représente un sous ensemble de configurations singulières.

chaîne cinématique du manipulateur, la position et l'orientation de l'organe terminal calculées à l'aide des équations de cinématique directe sont uniques. Cependant, le problème de cinématique inverse est plus complexe pour les raisons suivantes :

- Les équations à résoudre sont généralement non linéaires.
- Plusieurs classes de solutions peuvent exister (épaule avant / arrière, coude bas / haut ...).
- Un nombre infini de solutions peuvent exister (cas d'un manipulateur redondant).

Le nombre de solutions des équations d'$\mathcal{IK}$ est directement lié au nombre de degrés de liberté du manipulateur. En effet, un manipulateur trois axes plan, suivant ses limites articulaires, peut admettre jusqu'à deux solutions. Pour un manipulateur six axes de topologie et géométrie arbitraires jusqu'à seize solutions sont possibles. Le nombre de solutions de cinématique inverse est directement lié au nombre de sous-variétés de l'espace des configurations. Les ensembles de configurations dites singulières, qui composent les sous ensembles de dimension plus petite, permettent de passer d'une sous variété à une autre ce qui implique un changement de classe de solution de la cinématique inverse [Cortés 04a].

4.1.3 Configurations singulières

Une configuration singulière est une configuration où la mobilité de l'organe terminal est diminuée. Ce dernier ne peut plus se mouvoir ou tourner autour de certains axes. Deux types de singularités existent : des singularités de l'espace de travail et des singularités de l'espace articulaire. Les premières sont les plus communes et sont souvent causées par l'extension maxi-

male d'une des articulations. Les secondes surviennent quand certains axes du manipulateur sont alignés dans l'espace de travail, ou dans le cas d'alignements plus spécifiques à la topologie du manipulateur utilisé. Souvent, lorsqu'un manipulateur est dans une configuration singulière, une infinité de solutions aux équations de cinématique inverse existent.

Le domaine de la cinématique des robots, en particulier celui de l'étude des singularités est très complexe [Park 98]. On se limite à des notions basiques dans le contexte des travaux présentés dans ce chapitre.

4.1.4 Décomposition du système

Les systèmes considérés dans ce chapitre sont constitués d'un ensemble de manipulateurs maintenant un objet mobile. La liaison entre l'objet mobile et l'organe terminal peut être considérée comme fixe (ils forment un seul objet rigide) ou comme une articulation si certains mouvements entre l'objet et l'organe terminal sont autorisés. Les explications qui suivent sont limitées au cas d'une liaison rigide. Cependant, l'approche proposée reste générale.

Pour faciliter l'explication de l'approche, nous considérons le cas d'un système composé de deux manipulateurs. Par exemple, la Figure 4.3 illustre deux manipulateurs plans trois axes maintenant un objet. Le système ainsi composé (les deux bras plus la liaison au sol) peut être considéré comme une seule boucle cinématique (figure gauche). La structure du système peut également être vue sous la forme de deux boucles cinématiques (figure droite). Nous utilisons dans la suite la seconde décomposition qui permet la mise en œuvre d'une métrique plus simple dans les algorithmes de planification de mouvement.

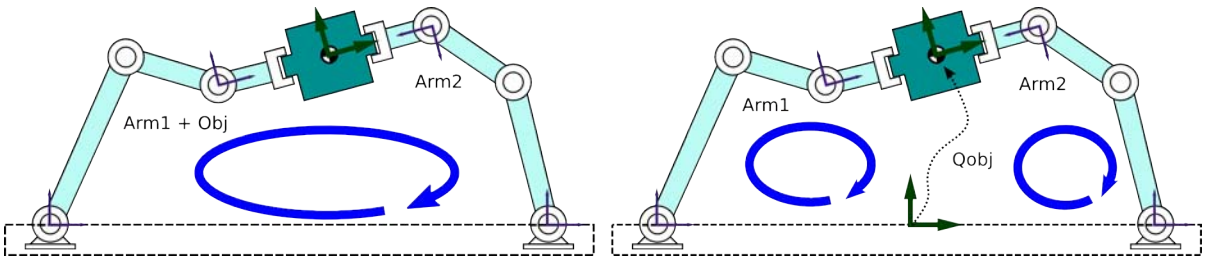

FIGURE 4.3 – Deux bras manipulateurs maintenant un objet. Le système représenté peut être considéré comme ne formant qu'une seule boucle cinématique (gauche), ou deux boucles cinématiques (droite)

La configuration $\mathbf{q}$ du système est donc définie par la position et l'orientation de l'objet manipulé, en plus de la configuration de chacun des deux manipulateurs : $\mathbf{q} = \{\mathbf{q}_{\text{Obj}}, \mathbf{q}_{\text{Arm1}}, \mathbf{q}_{\text{Arm2}}\}$. Ces paramètres sont les entrées des fonctions de cinématique inverse et une partie d'entre eux (paramètres dépendants), dans notre exemple $\mathbf{q}_{\text{Arm1}}$ et $\mathbf{q}_{\text{Arm2}}$, dépendront de la valeur des autres, $\mathbf{q}_{\text{Obj}}$. Dans le cas de mécanismes plans, le nombre de paramètres indépendants maximal pour une boucle cinématique simple est de trois. Si le système évolue dans l'espace, ce nombre est de six. En suivant la nomenclature proposée dans [Han 01, Cortés 04a] nous nommerons respectivement les variables dépendantes et indépendantes variables *passives* et *actives*. Les termes

actif et *passif* indiquent leur rôle dans l'approche expliquée ci-dessous.

En suivant la décomposition du système proposée, les paramètres actifs sont associés à la position et l'orientation ($\mathbf{q}_{\mathrm{Obj}}$) de l'objet manipulé. Si les manipulateurs ne sont pas redondants, leurs variables articulaires sont les variables passives du problème. Dans le cas contraire, en plus des paramètres de l'objet, les variables correspondant aux articulations redondantes sont considérées comme actives. On peut donc écrire :

$$\mathbf{q}^{\mathrm{a}} = \{\mathbf{q}_{\mathrm{Obj}}, \mathbf{q}^{\mathrm{a}}_{\mathrm{Arm1}}, \mathbf{q}^{\mathrm{a}}_{\mathrm{Arm2}}\} \ , \ \mathbf{q}^{\mathrm{p}} = \{\mathbf{q}^{\mathrm{p}}_{\mathrm{Arm1}}, \mathbf{q}^{\mathrm{p}}_{\mathrm{Arm2}}\}$$

Le calcul de $\mathbf{q}^{\mathrm{p}}$ à partir de $\mathbf{q}^{\mathrm{a}}$ requiert la résolution de la cinématique inverse pour chaque manipulateur, que l'on notera $\mathcal{IK}_{\mathrm{Arm1}}$ et $\mathcal{IK}_{\mathrm{Arm2}}$. Les travaux présentés ci dessous considèrent qu'une résolution analytique de la cinématique inverse des deux manipulateurs est disponible et que les différentes classes de solutions de la cinématique inverse (*e.g.* épaule avant / arrière, coude haut / bas, ...) peuvent être caractérisées. Nous noterons respectivement $\mathbf{q}^{\mathrm{p}}_{\mathrm{Arm1,\,i}}$ et $\mathbf{q}^{\mathrm{p}}_{\mathrm{Arm2,\,j}}$ les différentes solutions de $\mathcal{IK}_{\mathrm{Arm1}}$ et $\mathcal{IK}_{\mathrm{Arm2}}$. Les valeurs articulaires permettant de positionner les manipulateurs en configurations singulières sont également supposées connues. On notera $\mathcal{S}_{\mathrm{i,\,i'}}$ l'ensemble des configurations singulières reliant deux classes de solutions i et i' de la cinématique inverse d'un des deux bras.

4.2 Approche basée sur les réseaux probabilistes

La méthode proposée pour le calcul de mouvements de manipulation coordonnée utilise l'extension de la méthode [Han 01, Cortés 04a] basée sur les méthodes PRM pour les mécanismes à chaînes cinématique fermées. Notre contribution est le traitement explicite des configurations singulières, permettant de trouver des chemins solutions lorsque la reconfiguration du robot est nécessaire.

L'extension proposée pour la construction du réseau probabiliste comporte deux étapes. Dans un premier temps, des graphes *"parallèles"* sont construits sur les différentes *couches* de $\mathcal{C}$ (voir Section 4.2.1). En considérant la paramétrisation décrite dans la section précédente, chaque couche correspond à une combinaison unique d'une classe de solutions de la cinématique inverse de chacun des manipulateurs. Le nombre de couches dépend alors du nombre maximal de solutions de chacun des problèmes de cinématique inverse. Si la cinématique inverse du bras $Arm1$ et du bras $Arm2$ ont un nombre maximal de solutions, respectivement, n_{Arm1} et n_{Arm2}, le nombre de couches est $n_{\mathrm{Arm1}} \times n_{\mathrm{Arm2}}$. On notera $G_{\mathrm{i,j}}$ le graphe construit sur chacune des couches, avec $i \in 1 \ldots n_{\mathrm{Arm1}}$ et $j \in 1 \ldots n_{\mathrm{Arm2}}$. La seconde phase consiste à connecter tous les réseaux générés $G_{\mathrm{i,j}}$ à l'aide de configurations singulières (voir Section 4.2.2).

Une fois les réseaux multi-couches construits, la résolution des requêtes de planification consiste en la recherche de sous chemins dans ces couches afin d'obtenir des solutions qui requièrent la reconfiguration de l'un ou des deux manipulateurs. Cette étape est détaillée Section 4.2.3.

La Figure 4.4 illustre cette approche. Elle représente des réseaux probabilistes construits sur trois couches. Deux ensembles de configurations singulières sont représentés par les plans verti-

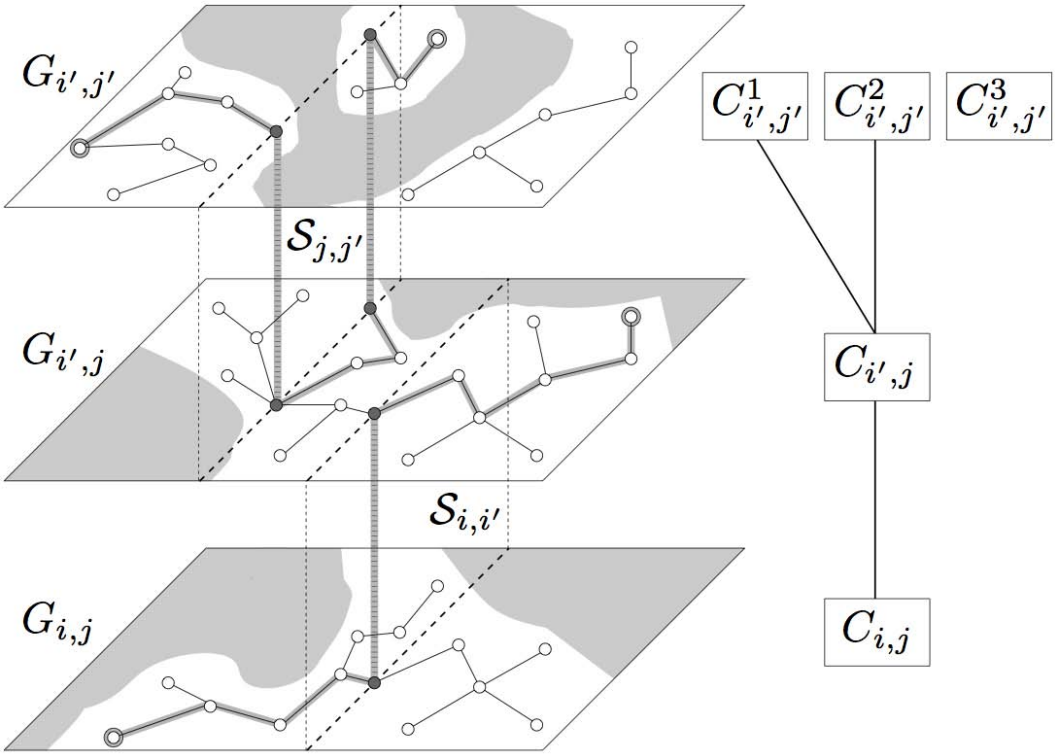

FIGURE 4.4 – Les graphes sont construits sur trois couches différentes correspondant aux sous variétés de $\mathcal{C}$. Ces graphes "parallèles" peuvent être connectés via des nœuds singuliers (gris foncé), illustrés à l'aide de plan perpendiculaires. Le graphe à gauche de la figure représente la connectivité des différentes composantes connexes de chaque couche via les configurations singulières.

caux intersectant les différentes couches. Le graphe sur la partie gauche de la figure, montre les connections entres les différentes composantes connexes de chaque couche via les configurations singulières.

4.2.1 Construction des couches de réseaux probabilistes

En suivant les principes décrits dans [Han 01, Cortés 04a], les couches $G_{i,j}$ sont construites en agissant sur les variables actives $\mathbf{q}^{\mathrm{a}}$ et en résolvant les problèmes de cinématique inverse pour obtenir les valeurs des variables passives $\mathbf{q}^{\mathrm{p}}$. Le détail de la génération des noeuds et des arêtes est donné ci-dessous.

Les nœuds

Les variables actives $\mathbf{q}^{\mathrm{a}}$, correspondant à la position et à l'orientation de l'objet mobile ainsi qu'aux degrés redondants de chaque bras, sont échantillonnées en utilisant l'extension de l'algorithme RLG décrit dans [Cortés 04a]. Au lieu d'utiliser un échantillonnage aléatoire uniforme, cet algorithme permet un guidage de l'échantillonnage des variables de $\mathbf{q}^{\mathrm{a}}$ dans des

régions possédant une grande probabilité de trouver des solutions aux problèmes de cinématique inverse pour les variables passives $\mathbf{q}^{\mathrm{p}}$. Une configuration peut être obtenue sur chaque couche $G_{i,j}$ en combinant les valeurs des paramètres actifs avec les différentes solutions de cinématique inverse : $\mathbf{q}_{i,j} = \{\mathbf{q}^{\mathrm{a}}, \mathbf{q}^{\mathrm{p}}_{i,j}\}$, avec $\mathbf{q}^{\mathrm{p}}_{i,j} = \{\mathbf{q}^{\mathrm{p}}_{\mathrm{Arm1},i}, \mathbf{q}^{\mathrm{p}}_{\mathrm{Arm2,j}}\}$.

Deux stratégies sont envisageables pour insérer des noeuds dans les réseaux multi-couches. La première est de générer systématiquement un noeud dans chacune des couches si la configuration correspondante est sans collision (variante parallèle). La seconde est de sélectionner (aléatoirement) une couche et de ne générer un noeud que dans cette dernière (variante une-par-une). Les résultats des deux stratégies sont discutés dans la Section 4.5.

Les arêtes

Les connexions sont testées entre les nœuds voisins dans chaque couche à l'aide d'un planificateur local agissant uniquement sur les variables actives. En l'absence de contraintes spécifiques sur les mouvements des manipulateurs, une interpolation linéaire peut être utilisée par le planificateur local. Le but de cette étape est de valider que la connexion entre les deux nœuds est libre de collision et que les contraintes de fermeture cinématique sont respectées tout le long de l'arête les reliant. Le chemin local créé entre les deux configurations $x_{i,j}$ et $y_{i,j}$ à connecter est discrétisé suivant un pas donné.

À chaque pas, la configuration intermédiaire de la partie active q^a du système est déterminée par interpolation linaire entre x^a et y^a. La configuration des parties passives $q^p_{i,j}$ sont obtenues par le calcul de la cinématique inverse des deux manipulateurs en sélectionnant les mêmes solutions que celle de $x_{i,j}$ et $y_{i,j}$ (*i.e.* la solution i pour le bras $Arm1$ et la solution j pour le bras $Arm2$). La configuration intermédiaire $q_{i,j}$ ainsi obtenue est testée en collision. Si $q_{i,j}$ respecte les contraintes de fermeture cinématique et est libre de collision, la configuration intermédiaire suivante est testée. Autrement, la connexion entre $x_{i,j}$ et yi, j ne peut être établie.

L'algorithme de construction des couches

L'algorithme 4.1 est le pseudo code de la phase de construction des différentes couches de réseaux probabilistes. Suivant la stratégie souhaitée, une ou plusieurs configurations aléatoires sont générées à l'aide de l'algorithme RLG. Si ces configurations sont valides vis à vis des contraintes internes du robot et sans collision, elles sont insérées puis reliées dans le graphe correspondant aux classes cinématiques sélectionnées. L'algorithme 4.2 détaille l'insertion et la connexion des nouvelles configurations à l'aide d'une méthode PRM. Les voisins les plus proches de la configuration générée appartiennent exclusivement à la même couche. Si une connexion avec l'un des voisins est également possible (*i.e. le chemin local est valide*), l'arête créée appartient à cette couche.

Notons que la plupart des heuristiques d'insertion et de connexion de nœuds proposées dans la littérature (*e.g.* [Kavraki 96, Amato 98, Hsu 03]) peuvent être utilisées pour accroître la performance du planificateur. Également, les algorithmes Visib-PRM [Siméon 00] ou PDR [Jaillet 08a]

Algorithm 4.1: PRM_LayersConstruction

Input: the robot R, the environment E
Output: the multi-layer roadmap G

```
for i ← 1 to n_Arm1 do
    for j ← 1 to n_Arm2 do
        G_i,j ← EmptyRoadmap;
while not StopCondition(G) do
    if Strategy = Parallel then                                   // parallel
        q[nSol] ← RLG_Sample(R, E);
        for i ← 1 to n_Arm1 do
            for j ← 1 to n_Arm2 do
                if ValidAndCollisionFreeConfig(R, E, q_I,J) then
                    PRM_InsertAndConnectConfig (R, E, G_i,j, q_i,j);
    else                                                          // one-by-one
        i ← RandomIkClass(Arm1);
        j ← RandomIkClass(Arm2);
        q_i,j ← RLG_Sample(R, E, i, j);
        if ValidAndCollisionFreeConfig(R, E, q_I,J) then
            PRM_InsertAndConnectConfig (R, E, G_i,j, q_i,j);
```

peuvent être utilisés pour générer des réseaux probabilistes compacts, respectivement avec ou sans cycles.

4.2.2 Connexion des différentes couches

La connexion des différentes couches du graphe nécessite un changement de classes de solutions de la cinématique inverse d'au moins un des deux bras. Suivant la paramétrisation introduite dans la Section 4.1, cette reconfiguration nécessite de passer à travers une configuration singulière du manipulateur. Cette stratégie de connexion entre couches ne considère pas la reconfiguration simultanée des deux manipulateurs. Cependant, ceci n'est pas une limitation pour la complétude (probabiliste) de la méthode. En effet un mouvement du robot nécessitant la

Algorithm 4.2: PRM_InsertAndConnectConfig

Input: the robot R, the environment E, the layer $G_{i,j}$, the config $q_{i,j}^{new}$
Output: the layer $G_{i,j}$ updated

```
AddNewNode(q_i,j^new, G_i,j);
L_best ← ListBestNodes(q_I,J^new, G_I,J);
forall q_i,j ∈ L_best do
    if not InSameComponent(q_i,j^new, q_i,j, G_i,j) then
        if FeasibleConnection(q_i,j^new, q_i,j) then
            AddNewEdge(q_i,j^new, q_i,j, G_i,j);
```

reconfiguration des deux manipulateurs simultanément peut être décomposé en mouvements nécessitant la reconfiguration d'un seul manipulateur à la fois. Pour illustrer la génération des noeuds dans les sous-variétés en utilisant les configurations singulières, prenons pour exemple la sous-variété $\mathcal{S}_{i,i'}$ (voir Figure 4.4), qui connecte les deux couches $G_{i,j}$ et $G_{i',j}$. Chaque configuration dans cet ensemble à une valeur fixe supposée connue pour certaines articulations du manipulateur ARM1.

Pour générer ces configurations singulières, il est donc nécessaire de fixer certaines articulations du manipulateur à des valeurs constantes et la position et l'orientation de l'objet mobile $\mathbf{q}_{\text{Obj}}$ tenu par ARM1 sont obtenues par cinématique directe. Les autres articulations du manipulateur sont échantillonnées à l'aide de l'algorithme RLG (voir Chapitre 2) afin de fermer la chaine cinématique. Une fois $\mathbf{q}_{\text{Obj}}$ calculée, $\mathbf{q}^{\text{a}}_{\text{Arm2}}$ (cas d'un manipulateur redondant) est échantillonnée à l'aide de l'algorithme RLG, et $\mathbf{q}^{\text{p}}_{\text{Arm2}}$ est obtenue par la résolution de la cinématique inverse correspondante. Les différentes étapes d'échantillonnage d'une configuration singulière sont illustrées Figure 4.5. Les configurations singulières sans collision générées, sont insérées dans le graphe puis connectées à leurs plus proches voisins dans les couches $G_{i,j}$ et $G_{i',j}$.

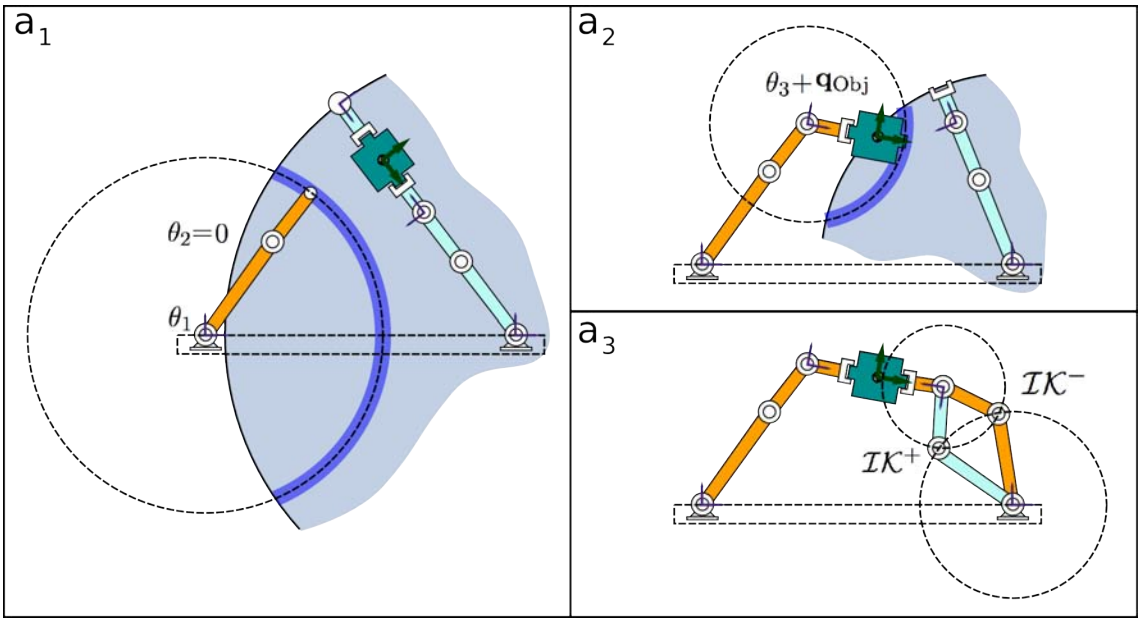

FIGURE 4.5 – Étapes d'échantillonnage d'une configuration singulière. La valeur de l'articulation θ_2 est fixée à 0. Les valeurs θ_1 et θ_3 sont échantillonnées à l'aide de l'algorithme RLG. La position ainsi que l'orientation de l'objet saisi sont calculées en utilisant la cinématique directe du bras gauche. Le reste des valeurs de la boucle cinématique sont calculées par résolution des équations de cinématique inverse du bras droit.

Un graphe est construit pour représenter, à plus haut niveau, les connexions possibles entre les différentes couches du réseau probabiliste global. Les noeuds de ce graphe correspondent aux composantes connexes de chaque couche et les arêtes aux configurations singulières connectant

ces composantes connexes entre elles (voir Figure 4.4 pour une illustration).

L'utilisation de la cinématique inverse pour fermer la chaîne cinématique, implique l'existence éventuelle de plusieurs solutions pour le second manipulateur. De ce fait, les configurations singulières sans collision générées relieront plusieurs paires de couches. En reprenant notre exemple, $\mathcal{S}_{i,i'}$ relie $G_{i,j}$ et $G_{i',j}$ si la classe de solution de la cinématique inverse de ARM2 est j. De même, $\mathcal{S}_{i,i'}$ relie $G_{i,j'}$ et $G_{i',j'}$ si la classe de solution de la cinématique inverse de ARM2 est j'. Suivant la stratégie sélectionnée par l'utilisateur ("parallèle" ou "une-par-une"), une ou toutes les configurations singulières générées seront insérées dans le graphe.

4.2.3 Résolution des requêtes de planification

Les requêtes de planification peuvent être spécifiées de deux façons : la première consiste à chercher un chemin solution d'une configuration initiale du système $\mathbf{q}_{\text{init}}$ à une configuration finale $\mathbf{q}_{\text{fin}}$. Dans ce cas, $\mathbf{q}_{\text{init}}$ et $\mathbf{q}_{\text{fin}}$ sont ajoutées aux couches correspondantes. La seconde ne considère que les configurations initiale et finale $\mathbf{q}^{\text{a}}_{\text{init}}$ et $\mathbf{q}^{\text{a}}_{\text{fin}}$ de l'objet à manipuler (*partie active*). Comme pour la génération des nœuds des différentes couches avec la méthode "parallèle", les configurations initiale et finale du système entier sont obtenues par résolution de la cinématique inverse des manipulateurs en testant toutes les classes de solution. Les configurations $\mathbf{q}_{\text{init}}$ et $\mathbf{q}_{\text{fin}}$ ainsi obtenues et sans collision sont insérées dans les couches correspondantes à chaque paire de solution. Des chemins locaux sont calculés pour connecter (les différents) $\mathbf{q}_{\text{init}}$ et $\mathbf{q}_{\text{fin}}$ avec leurs voisins dans les couches correspondantes.

Si $\mathbf{q}_{\text{init}}$ et $\mathbf{q}_{\text{fin}}$ peuvent être connectées à des noeuds de la même composante connexe sur la même couche, la requête de planification peut être résolue sans passer par des configurations singulières. Autrement, le graphe de connexion de haut niveau est exploré pour trouver le plus court chemin entre $\mathbf{q}_{\text{init}}$ et $\mathbf{q}_{\text{fin}}$. Ainsi, le chemin résultant est constitué par un nombre minimal de reconfigurations des manipulateurs pour résoudre la requête.

4.3 Approche basée sur les méthodes de diffusion

Nous présentons une variante "requête unique" de la méthode basée sur les réseaux probabilistes présentée Section 4.2. Dans cette approche, la construction de l'arbre ne peut pas être divisée en deux parties (construction des couches puis leur connexion). En effet, l'une des caractéristiques des méthodes de diffusion, type RRT, est que tout nœud ajouté à l'arbre étend forcément un nœud s'y trouvant déjà (hormis les nœuds initial et final). L'idée est donc d'échantillonner les configurations singulières et d'étendre l'arbre vers ces configurations. La Section 4.3.1 montre comment les différentes couches d'arbres de diffusion sont construites alors que la connexion des différentes couches est décrite Section 4.3.2.

4.3.1 Construction des couches

La méthode proposée est basée sur l'extension des algorithmes de type RRT aux mécanismes à chaîne cinématique fermée [Cortés 04a]. La première étape dans l'itération d'un algorithme

de type RRT est l'échantillonnage d'une configuration $\mathbf{q}_{rand}$. Contrairement aux méthodes de création de réseaux probabilistes (type PRM), cette configuration n'est pas ajoutée comme nouveau nœud de l'arbre. Elle est seulement utilisée comme but local de l'exploration. De ce fait, $\mathbf{q}_{rand}$ n'a pas besoin de satisfaire les contraintes de fermeture cinématique ou d'autres contraintes (*e.g.* évitement d'obstacles). Seules les variables actives du système sont alors échantillonnées et considérées lors du calcul de distance et de la création de chemins locaux entre deux configurations.

Généralement, les planificateurs de type RRT conçus pour des mécanismes à chaîne cinématique ouverte utilisent l'échantillonnage uniforme des variables articulaires. Cependant, sous des contraintes de fermeture cinématique, un échantillonnage spécifique des variables actives est nécessaire pour l'exploration. L'échantillonnage de ces variables actives est réalisé grâce à l'algorithme RLG.

Les deux stratégies proposées pour la construction des couches de réseau probabiliste peuvent être adoptées. Étant donné que seules les variables actives sont considérées lors de la phase d'expansion, une configuration $\mathbf{q}_{near}$ peut être sélectionnée sur chacune des couches (stratégie "parallèle"), ou seulement sur une seule couche aléatoirement ("une-par-une"). Le nœud racine de chaque couche est soit un nœud initial ou final (voir Section 4.2.3), soit une configuration singulière. L'Algorithme 4.3 illustre la différence entre ces deux stratégies et correspond à la variante RRT-Connect.

Algorithm 4.3: RRT_ConstructAndConnectLayers

Input: the robot R, the environment E, the root config q_{root}, the strategy *Strategy*
Output: the tree G

```
G ← InitTree(q_root);
while not StopCondition(G) do
    q_rand ← RLG_SampleActive(R, E);
    if Strategy = Parallel then                                   // parallel
        for i ← 1 to n_Arm1 do
            for j ← 1 to n_Arm2 do
                q_near ← NearestNeighbor(q_rand, G_i,j);
                if Exists(q_near) then
                    RRT_ExpandConnectAndInsert(q_rand, q_near, G_i,j);
    else                                                          // one-by-one
        i ← RandomIkClass(Arm1);
        j ← RandomIkClass(Arm2);
        q_near ← NearestNeighbor(q_rand, G_i,j);
        if Exists(q_near) then
            RRT_ExpandConnectAndInsert(q_rand, q_near, G_i,j);
```

4.3.2 Extension de l'arbre vers des configurations singulières

L'ajout de configurations singulières dans l'arbre de diffusion est nécessaire pour relier les différentes couches entre elles. L'Algorithme 4.4, montre le pseudo code de la phase d'expansion de la méthode. L'expansion de q_{near} est itérée jusqu'à ce que les contraintes internes ou externes du système ne soient plus valides ou que q_{rand} soit atteinte. La configuration q_{feas} résultante de cette expansion, est ajoutée à la couche $G_{\mathrm{i,j}}$ contenant q_{near}.

Algorithm 4.4: RRT_ExpandConnectAndInsert

Input: the direction config q_{rand}, the nearest config q_{near}, the layer $G_{\mathrm{i,j}}$
Output: the layer $G_{\mathrm{i,j}}$ updated

```
q_feas ← q_near;
state ← OK;
while state = OK do
    q_step ← MakeStep(q_feas, q_rand);
    if FeasibleConfiguration(q_step) then
        q_feas ← q_step;
    else
        state ← FAIL;
AddNewNode(q_feas, G_i,j);
AddNewEdge(q_feas, q_near, G_i,j);
ConnectToExistingCompCo(q_feas, G_i,j);
if IsCloseToSingularConfig(q_feas) then
    q_sing ← GetValidSingularConfig(q_feas);
    if AreConfigConnectable(q_sing, q_feas) then
        AddNewNode(q_sing, G_i,j);
        AddNewEdge(q_sing, q_feas, G_i,j);
        ConnectToExistingCompCo(q_sing, G_i',j);
```

L'échantillonnage des configurations singulières est une étape coûteuse en temps de calcul. Afin d'améliorer les performances de l'algorithme le test n'est effectué que lorsque une ou plusieurs articulations de la configuration q_{feas} sont proches d'une valeur singulière. Dans ce cas, une configuration singulière est générée en suivant la méthode décrite dans la Section 4.2.2. La valeur des articulations proche d'une valeur singulière est fixée à cette dernière. Les autres articulations sont échantillonnées de façon gaussienne autour de q_{feas}, au lieu d'être échantillonnées avec la méthode RLG. Le but de cette génération est de créer une configuration singulière q_{sing} du robot proche de q_{feas} afin de faciliter leur connexion. Si cette connexion est possible, q_{sing} est ajoutée aux couches $G_{\mathrm{i,j}}$ et $G_{\mathrm{i',j}}$, sinon q_{sing} est rejetée. Lorsqu'un arbre est déjà créé dans la couche $G_{\mathrm{i',j}}$, un test de connexion est effectué entre cet arbre et la configuration singulière. Cette connexion induit la présence de cycles dans le graphe de haut niveau (voir Section 4.2.2). Le test de connexion entre les différentes composantes connexes (fonction *ConnectToExistingCompCo*) d'une même couche est également réalisé lors de l'ajout d'un nœud non singulier.

4.4 Lissage des chemins solution

De par l'aspect probabiliste de cette approche, les chemins solutions générés sont irréguliers et parfois trop longs : ils doivent donc être lissés. La présence de configurations singulières le long du chemin ne permet pas l'application des méthodes de lissage (voir Chapitre 2) sur le chemin tout entier. En effet, la connexion entre les différentes couches ne peut être établie que par des configurations singulières. Le chemin solution est divisé en plusieurs sous-chemins, chacun se trouvant sur une couche, déterminés par les configurations singulières. La figure 4.6 illustre les différents sous-chemins obtenus. Chaque sous-chemin est lissé dans sa couche. Le chemin solution final est ensuite reconstitué.

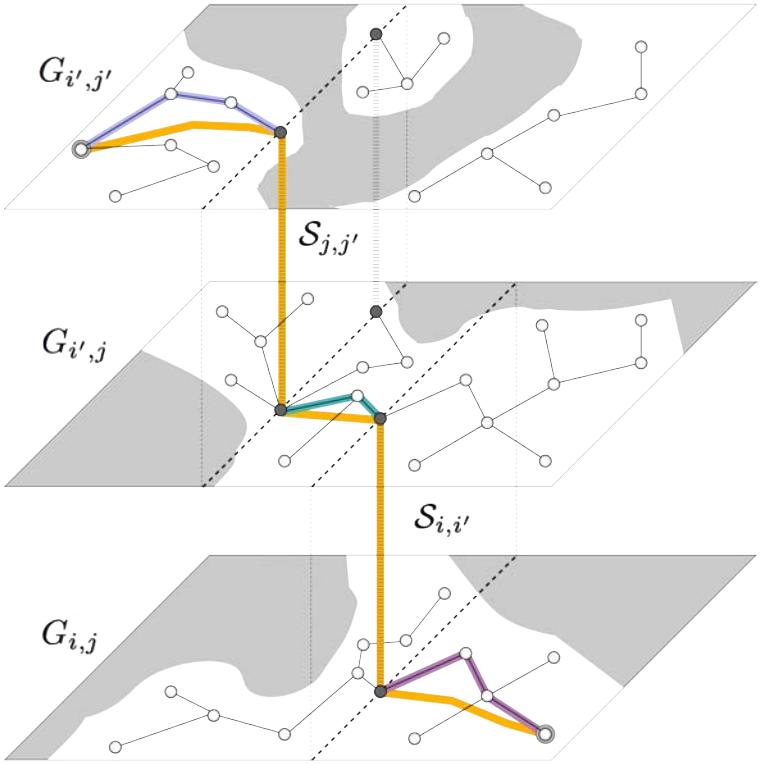

FIGURE 4.6 – Les sous-chemins, illustrés en bleu, vert et rose, composant le chemin solution initial sont lissés dans leurs couches respectives. Le chemin solution final (en jaune) est composé des sous-chemins lissés connectés par les configurations singulières.

4.5 Résultats

4.5.1 Implémentation

Cette section précise les algorithmes de planification utilisés dans les tests présentés ci-dessous. Pour le planificateur basé sur l'extension des réseaux probabilistes, l'approche implémentée est Visib-PRM [Siméon 00], qui permet de créer des graphes compacts (*i.e.* avec un faible nombre de nœuds). Pour le planificateur basé sur les méthodes de type RRT, l'approche BiRRT-Connect [Kuffner 00] a été utilisée. Cette méthode est très efficace pour la résolution de requêtes définies

par des configurations initiales et finales contraintes.

Pour la méthode basée sur les réseaux probabilistes, les phases de création et de connexion des couches sont appelées itérativement. Un paramètre définit le nombre d'appels consécutifs de la phase de construction avant une tentative de connexion des couches. Les tests présentés dans cette section ont été réalisés avec la valeur de ce paramètre à quatre.

4.5.2 Exemple académique

Cette section présente une analyse des performances de la méthode proposée sur plusieurs exemples. Le système mobile, représenté dans la Figure 4.1 est constitué de deux manipulateurs plans non redondants possédant trois articulation rotoïdes qui maintiennent un même objet. Les variables actives sont donc celles qui définissent la position de l'objet : $\mathbf{q}^{\mathrm{a}} = \mathbf{q}_{\mathrm{Obj}}$. Le nombre maximum de solutions des cinématiques inverses des deux manipulateurs, $\mathcal{IK}_{\mathrm{Arm1}}$ et $\mathcal{IK}_{\mathrm{Arm2}}$, est de deux. Le nombre total de couches sera donc de quatre. Pour les deux bras, les classes de solutions des cinématiques inverses s'intersectent dans un ensemble singulier caractérisé par une valeur articulaire de la seconde rotation (le coude) égale à zéro.

TABLE 4.1 – Résultats : Problème Académique

Problème	Stratégie	n_{nodes}	T
A	couche unique	11	0.93 s
A	parallèle	42	4.28 s
A	une-par-une	34	2.49 s
B	couche unique	–	–
B	parallèle	39	1.96 s
B	une-par-une	35	2.17 s

Le but de ces tests est de comparer la performance de l'algorithme proposé lorsque la reconfiguration d'un ou de plusieurs manipulateurs est nécessaire pour résoudre le problème de planification et lorsque le problème peut être résolu sans la reconfiguration d'aucun manipulateur. La méthode est testée sur les deux problèmes de la Figure 4.1. Pour le problème "A", qui ne requiert la reconfiguration d'aucun des deux manipulateurs, le chemin solution peut être obtenu en ne construisant un graphe que sur une seule couche définie par les classes de cinématique inverse des configurations initiale et finale. Les résultats numériques [1] de la première ligne de la Table 4.1 correspondent à ce test. Étant donné que la nécessité de reconfigurer un des manipulateurs pour résoudre un problème de planification n'est pas connue à priori, l'algorithme présenté (variantes "parallèle" et "une-par-une") explore toutes les couches pour trouver un chemin solution. Les résultats du problème "A" grâce à la variante "parallèle" sont reportés

1. Tous les résultats numériques dans cette section ont une moyenne d'au moins 20 exécutions du planificateur. Les temps de calculs correspondent à un processeur Intel Core2-Duo cadencé à 2.13Ghz.

dans la seconde ligne et dans la troisième ligne pour la variante "une-par-une". Comme prévu, le nombre de nœuds et le temps de calcul sont multipliés par quatre pour la variante "parallèle" vis-à-vis de la méthode ne générant pas de configurations singulières.

La solution du problème "B" nécessite la reconfiguration des deux manipulateurs (voir Figure 4.7). Un chemin solution ne peut donc être obtenu si on ne construit un graphe que sur une seule couche. La performance de la méthode pour la résolution du problème "parallèle" est reportée ligne cinq de la table. On remarque que le nombre de nœuds est proche du nombre de nœuds nécessaires à la résolution du problème "A". Notons que même si le problème "B" est plus difficile à résoudre, l'algorithme est plus rapide. La raison de cette meilleure performance est la présence de l'obstacle en bas à gauche de l'espace de travail qui réduit le volume de $\mathcal{C}_{feas}$ à explorer et donc le coût engendré par le test des chemins locaux. Ces résultats montrent que dans le pire des cas, le coût calculatoire augmente linéairement avec le nombre de couches.

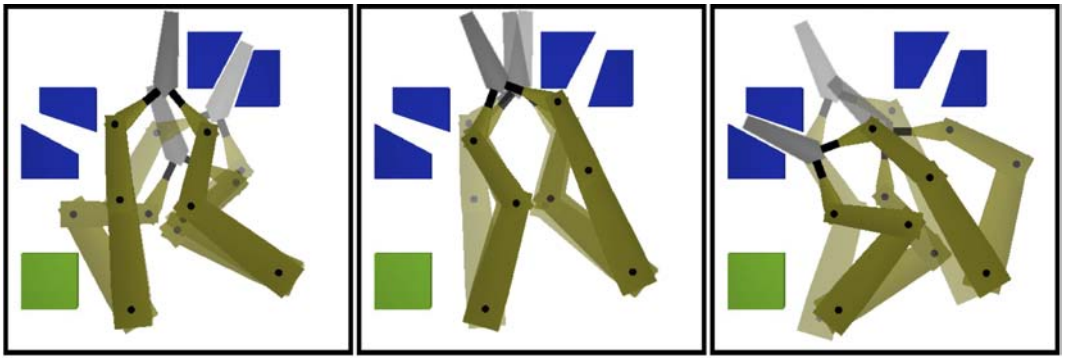

FIGURE 4.7 – Solution du problème "B". Les images montrent trois parties du chemin solution. La première partie (gauche) correspond au sous chemin entre la configuration initiale et la première configuration singulière, permettant la reconfiguration du manipulateur gauche. La seconde (milieu) montre le sous chemin pour atteindre la seconde singularité (reconfiguration du manipulateur droit). La troisième partie (droite) montre le dernier sous chemin afin d'atteindre la configuration finale.

Les tests ont également été effectués pour analyser la performance des deux stratégies de l'approche : une générant simultanément des nœuds sur toutes les couches pour chaque valeur $\mathbf{q}^a$ ("parallèle") et une autre ne générant qu'un seul nœud sur une seule couche ("une-par-une"). Pour le problème A, les résultats numériques montrent une meilleure performance de la stratégie "une-par-une". Pour le problème "B", le nombre de nœuds et le temps de calcul sont relativement similaires pour les deux stratégies. La raison est que dans le problème "A", toutes les couches n'ont pas besoin d'être explorées pour trouver un chemin solution, contrairement au problème "B" où une grande partie de l'espace des configurations des quatre couches doit être exploré afin d'obtenir un chemin solution contenant deux reconfigurations. Notons finalement que les résultats de la stratégie "une-par-une" sont presque identiques pour les deux problèmes présentés. Ceci démontre que la performance de la méthode proposée n'est pas affectée par la présence de configuration singulière dans le chemin solution.

4.5.3 Applications sur le robot Justin du DLR

Justin [Ott 06] est un robot humanoïde composé de deux bras DLR-Lightweight-Robot-III de sept degrés de liberté (DdL) chacun [Hirzinger 02], montés sur un torse de trois DdL. Le modèle géométrique inverse [Konietschke 09] considère comme articulation redondante la troisième rotation de chaque bras. Pour une valeur donnée de cette articulation, le nombre maximum de classes de solutions de la cinématique inverse pour les six autres rotations est de huit. Le nombre de couches est donc de soixante quatre. Quatre ensembles singuliers permettent de passer d'une couche à une autre. Ils sont caractérisés par la valeur (0 ou 90 degrés) d'un ou de deux angles de rotation.

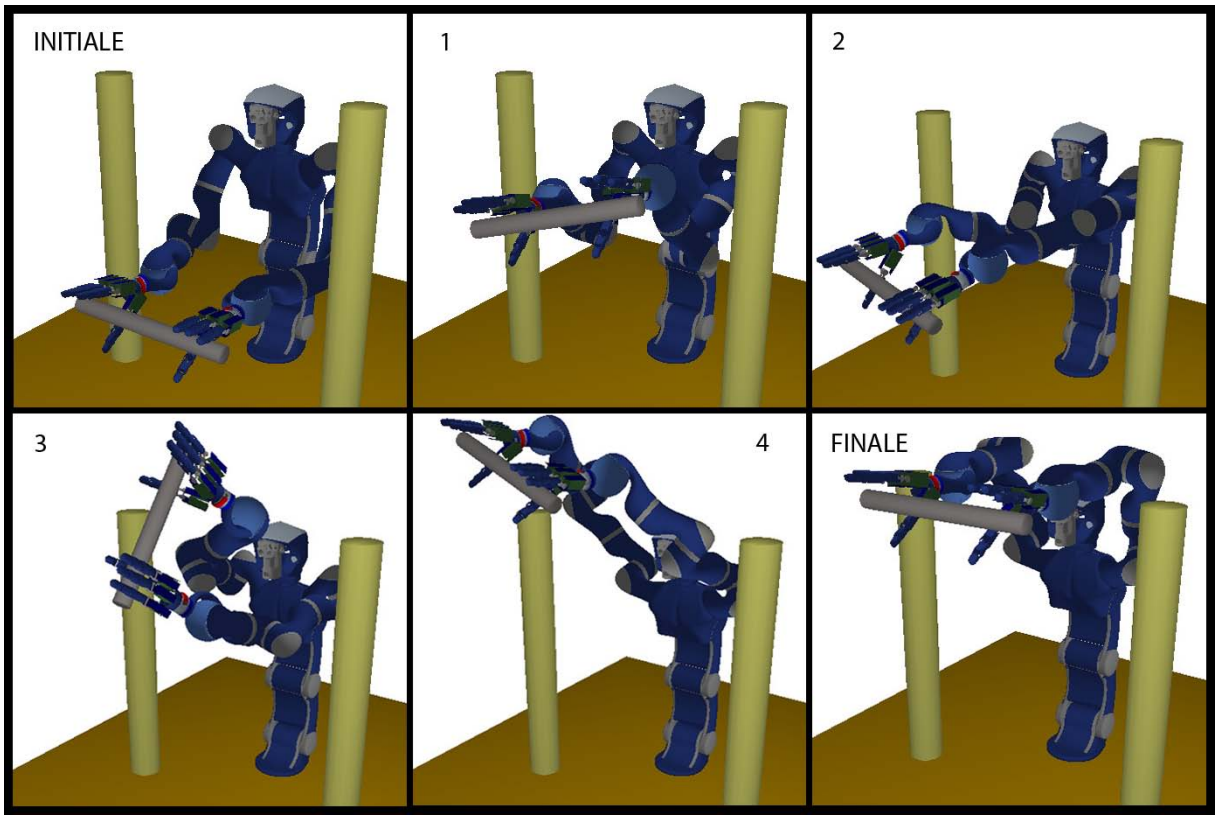

FIGURE 4.8 – Problème de manipulation bi-bras de Justin nécessitant deux reconfigurations. Les configurations singulières sont illustrées dans les vignettes deux et quatre de la séquence.

Dans le premier exemple présenté Figure 4.8, les articulations du torse et des mains ont été fixées. Les variables actives sont donc celles correspondant à la position et l'orientation de l'objet ainsi qu'aux articulations redondantes. Le problème est de calculer un mouvement de Justin permettant d'élever une barre. Les deux bras étant redondants, deux obstacles verticaux sont positionnés de par et d'autre du robot pour contraindre son mouvement et forcer les deux bras à traverser des singularités pour atteindre la configuration finale. La figure montre différentes configurations du chemin solution. En particulier, les vignettes deux et quatre correspondent aux deux configurations singulières permettant la reconfiguration du robot (la valeur du troisième

axe égale à zéro).

Le planificateur construit incrémentalement un graphe, à l'aide de la stratégie "parallèle" en utilisant l'algorithme Visib-PRM, qui permet de résoudre le problème en quarante six [2] secondes et contient près de 1220 nœuds. Pour la stratégie "une-par-une" le temps de calcul est approximativement de 41 secondes pour la génération de 845 nœuds. On remarque que le temps de calcul est sensiblement le même pour les deux stratégies. Cependant, la stratégie "une-par-une" produit des graphes plus "légers" (en terme de nombre de nœuds) que ceux produit par la stratégie "parallèle". Cependant quand le but est de construire un réseau probabiliste riche afin de résoudre plusieurs requêtes, la stratégie de construction "parallèle" est plus appropriée.

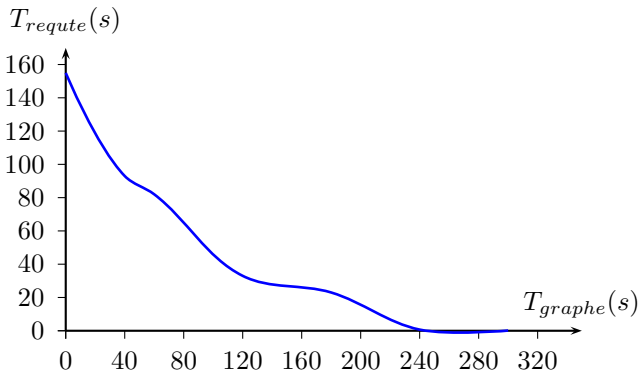

FIGURE 4.9 – Comparaison du temps de résolution des requêtes avec le temps de construction du graphe.

La Figure 4.9 montre la relation entre le temps de construction d'un réseau probabiliste et le temps nécessaire pour résoudre des requêtes de planification. Le temps de résolution de requêtes décroît rapidement quand la taille du graphe précalculé augmente. On peut noter que les requêtes de planification sont résolues quasi instantanément après quatre minutes de précalcul.

Dans le second exemple, Justin doit servir un verre à un humain. La Figure 4.10 illustre la configuration initiale et finale du robot, ainsi que les positions intermédiaires et de transitions représentatives du chemin solution de cet exemple. Un obstacle vertical a été ajouté à l'environnement pour contraindre Justin à reconfigurer son coude gauche. La configuration finale du bras gauche ne peut être sans collisions si le manipulateur ne change pas de classe de cinématique inverse. En effet l'utilisation seule du joint redondant ne permet pas de passer à une configuration coude haut à partir de la configuration de départ. De plus, dans cet exemple, une contrainte est ajoutée pour maintenir l'objet transporté (verre) horizontal. Cette contrainte est prise en compte lors de l'échantillonnage des configurations dans les différentes couches et les configurations singulières, où les paramètres d'orientation permis sont proches de zéro (état horizontal).

Dans cet exemple, la génération d'une trajectoire du robot avec la variante "parallèle" nécessite près de 250 secondes en générant 2380 nœuds sur les soixante quatre couches. Notons

2. Le temps publié dans [Gharbi 08] est de 140 sec. La différence de performance est due à des optimisations algorithmiques de l'implémentation.

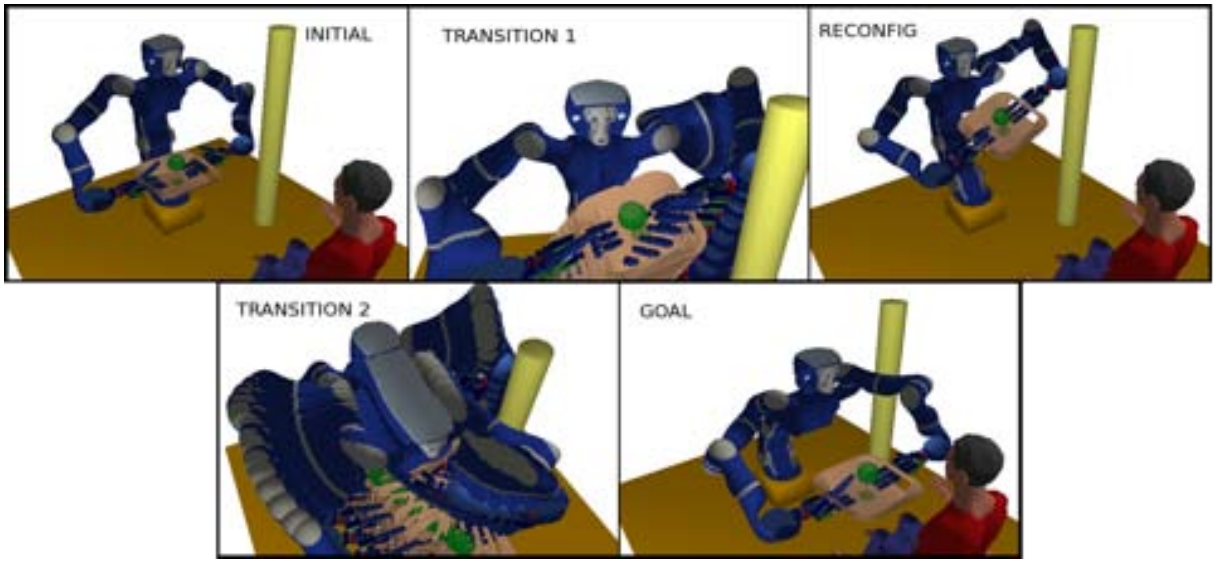

FIGURE 4.10 – Exemple de Justin servant un verre d'eau à l'humain. La présence de l'obstacle à droite des figures, force la reconfiguration du manipulateur gauche du robot illustrée dans la vignette "Reconfig".

qu'en l'absence de la contrainte de maintien du plateau horizontal, la trajectoire est générée en seulement 18 secondes pour 670 nœuds . Cette différence est principalement due à la complexité d'échantillonner des configurations valides et sans collisions répondant à cette contrainte additionnelle.

Le scénario présenté dans la Figure 4.11, illustre un exemple complexe de Justin monté sur sa plateforme mobile dans un environnement contraint. La tâche est de transférer un plateau se trouvant sur les étagères vers la table. Dans cet exemple, les dix-sept degrés de liberté de Justin sont planifiés, la base mobile est fixe. Plusieurs objets présents sur ces supports (maquette, étagères, lampe de bureau, livre) contraignent les mouvements du plateau alors que les lustres ainsi que la lampe à pied contraignent les mouvements du robot. Le chemin solution a été calculé par l'extension de la méthode RRT qui a été présentée en 458 secondes en générant 599 nœuds. L'utilisation de la méthode BiRRT-Connect [Kuffner 00], sans prise en compte des singularités, résout la requête de planification en 533 secondes en générant 667 nœuds. Notons que la gestion explicite des configurations singulières n'affecte pas la performance de la méthode.

4.6 Conclusion

Nous avons présenté une méthode pour la planification de mouvement de manipulation coordonnée pour un système composé de deux manipulateurs. La méthode permet d'étendre les algorithmes de planification des mécanismes à chaînes fermées existants pour la résolution de problème nécessitant la reconfiguration de l'un des bras manipulateurs. Cette approche est générale et peut directement être utilisée pour des systèmes multi-bras complexes. Les

résultats obtenus pour le robot Justin montrent l'efficacité calculatoire de l'algorithme face à des problèmes réalistes et complexes.

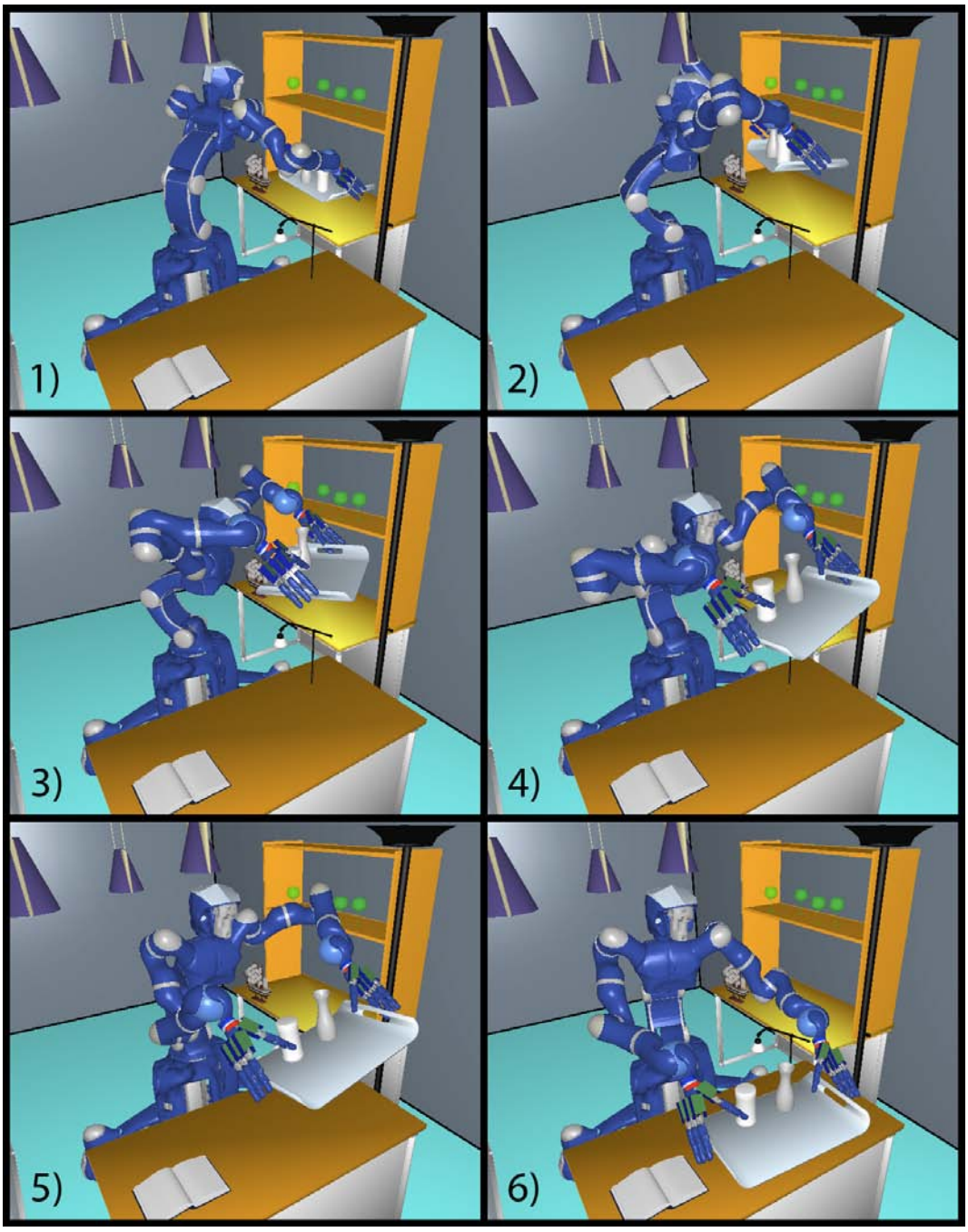

FIGURE 4.11 – Exemple complexe de Justin transportant un plateau posé sur une étagère à la table dans un environnement complexe (lustres, lampe de bureau, maquette, lampe à pied). Dans cet exemple, les dix-sept degrés de liberté du robot sont planifiés simultanément.

5

Tâche de prise et pose avec ressaisie

Dans ce chapitre nous présentons une méthode de planification de mouvement pour la résolution de tâches de prise et pose d'un objet par un torse humanoïde nécessitant un changement de prise de l'objet. Il s'agit donc d'une instance particulière de tâche de manipulation et la méthode proposée n'aborde pas le problème de décomposition automatique de la tâche de manipulation traitée dans [Siméon 04].

Cependant la manipulation d'objet par des torses humanoïdes pose de nouveaux problèmes. Par exemple, pour un torse humanoïde, lorsque les positions initiale et finale de l'objet à manipuler ne se trouvent pas dans l'espace de travail commun aux deux bras, le transfert de l'objet d'un bras à l'autre s'avère alors nécessaire pour accomplir la tâche. Ce changement de prise peut être réalisé en plaçant dans un premier temps l'objet dans une position intermédiaire située dans l'espace de travail commun des deux bras, puis en le reprenant à l'aide du second bras pour finaliser la tâche. Une stratégie plus efficace consiste en la planification d'une double prise de l'objet afin d'effectuer le changement sans pose préalable de l'objet manipulé. L'intérêt de cette stratégie est de produire des mouvements plus proches de ceux effectués par les humains lorsqu'ils manipulent des objets. Elle permet aussi de s'affranchir du calcul d'une position stable de l'objet dans l'espace de travail commun.

Un exemple de problème résolu par la méthode illustré Figure 5.1 est décrit dans ce chapitre. Le problème est défini uniquement par les configurations initiale et finale de l'objet et du robot. Le robot Justin, équipé de deux mains "Schunk Anthropomorphic Hands", a pour tâche de saisir la statuette de cheval à sa droite et de la placer sur sa gauche. Les vignettes de la Figure 5.10 montrent respectivement les configurations de saisie, d'échange et de pose générées par le planificateur pour résoudre la tâche. La principale difficulté du problème est de calculer

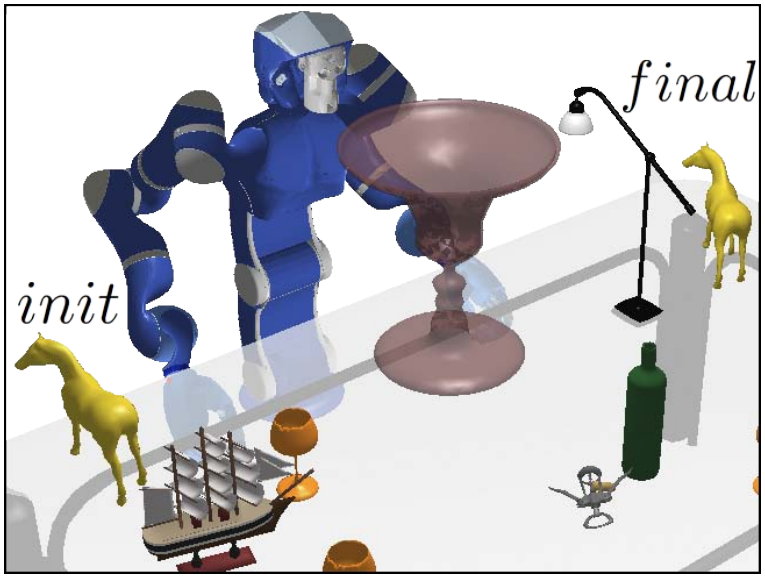

FIGURE 5.1 – Exemple de prise et pose nécessitant une ressaisie dans un environnement contraint. Le robot a pour tâche de déplacer la statuette en forme de cheval de sa configuration initiale à sa configuration finale.

la position d'échange de l'objet qui dépend des positions initiale et finale de ce dernier. Cette position est contrainte par les obstacles de l'environnent ainsi que par la cinématique du robot (elle doit être dans l'espace de travail commun aux deux bras).

5.1 Vue d'ensemble de l'approche

La méthode proposée pour la planification d'une tâche nécessitant un changement de main est basée sur une décomposition en quatre parties : mouvement de saisie de l'objet, mouvement de transport de l'objet jusqu'à la position d'échange entre les deux mains, mouvement de pose de l'objet et mouvement de retour à une position de repos du robot. Chaque partie représente une requête de planification de mouvement.

L'approche proposée utilise un ensemble de données précalculé afin de réduire le temps de calcul en ligne. Cet ensemble est composé d'une liste triée de prises possibles de l'objet à manipuler, générée pour chaque main, en utilisant la méthode développée dans [Saut 07a] (Section 5.3.1). Il est également composé d'un réseau probabiliste ne prenant pas en compte l'objet à manipuler. Ce réseau est calculé à l'aide de la méthode de coordination de réseaux proposée dans le Chapitre 3.

Afin de résoudre une tâche de prise et pose spécifique (planification en ligne), une liste de configurations de prise sans collision est générée et ordonnée suivant des critères décrits dans la Section 5.4.2. Ces deux listes sont ensuite utilisées pour filtrer le grand nombre de candidats pour les doubles prises et ainsi d'en générer une liste ordonnée (Section 5.3.2). Les doubles prises les mieux classées déterminent les différentes configurations de saisie, de pose et d'échange

nécessaires à la résolution de la requête de planification.

Le réseau probabiliste précalculé (Section 5.4.1) est utilisé pour résoudre ces requêtes de planification (Section 5.4.4). Cependant, vu que lors de la construction du réseau l'objet à manipuler n'a pas été pris en compte, une validation des chemins solutions est nécessaire. Lorsque le chemin est en collision, une reconnexion locale est effectuée afin de trouver un chemin alternatif (Section 5.4.5). En cas d'échec de la planification, de nouvelles configurations de saisie, de pose et d'échange sont utilisées pour résoudre la tâche. La Figure 5.2 représente l'architecture globale du planificateur ainsi que l'enchainement des différentes étapes mises en œuvre par les sous-modules de saisie et de planification de mouvement. Les pictogrammes illustrent les éléments pris en compte et les résultats des sous-modules.

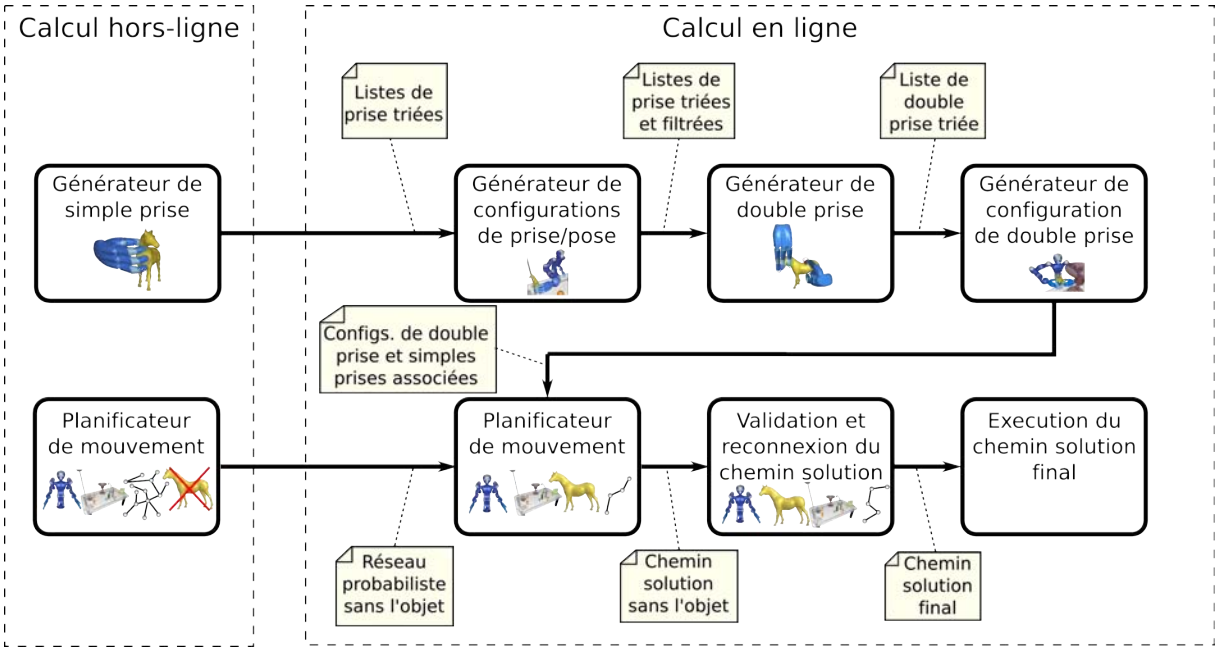

FIGURE 5.2 – Différents modules utilisés pour planifier une tâche de prise et pose avec ressaisie.

5.2 Formulation du problème

La méthode décrite dans ce chapitre considère un robot muni de deux bras ayant chacun comme organe terminal une main, qui doit manipuler un objet dans un environnement composé d'obstacles statiques. Les entrées du problème sont les configurations initiale et finale du robot (q_{robot}^{init} et q_{robot}^{final}) ainsi que les positions initiales et finales de l'objet (p_{object}^{init} et p_{object}^{final}). Les positions p_{object}^{init} et p_{object}^{final} correspondent à des placements stables de l'objet sur un support (une table par exemple). La position p_{object}^{init} est considérée telle que l'objet ne peut être saisi que par un des deux bras, noté "bras 1". De même la position finale p_{object}^{final} est telle que l'objet ne peut être atteint que par l'autre bras, noté "bras 2". Le robot devra donc échanger l'objet entre ses deux mains. La position d'échange de l'objet est notée p_{object}^{exch}. La configuration globale du robot (base ou torse et les bras/mains) lors de cet échange est notée q_{robot}^{exch}. Le robot saisira l'objet en p_{object}^{init}

avec la configuration q_{robot}^{grasp} et le placera en p_{object}^{final} avec la configuration q_{robot}^{place}.

La planification de la tâche prise et pose avec reprise, nécessite la combinaison de plusieurs aspects, généralement étudiés séparément dans la littérature : la génération automatique de prises de l'objet [Shimoga 96, Bicchi 00] et la planification des mouvements du robot [Choset 05, LaValle 06]. Les opérations de saisie et de ressaisie de l'objet nécessitent un planificateur de prise capable de générer des prises stables et sans collision. Les prises doivent également satisfaire les contraintes cinématiques du robot. Le planificateur de mouvements doit être apte à générer des chemins solutions pour un système robotique complexe dans un environnement contraint par des obstacles statiques et mobiles (objet manipulé). Les sections suivantes détaillent le fonctionnement de la méthode.

5.3 Planification de prise

La planification de prise consiste à trouver une configuration d'une main ou de l'organe terminal permettant de saisir l'objet. Dans le contexte présenté ci-dessus, nous nous intéressons à deux types de prises : des prises à une main (nommées "simples prises") et des prises à deux mains (nommées "doubles prises"). Nous considérons le cas de prises dites prises en précision, (*i.e.* les contacts sont réalisés par le bout de doigts). Cette supposition permet de ne raisonner qu'avec des points de contact (cas le plus traité dans la littérature). Elle donne également plus de possibilités de prises car de petites parties peuvent être saisies, mais au détriment d'une stabilité plus faible que celle fournie par une prise à pleine main. Les travaux présentés dans cette section ont été détaillés dans [Saut 10]. Cet algorithme a été implémenté pour la main Schunk Anthropomorphic Hand ("SAHand" illustrée Figure 5.4) mais il convient de noter que la méthode n'est pas spécifique à cette main.

Le planificateur repose sur le précalcul d'une liste de prises, pour chacune des mains du robot et pour l'objet à manipuler. Ces listes sont ensuite utilisées en ligne pour générer les doubles prises.

5.3.1 Planification de simples prises

Une simple prise est caractérisée par : 1) un repère de prise qui représente la position relative du repère de référence de la main (*e.g.* repère de la paume ou du poignet) par rapport au repère de l'objet. 2) Un ensemble de points de contact (un point sur la surface de l'objet et le doigt réalisant le contact). 3) Une configuration de la main (ensemble des paramètres articulaires de la main/doigts).

La méthode utilisée dans nos travaux permet de générer une liste ordonnée de prises suivant plusieurs critères (*e.g.* stabilité, fermeture en force, courbure de la zone de contact, ...). Cette méthode génère des prises de façon uniforme autour de l'objet et ne privilégie pas une direction donnée (*e.g.* ne générer que des prises saisissant l'objet par le haut). Cet algorithme, décrit dans [Saut 07a] et dont nous rappelons ci-dessous les grandes lignes, repose sur les quatre étapes suivantes :

– Échantillonnage d'un ensemble de repères de prises.
– Calcul d'une liste de prises à partir de l'ensemble des repères de prises.
– Application d'un filtre de stabilité.
– Calcul d'un score représentant la qualité de chaque prise.

5.3.1.1 Échantillonnage des repères de prises

Les repères de prises sont échantillonnés uniformément autour de l'objet à l'aide d'une grille afin d'éviter de privilégier des directions d'approche de la main lors du calcul d'une prise. Dans le cas de la main Schunk utilisée dans nos travaux, les repères de prises sont centrés sur l'intersection des espaces de travail des doigts. Le nombre de positions et d'orientations souhaité est donné en entrée de l'algorithme : chaque paire position-orientation définissant un repère. Les positions sont échantillonnées uniformément à l'intérieur de la boîte englobante de l'objet. Les orientations sont calculées à l'aide d'une méthode déterministe générant des échantillons uniformes dans $SO(3)$, comme celle présentée dans [Yershova 04]. Pour chacun des repères générés, un ensemble de prises est calculé.

5.3.1.2 Calcul d'une liste de prise

Comme la méthode ne restreint pas les positions de la main ni les portions de la surface de l'objet sur lesquelles les contacts peuvent avoir lieu, le calcul de la liste de prises peut être coûteux. Mis à part le test de collision, le calcul le plus coûteux est la résolution de la cinématique inverse des doigts. Un contact peut uniquement avoir lieu dans l'intersection de l'espace de travail d'un doigt et la surface de l'objet. Une méthode a donc été mise au point pour savoir au plus tôt si un contact entre le doigt et l'objet est possible (voir Figure 5.3).

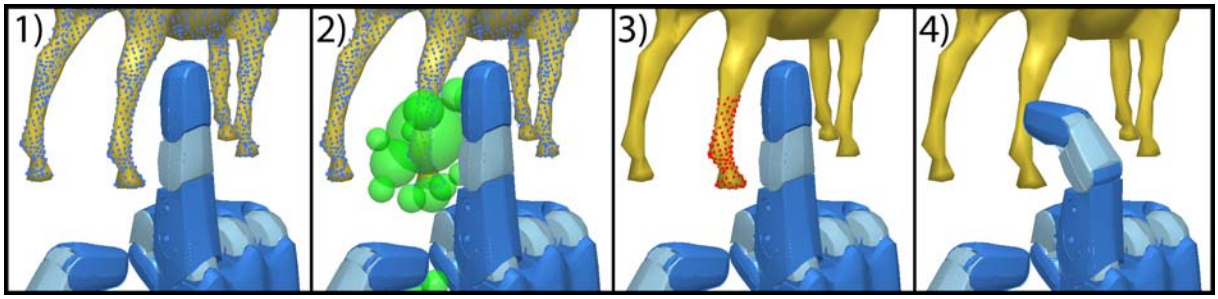

FIGURE 5.3 – À partir de l'approximation de la surface de l'objet (1) et de l'espace de travail du doigt (2), un ensemble de points atteignables est calculé (3) avant d'en sélectionner un comme point de contact (4).

La surface de l'objet est approximée à l'aide d'un ensemble de points uniformément échantillonnés (Figure 5.3 (1)). Le pas d'échantillonnage est déterminé par le rayon du bout des doigts, partie qui assurera le contact avec l'objet.

Une approximation de l'espace de travail de chaque doigt est également calculée. Partant d'une grille approximant l'espace de travail des doigts (Figure 5.4 à gauche), un ensemble de

sphères contenues dans cet espace est calculé de façon incrémentale. L'idée est de trouver les points internes de la grille maximisant la distance d avec l'enveloppe de l'espace de travail et les sphères générées. À chaque itération, cette distance est calculée pour tous les points internes de la grille non inclus dans une sphère. Une sphère de rayon d centrée sur le point de la grille ayant la plus grande distance d est ensuite créée. Ce processus est réitéré jusqu'à ce que le nombre maximal de sphères désiré soit atteint, ou que la dernière sphère calculée ait un rayon inférieur à un certain seuil (Figure 5.4 à droite). L'ordre de création des sphères est sauvegardé de telle sorte que la hiérarchie de sphères commence par celle possédant le plus grand rayon, laquelle correspond aux parties de l'espace de travail les plus éloignées des limites articulaires des doigts.

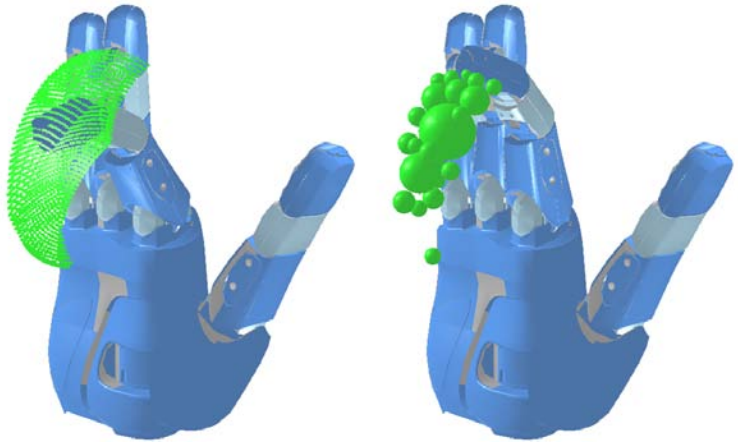

FIGURE 5.4 – Illustration d'une main Schunk Anthropomorphic Hand. L'espace de travail des doigts est discrétisé à l'aide d'une grille (à gauche pour l'index). La grille est convertie en une approximation volumétrique avec un ensemble de sphères (à droite)

Une fois les points sur la surface de l'objet et les sphères calculés, l'intersection des deux ensembles détermine les points accessibles par le doigt (Figure 5.3 (3)). Le point accessible minimisant la distance avec le centre de la sphère la plus grande est sélectionné comme point de contact (Figure 5.3 (4)).

Pour un repère de prises donné, la prise est calculée doigt par doigt. Cela signifie que le point de contact n'est cherché que pour un doigt à la fois et le test de collision n'est effectué qu'avec les doigts précédemment traités. Le premier doigt testé est le pouce étant donné qu'aucune prise stable ne peut être obtenue sans lui. Si un doigt ne peut établir le contact avec l'objet, il est laissé en configuration de "repos" (droit). Le test de stabilité n'est exécuté que si trois contacts au moins sont établis. Notons qu'à cette étape, les prises sont sans collision, *i.e.* il n'y a pas de collision entre la main et l'objet ; les collisions avec l'environnement, les bras ou le torse du robot ne sont pas encore prises en compte.

5.3.1.3 Filtre de stabilité et qualité des prises

Le test de stabilité est basé sur des contacts ponctuels avec frottement, ce qui explique pourquoi au moins trois contacts sont nécessaires. Ce test est basé sur un test de fermeture en force (*force closure*) ainsi que sur un critère de stabilité [Bounab 08]. Toutes les prises ne vérifiant pas la fermeture en force sont éliminées.

De nombreux critères peuvent être utilisés pour calculer la qualité d'une prise [Suarez 06]. Un critère de stabilité n'étant pas suffisant pour déterminer les bonnes prises, quatre critères sont pris en compte : 1) le score de fermeture en force présenté ci-dessus. 2) La distance entre le barycentre des points de contact et le centre de masse de l'objet. 3) la somme des angles entre le vecteur normal au contact et l'axe directeur de l'ellipsoïde de force de chaque doigt. Ce critère privilégie les prises ayant une meilleure transmission de force (plus de détails peuvent être trouvé dans [Suarez 06]). 4) Un score de courbure aux contacts privilégiant les contacts où la courbure de l'objet est faible. Le score d'une simple prise sg s'écrit donc :

$$\mathcal{S}_{grasp}(sg) = \mathcal{S}_{forceClosure}(sg) + \mathcal{S}_{centerOfMass}(sg) + \mathcal{S}_{contactNormal}(sg) + \mathcal{S}_{curvature}(sg) \quad (5.1)$$

5.3.2 Planification de double prise

Les double prises sont calculées à partir de deux listes de simple prise L_1 et L_2, obtenues pour chacune des mains. Chaque paire de simples prises sg_1 et sg_2, appartenant respectivement à L_1 et L_2, est testée afin de ne retenir que les paires libres de collision. Pour réduire le nombre de paires à tester, les listes L_1 et L_2 sont tout d'abord filtrées pour éliminer les simples prises en collision avec les obstacles pour une position initiale et finale de l'objet. Par exemple, toutes les prises saisissant l'objet par le "bas" sont éliminées étant donné qu'il y a collision entre le support de l'objet (la table) et la main. Pour chaque double prise, un score est calculé à partir des deux scores de qualité des simples prises et des deux scores des configurations du robot associés à ces simples prises (*i.e.* les configurations de saisie/pose de l'objet). La génération de ces configurations est présentée dans la Section 5.4.2.

Le score de la configuration du robot prend en compte la configuration des deux bras ainsi que celle du torse. Le score du bras saisissant l'objet est déterminé par l'angle entre la direction de la prise sélectionnée et la direction formée par la base du robot et l'objet (Figure 5.5 (1)). Il est calculé comme suit :

$$\mathcal{S}_{graspDirection}(q_{sg}^{arm1}) = \begin{cases} 0 & \text{si } |\theta| < 90 \\ \frac{|\theta|}{90} - 1 & \text{sinon} \end{cases} \quad \text{avec } \theta = \widehat{\overrightarrow{P_{torso}P_{object}}\ \overrightarrow{Z_{grasp}}}$$

Ce critère pénalise les prises ayant des directions éloignées de l'axe base du robot / objet. Le score du bras libre est ajouté afin de favoriser les configurations "naturelles" du robot. Ce score est composé de trois critères :

– La distance articulaire entre la configuration échantillonnée du bras et une configuration de repos du bras définie par l'utilisateur (Figure 5.5 (2)). Ce critère pénalise les configurations

du bras libre éloignée de la configuration de repos du bras dans $\mathcal{C}$:

$$\mathcal{S}_{jointDist}(q_{sg}^{arm2}) = \sum_{i=1}^{7}(\delta q_i^{arm2})$$

– La différence de hauteur entre la configuration échantillonnée et celle de repos (Figure 5.5 (3)). Ce critère permet de pénaliser les configurations où le bras libre est en position haute favorisant les configurations à la même hauteur que la configuration de repos. La différence de hauteur est calculée au niveau du coude (h_{elbow}) et du poignet (h_{wrist}) :

$$\mathcal{S}_{height}(q_{sg}^{arm2}) = \|h_{elbow} + h_{wrist}\|$$

– La distance [1] $d_a(Pl_{torso}, P_{wrist})$ entre la position P_{wrist} du poignet et le plan Pl_{torso} (Figure 5.5 (4)). Ce critère favorise les configurations du bras libre dont le poignet se trouve devant le robot :

$$\mathcal{S}_{torsoDist}(q_{sg}) = \begin{cases} 0 & \text{si } d_a(Pl_{torso}, P_{wrist}) > 0 \\ \|d_a(Pl_{torso}, P_{wrist})\| & \text{sinon.} \end{cases}$$

Le score d'une configuration q_{sg} saisissant l'objet avec le bras $arm1$ et une prise sg s'écrit :

$$\mathcal{S}_{config}(q_{sg}) = \mathcal{S}_{graspDirection}(q_{sg}^{arm1}) + \mathcal{S}_{jointDist}(q_{sg}^{arm2}) + \mathcal{S}_{height}(q_{sg}^{arm2}) + \mathcal{S}_{torsoDist}(q_{sg}) \quad (5.2)$$

Le plus petit score des simples prises et de leurs configurations associées sont normalisés puis additionnés pour obtenir le score de la double prise :

$$\mathcal{S}_{doubleGrasp}(dg_{1,2}) = \|min(\mathcal{S}_{grasp}(sg_1), \mathcal{S}_{grasp}(sg_2)\| + \|min(\mathcal{S}_{config}(q_{sg_1}), \mathcal{S}_{config}(q_{sg_2})\| \quad (5.3)$$

5.4 Planification de mouvement

La tâche de prise et pose avec ressaisie est décomposée en quatre étapes consécutives :
– Mouvement de "saisie" de l'objet à partir de la configuration initiale du robot(de q_{robot}^{init} à q_{robot}^{grasp}).
– Mouvement de "transport" de l'objet jusqu'à la position d'échange (de q_{robot}^{grasp} à q_{robot}^{exch})
– Mouvement de "pose" de l'objet (de q_{robot}^{exch} à q_{robot}^{place})
– Mouvement de retour à une position de "repos" (de q_{robot}^{place} à q_{robot}^{final})

Les configurations q_{robot}^{init} et q_{robot}^{final} et les positions initiale et finale de l'objet (p_{object}^{init} et p_{object}^{final}) sont les données du problème. La méthode proposée génère les trois autres configurations (q_{robot}^{grasp}, q_{robot}^{exch} et q_{robot}^{place}) à partir de ces données et des listes des simples et doubles prises. Un réseau

1. Notons que cette distance est considérée positive lorsque le poignet est devant le robot et négative lorsque le poignet est derrière.

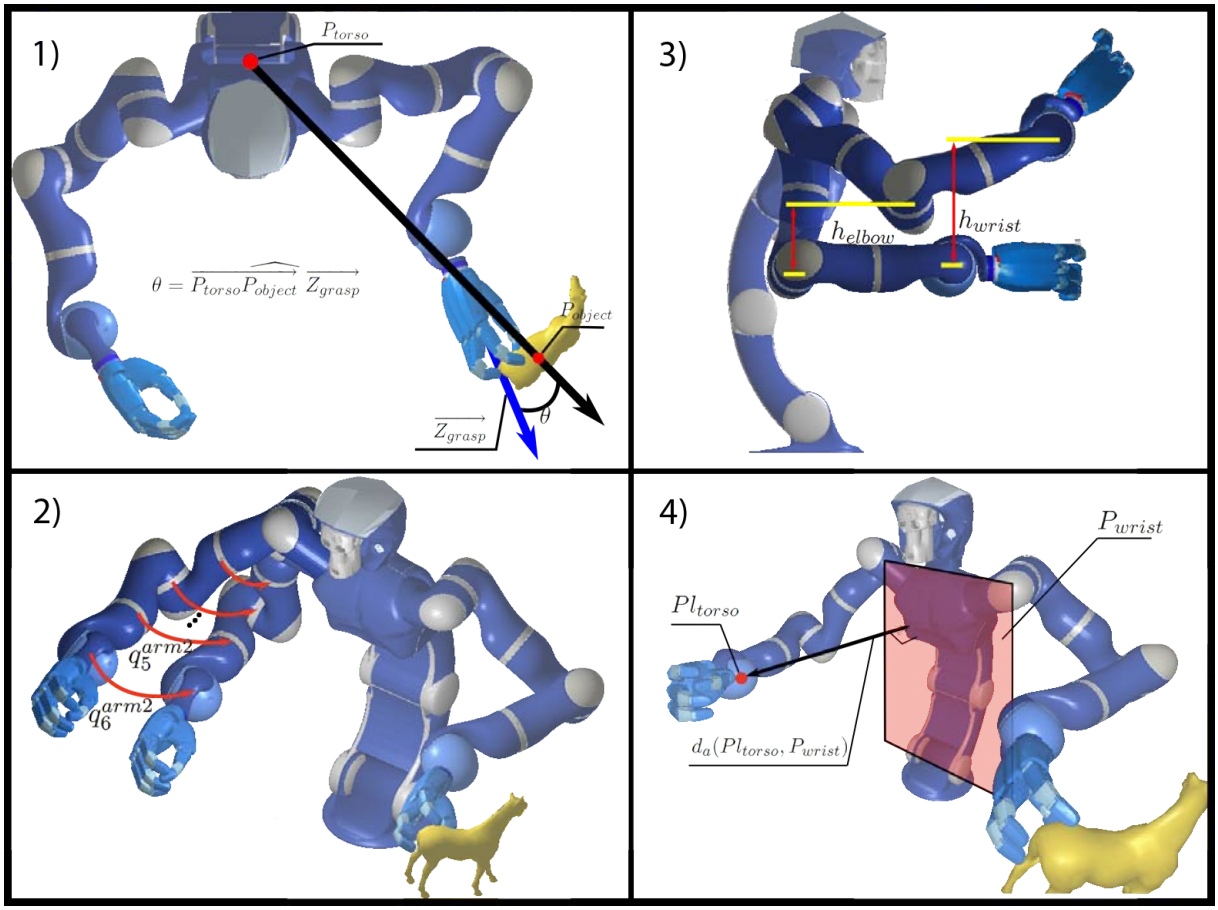

FIGURE 5.5 – Différents critères utilisés pour déterminer la qualité d'une configuration de saisie ou de pose.

probabiliste précalculé pour le robot dans son environnement sans tenir compte de l'objet à manipuler est utilisé pour résoudre les requêtes de planification de chaque étape. Les chemins solutions sont ensuite validés prenant en compte l'objet à manipuler. Nous décrirons dans cette section ces différentes étapes.

5.4.1 Précalcul du réseau probabiliste

Afin d'accélérer la résolution des tâches de prise et pose, le précalcul d'un réseau probabiliste est nécessaire. Durant cette construction de réseau, seules les auto-collisions et les collisions avec les obstacles statiques sont prises en compte, ignorant ainsi l'objet à transporter. Ce choix est réalisé pour permettre l'utilisation du réseau pour plusieurs objets pouvant avoir une forme différente. Une autre stratégie serait de précalculer le réseau en matérialisant l'objet par sa sphère englobante. Elle serait plus performante lors des phases de requête mais ne permettrait d'utiliser le réseau que pour les objets ayant une sphère englobante plus petite que celle utilisée lors du précalcul.

Le réseau probabiliste est précalculé à l'aide de l'algorithme de fusion de réseaux, présenté

dans le Chapitre 3. Un réseau probabiliste est d'abord calculé pour chaque bras avec le torse du robot. Ces deux réseaux sont ensuite fusionnés pour obtenir le réseau du robot tout entier.

5.4.2 Configurations de saisie et de pose

La génération des configurations de saisie q_{robot}^{grasp} et de pose q_{robot}^{place} est utilisée d'une part pour le filtrage des listes de simples prises qui ont été précalculées pour ne retenir que les prises valides étant donné les placements initial et final de l'objet, et d'autre part pour définir les requêtes de planification de mouvement correspondant aux différentes étapes de la tâche de prise et pose.

Partant des listes de simples prises précalculés, des positions initiale et finale de l'objet à manipuler, du bras devant le saisir et du bras devant le déposer, les configurations de saisie et de pose sont générées par échantillonnage aléatoire des degrés de liberté du torse et du bras libre. La configuration du bras saisissant l'objet est ensuite déterminée en résolvant le problème de cinématique inverse, caractérisé par une prise et la position de l'objet d'une part et la configuration du torse d'autre part. Un test de collision de la configuration calculée est ensuite effectué. Si après un nombre défini de générations aucune configuration n'est valide (sans collision et respectant la contrainte de cinématique inverse) la simple prise est rejetée, sinon, dès qu'une prise valide est générée, un score de la configuration est calculé (voir Section 5.3.2) et la simple prise est retenue pour la génération des double prises.

5.4.3 Configuration d'échange

Afin de générer la configuration d'échange de l'objet à partir des listes de doubles prises, il est nécessaire de calculer où cet échange doit avoir lieu. La position d'échange est d'abord calculée. L'orientation de l'objet est ensuite déterminée pour faciliter la génération de la configuration d'échange.

5.4.3.1 Position de l'objet

Parmi l'ensemble des positions d'échange possibles, nous proposons une méthode permettant d'en sélectionner une selon un critère spécifique. Un critère simple serait de sélectionner une des positions d'échange se trouvant sur la droite formée par les positions initiale et finale de l'objet. Cependant, ce critère ne prend pas en compte la cinématique du robot : la position d'échange peut être hors de l'espace de travail commun des deux bras. Nous proposons de choisir une position d'échange minimisant les distances (dans l'espace de travail) parcourues par les poignets lors de la réalisation de la tâche. En effet, ce critère permet d'éviter des positions d'échange trop éloignées du robot. Dans cette section, le centre géométrique de l'objet à manipuler et le poignet sont considérés confondus.

La Figure 5.6 montre les mouvements des poignets dans l'espace cartésien à minimiser. Nous supposons dans l'exemple que le bras Arm_1 saisit l'objet et que le bras Arm_2 le dépose. La distance parcourue par le poignet du bras Arm_1 est :

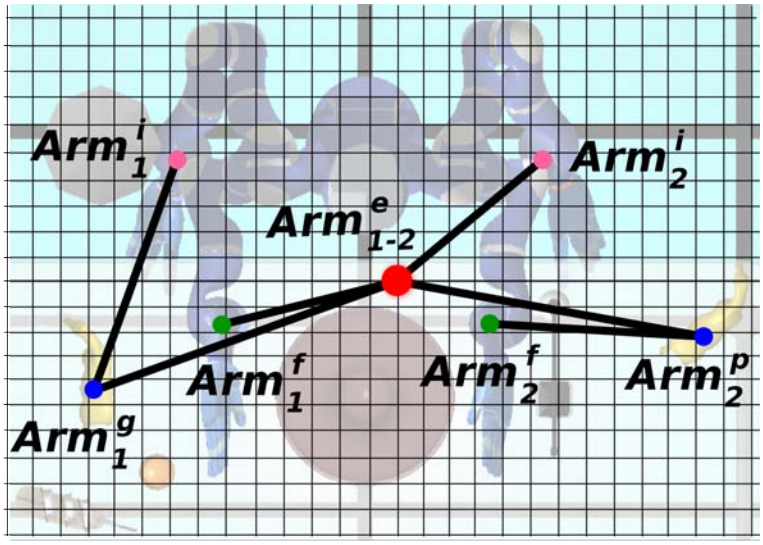

FIGURE 5.6 – Vue de dessus d'une tâche de prise et pose montrant les distances élémentaires à minimiser à l'aide d'une grille 3D afin de calculer la position d'échange (le cercle rouge au centre). Les notations correspondent à des configurations clés des bras (configuration [i = *initiale*, f = *finale*, g = *de prise*, e = *d'échange*, p = *de pose*])

$$d(Arm_1) = \|Arm_1^g - Arm_1^i\| + \|Arm_{1,2}^e - Arm_1^g\| + \|Arm_1^f - Arm_{1,2}^e\| \tag{5.4}$$

Avec Arm_1^i, Arm_1^g, $Arm_{1,2}^e$ et Arm_1^f la position du poignet du bras Arm_1 lorsque le robot est respectivement en configuration initiale, de saisie, d'échange et finale. De même pour le poignet du bras $Arm2$ la distance est :

$$d(Arm_2) = \|Arm_{1,2}^e - Arm_2^i\| + \|Arm_2^p - Arm_{1,2}^e\| + \|Arm_2^f - Arm_2^p\| \tag{5.5}$$

Avec Arm_2^i, $Arm_{1,2}^e$, Arm_2^p et Arm_2^f la position du poignet du bras Arm_2 lorsque le robot est en configuration initiale, d'échange, de pose et finale. La recherche de $Arm_{1,2}^e$ revient à résoudre le problème d'optimisation suivant :

$$\min_{Arm_{1,2}^e \in \mathbb{R}^3} (d(Arm_1) + d(Arm_2)) \tag{5.6}$$

Comme la dimension de l'espace de recherche est petite, l'utilisation d'une grille 3D est suffisante pour minimiser ce critère. La distance $d(Arm_1) + d(Arm_2)$ est calculée considérant chaque point de la grille comme une position d'échange de l'objet. Afin de valider une position d'échange de l'objet dans la grille, un test de collision de la sphère englobante de l'objet centrée sur cette position avec les obstacles de l'environnement est effectué. Lorsque la sphère est en collision, la position est rejetée. Les positions d'échange sont sélectionnées dans l'ordre croissant du critère de distance auquel elles sont associées. Le processus est arrêté une fois qu'une position d'échange valide de l'objet est obtenue.

5.4.3.2 Orientation de l'objet

L'orientation de l'objet est calculée afin de faciliter la génération de la configuration d'échange contrainte par une fermeture cinématique formée par les bras et l'objet (*e.g.* éviter une configuration d'échange où le robot croise les bras). L'idée est d'utiliser les informations fournies par la double prise (le repère $R_{doubleGrasp}$ dans la Figure 5.7 (4)) et par la position relative de l'objet par rapport au torse du robot (le repère R_{torso} dans la Figure 5.7 (4)).

L'orientation de l'objet est obtenue en alignant le repère orthonormé de la double prise $R_{doubleGrasp}$ avec le repère orthonormé du torse R_{torso}. L'axe des abscisses "X" du repère R_{torso} est déterminé par la position du torse et la position d'échange de l'objet. L'axe "Z" de ce repère est pris vertical allant de la base à la tête du robot. Le repère de double prise $R_{doubleGrasp}$ est déterminé par ceux des simples prises associés (Figure 5.7 (1)). Dans le cas général (Figure 5.7 (2)), l'axe des ordonnées "Y" est déterminé par la position relative des deux simples prises. L'axe "Z" est obtenu en sommant ceux des repères de simple prise. Deux cas particuliers illustrés Figure 5.7 (3), sont également traités. Ces cas apparaissent lorsque les axes "Z" des deux simples prises sont colinéaires.

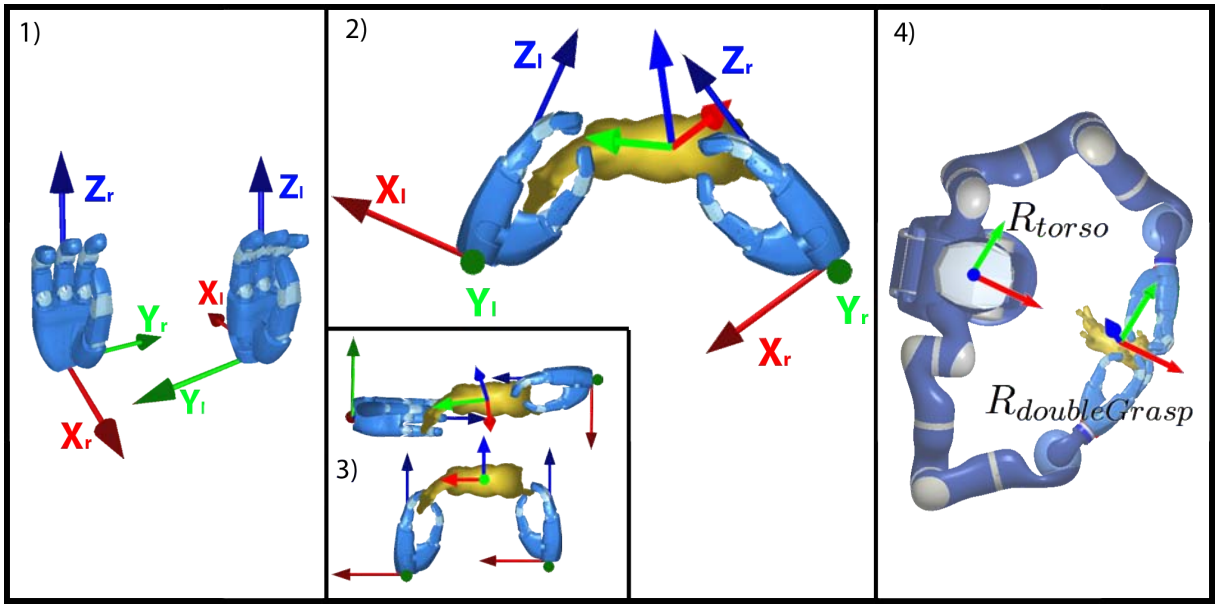

FIGURE 5.7 – Vecteurs directeurs des mains (1), le calcul d'une direction de double prise dans le cas général (2) ainsi que deux cas particuliers (3) et une configuration d'échange de l'objet (4).

5.4.3.3 Génération de la configuration d'échange

Une fois une position d'échange p^{exch}_{object} calculée (position + orientation de l'objet), la configuration du robot est générée par échantillonnage aléatoire des degrés de liberté du torse. La configuration des bras est ensuite déterminée en résolvant le problème de cinématique inverse pour chaque bras, caractérisé par la prise et la position d'échange de l'objet d'une part et

la configuration du torse d'autre part. La position et l'orientation de l'objet sont également échantillonnées dans le proche voisinage de la meilleure p_{object}^{exch} calculée. Cet échantillonnage de l'objet permet de trouver une configuration du robot valide et sans collision plus rapidement tout en restant proche de la position d'échange optimale. La Figure 5.7 (4) illustre une configuration d'échange générée.

Les configurations d'échange ainsi générées sont triées suivant le score des doubles prises associées.

5.4.4 Résolution des requêtes de planification

Une fois les configurations de prises générées (voir Section 5.4.2), les quatre requêtes présentées sont séquentiellement exécutées par le module de planification de mouvement. Les configurations d'échanges sont sélectionnées itérativement dans la liste calculée. Dans le cas où le planificateur échoue à résoudre une requête, la configuration suivante est sélectionnée.

Partant d'une configuration de double prise et les configurations de prise et pose associées, un chemin solution aux quatre requêtes de planification est cherché dans le réseau précalculé. Nous rappelons que ce réseau probabiliste est calculé sans la prise en compte de l'objet. Durant cette phase de recherche, il se peut qu'aucun chemin solution ne soit trouvé dans le graphe. Une étape d'enrichissement du graphe est alors effectuée cette fois-ci en prenant en compte l'objet lors des tests de collision. Les nœuds et les arêtes ajoutés pendant la méthode d'enrichissement sont sauvegardés dans le réseau probabiliste pour une future utilisation (lorsque l'objet manipulé est le même que celui qui a servi à leur création). Le graphe ainsi construit contient à la fois des chemins locaux valides pour le robot et pour l'objet manipulé (enrichissement du graphe) et d'autres ne considérant que le robot (réseau précalculé). La validation des chemins solutions avec l'objet est détaillée dans la Section 5.4.5.

Lorsqu'une condition d'arrêt du planificateur (*e.g.* nombre limite de nœuds, temps de calcul, ...) est atteinte sans avoir trouvé de chemin solution pour une des requêtes, la résolution de la tâche à l'aide de la double prise choisie est considéré comme impossible et la double prise suivante est considérée pour la résolution de la tâche.

5.4.5 Validation des chemins solution pour l'objet

Afin d'obtenir des chemins solutions valides, les chemins locaux, les composants, doivent être testés vis-à-vis de l'objet. En effet, l'ajout d'un objet dans l'espace de travail du robot modifie l'espace de configuration sans collision du robot. L'espace de configuration change également lorsque le robot transporte l'objet avec une prise donnée. Le changement de la prise influe également (prise de l'objet de deux façons différentes avec la même main ou avec l'autre main). La validation du réseau probabiliste précalculé en entier est une opération très coûteuse. Une méthode de validation paresseuse des nœuds et des arêtes [Bohlin 00, Sánchez 02, Jaillet 04] est utilisée.

Les chemins locaux composant le chemin solution sont testés en collision. Ce test de collision est uniquement effectué entre le robot et l'objet dans le cas des mouvements de saisie et de

retour à la configuration de repos, ainsi qu'entre l'objet et l'environnement dans le cas des mouvements de transport et de pose. Si une collision est détectée sur certains chemins locaux, une replanification locale de la portion en collision est effectuée.

Les configurations libres du chemin solution autour des portions en collision sont déterminées. Un chemin alternatif entre ces deux configurations déconnectées est ensuite cherché dans le réseau précalculé. Si aucune solution n'est trouvée, une reconnexion locale est effectuée en utilisant des planificateurs du type RRT [LaValle 01, Cortés 08]. Lors de cette reconnexion locale, l'objet est pris en compte et le test de collision est effectué pour le robot et l'objet avec l'environnement ainsi que pour le robot avec l'objet.

Le chemin est itérativement modifié jusqu'à ce qu'il ne soit plus composé que de chemins locaux valides. La Figure 5.8 illustre ces deux stratégies. Dans le Cas A, la portion du chemin est invalidé par un obstacle du au changement de l'espace de travail introduit par l'objet. Un chemin solution alternatif a été trouvé sans avoir recours à la méthode de reconnexion locale nécessaire dans le Cas B.

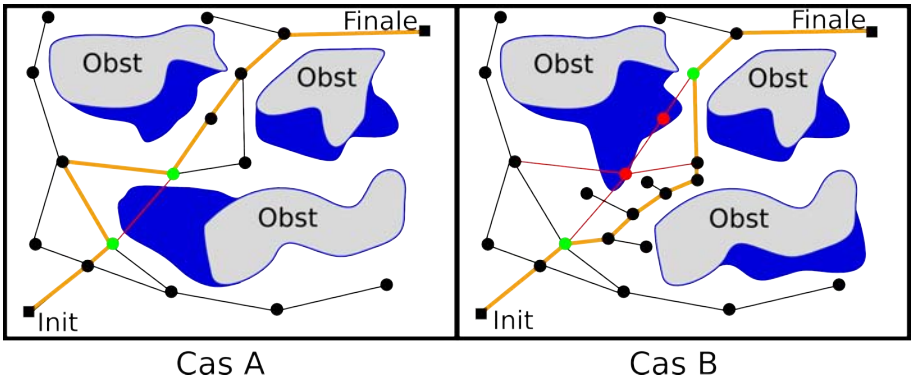

FIGURE 5.8 – Deux cas illustrant le choix des configurations le long du chemin autour de portion en collision (Points Verts) et les stratégies de replanification durant l'exécution d'une requête. Les obstacles en gris sont ceux lorsque l'objet n'est pas pris en compte lors de la planification et les obstacles en bleu lorsqu'il est pris en compte. Dans le Cas A, l'obstacle invalide seulement un chemin local de la trajectoire. Un chemin alternatif peut directement être trouvé dans le graphe précalculé. Dans le cas B, l'obstacle bloque un ensemble de configurations et de chemins locaux. Aucun chemin alternatif n'est présent dans le graphe. La phase d'enrichissement est nécessaire pour trouver un chemin contournant l'obstacle.

Notre implémentation utilise la même structure de données pour sauvegarder les réseaux probabilistes et les arbres de diffusion. Il est donc facile d'enrichir un graphe calculé par des méthodes du type PRM avec des méthodes du type RRT pour efficacement reconnecter les différentes composantes connexes.

5.5 Résultats

Afin d'évaluer la performance du planificateur, nous considérons l'exemple d'une tâche de prise et pose avec le robot Justin [Ott 06] équipé de deux mains SAHands, illustré par la Figure 5.1. Les bras du robot sont composé de sept degrés de liberté chacun et sont montés sur un torse de trois degrés de liberté. Les mains sont composées de quatre doigts à trois degrés de liberté chacun, plus une base de pouce mobile (treize degrés de liberté pour chaque main). En considérant que les articulations du cou sont fixes, le robot comporte quarante trois degrés de liberté. L'objet à manipuler est un corps non convexe ayant plusieurs parties (une statuette de cheval, dont le modèle 3D a été simplifié à 672 vertices et 1334 triangles).

TABLE 5.1 – Temps de calcul des différentes étapes de la méthode

Calcul	Hors-ligne			En ligne				
	Simple prise		Réseau	Config. de saisie/pose		Config.	Planification	Validation
	Droite	Gauche	proba.	Droite	Gauche	d'échange		des chemins
Temps	1 min	1 min	5 min	4.2 sec		2.6 sec	8.4 sec	3.2 sec
Prises	17	22						
Nœuds			1700				199	36
Configs.				4	7	4		

Durant cette tâche, Justin saisit l'objet à sa droite et le place derrière la lampe de bureau à sa gauche. Le lampadaire et le vase contraignent les mouvements du robot et la configuration d'échange. Les temps de calcul des différentes étapes de la méthode sont présentés dans la Table 5.1. La génération de prises étant indépendante de l'environnement, les tests sont exécutés avec les mêmes listes (une pour chaque main) de simples prises précalculées. La liste de prises de la main droite contient dix-sept prises valides et celle de gauche en contient vingt-deux, chaque liste étant générée (hors-ligne) en une minute [2] environ. Le nombre de repères de prise échantillonnés sur la grille autour de l'objet est de 2500. La différence entre le nombre de prises générées et le nombre de repères échantillonnés vient principalement du nombre de degrés de liberté de la main ainsi que des collisions possibles entre les doigts. La forme complexe de l'objet est également un facteur. La différence de nombre de prises généré entre la main droite et la main gauche, résulte de la non symétrie de l'objet contraignant d'avantage la main droite que la main gauche. Dans le scénario présenté, étant donné les listes de prises, le planificateur ne retient en ligne que quatre (droite) et sept (gauche) configurations de simples prises valides (accessibles et sans collision) en 4.2 secondes. La position d'échange de l'objet est calculée dans une grille 3D de dimension 30^3. Les doubles prises ainsi que leur configurations respectives sont générées en 2.6 secondes. La Figure 5.9 illustre un ensemble représentatif de configurations d'échange

2. Les résultats numériques sont une moyenne de 20 exécutions du planificateur. Le temps de calcul correspond à un processeur Dual-Core AMD Opteron 2222 cadencé à 3.0 GHz

générées par le planificateur. On remarque que la position d'échange de l'objet se trouve dans la même zone (près du torse du robot) quel que soit les simples prises sélectionnées. La Figure 5.10, (1 et 3) représentent les configurations de prise et de pose associées à la configuration de prise Figure 5.10 (2).

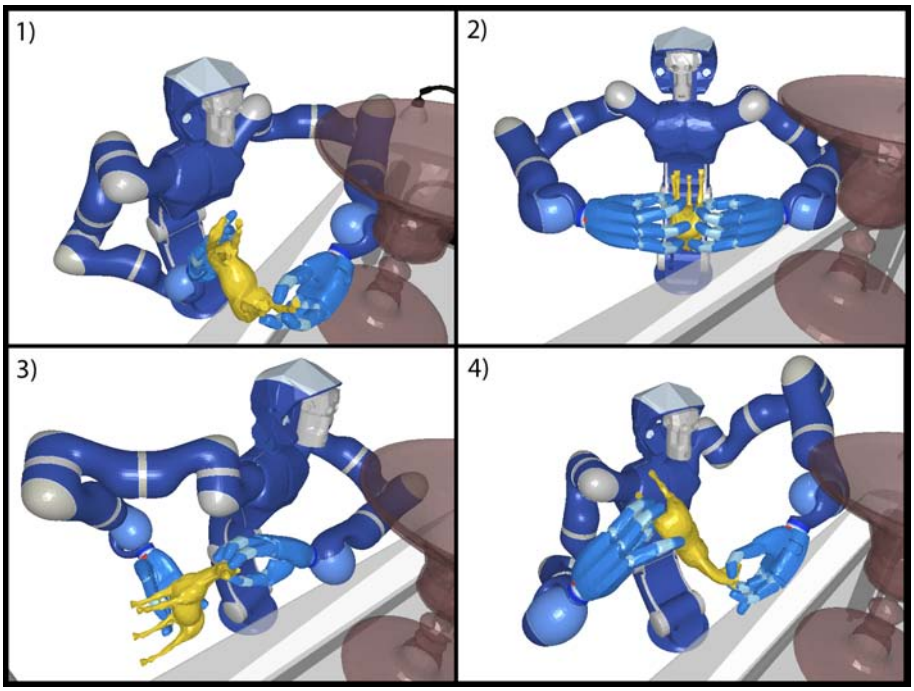

FIGURE 5.9 – Ensemble représentatif des configurations d'échange générées par le planificateur.

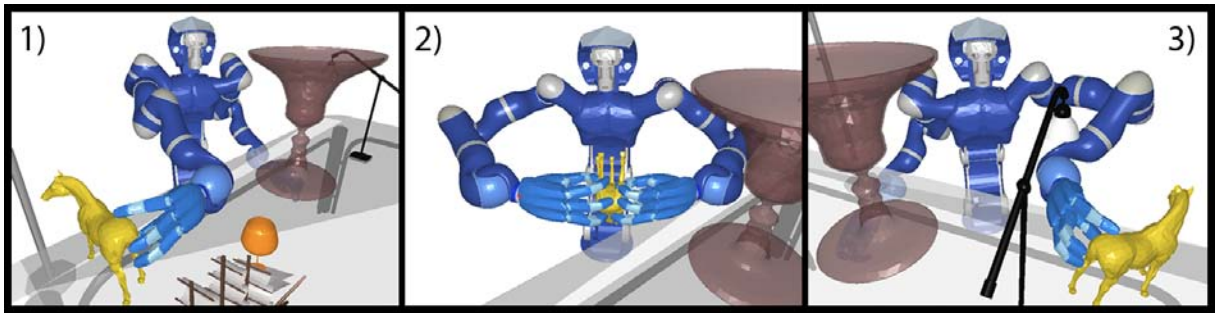

FIGURE 5.10 – Les configurations de prise et pose associées à la configuration de prise (2) sont illustré (1) et (2).

La Figure 5.11 montre deux exemples des meilleures doubles prises calculées pour l'échange de l'objet. Elles ont été calculées pour le même objet mais avec des positions initiales et finales de l'objet différentes (Vignettes (1 et 3) vs. (2 et 4)) et dans deux environnements différents (Vignettes (1 et 2) vs. (3 et 4)). Comme on peut le constater, l'objet est près du torse du robot en configuration d'échange et offre une meilleure maniabilité de l'objet quel que soit l'environnement et les positions initiale et finale de l'objet. La position d'échange diffère suivant les positions

initiale et finale de l'objet. L'ajout d'obstacles dans l'environnement influe également sur la position d'échange.

La Table 5.2 présente les résultats numériques obtenus pour la résolution des différentes requêtes proposées (saisie, transport, placement et repos). Le réseau probabiliste est précalculé en près de 5 minutes et contient 1700 nœuds. Les nœuds sont obtenus par la fusion de deux réseaux élémentaires générés à l'aide de la méthode Visib-PRM [Siméon 00], chacun contenant 50 nœuds. La Figure 5.12 montre les réseaux élémentaires précalculés (1 et 2) ainsi que le Super Graphe résultant (3). En utilisant ce réseau, le planificateur résout la tâche complète de manipulation en 11.6 secondes. En comparaison, l'utilisation d'un planificateur à requête unique [LaValle 01] permet de résoudre le même problème en 35 secondes. Ces deux résultats n'incluent pas le temps nécessaire pour le calcul des configurations de simples et doubles prises.

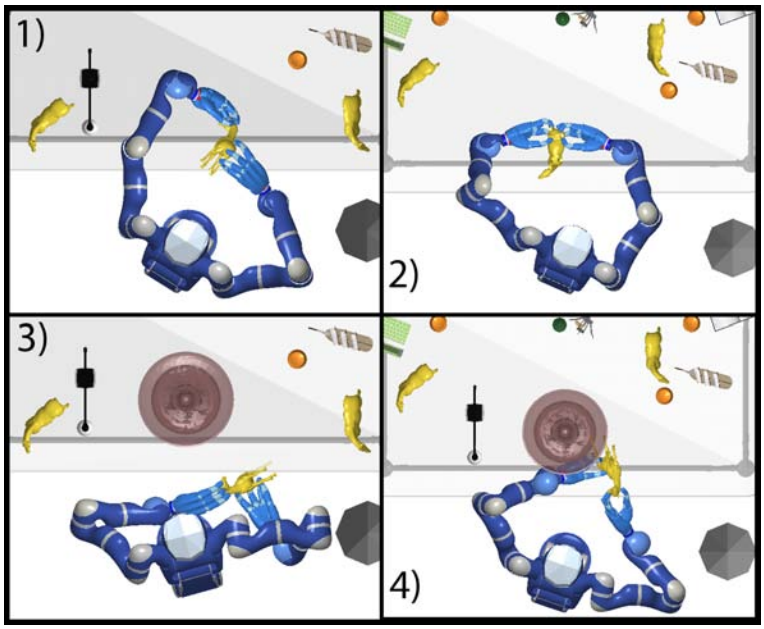

FIGURE 5.11 – Exemple illustrant des positions d'échange pour différentes positions initiales et finales de l'objet, prenant en compte le coût à minimiser et l'évitement d'obstacle.

La Table 5.2 montre également que le temps passé dans la validation des chemins pour les phases de transport et de pose est plus important que celui des deux autres phases. Ceci est dû à la modification plus importante de l'espace de configuration du robot lorsque l'objet est saisi par rapport à celle où l'objet est considéré comme un obstacle statique. En effet, un obstacle statique ajouté dans l'espace de travail du robot n'implique l'invalidation que d'un petit ensemble d'arêtes du réseau précalculé. Cependant, lorsque l'obstacle est transporté par le robot, de nombreuses arêtes sont concernées par ce changement.

Le temps nécessaire pour la planification (avant la validation des chemins) des phases de saisie et de repos est plus important à cause des configurations de prise et de pose qui sont fortement contraintes par les obstacles et par l'objet à manipuler.

TABLE 5.2 – Détails des temps de calcul pour la résolution des requêtes de planification

Problème		Requête	Validation	Total
Saisie	$n_{\text{nœuds}}$	65	3	68
	T (sec)	2.7	0.3	3
Transport	$n_{\text{nœuds}}$	53	18	71
	T (sec)	2.6	1.5	4.1
Pose	$n_{\text{nœuds}}$	32	12	44
	T (sec)	1.2	1.1	2.3
Repos	$n_{\text{nœuds}}$	49	3	51
	T (sec)	1.9	0.3	2.1

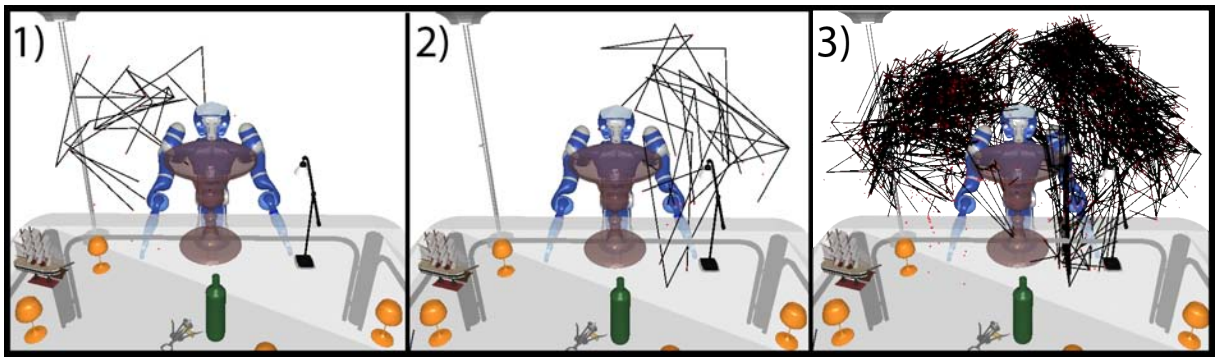

FIGURE 5.12 – Exemple de réseau probabiliste calculé grâce à l'approche présentée dans le Chapitre 3. Les réseaux élémentaires précalculés respectivement pour le bras droit et le bras gauche sont illustrés dans (1 et 2). Le Super Graphe résultant de la fusion de ces deux réseaux est présenté dans (3).

5.6 Conclusion

Nous avons présenté un planificateur permettant le calcul automatique de mouvement d'un robot muni de deux bras/mains pour la résolution d'une tâche de prise et pose nécessitant un échange de l'objet entre les deux mains. Le planificateur calcule également les configurations intermédiaires requises. L'intégration de plusieurs structures de données précalculées et réutilisables, comme les listes de prises et les réseaux probabilistes de bras, ont permis de réduire les temps de calcul en comparaison des méthodes à requêtes uniques. Les résultats en simulation montrent que le planificateur résout dans des temps raisonnables les problèmes de manipulation complexes impliquant le torse humanoïde Justin.

Une amélioration possible de l'algorithme est de traiter des problèmes où plusieurs poses intermédiaires sont nécessaires pour achever la tâche. Cette amélioration reposerait sur la formulation générale du problème "GRASP ∩ PLACEMENTS" [Siméon 04] en y introduisant la notion de doubles prises.

Une autre amélioration serait d'intégrer la possibilité d'effectuer des mouvements en chaîne cinématique fermée et des mouvements de glissement lors des phases de transport de l'objet. Ces mouvements permettraient de traiter des problèmes avec des objets lourds et/ou encombrants à manipuler (le robot saisi l'objet à une main le fait glisser jusqu'à la zone de travail commune aux deux bras avant de le transporter jusqu'à son placement final à l'aide des deux bras).

6 Expérimentations

Ce chapitre présente le planificateur de mouvement du LAAS-CNRS (Move3D [Siméon 01]) dans lequel nous avons intégré les méthodes décrites dans cette thèse, ainsi que deux expérimentations mises en œuvre sur des plate-formes robotiques réelles. Le planificateur Move3D propose des méthodes génériques de planification de mouvement capables de résoudre des problèmes impliquant des robots ayant des structures cinématiques complexes dans des environnements variés. Les principaux modules composant ce planificateur sont présentés dans la Section 6.1.

Deux fonctionnalités du planificateur Move3D, l'une traitant des tâches de prise et pose pour un manipulateur mobile et l'autre illustrant les capacités des méthodes probabilistes de base pour la prise en compte des modifications de l'environnement, ont été démontrées sur les robots Justin du DLR et le robot Jido du LAAS. Les spécificités de ces deux planificateurs sont respectivement détaillées dans la Section 6.2 et la Section 6.3.

6.1 Le planificateur de mouvement Move3D

Move3D est un outil générique dédié à la planification de mouvement dans des environnements tridimensionnels. Il nous a servi comme support pour développer les algorithmes de planification présentés dans les chapitres précédents. Move3D est composé de trois couches : une couche géométrique, une couche cinématique et une couche algorithmique.

La couche géométrique englobe deux modules essentiels au fonctionnement du planificateur : un module de modélisation et un module de détection de collision. Le module de modélisation permet, à partir de données utilisateur, de représenter les systèmes mécaniques ainsi que les scènes 3D où ces systèmes vont évoluer. Ces modèles sont utilisés par le détecteur de collision lors de la planification de mouvement pour déterminer si les configurations et les chemins locaux

générés par la couche cinématique sont en collision (auto-collisions et avec l'environnement). Plusieurs algorithmes de détection de collision sont implémentés dans ce module dont PQP [Larsen 00] et KCD [Geem 01] utilisés dans nos travaux.

La couche cinématique représente la bibliothèque des méthodes locales et des méthodes d'échantillonnage utilisables lors de la planification de mouvements. Ces méthodes assurent le respect des contraintes cinématiques du robot (*e.g.* non-holonomie, fermeture cinématique, butées articulaires, ...). Plusieurs méthodes locales ont été implémentées dans le planificateur Move3D dont une méthode d'interpolation "linéaire", une méthode de mouvements souples "soft-motion" [Broquère 10] et une méthode basée sur les courbes de "Reeds et Shepp" [Reeds 90]. De même, Move3D propose un ensemble de méthodes d'échantillonnage des configurations des systèmes comme la méthode "random" sélectionnant la valeur des articulations de façon aléatoire, la méthode "gaussian" [Boor 99] échantillonnant les configurations près des obstacles ou autour d'une configuration de référence, etc.

La couche algorithmique est composée des méthodes de planification de mouvements probabilistes de la littérature (*e.g.* PRM [Kavraki 96], RRT [LaValle 98], RRT-Connect [Kuffner 00], Visib-PRM [Siméon 00] ...). On y trouve également les algorithmes présentés dans les chapitres précédents. Le planificateur Move3D utilise la même structure de données pour sauvegarder les graphes et les arbres générés par les différentes méthodes de planification, ce qui permet le développement de méthodes de planification combinant ces deux types d'approche.

Les méthodes présentes dans les couches cinématique et algorithmique peuvent être vues comme des modules pouvant être combinés entre eux, suivant le problème et le système robotique à traiter. L'utilisateur peut donc décider de combiner la méthode PRM avec un échantillonnage gaussien, une méthode locale Reeds et Shepp et de détecter les collisions avec l'algorithme KCD. Cette modularité permet de créer des planificateurs, spécialisés dans la résolution de certains problèmes comme les planificateurs de prises, faisant appel aux différents modules de Move3D.

Move3D est également muni d'une interface graphique permettant à l'utilisateur de définir le problème de planification de mouvement à traiter (*e.g.* configurations initiale et finale du robot et de l'objet à manipuler) ainsi que de choisir la combinaison de modules à utiliser pour la planification.

Nous proposons dans les sections suivantes deux planificateurs de mouvement l'un traitant des problèmes de prise et pose à l'aide de manipulateurs mobiles (Section 6.2), et l'autre utilisant les différents modules de planification (couche algorithmique) pour prendre en compte des modifications de l'environnement (Section 6.3).

6.2 Planificateur intégré sur le robot Justin

Cette section présente un planificateur de mouvement pour les tâches de prise et pose à l'aide d'un manipulateur mobile intégré sur le robot mobile Justin du DLR dans le cadre du projet Phriends. Ce planificateur utilise les différents modules présents dans Move3D et en contient trois nouveaux afin de générer les chemins solutions de ces tâches de manipulation. Les modules du

planificateur sont génériques et peuvent être combinés entre eux suivant le problème à traiter : holonomie de la plate-forme, planification séparée de la plate-forme et de la partie haute du robot, manipulateur à un ou deux bras, génération des configurations de saisie et de pose de l'objet.

Le planificateur de tâche de prise et pose que nous avons développé est composé d'un module de génération de configurations de prise et pose, d'un module de planification de mouvement de la plate-forme et d'un module de planification de mouvement de la partie haute du robot (manipulateur). Le fonctionnement de ces trois modules sera détaillé respectivement dans les Sections 6.2.2, 6.2.3 et 6.2.3. La Section 6.2.1 présente l'enchainement des différents appels aux modules du planificateur nécessaires pour résoudre les tâches de prise et pose à l'aide d'un manipulateur mobile. L'intégration et les spécificités du robot mobile Justin sont présentées dans la Section 6.2.6

6.2.1 Vue globale du planificateur

Le planificateur proposé est basé sur la décomposition de la tâche en trois mouvements élémentaires :

1. Mouvement "d'approche" où le robot se déplace jusqu'à une configuration de la plate-forme à partir de laquelle l'objet peut être saisi ou posé.
2. Mouvement de "saisie" où le robot saisit l'objet en son placement initial.
3. Mouvement de "pose" où le robot dépose l'objet en son placement final.

Le planificateur enchaîne ces trois mouvements séquentiellement : un premier mouvement d'approche, puis de saisie, puis un second mouvement d'approche et finalement un mouvement de pose (voir la Section 6.2.5). La différence entre les deux mouvements d'approche réside dans le fait que le robot ne transporte pas d'objet lors de la première étape. Partant des positions initiale et finale de l'objet ainsi que des configurations initiale et finale du robot, le planificateur génère des configurations de saisie et de pose nécessaires à la résolution de la tâche. Les prises ne sont pas calculées automatiquement comme dans le Chapitre 5 mais sont supposées connues.

La Figure 6.1 représente l'architecture globale du planificateur ainsi que l'enchainement des différentes phases. Les configurations de saisie et de pose sont d'abord calculées par le module de génération de configurations. À partir de ces configurations, des appels aux modules de planification de la plate-forme et du manipulateur sont effectués. Si l'on souhaite avoir des mouvements coordonnés de la plate-forme et du manipulateur, les deux modules sont exécutés simultanément. Dans le cas contraire (mouvements découplés), les modules sont appelés séquentiellement par le planificateur. Les sous-chemins solutions générés par ces modules sont fusionnés afin d'obtenir le chemin solution final.

Ce planificateur a été intégré au robot Justin du DLR dans le cadre du projet Phriends. La Section 6.2.6 décrit cette intégration et montre des exemples de problèmes résolus.

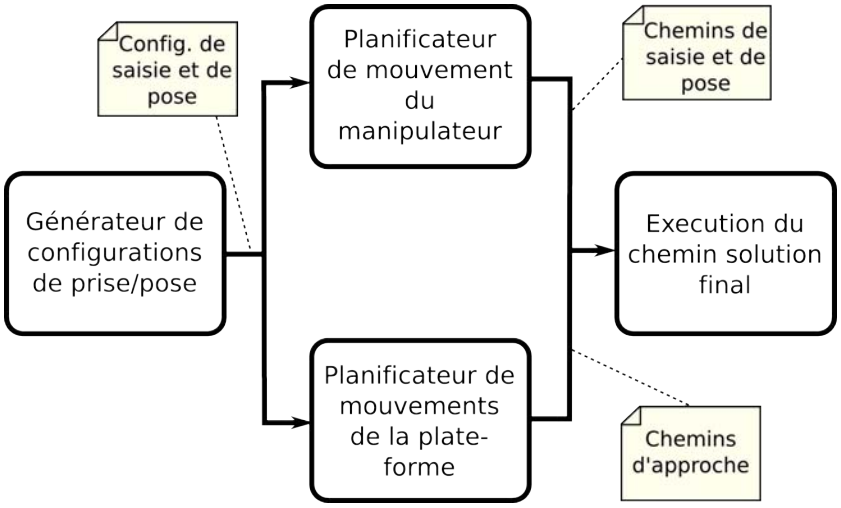

FIGURE 6.1 – Différents modules utilisés pour réaliser une tâche de prise et pose avec un manipulateur mobile

6.2.2 Générateur des configurations de saisie et pose

Afin de définir les requêtes de planification de mouvement correspondant aux différentes étapes de la tâche de prise et pose, des configurations valides (*i.e.* sans collision et respectant les contraintes internes du robot) de saisie et de pose doivent être générées. Ces configurations sont calculées à partir des positions initiale et finale de l'objet à manipuler et de la position des prises pour les bras du robot. Ce calcul est effectué par échantillonnage aléatoire des degrés de liberté du système. Il se décompose en deux étapes (détaillées ci-dessous) : la génération de la configuration de la plate-forme (position et orientation), puis de la configuration du manipulateur. Tant qu'une configuration valide du système en entier (plate-forme + manipulateur) n'est pas obtenue, le module de génération des configurations de saisie (ou de pose) itère l'appel à ces deux étapes. Notons que dans le cas d'un torse humanoïde fixe, seule la seconde étape est nécessaire.

6.2.2.1 Configuration de la plate-forme

La génération de la configuration de la plate-forme permet de favoriser les positions et orientations où le robot est face à l'objet. La position de la plate-forme est obtenue par échantillonnage aléatoire dans un domaine défini par une approximation des configurations limites du manipulateur (extensions maximale et minimale) et centré sur les positions initiale et finale de l'objet à manipuler (Figure 6.2). L'orientation de la plate-forme est également déterminée par échantillonnage aléatoire dans un domaine centré autour de la direction formée par la position de la plate-forme et la position de l'objet (*e.g.* dans la Figure 6.2, Cas A, cet angle est limité à $\pm 10^o$).

Suivant que l'objet doit être saisi (ou déposé) à un ou deux bras, le domaine d'échantillonnage de la position de la plate-forme est calculé différemment. Dans le cas d'un objet simple (*e.g.* un verre), la saisie (ou la pose) ne nécessite qu'un seul bras et peut être effectuée à partir de toutes

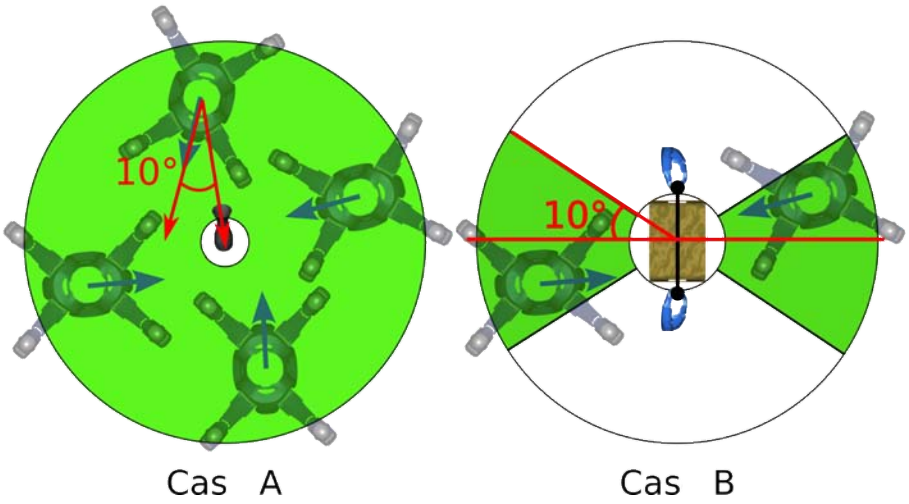

FIGURE 6.2 – Échantillonnage de la position de la plate-forme dans un domaine centré autour de la position de l'objet. Ce domaine est en forme d'anneau si la saisie de l'objet ne nécessite qu'une prise à une main (Cas A). Autrement, la plate-forme est échantillonnée sur des portions de cet anneau définies par la position des prises (Cas B). Dans les deux cas, l'orientation de la plate-forme est échantillonnée aléatoirement autour de la direction formée par la position de la plate-forme et celle de l'objet.

les directions autour de l'objet (sans considérer à ce niveau les obstacles de l'environnement). Le domaine d'échantillonnage (centré sur la position de l'objet) a donc une forme annulaire (Figure 6.2 Cas A). Cet anneau est défini par la taille de l'objet à manipuler (petit rayon) ainsi que par l'extension maximale du torse du robot (Figure 6.3, 1,46 m pour Justin). Dans le cas où la saisie (ou la pose) de l'objet nécessite deux bras (*e.g* un plateau), une contrainte de fermeture cinématique apparaît dans le système. Afin de faciliter l'échantillonnage des configurations satisfaisant les contraintes de fermeture cinématique du manipulateur, le domaine annulaire précédemment défini est restreint (Figure 6.2 Cas B) afin de garantir une position de la plate-forme face à l'objet. Les portions de l'anneau sont obtenues en limitant l'angle formé par la médiatrice du segment défini par le centre des deux prises et par la droite reliant le centre de la plate-forme au centre de l'objet. Dans le cas du robot mobile Justin, cet angle est limité à $\pm 10^o$.

Une fois la position et l'orientation de la plate-forme obtenues, un test de collision de cette dernière avec les obstacles de l'environnement est effectué afin de détecter au plus tôt les configurations invalides.

6.2.2.2 Configuration du torse

Pour une configuration de la plate-forme calculée, la seconde étape consiste à déterminer une configuration du manipulateur. Elle est générée par échantillonnage aléatoire des degrés de liberté du torse (si le manipulateur en possède un). La configuration des bras est ensuite obtenue en résolvant le problème de cinématique inverse pour chaque bras, étant donnés les degrés de

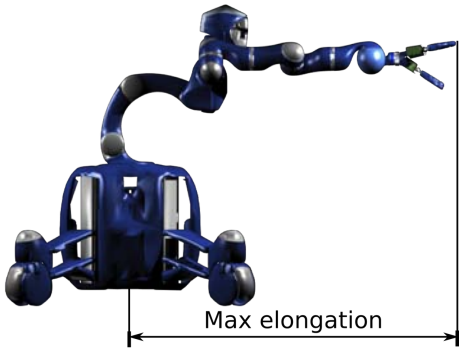

FIGURE 6.3 – L'extension maximale du robot Justin. Cette distance détermine le grand rayon de l'anneau sur lequel la position de la plate-forme est échantillonnée.

liberté échantillonnés (torse) et la position de l'effecteur pour la prise. Une fois obtenue, la configuration du manipulateur est testée en collision avec la plate-forme et les obstacles de l'environnement. Lorsque ce test échoue, ou que la configuration de la plate-forme ne permet pas au manipulateur d'atteindre l'objet, la génération de la configuration du système en entier est réitérée jusqu'au nombre limite d'itérations fixé par l'utilisateur au delà duquel la tâche est considérée non résoluble.

6.2.3 Planification de mouvement de la plate-forme mobile

Le module de planification de mouvement de la plate-forme mobile est basé sur les différents composants du planificateur Move3D. L'extension des techniques de planification de mouvement aux robots manipulateurs mobiles nécessite la prise en compte de contraintes cinématiques spécifiques (*e.g.* non-holonomie).

La planification de mouvement de la plate-forme est réalisée avec une des méthodes probabilistes fournies par Move3D (*e.g.* Visib-PRM [Siméon 00], Bi-RRT [Kuffner 00], ...). Afin de prendre en compte les contraintes de non-holonomie de la plate-forme, des méthodes locales dédiées (*e.g.* Reeds and Shepp [Reeds 90]) sont utilisées (Figure 6.4 (2)) à la place d'une simple interpolation linéaire des paramètres de la plate-forme (Figure 6.4 (1)). Certaines plates-formes holonomes ne permettent pas la réalisation de mouvements de translation et de rotation simultanés (*e.g.* la plate-forme mobile de Justin) mais permettent d'exécuter ces mouvements de façon découplée (translations et rotations pures). Le module de planification de la plate-forme propose, en plus des deux méthodes locales citées ci-dessus, une méthode locale permettant de générer ce type de mouvement (Figure 6.4 (3)).

6.2.4 Planification de mouvement du manipulateur

La planification de mouvement du manipulateur est réalisée avec une des méthodes probabilistes fournies par Move3D (*e.g.* Visib-PRM [Siméon 00], Bi-RRT [Kuffner 00], ...) ou avec la méthode "Multi-Graphe" du Chapitre 3 pour le cas de torses humanoïdes. Lorsque le robot

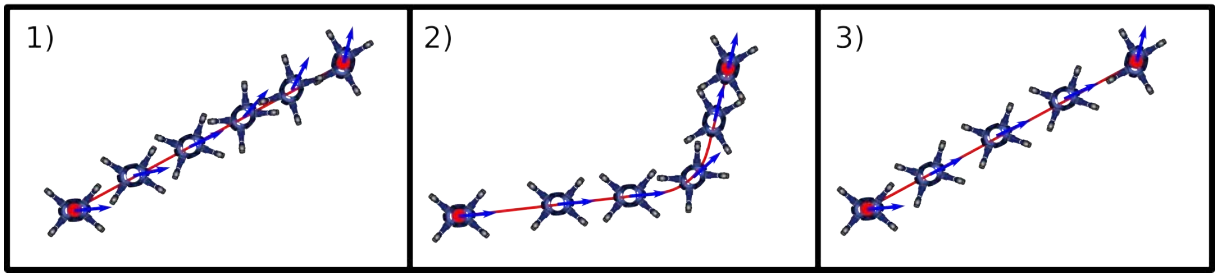

FIGURE 6.4 – Trois exemples de méthodes locales utilisées pour la planification d'une plate-forme mobile. Une méthode locale linéaire (1), une méthode locale basée sur les courbes de Reeds and Shepp [Reeds 90] (2) et une méthode locale basée sur les courbes de Reeds and Shepp avec un rayon de giration nul (3).

saisit l'objet à deux bras, une contrainte de fermeture cinématique de la chaîne apparaît (formée par les bras et l'objet). Dans ce cas, la méthode de planification pour chaînes fermées décrite au Chapitre 4 est utilisée.

6.2.5 Mouvements d'approche de saisie et de pose

Dans cette section nous décrivons l'enchaînement des différentes étapes pour planifier les mouvements d'approche, de saisie et de pose. À ce stade, on considère que les configurations de saisie et de pose ont été calculées comme indiqué dans la Section 6.2.2.

Le mouvement d'approche consiste, à partir de la configuration courante du robot (plate-forme + manipulateur), à positionner la plate-forme de telle sorte que le manipulateur puisse saisir ou déposer l'objet (configurations de saisie ou de pose). Suivant que l'on souhaite découpler les mouvements de la plate-forme et du manipulateur, les modules de planification sont appelés séquentiellement ou simultanément.

Dans le cas où les mouvements de la plate-forme et du manipulateur sont planifiés simultanément, la résolution de la tâche de prise et pose consiste alors à résoudre deux requêtes de planification :

– De la configuration initiale à la configuration de prise pour le robot seul
– De la configuration de prise à la configuration de pose pour le robot tenant l'objet

Dans l'autre cas (découplage des mouvements, voir Figure 6.5), la configuration du manipulateur est fixée afin d'accélérer la planification de mouvement de la plate-forme (configuration de roulement). Deux configurations de roulement sont nécessaires : une pour le mouvement d'approche pour la saisie (le robot ne maintient pas l'objet, Figure 6.5 (3 et 5)) et une pour le mouvement d'approche pour la pose (le robot maintient l'objet, Figure 6.5 (9 et 11)). Ces configurations de roulement sont renseignées par l'utilisateur lors de la définition de la tâche de prise et pose. Lors de la génération de configurations de prise et de pose, les configurations de roulement sont testées en collision avec l'environnement au même titre que les configurations de saisie et de pose du manipulateur. Les mouvements d'approche sont définis alors par deux requêtes de planification :

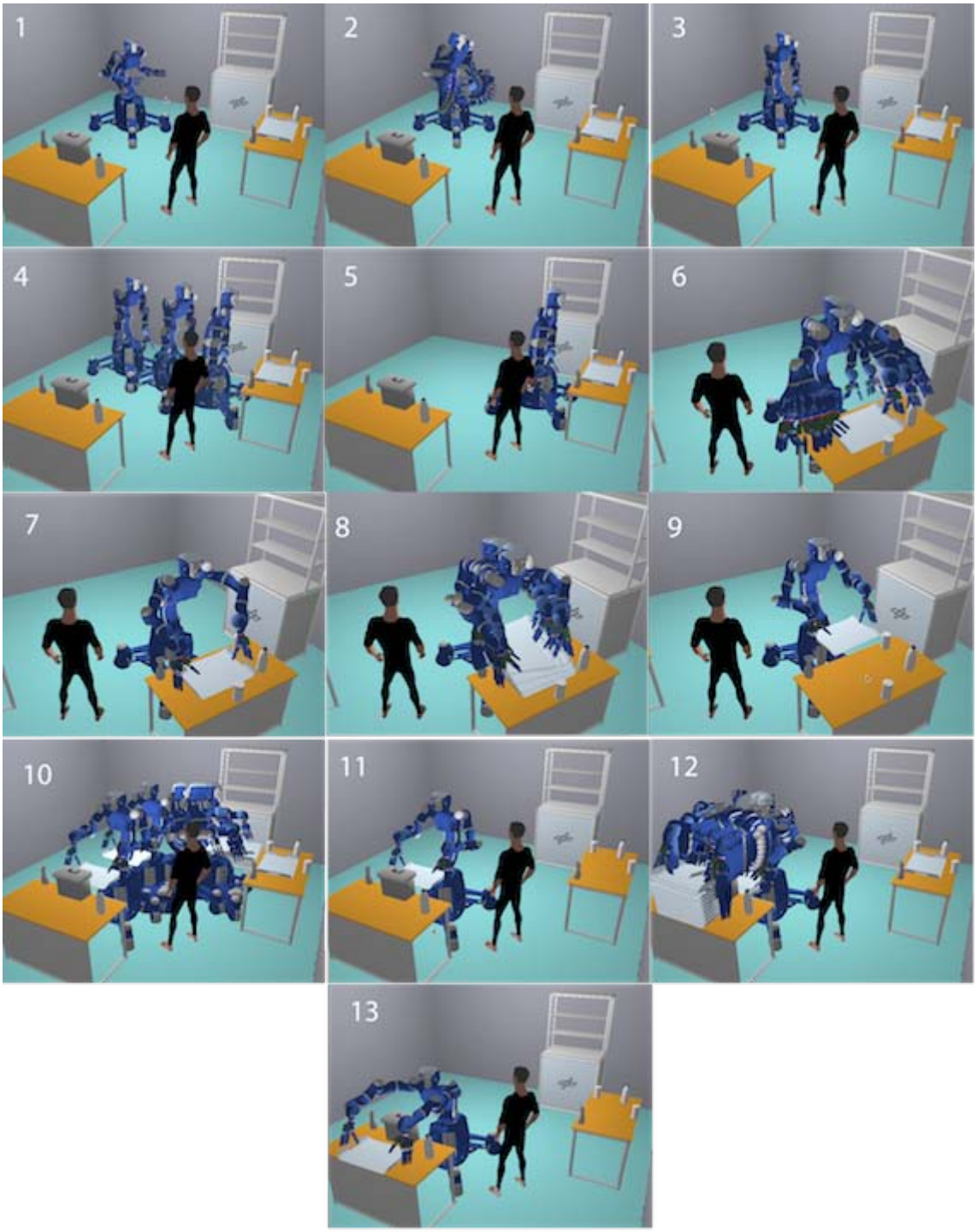

FIGURE 6.5 – Exemple de tâche de prise et pose automatiquement résolue par le planificateur présenté. La tâche consiste à transporter le plateau du bureau droit à l'autre (1 et 13). Les Vignettes 2, 4, 8 et 10 illustrent les phases d'approche, la Vignette 6 la phase de prise et la Vignette 12 la phase de pose. Les Vignettes 3, 5, 9 et 11 sont les configurations de roulement correspondant à la position initiale de la base, à la position de la base en position de prise (7) et à la position de pose (13).

– De la configuration courante du robot à la configuration de roulement associée à la position/orientation courante de la plate-forme. Durant cette requête seuls les degrés de liberté du manipulateur sont planifiés (Figure 6.5 (2 et 8)).
– De la configuration de roulement courante à la configuration de roulement associée à la configuration de saisie ou de pose. Durant cette requête seuls les degrés de liberté de la plate-forme sont planifiés (Figure 6.5 (4 et 10)).

Les mouvements de saisie et de pose consistent à planifier les mouvement du manipulateur d'une configuration de roulement à une configuration de saisie ou de pose (Figure 6.5 (6 et 12)). La tâche de prise et pose est ainsi résolue en six requêtes.

Le chemin solution de la tâche de prise et pose par le manipulateur mobile est obtenu en fusionnant les différents sous-chemins créés. L'utilisation des méthodes probabilistes pour générer ces sous-chemins impose une phase de lissage. Compte tenu de la séparation de la planification de la plate-forme et du torse du robot, le lissage ne peut s'effectuer sur le chemin solution final en entier. Il est effectué à l'aide de la méthode "ShortCut" [Laumond 98] sur chacun des sous-chemins séparément avant qu'ils ne soient fusionnés pour former le chemin solution final.

La Figure 6.5 montre un exemple de tâche résolue automatiquement par le planificateur Justin mobile avec une prise de l'objet à deux bras.

6.2.6 Expérimentations sur le robot Justin Mobile

Dans le cadre du projet Phriends, le planificateur proposé dans la Section 6.2 a été intégré lors d'un séjour effectué au DLR au robot Justin (Figure 6.6) dans le but de planifier des tâches de prise et de pose, ainsi que d'autres mouvements simples comme de revenir à une position de repos. Justin est un robot humanoïde développé comme plate-forme de recherche pour l'étude de la manipulation dextre dans des environnements humains standards.

Ce torse d'humanoïde mobile est doté de deux bras modulaires de sept degrés de libertés DLR-Lightweight-Robot-III (DLR-LWR-III) [Hirzinger 02] où des capteurs de couple ont été intégrés dans chacune des articulations. À vitesse maximale, ces bras peuvent supporter jusqu'à 7 Kg de charge ; à faible vitesse, 15 Kg sont admissibles par le système alors que la masse du bras ne dépasse pas 14 Kg. Chacun de ces bras est muni à son extrémité d'une main à quatre doigts DLR-Hand-II [Butterfaß 01]. Comme pour les bras, les articulations, au nombre de trois dans chaque doigt, sont équipées d'un capteur de couple et de capteurs de force / couple pour les articulations du bout des doigts. Le robot est également équipé d'une tête robotique à deux degrés de liberté utilisée comme capteur visuel de l'environnement du robot. Elle permet de détecter puis de suivre les objets, et de générer un modèle de l'environnement. Les manipulateurs ainsi que la tête sont montés sur un torse à quatre degrés de liberté. Seulement trois d'entre eux sont contrôlables, la quatrième servant à garder le support des bras manipulateurs perpendiculaire au sol. Contrairement aux autres articulations du robot, celles du torse sont entraînées par des courroies mais possèdent également des capteurs de couple.

Ce torse humanoïde est monté sur une plate-forme mobile afin d'augmenter son rayon d'action. Cette plate-forme est composée de 4 roues indépendantes et rétractables permettant au



FIGURE 6.6 – Le robot mobile Justin conçu au DLR

robot de passer par des ouvertures étroites. Les supports de roues sont munies d'amortisseurs offrant une meilleure stabilité à la partie supérieure du robot. À l'intérieur de la base se trouvent des télémètres lasers, les batteries ainsi que les ordinateurs commandant le robot.

L'architecture logicielle de Justin est construite autour d'un concept logiciel nommé aRD pour "agile Robot Development" [B.Bäuml 06]. Il fournit un accès aisé aux ressources système et est basé sur la vue abstraite du système robotique en tant qu'un "réseau de blocs de calcul et de liens de communication". Grâce aux bus de communication présents dans chacune des parties du robots (bras, torse, tête, plate-forme et mains), les instructions envoyées par les blocs temps réel aux différents moteurs du système peuvent être cadencées à la fréquence de 1 KHz. Les blocs non temps réel sont par exemple ceux gérant l'interface 3D et l'interface utilisateur ou encore ceux d'intelligence de haut niveau comme les planificateurs de mouvement ou le système de vision. D'autres blocs servent à la surveillance et à la simulation statique et dynamique des différentes parties du système, permettant ainsi au réseau de blocs d'émuler le fonctionnement du robot pour des tests en simulation.

Le planificateur a donc été intégré à cette structure par le biais d'un bloc de contrôle utilisateur en communiquant via des fichiers formatés. Les données transmises par le bloc au planificateur concernent la position des obstacles et de l'objet à manipuler, la configuration courante du robot, les repères de prise à utiliser, etc. Il est également possible de spécifier le type de tâche à réaliser dans le cas où seule une partie de la séquence de prise et pose doit être planifiée. Les positions des objets dans l'environnement et la géométrie du robot sont supposées connues par le planificateur. En retour, le planificateur transmet, également par l'intermédiaire d'un fichier, les configurations successives que le robot doit exécuter pour réaliser sa tâche.

Plusieurs problèmes se sont posés lors de l'intégration du planificateur au système. Toutes

les articulations de Justin sont conçues pour avoir une raideur variable. Cette variation de raideur permet au robot d'être peu dangereux pour un humain en cas de contact physique. Cependant, même si la raideur des articulations est maximale, un petit jeu subsiste et suivant les configurations du robot : la position des organes effecteurs peut être erronée. La localisation des objets par les caméras de la tête est également concernée par ce problème. Par exemple, la position d'un objet calculée peut enregistrer des écarts allant jusqu'à 2 centimètres suivant l'inclinaison du buste du robot. La solution trouvée pour palier ce problème est de détecter les obstacles en position du torse inclinée au maximum. De cette façon l'erreur est prise en compte lors de la localisation des objets.

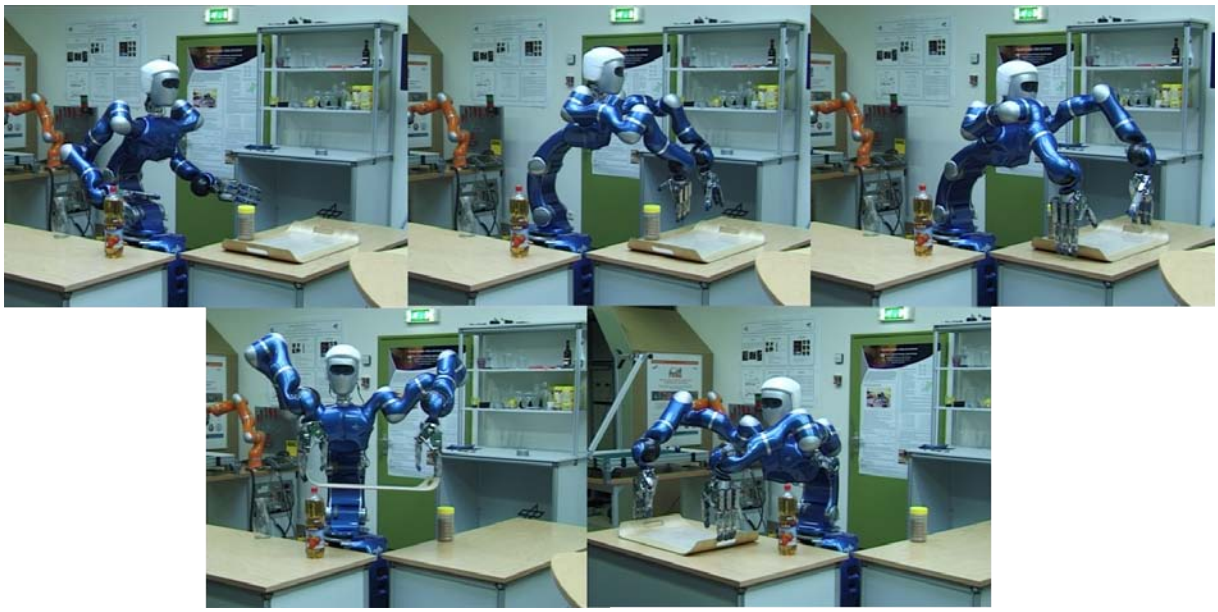

FIGURE 6.7 – Intégration du planificateur de tâche de manipulation mobile dans le robot Justin du DLR.

Un autre problème concerne la plate-forme mobile. La présence d'amortisseurs confère une bonne stabilité au torse robotique. Cependant le planificateur ne considère que les trois degrés de liberté actifs. Cet amortissement peut être bloqué, mais pas forcément parallèlement au sol. Quelques degrés de tangage ou de roulis au niveau de la plate-forme se transforment en quelques centimètres d'écart des organes effecteurs. Le réglage de la plate-forme a donc été empirique mais a néanmoins fonctionné.

La Figure 6.7 montre un exemple de tâche exécutée par Justin qui doit transporter le plateau d'un bureau à l'autre dans un environnement peu contraint. Pour des raisons de précisions évoquées ci-dessus, la base mobile est restée fixe lors de cette expérimentation.

6.2.7 Conclusion

Nous avons présenté dans cette section une méthode de planification de tâche de manipulation impliquant des manipulateurs mobiles dans des environnements variés. Le planificateur est vu

comme un ensemble de modules de planification et de génération de configurations pouvant être utilisés séparément ou simultanément pour résoudre différents problèmes de planification de tâche de prise et pose.

La validité de la méthode proposée a été démontrée par l'intégration du planificateur sur le robot Justin du DLR et a été démontrée lors de la revue finale du projet en exécutant avec succès une tâche de prise et pose dans différents environnements.

6.3 Planificateur intégré sur le robot Jido

Cette section présente un planificateur de mouvement, utilisant des méthodes probabilistes de base fournies par Move3D afin de prendre en compte des modifications de l'environnement, intégré au robot Jido. Ce planificateur étend la méthode de validation des chemins présentée dans la Section 5.4.5 afin de gérer des environnements faiblement dynamiques (apparition, disparition et déplacement des obstacles).

L'idée est d'utiliser un réseau probabiliste calculé hors ligne pour la partie statique de l'environnement. Ceci permet de trouver plus efficacement un chemin solution alternatif lorsqu'un nouvel obstacle est ajouté où que sa position change. L'opération de replanification peut être très rapide si le réseau probabiliste contient des chemins alternatifs ; autrement, la replanification consiste en la reconnexion des parties invalidées du chemin en utilisant un planificateur de type RRT. Le réseau précalculé combiné avec l'adaptation locale du chemin permet une replanification plus efficace. Un rappel de la méthode utilisée est présenté dans la Section 6.3.1 ainsi que les résultats de la meilleure performance de l'algorithme. L'intégration au robot Jido est présentée dans la Section 6.3.2.

6.3.1 Prise en compte de l'environnement changeant

La méthode présentée dans la Section 5.4.5 construit dans un premier temps, un réseau probabiliste à l'aide d'une des méthodes de planification de mouvement fournie par Move3D (*e.g.* PRM [Kavraki 96], Visib-PRM [Siméon 00], PDR [Jaillet 08a], ...). Un chemin solution au problème de planification de mouvement est ensuite cherché dans le graphe calculé.

Lors de l'apparition d'un obstacle dans l'environnement, les différents chemins locaux et configurations formant le chemin solution sont testés en collision avec ce nouvel obstacle. Dans le cas ou certains chemins locaux sont en collision, une stratégie de reconnexion locale vise à réparer les arêtes invalides et à trouver un chemin alternatif. Les configurations sans collision bordant ces chemins locaux invalides sont d'abord déterminées (les nœuds verts dans la Figure 6.8). Ensuite, une recherche dans le réseau probabiliste d'un chemin alternatif reliant ces deux configurations est effectuée. Si un chemin alternatif existe, le chemin solution est reconstruit à partir des chemins locaux valides le composant initialement et des chemins locaux alternatifs (Figure 6.8, Cas A). Sinon, l'enrichissement du réseau probabiliste est nécessaire (Figure 6.8, Cas B). Une méthode de diffusion de type RRT, est utilisée pour cet enrichissement. La généricité des méthodes et des structures de données fournies par Move3D, permettent de combiner l'utilisation de méthodes

créant des réseaux probabilistes et de méthodes de diffusion.

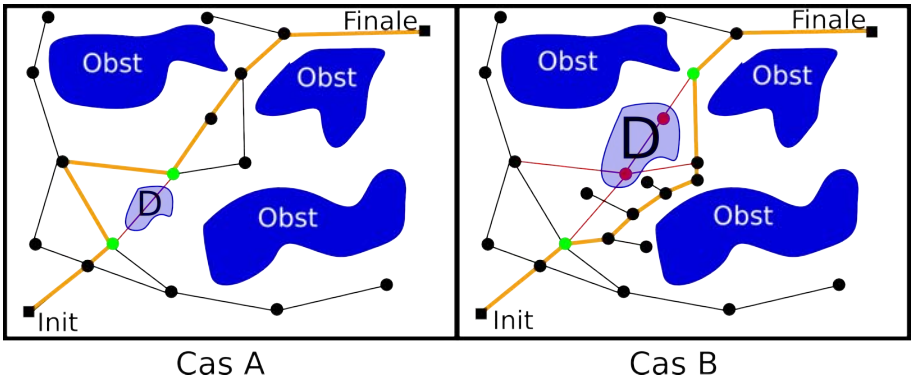

FIGURE 6.8 – Deux cas illustrant le choix des configurations bordant la collision (Points Verts) et les stratégies de replanification durant l'exécution d'une requête. Dans le premier cas, l'obstacle "D" invalide seulement un chemin local de la trajectoire. Un chemin alternatif peut directement être trouvé dans le graphe précalculé. Dans le cas B, l'obstacle "D" bloque un ensemble de configurations et de chemins locaux. Aucun chemin alternatif n'est présent dans le graphe. La phase d'enrichissement est nécessaire pour trouver un chemin contournant l'obstacle.

La capacité de replanification rapide de la méthode, suite à une modification de l'environnement, est démontrée sur un exemple avec le robot Justin. Dans le scénario proposé, Justin doit prendre un plateau sur une table et le donner à un humain assis près de lui. Ce scénario ne prend en compte que les mouvements du torse et des bras du robot (la plate-forme est fixe). Un second humain, considéré comme un obstacle, apparaît dans la scène. La Figure 6.9 illustre ce scénario. Dans cet exemple, le premier chemin calculé se trouve en collision avec le second humain. Une replanification est donc nécessaire pour obtenir un chemin alternatif.

Dans cet exemple, le précalcul d'un graphe pendant une minute et demi (générant 400 nœuds) permet de résoudre directement la requête (0.32 secondes) et de produire un nouveau chemin solution en près de deux secondes si une mise à jour du graphe est nécessaire. Les méthodes de base (PRM ou RRT) recalculant le chemin sont trois à sept fois plus lentes. Notons que les temps présentés correspondent à des mouvements de Justin formant une chaîne cinématique fermée imposée par la saisie du plateau des deux mains (voir Chapitre 4).

6.3.2 Intégration sur le robot Jido

Le planificateur en environnements changeants a été intégré à la plate-forme robotique Jido du LAAS-CNRS. Jido (Figure 6.10) est un robot mobile doté d'une plate-forme mobile MP-L655 non-holonome fabriquée par Neobotix. Elle possède deux roues motrices et trois roues de support multidirectionnelles. Cette plate-forme est munie de plusieurs capteurs dont des sonars, deux télémètres lasers ainsi que des odomètres. Sur la plate-forme, un bras KUKA Lightweight-Robot-IV monté comporte sept degrés de liberté. Comme pour les bras (DLR-LWR-III) équipant le robot Justin du DLR, chaque articulation est dotée d'un capteur de couple. Pour notre expérimentation, le bras est équipé d'une pince à deux doigts. Le robot est doté d'un banc

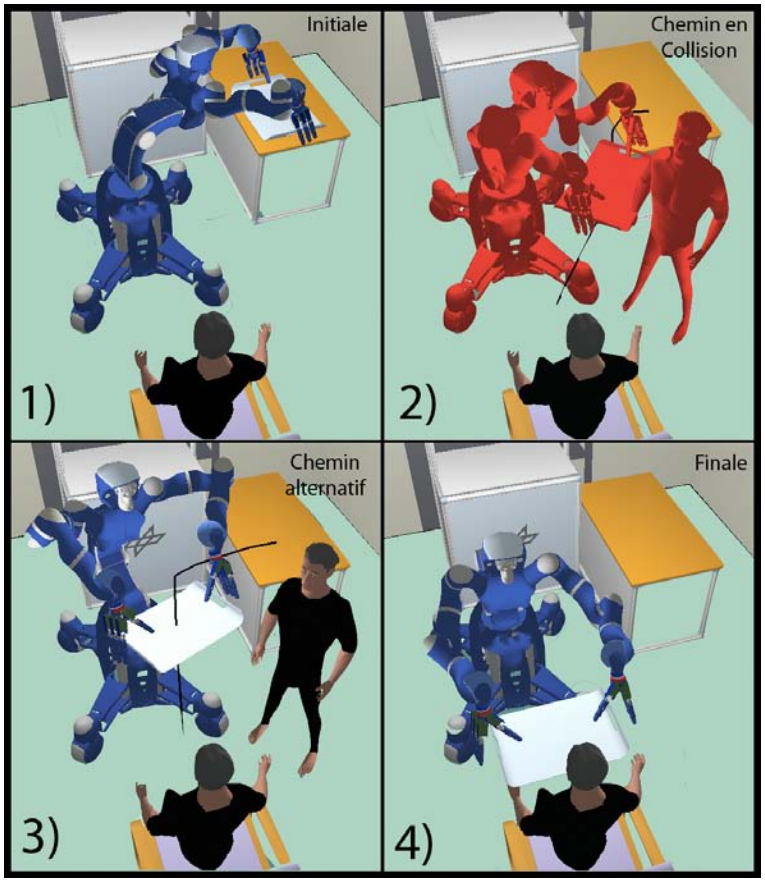

FIGURE 6.9 – Scénario de replanification. La première et dernière vignettes représentent les configurations initiale et finale de la requête. Lorsque l'humain bloque le chemin solution (2), le processus de replanification est lancé jusqu'à obtention d'un chemin alternatif valide (3).

de stéréovision ainsi que d'un capteur vidéo 3D (SwissRanger). La puissance de calcul est fournie par trois ordinateurs (cinq coeurs) utilisant un système d'exploitation Linux.

L'architecture logicielle de Jido est basée sur l'architecture du LAAS [Alami 98] composée de plusieurs modules GenoM (Générateur de Modules) [Fleury 97]. L'architecture du LAAS a été développée au fil des années, fournissant aujourd'hui un haut niveau de généricité et de modularité simplifiant ainsi le travail nécessaire pour l'intégration. Elle est décomposée en deux couches :

Couche Fonctionnelle : cette couche, contient l'intégralité des fonctions de perception et d'action du robot. Les boucles de contrôle et l'interprétation des données sont encapsulées dans des modules GenoM. Certains modules de cette couche ont un accès direct à la partie matérielle du robot (les capteurs et les moteurs) et offrent des services contrôlables via des requêtes. Chaque module communique à l'aide d'un bloc mémoire partagé nommé "poster".

Couche Décisionnelle : la couche décisionnelle contient les composants fournissant les capacités de décision du robot. On y trouve un planificateur de tâches générant des plans



FIGURE 6.10 – Le robot du LAAS-CNRS Jido

symboliques donnés au système de supervision. Le système de supervision présent dans cette couche, sélectionne le plan à suivre et s'assure de sa bonne exécution par les modules de la couche fonctionnelle.

Le module GenoM contenant le planificateur de mouvement présenté ci-dessus se trouve dans la couche fonctionnelle du système. L'avantage de cette architecture est qu'une grande partie des modules sont indépendants de l'architecture matérielle du robot et peuvent être utilisés sur plusieurs plate-formes. Une trentaine de modules GenoM sont disponibles sur Jido. Les plus importants pour notre expérimentation sont les suivants :

ViMan : Ce module permet la détection des objets par vision. Les objets sont équipés de "tags" (code barre 2D) sur certaines faces. Un traitement d'image et une stéréovision permettent au module de déterminer la position et l'orientation des objets connus de la scène.

Lwr : Les trajectoires générées pour le bras par le planificateur de mouvement sont exécutées par ce module. Les trajectoires sont reçues sous la forme d'une suite de configurations. Une interpolation linéaire entre les configurations est effectuée lors de l'exécution.

Le planificateur de mouvement a été intégré dans cette architecture et communique avec les modules présentés ci-dessus. Il est basé sur le planificateur Move3D [Siméon 01] développé au LAAS. Deux boucles d'exécution tournent simultanément dans le module. La première assure la mise à jour de la représentation de l'environnement dans le planificateur : les données relatives à la position du robot dans l'environnement et la configuration du bras ainsi que la position

des autres objets sont lues à chaque temps de cycle par le module dans les différents posters du système. La seconde boucle concerne la planification des mouvements et la détection de collision. Aussi, elle permet de construire les réseaux probabilistes utilisés pour résoudre les requêtes de planification et lors de l'exécution de s'assurer que le chemin réalisé par le robot est effectivement libre de collision.

La Figure 6.11, illustre deux trajectoires (1–3 et 4–9) exécutées par le robot Jido solutions du même problème de planification. Jido doit sortir son bras de l'obstacle bleu pour le positionner entre les tables haute et basse. La première trajectoire est exécutée sans changement de l'environnement physique du robot. Sans précalcul préalable, la résolution de cette requête nécessite 11 secondes [1] de calcul. Autrement avec un réseau probabiliste calculé en trois minutes, seules 3,5 secondes sont nécessaires. Lors de l'exécution de la seconde trajectoire, une boîte apparait dans l'espace de travail du robot et obstrue la trajectoire. Une trajectoire alternative est calculée afin d'éviter l'obstacle. À l'aide de la méthode présentée dans la Section 6.3.1, cette replanification nécessite 2,2 secondes. Notons que ces temps de calcul incluent le lissage de la trajectoire (2 secondes).

Le diagramme de la Figure 6.12 illustre le déroulement de la stratégie de replanification par le module Move3D. Un réseau probabiliste est considéré comme calculé. L'environnement du robot est continuellement mis à jour jusqu'à ce que la résolution d'une requête de planification soit demandée. La trajectoire solution est calculée puis envoyée au module Lwr pour son exécution. Tant que la fin de la trajectoire n'est pas atteinte, durant son exécution, le module de planification actualise l'environnement 3D du robot (position et orientation des objets de l'environnement ainsi que la configuration du robot) à partir des données fournies par les modules ViMan et Lwr. La trajectoire est également testée en collision après cette mise à jour. Afin d'accélérer ce test, seule une portion de la trajectoire suivant la configuration courante du robot est testée. Lorsque cette portion de trajectoire est détectée en collision, le robot est arrêté puis l'environnement du robot est mis à jour afin d'obtenir sa configuration à l'arrêt. Une trajectoire alternative est calculée suivant la méthode présentée dans la Section 6.3.1 partant de la configuration courante du robot. Cette trajectoire alternative est envoyée au module Lwr afin d'être exécutée en remplacement de celle préalablement détectée en collision.

6.3.3 Conclusion

Nous avons présenté dans cette section un planificateur de mouvement que nous avons développé et intégré dans l'architecture du robot afin de traiter des problèmes dans des environnements changeants. Ce planificateur est basé sur les méthodes de planification fournies par Move3D, y est intégré et peut être combiné avec les autres modules.

La validité de la méthode proposée a été démontrée en simulation sur le robot Justin et a été intégrée sur le robot Jido du LAAS-CNRS afin de planifier les mouvements du robot dans

1. Les temps de calcul correspondent au temps processeur du module de planification. La communication avec les autres modules (mise à jour de l'environnement, écriture et lecture de la trajectoire à exécuter) nécessite 2 secondes supplémentaires.

des environnements changeants.

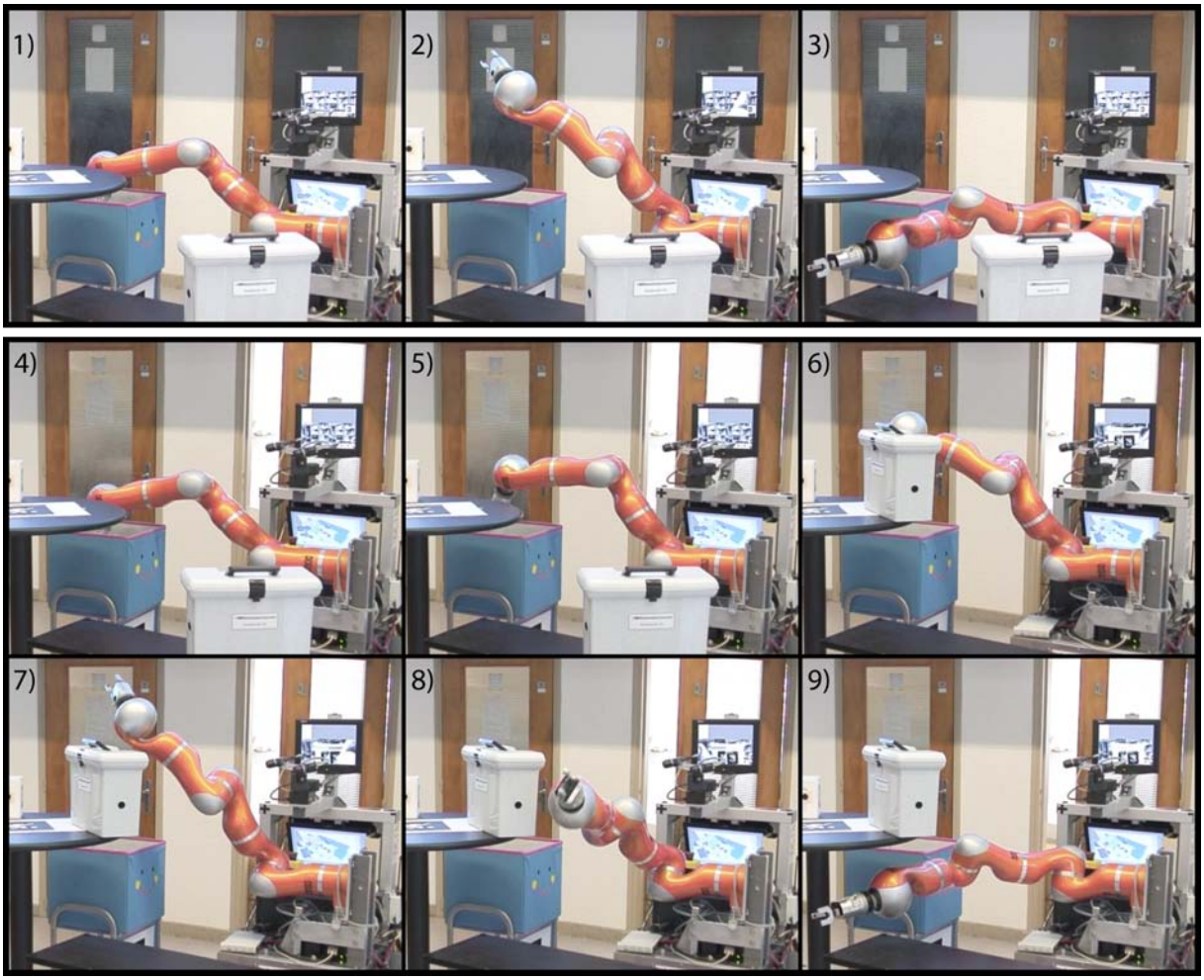

FIGURE 6.11 – Expérimentation réalisée sur le robot Jido. Le robot doit sortir son bras de l'obstacle bleu et le positionner entre les tables haute et basse. La trajectoire est calculée par le planificateur (1–3). Si un obstacle (ici une boîte) intersecte la trajectoire du robot, une trajectoire alternative est replanifiée puis exécutée sur le robot (6–9).

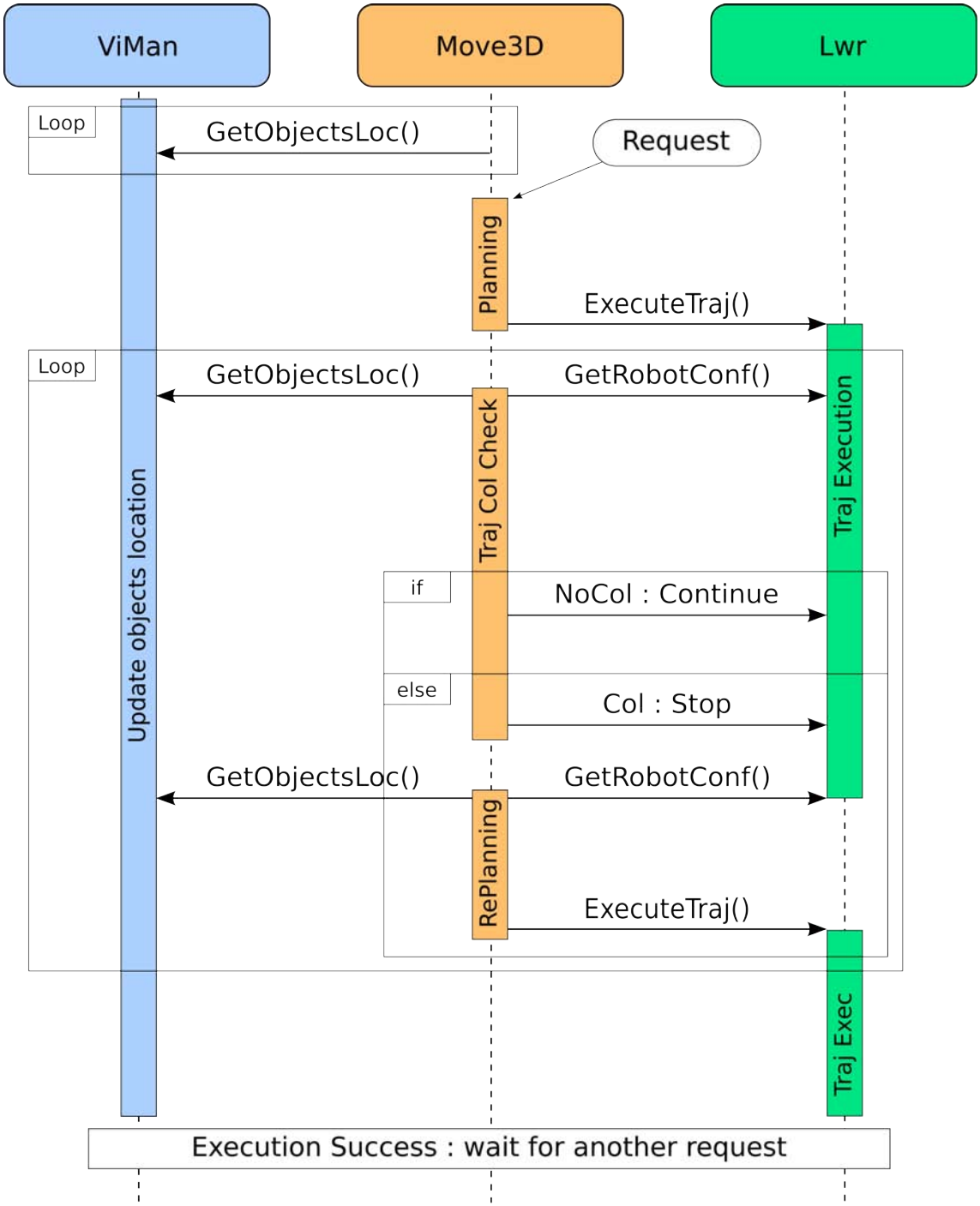

FIGURE 6.12 – Algorithme de replanification exécuté par le module Move3D.

7

Conclusion et perspectives

7.1 Conclusion

Les travaux présentés dans ce manuscrit portent sur la planification de mouvement pour la réalisation de tâches de manipulation d'objet par des torses humanoïdes. Les méthodes proposées conduisent à un planificateur de mouvement performant, capable d'exécuter des opérations de prise et pose d'objets avec un ou plusieurs bras.

Les méthodes développées sont génériques. Elles considèrent des systèmes articulés multi-bras de complexité variable, dans des environnements géométriques 3D statiques ou peu changeants. La généricité de nos approches vient de l'utilisation des méthodes probabilistes dont la performance a été démontrée dans de nombreuses applications. Les contributions de ce travail reposent sur l'extension de ces méthodes à des systèmes multi-bras tels que des torses humanoïdes afin de résoudre des tâches de manipulation d'objets.

Nous proposons tout d'abord une technique de planification de mouvement performante basée sur la coordination des mouvements du système multi-bras. Elle exploite au mieux la structure du système en la divisant en parties élémentaires. Chaque partie est planifiée sans prendre en compte le reste du système à l'aide de méthodes probabilistes générant des graphes compacts du type de Visib-PRM [Siméon 01] ou PDR [Jaillet 08a]. Ces réseaux élémentaires sont ensuite fusionnés dans le but d'obtenir un graphe prenant en compte toutes les parties du robot. La recherche du chemin solution est ensuite effectuée dans ce graphe. En l'absence de solution, le graphe est enrichi. Les résultats obtenus sur le robot Justin du DLR montrent l'efficacité de cette méthode, qui s'avère jusqu'à soixante-dix fois plus rapide que des méthodes considérant toutes les parties du robot en même temps.

Une seconde contribution porte sur l'extension des méthodes de planification pour des robots présentant des chaînes cinématiques fermées. Ces boucles cinématiques apparaissent dans le système lorsque, par exemple, le torse humanoïde saisit un objet avec plusieurs bras. Cette méthode traite explicitement les configurations singulières des manipulateurs, offrant ainsi une meilleure maniabilité de l'objet. Deux versions sont proposées : la première utilise des réseaux probabilistes du type PRM et la seconde des arbres de diffusion du type RRT. La méthode résout de façon efficace des problèmes nécessitant la reconfiguration des manipulateurs, comme le montrent les simulations effectuées sur le torse humanoïde Justin.

Finalement une approche est proposée pour la planification de tâches de prise et pose d'objets par un torse humanoïde dans le cas où les contraintes imposées par la tâche nécessitent le passage par une double prise afin de transférer l'objet d'une main à l'autre. La méthode génère automatiquement les configurations intermédiaires du robot, les positions initiale et finale de l'objet ainsi que la configuration d'échange de l'objet étant connus.

Deux planificateurs de mouvement ont été intégrés sur le robot Justin du DLR et le robot Jido du LAAS.

Le premier porte sur la résolution de tâches de prise et pose avec un manipulateur mobile. Le planificateur est suffisamment modulaire pour traiter les différentes contraintes posées par ces tâches comme la (non-)holonomie de la plate-forme ou encore la planification séparée de la plate-forme et du manipulateur. L'intégration de cette méthode a été effectuée sur le robot Justin au DLR, lequel est le démonstrateur du projet européen Phriends dans le cadre duquel s'inscrit cette thèse.

Le second planificateur s'adresse à la planification de mouvement en environnements faiblement changeants. En cas de collision du chemin solution originel, le graphe préalablement construit est utilisé pour trouver des chemins solution alternatifs. La performance de ce planificateur a été démontrée sur le robot Jido du LAAS-CNRS.

7.2 Améliorations et extensions

Améliorations algorithmiques

La méthode présentée dans le Chapitre 3 est séquentielle : les réseaux élémentaires sont construits pour un bras puis pour l'autre. Une extension possible de ces méthodes est la parallélisation de ces tâches. Chaque réseau élémentaire serait généré par une instance du planificateur. La parallélisation de l'étape de fusion pourrait également accroître les performances de l'algorithme. Dans la même idée, pour la méthode présentée dans le Chapitre 4, la parallélisation de la construction des couches peut être envisagée.

L'utilisation de méthodes générant des réseaux probabilistes pour planifier hors-ligne les mouvements de manipulateurs mobiles reste problématique particulièrement lorsque le manipulateur possède un grand nombre de degrés de libertés. Par exemple, pour une tâche où un torse humanoïde mobile doit se déplacer pour saisir un objet, la région de l'espace autour de l'objet doit être suffisamment explorée pour générer efficacement le mouvement de prise. Par contre,

une exploration plus grossière est suffisante pour l'espace réservé à la navigation de la plate-forme (le robot étant moins contraint dans cet espace). La prise en compte de ces paramètres lors de la planification pourrait accroître les performances des algorithmes.

Vers des tâches de manipulation plus complexes

Les méthodes de planification présentées dans ce manuscrit peuvent servir de base pour la résolution de tâches de manipulation plus complexes. Le planificateur décomposerait automatiquement la tâche de manipulation en plusieurs sous-tâches élémentaires (*e.g.* les tâches de manipulation présentées dans les Chapitres 3, 4 et 5) afin de réduire sa complexité. Suivant la tâche considérée, plusieurs paramètres peuvent entrer en compte comme par exemple : faut-il exécuter la tâche de manipulation avec un ou deux bras ? Les paramètres physiques de l'objet peuvent également influer sur la décomposition de la tâche : est-il préférable de soulever l'objet ou de le faire glisser à sa position d'arrivée ?

Une extension de la méthode présentée dans le Chapitre 5 serait de pouvoir traiter le cas d'un arrangement de plusieurs objets à transférer. Un ordre de transfert doit être déterminé suivant l'accessibilité des objets en leur position initiale ainsi qu'en leur position finale. Des positions intermédiaires peuvent être utilisées pour certains objets afin de pouvoir atteindre d'autres objets plus contraints en position finale. L'intégration de la méthode à la formulation générale du problème de manipulation [Siméon 04] serait également une solution.

Manipulation en environnement dynamique

Les environnements où les robots de service sont menés à évoluer sont composés d'obstacles statiques et/ou mobiles. Lors de l'exécution d'une tâche de manipulation, le robot modifie son environnement en manipulant les objets. Les graphes et les arbres générés par les méthodes proposées ne prennent que partiellement en compte ces environnements : c'est le cas pour la méthode présentée dans le Chapitre 5. Afin de tenir compte de cette dynamicité certaines méthodes [Brock 02, Jaillet 08a] peuvent être utilisées. Une extension de ces méthodes pourrait permettre de précalculer des graphes capturant la connexité de l'environnement statique ainsi que les chemins alternatifs. Ces chemins tout en étant "souples" pourront se déformer lorsque un objet mobile intersecte le chemin solution ou lorsque le robot saisit un objet induisant un changement de la topologie de l'espace des configurations.

Manipulation en présence de l'homme

Les méthodes de planification présentées traitent de problèmes où les robots évoluent dans des environnements "contrôlés". Le but de la robotique de service étant d'aider l'homme à réaliser certaines tâches ou de les réaliser à sa place, la prise en compte de l'homme lors des phases de planification est nécessaire. Une représentation cinématique et géométrique de l'homme pourrait être ajoutée à l'environnement du robot. Ainsi des tâches de manipulation coordonnées avec l'homme pourrait être réalisées par exemple pour transporter des objets encombrants ou

lourds. La méthode présentée dans le Chapitre 4 serait alors utilisée pour trouver des chemins solution : le système composé de l'homme et du robot contiendra des chaînes cinématiques fermées. D'autres tâches de manipulation tel que donner un objet à un humain peuvent être réalisées en s'appuyant sur les premiers travaux de planification de mouvement considérant les contraintes d'interaction homme-robot [Sisbot 08]. Des coûts relatifs à des critères de confort pour l'homme peuvent être pris en compte. Les méthodes d'échantillonnage et de recherche de chemins se baseront sur ces coûts pour produire des mouvements confortables pour l'homme.

Bibliographie

[Ahuactzin 98] J.M. Ahuactzin, K.K. Gupta & E. Mazer. Manipulation planning for redundant robots : a practical approach. J. Wiley, 1998. 17

[Alami 90] R. Alami, T. Siméon & J.-P. Laumond. *A geometrical approach to planning manipulation tasks. The case of discrete placements and grasps.* In International Symposium on Robotics Research, pages 453–463, Cambridge, MA, USA, 1990. MIT Press. 17

[Alami 95] R. Alami, F. Robert, F. Ingrand & S. Suzuki. *Multi-robot cooperation through incremental plan-merging.* In IEEE International Conference on Robotics and Automation, volume 3, pages 2573–2579, may 1995. 14, 17

[Alami 98] R. Alami, R. Chatila, S. Fleury, M. Ghallab & F. Ingrand. *An Architecture for Autonomy.* International Journal of Robotics Research, vol. 17, pages 315–337, 1998. 90

[Albu-Schäffer 07] A. Albu-Schäffer, S. Haddadin, Ch. Ott, A. Stemmer, T. Wimbock & G. Hirzinger. *The DLR lightweight robot : design and control concepts for robots in human environments.* Industrial Robot : An International Journal, vol. 34, no. 5, pages 376–385, 2007. 35

[Amato 98] N.M. Amato, O.B. Bayazit, L.K. Dale, C. Jones & D. Vallejo. *OBPRM : an obstacle-based PRM for 3D workspaces.* In International Workshop on Algorithmic Foundations of Robotics, pages 155–168. A. K. Peters, Ltd., Natick, MA, USA, 1998. 9, 45

[Amato 00] N.M. Amato, O.B. Bayazit, L.K. Dale, C. Jones & D. Vallejo. *Choosing good distance metrics and local planners for probabilistic roadmap methods.* IEEE Transactions on Robotics and Automation, vol. 16, no. 4, pages 442–447, August 2000. 10

[Barraquand 90] J. Barraquand & J.-C. Latombe. *A Monte-Carlo algorithm for path planning with many degrees of freedom.* In IEEE International Conference on Robotics and Automation, volume 3, pages 1712–1717, may 1990. 13

[Barraquand 91] J. Barraquand & J.-C. Latombe. *Robot Motion Planning : A Distributed Representation Approach.* International Journal of Robotics Research, vol. 10, no. 6, pages 628–649, December 1991. 7, 11, 13, 14, 18

[B.Bäuml 06] B.Bäuml & G. Hirzinger. *Agile Robot Development (aRD) : A Pragmatic Approach to Robotic Software*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3741–3748, 2006. 86

[Berenson 08] D. Berenson, J. Kuffner & H. Choset. *An optimization approach to planning for mobile manipulation*. In IEEE International Conference on Robotics and Automation, pages 1187–1192, may 2008. 18

[Bessiere 93] P. Bessiere, J.-M. Ahuactzin, E.-G. Talbi & E. Mazer. *The "Ariadne's clew" algorithm : global planning with local methods*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 2, pages 1373–1380, July 1993. 12

[Bicchi 00] A. Bicchi & V. Kumar. *Robotic grasping and contact : a review*. In IEEE International Conference on Robotics and Automation, volume 1, pages 348–353, 2000. 62

[Bohlin 00] R. Bohlin & L.E. Kavraki. *Path planning using lazy PRM*. In IEEE International Conference on Robotics and Automation, volume 1, pages 521–528, 2000. 10, 11, 12, 38, 71

[Boor 99] V. Boor, M.H. Overmars & A.F. Van Der Stappen. *The Gaussian sampling strategy for probabilistic roadmap planners*. In IEEE International Conference on Robotics and Automation, volume 2, pages 1018–1023, 1999. 9, 78

[Bounab 08] B. Bounab, D. Sidobre & A. Zaatri. *Central axis approach for computing n-finger force-closure grasps*. IEEE International Conference on Robotics and Automation, pages 1169–1174, May 2008. 64

[Brock 02] O. Brock, O. Khatib & S. Viji. *Task-consistent obstacle avoidance and motion behavior for mobile manipulation*. In IEEE International Conference on Robotics and Automation, volume 1, pages 388–393, 2002. 97

[Broquère 10] X. Broquère, D. Sidobre & K. Nguyen. *From Motion Planning to Trajectory Control with Bounded Jerk for Service Manipulator Robots*. In IEEE International Conference on Robotics and Automation, pages 4505–4510, May 2010. 78

[Butterfaß 01] J. Butterfaß, M. Grebenstein, H. Liu & G. Hirzinger. *DLR-Hand II Next Generation of a Dextrous Robot Hand*. In IEEE International Conference on Robotics and Automation, pages 109–120, October 2001. 86

[Canny 88] J.F. Canny. The complexity of robot motion planning. MIT Press, Cambridge, MA, USA, 1988. 7, 14

[Caurin 09] G.A.P. Caurin & L.M. Pedro. *Hybrid motion planning approach for robot dexterous hands*. Journal of the Brazilian Society of Mechanical Sciences and Engineering, vol. 31, no. 4, pages 289–296, October 2009. 19

[Chang 95] H. Chang & T.-Y. Li. *Assembly maintainability study with motion planning.* In IEEE International Conference on Robotics and Automation, volume 1, pages 1012–1019, may 1995. 8

[Chen 91] P.C. Chen & Y.K. Hwang. *Practical path planning among movable obstacles.* In IEEE International Conference on Robotics and Automation, volume 1, pages 444 –449, apr 1991. 17

[Choset 05] H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki & S. Thrun. Principles of robot motion : Theory, algorithms, and implementations (intelligent robotics and autonomous agents). The MIT Press, June 2005. 5, 7, 62

[Cortés 03] J. Cortés & T. Siméon. *Probabilistic motion planning for parallel mechanisms.* In IEEE International Conference on Robotics and Automation, volume 3, pages 4354–4359, sept. 2003. 15, 16

[Cortés 04a] J Cortés & T. Siméon. *Sampling-based motion planning under kinematic loop-closure constraints.* In International Workshop on Algorithmic Foundations of Robotics, Utrecht, Netherland, 2004. 15, 16, 39, 40, 41, 42, 43, 44, 48

[Cortés 04b] J. Cortés, T. Siméon, M. Remaud-Siméon & V. Tran. *Geometric Algorithms for the Conformational Analysis of Long Protein Loops.* Journal of Computational Chemistry, vol. 25, 2004. 16

[Cortés 08] J. Cortés, L. Jaillet & T. Siméon. *Disassembly Path Planning for Complex Articulated Objects.* IEEE Transactions on Robotics and Automation, vol. 24, no. 2, pages 475–481, april 2008. 13, 72

[Diankov 08] R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa & J. Kuffner. *BiSpace Planning : Concurrent Multi-Space Exploration.* In Robotics : Science and Systems, volume 4, Zurich, Switzerland, June 2008. 18

[Ding 07] D. Ding, H. Liu, X. Deng & H. Zha. *A dynamic bridge builder to identify difficult regions for path planning in changing environments.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2925–2931, November 2007. 9

[Erdmann 86] M. Erdmann & T. Lozano-Perez. *On multiple moving objects.* In IEEE International Conference on Robotics and Automation, volume 3, pages 1419–1424, april 1986. 14

[Esteves 06] C. Esteves, G. Arechavaleta, J. Pettré & J.-P. Laumond. *Animation planning for virtual characters cooperation.* ACM Transactions on Graphics, vol. 25, no. 2, pages 319–339, 2006. 15

[Faverjon 84] B. Faverjon. *Obstacle avoidance using an octree in the configuration space of a manipulator.* In IEEE International Conference on Robotics and Automation, volume 1, pages 504–512, march 1984. 7

[Fleury 97] S. Fleury, M. Herrb & R. Chatila. *GenoM : A Tool for the Specification and the Implementation of Operating Modules in a Distributed Robot Architecture*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 842–848, 1997. 90

[Garrido 08] S. Garrido, L. Moreno & D. Blanco. *Exploration of a cluttered environment using Voronoi Transform and Fast Marching*. Robotics and Autonomous Systems, vol. 56, no. 12, pages 1069–1081, 2008. 11

[Geem 01] C. Van Geem, T. Siméon & J. Cortés. *Progress Report on Collision Detection. In Second Year Delivrables of the MOLOG Project*. Rapport technique, LAAS, February 2001. 10, 78

[Gharbi 08] M. Gharbi, J. Cortés & T. Siméon. *A sampling-based path planner for dual-arm manipulation*. In IEEE/ASME International Conference on Advanced Intelligent Mechatronics, july 2008. 54

[Gil-Pinto 07] A. Gil-Pinto, P. Fraisse & R. Zapata. *Decentralized strategy for car-like robot formations*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4176–4181, October 2007. 14

[Guernane 09] R. Guernane & N. Achour. *An Algorithm for Generating Safe and Execution-Optimized Paths*. In International Conference on Autonomic and Autonomous Systems, pages 16–21, april 2009. 11

[Guitton 09] J. Guitton, J.-L. Farges & R. Chatila. *Cell-RRT : Decomposing the environment for better plan*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5776–5781, oct. 2009. 13

[Halton 64] J.H. Halton & G.B. Smith. *Algorithm 247 : Radical-inverse quasi-random point sequence*. Commun. ACM, vol. 7, no. 12, pages 701–702, 1964. 9

[Hammersley 56] J.M. Hammersley & K.W. Morton. *A new Monte Carlo technique : antithetic variates*. In Cambridge University Press, editeur, Mathematical Proceedings of the Cambridge Philosophical Society, volume 52, pages 449–475, July 1956. 9

[Han 01] L. Han & N.M. Amato. *A Kinematics-Based Probabilistic Roadmap Method for Closed Chain Systems*. In B. R. Donald, K. M. Lynch & D. Rus, editeurs, International Workshop on Algorithmic Foundations of Robotics, pages 233–246. A K Peters, Wellesley, MA, 2001. 15, 42, 43, 44

[Hirzinger 02] G. Hirzinger, N. Sporer, A. Albu-Schäffer, M. Hähnle, R. Krenn, A. Pascucci & M. Schedl. *DLR's Torque-Controlled Light Weight Robot III - Are We Reaching the Technological Limits Now ?* In IEEE International Conference on Robotics and Automation, pages 1710–1716. IEEE, 2002. 35, 36, 53, 86

[Hsu 97] D. Hsu, J.-C. Latombe & R. Motwani. *Path planning in expansive configuration spaces.* In IEEE International Conference on Robotics and Automation, volume 3, pages 2719–2726, april 1997. 12

[Hsu 98] D. Hsu, L.E. Kavraki, J.C. Latombe, R. Motwani & S. Sorkin. *On finding narrow passages with probabilistic roadmap planners.* In P.K. Agarwal *et al*, editeurs, wafr, pages 141–154. A. K. Peters, Wellesley, MA, 1998. 9

[Hsu 99] D. Hsu, J. C. Latombe & R. Motwani. *Path Planning in expansive configuration spaces.* International Journal Computational Geometry and Applications, vol. 9, no. 4-5, pages 495–512, 1999. 8

[Hsu 00] D. Hsu. *Randomized Single-query Motion Planning in Expansive Spaces.* PhD thesis, Deptartement of Computer Science, Stanford University, 2000. 11

[Hsu 03] D. Hsu, T. Jiang, J. Reif & Z. Sun. *The bridge test for sampling narrow passages with probabilistic roadmap planners.* In IEEE International Conference on Robotics and Automation, volume 3, pages 4420–4426, sept. 2003. 9, 45

[Hsu 06] D. Hsu, G. Sanchez-Ante, H-L. Cheng & J.-C. Latombe. *Multi-level free-space dilation for sampling narrow passages in PRM planning.* In IEEE International Conference on Robotics and Automation, pages 1255–1260, may 2006. 9

[Jaillet 04] L. Jaillet & T. Siméon. *A PRM-based motion planner for dynamically changing environments.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 2, pages 1606–1611, Sept. 2004. 71

[Jaillet 05a] L. Jaillet. *Méthodes Probabiliste pour la Planification Réactive de Mouvements.* PhD thesis, Universite Paul Sabatier, 2005. 10

[Jaillet 05b] L. Jaillet, A. Yershova, S.M. LaValle & T. Siméon. *Adaptive tuning of the sampling domain for dynamic-domain RRTs.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2851–2856, August 2005. 12

[Jaillet 08a] L. Jaillet, J. Cortés & T. Siméon. *Transition-based RRT for path planning in continuous cost spaces.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2145–2150, sept. 2008. 13, 24, 25, 45, 88, 95, 97

[Jaillet 08b] L. Jaillet & T. Siméon. *Path Deformation Roadmaps : Compact Graphs with Useful Cycles for Motion Planning.* International Journal of Robotics Research, vol. 27, no. 11–12, pages 1175–1188, 2008. 10, 11

[Jiménez 98] P. Jiménez, F. Thomas & C. Torras. *Collision Detection Algorithms for Motion Planning. Robot Motion Planning and Control.* In J.-P. Laumond, editeur, Lecture Notes in Control and Information Sciences, 229, pages 305–343. Springer, 1998. 11

[Kant 86] K. Kant & S.W. Zucker. *Toward efficient trajectory planning : The path-velocity decomposition.* International Journal of Robotics Research, vol. 5, no. 3, pages 72–89, 1986. 13

[Kavraki 95] L.E. Kavraki. *Random Networks in Configuration Space for Fast Path Planning.* PhD thesis, Stanford University, Stanford, CA, USA, 1995. 9

[Kavraki 96] L.E. Kavraki, P. Svestka, J.-C. Latombe & M.H. Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces.* IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pages 566 –580, aug 1996. 2, 7, 8, 9, 10, 45, 78, 88

[Khanmohammadi 08] S. Khanmohammadi & A. Mahdizadeh. *Density Avoided Sampling : An Intelligent Sampling Technique for Rapidly-Exploring Random Trees.* In International Conference on Hybrid Intelligent Systems, pages 672–677, sept. 2008. 12

[Khatib 86] O Khatib. *Real-time obstacle avoidance for manipulators and mobile robots.* International Journal of Robotics Research, vol. 5, no. 1, pages 90–98, 1986. 7

[Koga 94] Y. Koga & J.-C. Latombe. *On multi-arm manipulation planning.* In IEEE International Conference on Robotics and Automation, volume 2, pages 945–952, may 1994. 14, 18

[Konietschke 09] R. Konietschke & G. Hirzinger. *Inverse kinematics with closed form solutions for highly redundant robotic systems.* In IEEE International Conference on Robotics and Automation, pages 2945–2950, December 2009. 53

[Kuffner 00] Jr. Kuffner J.J. & S.M. LaValle. *RRT-connect : An efficient approach to single-query path planning.* In IEEE International Conference on Robotics and Automation, volume 2, pages 995–1001, 2000. 12, 51, 56, 78, 82

[Larsen 00] E. Larsen, S. Gottschalk, M.C. Lin & D. Manocha. *Fast distance queries with rectangular swept sphere volumes.* In IEEE International Conference on Robotics and Automation, volume 4, pages 3719–3726, 2000. 11, 78

[Latombe 91] J.-C. Latombe. Robot motion planning. Kluwer Academic Publishers, Boston, MA, 1991. 1, 5, 7, 17

[Laumond 98] J. P. Laumond, S. Sekhavat & F. Lamiraux. *Guidelines in nonholonomic motion planning for mobile robots.* In J.-P. Laumond, editeur, Robot

Motion Plannning and Control, volume 229/1998, pages 1–53. Springer-Verlag, 1998. 10, 85

[LaValle 96] S.M. LaValle & S.A. Hutchinson. *Optimal motion planning for multiple robots having independent goals*. In IEEE International Conference on Robotics and Automation, volume 3, pages 2847–2852, april 1996. 14

[LaValle 98] S.M. LaValle. *Rapidly-exploring random trees : A new tool for path planning*. Rapport technique 98–11, Computer Science Departement, Iowa State University, October 1998. 7, 8, 12, 78

[LaValle 99] S.M. LaValle, J.H. Yakey & L.E. Kavraki. *A probabilistic roadmap approach for systems with closed kinematic chains*. In IEEE International Conference on Robotics and Automation, volume 3, pages 1671–1676, 1999. 15

[LaValle 01] Steven M. LaValle & J. Kuffner. *Rapidly-Exploring Random Trees : Progress and Prospects*. In Algorithmic and Computational Robotics : New Directions, pages 293–308, Wellesley, MA, 2001. 12, 72, 75

[LaValle 02] S.M. LaValle. *From Dynamic Programming to RRTs : Algorithmic Design of Feasible Trajectories*. In Springer Berlin /Heidelberg, editeur, Control Problems in Robotics, volume 4/2003, pages 19–37. Springer-Verlag, 2002. 10

[LaValle 04] S.M. LaValle, M.S. Branicky & S.R. Lindemann. *On the Relationship between Classical Grid Search and Probabilistic Roadmaps*. International Journal of Robotics Research, vol. 23, pages 673–692, 2004. 9

[LaValle 06] S.M. LaValle. Planning algorithms. [Online], 2006. 1, 5, 7, 62

[Le 09] D.T. Le, J. Cortés & T. Siméon. *A path planning approach to (dis)assembly sequencing*. In IEEE International Conference on Automation Science and Engineering, pages 286–291, aug. 2009. 13

[Lien 03] J.-M. Lien, S.L. Thomas & N.M. Amato. *A general framework for sampling on the medial axis of the free space*. In IEEE International Conference on Robotics and Automation, volume 3, pages 4439–4444, sept. 2003. 9

[Lin 03] M.C. Lin & D. Manocha. *Collision and Proximity Queries*, 2003. 11

[Lozano-Pérez 83] T. Lozano-Pérez. *Spatial Planning : A Configuration Space Approach*. IEEE Transactions on Computers, vol. C–32, no. 2, pages 108–120, Feb. 1983. 1, 5, 6, 7

[Lozano-Pérez 87] T. Lozano-Pérez. *A simple motion-planning algorithm for general robot manipulators*. IEEE Journal of Robotics and Automation, vol. 3, no. 3, pages 224–238, june 1987. 7

[Mazer 98] E. Mazer, J.-M. Ahuactzin & P. Bessiere. *The Ariadne's Clew Algorithm.* JAIR, vol. 9, pages 295–316, 1998. 12

[Nielsen 00] C.L. Nielsen & L.E. Kavraki. *A two level fuzzy PRM for manipulation planning.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 3, pages 1716–1721, 2000. 11, 17

[Nieuwenhuisen 04] D. Nieuwenhuisen & M.H. Overmars. *Useful cycles in probabilistic roadmap graphs.* In IEEE International Conference on Robotics and Automation, volume 1, pages 446–452, april 2004. 10

[Nilsson 69] N.J. Nilsson. *A mobius automation : an application of artificial intelligence techniques.* In International Joint Conference on Artificial Intelligence, volume 1, pages 509–520, San Francisco, CA, USA, 1969. Morgan Kaufmann Publishers Inc. 7

[Nissoux 99] C. Nissoux. *Visibilité et méthodes probabilistes pour la planification de mouvements en robotique.* PhD thesis, Université de Toulouse 3, Toulouse, FRANCE, 1999. 10, 11

[O'Donnell 89] P.A. O'Donnell & T. Lozano-Pérez. *Deadlock-free and collision-free coordination of two robot manipulators.* In IEEE International Conference on Robotics and Automation, volume 1, pages 484–489, may 1989. 14

[Okada 04] K. Okada, A. Haneda, H. Nakai, M. Inaba & H. Inoue. *Environment manipulation planner for humanoid robots using task graph that generates action sequence.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 2, pages 1174–1179, sept. 2004. 17

[Ott 06] C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schaffer, B. Brunner, H. Hirschmuller, S. Kielhofer, R. Konietschke, M. Suppa, T. Wimbock, F. Zacharias & G. Hirzinger. *A Humanoid Two-Arm System for Dexterous Manipulation.* In IEEE/RAS International Conference on Humanoid Robots, pages 276–283, December 2006. 21, 36, 53, 72

[Park 98] F.C. Park & Jin Wook Kim. *Manipulability and singularity analysis of multiple robot systems : a geometric approach.* In Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on, volume 2, pages 1032–1037, 1998. 42

[Porta 07] J.M. Porta, J. Cortés, L. Ros & F. Thomas. *A Space Decomposition Method for Path Planning of Loop Linkages.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007. 14

[Ranganathan 04] A. Ranganathan & S. Koenig. *PDRRTs : integrating graph-based and cell-based planning.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 3, pages 2799–2806, September 2004. 13

[Reeds 90] J.A. Reeds & L.A. Shepp. *Optimal paths for a car that goes both forwards and backwards.* Pacific J. Math., vol. 145, no. 2, pages 367–393, 1990. 78, 82, 83

[Reif 79] J.H. Reif. *Complexity of the mover's problem and generalizations.* In Symposium on Foundations of Computer Science, pages 421–427, Washington, DC, USA, 1979. IEEE Computer Society. 7

[Rodriguez 06] S. Rodriguez, X. Tang, J.-M. Lien & N.M. Amato. *An obstacle-based rapidly-exploring random tree.* In IEEE International Conference on Robotics and Automation, pages 895–900, may 2006. 12

[Saha 05] M. Saha & J.-C. Latombe. *Finding narrow passages with probabilistic roadmaps : the small step retraction method.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 622–627, aug. 2005. 9

[Saha 06] M. Saha & P. Isto. *Multi-Robot Motion Planning by Incremental Coordination.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5960–5963, oct. 2006. 14

[Sánchez 02] G. Sánchez & J.-C. Latombe. *On Delaying Collision Checking in PRM Planning : Application to Multi-Robot Coordination.* International Journal of Robotics Research, vol. 21, no. 1, pages 5–26, 2002. 71

[Sánchez 03] G. Sánchez & J.C. Latombe. *A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking.* In International Symposium on Robotics Research, pages 403–417, Lorne, Victoria, Australia, November 2003. R.A. Jarvis and A. Zelinsky (eds.), Springer Tracts in Advanced Robotics. 12

[Saut 07a] J.-P. Saut. *Planification de Mouvement pour la Manipulation Dextre d'Objets Rigides.* PhD thesis, Université Pierre et Marie Curie, November 2007. 60, 62

[Saut 07b] J.-P. Saut, A. Sahbani, S. El-Khoury & V. Perdereau. *Dexterous manipulation planning using probabilistic roadmaps in continuous grasp subspaces.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2907–2912, oct. 2007. 19

[Saut 10] J.-P. Saut & D. Sidobre. *Efficient Models for Grasp Planning With A Multi-fingered Hand.* In IEEE/RSJ International Conference on Intelligent Robots and Systems. Workshop on Grasp Planning and Task Learning by Imitation, October 2010. 62

[Schwartz 83] J. T. Schwartz & M. Sharir. *On the Piano Movers' Problem : III. Coordinating the Motion of Several Independent Bodies.* International Journal of Robotics Research, vol. 2, pages 46–75, September 1983. 5, 7, 13, 14

[Shibata 93] T. Shibata & T. Fukuda. *Coordinative behavior in evolutionary multi-agent system by genetic algorithm.* In IEEE International Conference on Neural Networks, volume 1, pages 209–214, 1993. 14

[Shimoga 96] K.B. Shimoga. *Robot Grasp Synthesis Algorithms : A Survey.* International Journal of Robotics Research, vol. 15, no. 3, pages 230–266, 1996. 62

[Siméon 00] T. Siméon, J.-P. Laumond & C. Nissoux. *Visibility-Based Probabilistic Roadmaps for Motion Planning.* Advanced Robotics Journal, vol. 14(6), pages 477–494, 2000. 11, 24, 25, 28, 45, 51, 74, 78, 82, 88

[Siméon 01] T. Siméon, J.-P. Laumond & F. Lamiraux. *Move3D : a generic platform for path planning.* In International Symposium on Assembly and Task Planning, volume 4, pages 25–30, 2001. 35, 77, 91, 95

[Siméon 02] T Siméon, S. Leroy & J.-P. Laumond. *Path coordination for multiple mobile robots : a resolution-complete algorithm.* IEEE Transactions on Robotics and Automation, vol. 18, no. 1, pages 42–49, February 2002. 14

[Siméon 04] T. Siméon, J.P. Laumond, J. Cortés & A. Sahbani. *Manipulation planning with probabilistic roadmaps.* International Journal of Robotics Research, vol. 23, no. 7-8, pages 729–746, August 2004. 17, 59, 76, 97

[Sisbot 08] E.A. Sisbot. *Towards Human-Aware robot Motions.* PhD thesis, LAAS/CNRS, Universite Paul Sabatier, October 2008. 98

[Stilman 04] M. Stilman & J. Kuffner. *Navigation among movable obstacles : real-time reasoning in complex environments.* In IEEE/RAS International Conference on Humanoid Robots, volume 1, pages 322–341, nov. 2004. 17, 18

[Stilman 07] M. Stilman, J. Schmburek, J. Kuffner & T. Asfour. *Manipulation Planning Among Movable Obstacles.* In IEEE International Conference on Robotics and Automation, 2007. 17

[Suarez 06] R. Suarez, M. Roa & J. Cornella. *Grasp quality measures.* In Universitat Politecnica de Catalunya (UPC), Technical Report, 2006. 65

[Tournassoud 86] P. Tournassoud. *A strategy for obstacle avoidance and its application to mullti-robot systems.* In IEEE International Conference on Robotics and Automation, volume 3, pages 1224–1229, april 1986. 13

[Trinkle 02] J.C. Trinkle & R.J. Milgram. *Complete Path Planning for Closed Kinematic Chains with Spherical Joints.* International Journal of Robotics Research, vol. 21, no. 9, pages 773–789, 2002. 14

[Urmson 03] C. Urmson & R. Simmons. *Approaches for heuristically biasing RRT growth.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 2, pages 1178–1183, oct. 2003. 13

[Vahrenkamp 09] N. Vahrenkamp, D. Berenson, T. Asfour, J. Kuffner & R. Dillmann. *Humanoid motion planning for dual-arm manipulation and re-grasping tasks.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2464–2470, oct. 2009. 18

[Švestka 95] P. Švestka & M.H. Overmars. *Coordinated motion planning for multiple car-like robots using probabilistic roadmaps.* In IEEE International Conference on Robotics and Automation, volume 2, pages 1631–1636, may 1995. 13, 21

[Švestka 97] P. Švestka. *Robot Motion Planning using Probabilistic Roadmaps.* PhD thesis, Universiteit Utrecht, 1997. 2, 7, 8, 9, 14, 24

[Wilfong 88] G. Wilfong. *Motion planning in the presence of movable obstacles.* In Symposium on Computational Geometry, pages 279–288, New York, NY, USA, 1988. ACM. 17

[Wilmarth 98] S.A. Wilmarth, N.M. Amato & P.F. Stiller. *MAPRM : A Probabilistic Roadmap Planner with Sampling on the Medial Axis of the Free Space.* In IEEE International Conference on Robotics and Automation, pages 1024–1031, 1998. 9

[Yang 04] Y. Yang & O. Brock. *Adapting the sampling distribution in PRM planners based on an approximated medial axis.* In IEEE International Conference on Robotics and Automation, volume 5, pages 4405–4410, april 2004. 9

[Yashima 02] M. Yashima & H. Yamaguchi. *Dynamic Motion Planning Whole Arm Grasp Systems Based on Switching Contact Modes.* In IEEE International Conference on Robotics and Automation, Washington D.C., États-Unis, mai 2002. 19

[Yashima 03] M. Yashima, Y. Shiina & H. Yamaguchi. *Randomized Manipulation Planning for A Multi-Fingered Hand by Switching Contact Modes.* In IEEE International Conference on Robotics and Automation, Taiwan, septembre 2003. 19

[Yershova 04] A. Yershova & S. LaValle. *Deterministic Sampling Methods for Spheres and SO(3).* In IEEE International Conference on Robotics and Automation, 2004. 63

Résumé

L'apparition de robots de service de plus en plus complexes ouvre de nouvelles perspectives quant aux tâches de manipulation d'objets. Malgré les progrès récents des techniques de planification de mouvement, peu d'entre elles s'intéressent directement à des systèmes multi-bras comme les torses humanoïdes. Notre contribution à travers cette thèse porte sur trois aspects.

Nous proposons une technique de planification de mouvement performante basée sur la coordination des mouvements du système multi-bras. Elle exploite au mieux la structure du système en la divisant en parties élémentaires dont les mouvements sont planifiés indépendamment du reste du système. La fusion des différents réseaux élémentaires générés est ensuite réalisée dans le but d'obtenir un graphe prenant en compte le robot tout entier.

Une seconde contribution porte sur l'extension des méthodes de planification pour des robots présentant des chaînes cinématiques fermées. Ces boucles cinématiques apparaissent dans le système lorsque, par exemple, le torse humanoïde saisit un objet avec plusieurs bras. Cette méthode traite explicitement les configurations singulières des manipulateurs, offrant ainsi une meilleure maniabilité de l'objet.

Finalement, nous proposons deux approches pour la planification de tâches de manipulation d'objets par un torse humanoïde. La première concerne la résolution d'une tâche de prise et pose d'objets par un torse humanoïde à deux bras dans le cas où les contraintes imposées par la tâche nécessitent le passage par une double prise afin de transférer l'objet d'une main à l'autre. La seconde porte sur la résolution du même type de tâche par un manipulateur mobile.

La thèse, effectuée dans le cadre du projet européen Phriends, présente les résultats d'expérimentations réalisées sur le robot Justin, démonstrateur du projet.

Abstract

The emergence of new more and more complex service robots opens new research fields on objet manipulation. Despite the recent progresses in motion planning techniques, few of them deal directly with multi-arm systems like humanoid torsos. Our contribution through this thesis focuses on three aspects.

We present an efficient motion planning technique based on the multi-arm system motion coordination. It takes advantage of the system's structure by dividing it into elementary parts of which movements are planned independently of the rest of the system. Generated elementary networks are then fused to obtain a roadmap that takes into account the whole robot.

The second contribution consists of the extension of motion planning methods for a robot under loop closure constraints. These kinematic loops appear in the system when, for example, the humanoid torso grasps an objet with two arms. This method treats explicitly the singular configurations of the manipulators, providing better handling of the object.

Finally, we present two approaches for planning object manipulation tasks by humanoid torsos. The first concerns solving pick and place task by humanoid torso where the imposed task constraints require a passage through a double grasp to transfer the object from one hand to the other. The second approach concerns the resolution of the same type of task by a mobile manipulator.

The presented methods have been integrated on a real platform, Justin, and validated with experiments in the frame of E.U. FP-6 PHRIENDS project.