Planification de tâche de manipulation par pivotement pour un robot humain

Mathieu Poirier

▶ **To cite this version:**

Mathieu Poirier. Planification de tâche de manipulation par pivotement pour un robot humain. Automatique / Robotique. INSA de Toulouse, 2009. Français. NNT : . tel-00450869

HAL Id: tel-00450869

https://theses.hal.science/tel-00450869v1

Submitted on 27 Jan 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par *INSA Toulouse*
Discipline ou spécialité : *Systèmes Informatiques*

Présentée et soutenue par *Mathieu Poirier*
Le *21 Septembre 2009*

Titre : *Planification de tâches de manipulation par pivotement pour un robot humanoïde*

JURY
Rachid Alami - Directeur de Recherche CNRS
Philippe Fraisse - Professeur des Universités
Christian Laugier - Directeur de Recherche INRIA
Jean-Paul Laumond - Directeur de Recherche CNRS
Eiichi Yoshida - Chercheur AIST

Ecole doctorale : *Ecole Doctorale Systèmes*
Unité de recherche : *Gepetto*
Directeur(s) de Thèse : *Jean-Paul Laumond / Eiichi Yoshida*
Rapporteurs : *Philippe Fraisse / Christian Laugier*

Abstract

This manuscript highlights the ability of a humanoid robot to perform a task hard to achieve by other types of robots. Here we are interested in manipulation of bulky objects. Such manipulation tasks are performed with great difficulty and involve several skills, such as taking into account motion of the body as one whole system (or whole-body motion) and perfect synchronization between the different limbs, arms and legs. We introduce here a whole-body motion planner that gives a humanoid robot the ability to automatically find a strategy for moving bulky objects by pivoting, taking into account a number of constraints : collision avoidance, limbs coordination, arms and legs, and stability control throughout the motion. Based on formal controllability results, defined upstream, the planner also inherits completeness of the probabilistic random sampling methods on which it is built. The geometric and kinematic capabilities of the planner proposed are also demonstrated through simulations and real experiments. Then we focus on solving more complex problems, giving the robot the ability to release and regrasp the object to manipulate if it is stuck in a narrow passage.

Résumé

Ce manuscrit met en avant la capacité d'un robot humanoïde à effectuer une tâche difficilement réalisable par d'autres types de robots. On s'intéresse ici à la manipulation d'objets dits encombrants. De telles tâches de manipulation s'effectuent avec beaucoup de difficultés et font appel à plusieurs aptitudes, telles la prise en compte du mouvement du corps dans son ensemble (ou mouvement corps complet) et une parfaite synchronisation entre les différents membres, bras et jambes. Nous introduisons ici un planificateur de mouvements corps complet qui donne à un robot humanoïde la capacité de mettre en place, automatiquement, une stratégie de déplacement d'objets encombrants par pivotement, tout en prenant en compte un certain nombre de contraintes : évitement de collisions, coordination des membres, bras et jambes, et contrôle de la stabilité pendant tout le déplacement. Basé sur des résultats formels, définis en amont, prouvant la contrôlabilité en temps petit d'un système se déplaçant par pivotement. Le planificateur hérite aussi de la complétude probabiliste des méthodes d'échantillonnage aléatoire de planification sur lesquelles il est construit. Les capacités géométriques et cinématiques du planificateur proposé sont aussi démontrées à travers des simulations et des expérimentations réelles. Nous nous intéressons ensuite à résoudre des problèmes plus complexes, en offrant au robot la possibilité de lâcher et de reprendre l'objet à manipuler si celui-ci est bloqué dans un passage étroit.

Remerciements

Je tiens à remercier mes directeurs de thèse Jean-Paul Laumond et Eiichi Yoshida pour leur soutien, leur compréhension, leur rigueur scientifique, leur temps, leurs précieux conseils et pour leurs qualités humaines qui apportent cette formidable cohésion au sein du groupe Gepetto.

Merci également aux rapporteurs Philippe Fraisse (LIRMM) et Christian Laugier (INRIA) et à l'examinateur Rachid Alami (LAAS-RIA) d'avoir accepté de rapporter et de participer au jury de ma thèse.

Je remercie aussi tous les responsables et permanents du pôle RIA, ainsi que ceux du JRL-France et JRL-Japan qui m'ont apporté leur soutien sur certains points techniques et qui m'ont permis de partir en mission au Japon.

Je remercie Anthony M. pour son support technique formidable, sa bonne humeur et ses réponses à nos questions les plus farfelues. Mais aussi pour ces très bons moments partagés au Japon autour d'un bon repas et d'un bon scotch.

Ensuite je tiens à remercier tous mes meilleurs amis du monde entier, sans qui ces trois années n'auraient pu se réaliser. Merci pour ces bonnes rigolades, ces bons moments d'échanges internationaux, ces solutions trouvées autour d'un café ... ces solutions avortées et restées au café. Merci à Alireza N.(meilleur Iranien), Sébastien D.(meilleur Parisien), Minh T.(meilleur Vietnamien du nord), Brice B.(meilleur Alsacien), Anh T.(meilleur Vietnamien du sud), Oussama K.(meilleur Tunisien), Francesco M.(meilleur Mexicain), Manish S.(meilleur Indien), David F.(meilleur Réunionnais), Panos T.(meilleur Grec), Mokhtar G.(meilleur Marocain). Thank you all bros !

Merci aussi à tous les doctorants RIA pour cette ambiance inoubliable.

J e remercie aussi Sébastien Jeanne-Brou, Adèle Panouillot, Céline Ouali et Sébastien Dalibar pour avoir relu et corrigé ce manuscrit.

Merci à mes parents, frère et soeur, ainsi que toute ma Famille et Amis pour leur soutien moral et leur capacité à me donner l'envie de toujours aller plus loin dans la vie. Vous êtes la meilleur énergie qui soit.

Et finalement je remercie HRP-2 n14 mon premier meilleur ami entièrement mécanique sans qui rien n'aurait été possible.

A mes parents et mes amis,
qui ont su me soutenir pendant ces 3 années.

(>’-’)> <(’-’)> <(’-’<)

”The World has to keep in touch”

Table des matières

1 Introduction

1.1 Le contexte général de la robotique

La robotique

Jusque dans les années 60, la robotique était considérée plus comme un thème de science-fiction que comme un fait de la réalité. D'ailleurs le terme robotique est introduit dans la littérature en 1942 par *Isaac Asimov* dans son livre Runaround. Celui-ci y énonce ce que l'on appelera par la suite les *trois lois de la robotique*.

De cette science-fiction à la réalité, la robotique a bien évolué. Après avoir été essentiellement un domaine de recherche scientifique, la robotique a fait son apparition dans l'industrie.

On peut définir la robotique comme la science qui étudie l'automatisation d'un système mécanique. En dotant ce système de capacités de perception, de décision et d'action, nous lui permettons d'interagir avec son environnement de façon autonome. La robotique est un domaine pluridisciplinaire au carrefour de la mécanique, de l'automatique, de l'intelligence artificielle, de l'informatique ou encore du traitement de l'information (capteurs, moteurs ...).

Nous définissons maintenant le terme *robot*, inventé en 1920 par *Karel Capek* en faisant référence, à cette époque, à des humains organiques artificiels, comme : un système automatique mécanisé capable d'effectuer une ou plusieurs tâches, dans un environnement donné, de manière autonome, par l'exécution d'un programme ou d'une séquence d'instructions.

L'autonomie suppose que le programme d'instruction prévoit la venue de certains événements, puis la ou les réactions appropriées à ceux-ci. On associe donc toujours au robot une boucle Perception → Décision → Action :

- **La perception** représente le flot d'informations en entrée du système. La plupart du temps, ces informations sont fournies par des capteurs. Suivant la nature du capteur (caméras, lasers, sons, ultra-sons, etc.), le système peut obtenir et traiter différentes informations sur son environnement.
- **La décision** représente le programme (informatique ou non) qui en fonction des entrées reçues, va pouvoir analyser la situation et prendre ou construire la solution qui correspondra le mieux à ce que le robot souhaite réaliser.
- **L'action** représente le flot d'informations en sortie du système. Dans la majorité des cas, ces informations sont souvent des consignes envoyées aux actionneurs ou aux moteurs du robot. C'est le résultat de la décision.

La robotique dans la société actuelle

La société japonaise en tête, en raison de sa démographie en forme de pyramide inversée, tente de développer très rapidement des solutions robotisées pour venir en aide à la population vieillissante du pays. On y compte près de 300 robots pour 10000 ouvriers. Une densité qui est 10 fois supérieure à la moyenne mondiale.

De nos jours, les robots sont intensivement et principalement utilisés dans l'industrie. Ils y effectuent des tâches répétitives souvent fastidieuses et/ou dangereuses pour un être humain et le tout avec plus de rigueur et de précision. Ils remplacent les ouvriers sur les chaînes de montage de l'industrie automobile pour les tâches de peinture, soudage, etc. Mais ils ont des applications dans des domaines aussi variés que l'industrie manufacturière, le spatial, l'automobile, le médical ou plus récemment les loisirs. Ceci démontre leur intérêt économique et social, notamment pour la recherche.

La robotique mobile autonome vise plus spécifiquement à concevoir des systèmes capables de se déplacer de façon autonome. Les applications directes se situent dans le domaine de l'automobile (avec par exemple les nouvelles options proposées comme le parking automatique), de l'exploration planétaire ou de la robotique de service.

La robotique médicale est également très active. De nouveaux robots sont développés pour la chirurgie mini-invasive et la téléchirurgie. De nouvelles techniques sont exploitées, comme les actionneurs AMF (alliages à mémoire de forme), la microrobotique et les interfaces haptiques. Des algorithmes d'analyse d'images sont développés dans la même voie.

La robotique humanoïde et le projet *Humanoïd Robot Project - HRP*

Nous avons assisté, il y a quelques années à une intensification de la recherche sur la robotique à locomotion bipède dont font partie les robots humanoïdes.

Le projet HRP est un projet mené en collaboration entre la société Kawada (pour la fabrication) et Humanoid Research Group (*HRG*) de l'institut de recherche japonais *Advanced Industrial Science and Technology - AIST* pour la conception et la recherche. Le projet s'intéresse à l'étude des robots humanoïdes incluant le design, la simulation dynamique, le contrôle ou encore la planification de mouvement.

Malheureusement, malgré un premier âge d'or avec des robots comme HRP-2 (voir Fig. 1.1), la recherche humanoïde coûte cher et les retombées économiques ne sont pas forcement encore perceptibles. Son successeur HPR-3 n'ayant pas eu la reconnaissance de son aîné, les instituts de recherches japonais se sont lancés alors dans la robotique de « loisirs »pour toucher un plus large public. Avec l'androïde HRP-4 (Fig. 1.1), en plus des programmes de recherches centrés sur la locomotion humaine, les expressions faciales, etc. Les instituts japonais tentent de poursuivre dans la brèche entamée par la société Sony et son chien robot Aïbo ou le petit robot Nao de la société française Aldebaran Robotics.

FIGURE 1.1 – De gauche à droite HRP-2, HRP-3 et le nouvel androïde HRP-4

HRP-4 reste pourtant un acteur majeur pour la suite de la recherche humanoïde. Nous sommes en droit de nous demander si nous sommes au commencement d'une nouvelle ère ou en train de refermer provisoirement une page de la robotique. En tout cas, la recherche française y croit encore, en lançant en janvier 2009 le projet Romeo (*www.projetromeo.com*) visant à construire pour 2011 le premier robot humanoïde français.

De ce fait de nombreuses personnes nous posent la question : pourquoi étudier des machines bipèdes qui sont, par nature, plus instables que des robots hexapodes, quadrupèdes, etc. ? Ces derniers étant statiquement stable. Et pourquoi ne pas ajouter plusieurs bras ou outils ?

D'un point de vue pragmatique, tout ceci pourrait facilement se justifier. Mais d'un point de vue scientifique, il ne faut pas oublier que la recherche en robotique humanoïde est aussi un très beau moyen d'étudier l'homme et le comportement humain : comment la locomotion humaine est produite, quels repères utilisons nous, nous servons-nous de tous nos sens pendant le déplacement, etc. ? Beaucoup de

ces questions sont encore discutées de nos jours. C'est pourquoi notre groupe travaille en collaboration avec des laboratoires de neurosciences tels que le *CERCO* de Toulouse ou le *Collège de France* avec le Pr. Berthoz.

Grâce aux progrès, les robots humanoïdes peuvent maintenant exécuter des tâches de plus en plus complexes pour remplacer l'homme dans des environnements dangereux ou l'aider à accomplir des tâches fastidieuses. L'une de ces applications est la manipulation d'objets divers, et tout particulièrement, celle des objets lourds et/ou volumineux sollicitant les propriétés d'un corps anthropomorphe. Pour cela, nous devons utiliser toutes les possibilités mécaniques offertes par le robot humanoïde. Tous les degrés de liberté du robot sont sollicités lors de la manipulation. Dans nos travaux, nous nous intéressons à la question suivante : pouvons-nous déplacer des objets encombrants par une méthode de pivotement avec un robot humanoïde, comme le ferait un humain pour déménager une commode, un réfrigérateur etc. ?

1.2 Approche générale du problème

1.2.1 Définition du problème

Dans cette thèse nous cherchons à nous démarquer des méthodes de décomposition fonctionnelle (chapitre 2.2.5) de [Yoshida et al. 2008] en proposant une manipulation d'objets encombrants nécessitant obligatoirement un mouvement synchronisé des bras et des jambes. Nous voulons ici, utiliser au mieux les propriétés anthropomorphes que nous offre le type de robot humanoïde.

On appellera, un *mouvement corps complet* (en l'anglais *whole-body motion*) un mouvement qui implique l'activation de l'ensemble des degrés de liberté d'un système anthropomorphe tel que le robot humanoïde HRP-2.

Il s'agira donc de planifier des mouvements corps complet pour un système anthropomorphe de type robot humanoïde afin que celui-ci puisse déplacer un objet encombrant dans un environnement complexe à l'aide d'une méthode de pivotement. La figure 1.2 nous montre un exemple de problèmes à résoudre à l'aide de notre méthode. Le robot doit déplacer l'objet de sa position initiale à sa position finale.

(a) Le robot et l'objet à leurs emplacements initiaux (b) Le robot et l'objet à leurs emplacements finaux

FIGURE 1.2 – Exemple de problème à résoudre à l'aide de notre méthode

1.2.2 Notre approche

L'approche que nous avons définie suit les principes suivants :

1.2.2.1 La Planification de mouvement : calcul de l'itinéraire à suivre

Il s'agit dans un premier temps de planifier le déplacement de l'objet à manipuler ainsi que celui du robot. Nous utilisons des techniques robustes et reconnues qui sont issues de la planification de mouvement pour robot mobile à roues. A cette étape, les propriétés anthropomorphes du système et ses 40 degrés de liberté ne seront pas prises en compte. Nous calculons un chemin en trois dimensions correspondant à un volume de l'espace de travail permettant au robot de déplacer l'objet de sa configuration initiale à sa configuration finale par une action de pivotement et ce, sans entrer en collision avec son environnement. Pour cela, nous réduisons l'ensemble du système robot-objet à un simple objet parallélépipède rigide. C'est à ce stade que nous introduisons une première contrainte issue de la locomotion humaine. Nous souhaitons privilégier des mouvements de locomotion en marche avant (ou arrière) et éviter des déplacements "en crabe". Ce type de mouvement induit une contrainte non holonome sur le chemin global suivi par l'objet.

Nous donnons à cet objet rigide un comportement de déplacement par glissement soumis à des contraintes définies chap. 4.1.1 et nous planifions le déplacement de celui-ci dans son environnement afin de trouver une trajectoire optimisée libre de collision.

1.2.2.2 Calcul de la séquence de pivotement, du mouvement des mains et de la trajectoire des pieds

Une fois que nous disposons d'un premier chemin, nous obtenons un volume de l'espace de travail dans lequel le robot peut manipuler l'objet sans entrer en collision avec son environnement. Nous animons l'objet à manipuler le long du chemin. Cette animation est créée sur la base du déplacement par pivotement comme définie chap. 3. Nous obtenons ainsi une séquence de pivotement de la configuration initiale à la configuration finale. Les points de contact entre l'objet à manipuler et les mains du robot sont

alors extraits. La trajectoire que chacune des deux mains du robot devra suivre afin de déplacer l'objet est ainsi calculée. De la même manière nous calculons la position des pieds du robot en fonction de la position de l'objet le long de la trajectoire initiale.

1.2.2.3 Génération de mouvements corps complet

Ces données sont ensuite transformées sous forme de contraintes, avec une notion de priorité entre elles et ainsi utilisées en entrée des algorithmes de cinématique inverse généralisée développés au sein de notre équipe de recherche par [Yoshida et al. 2006], sur les bases du générateur de marche prédictif de [Kajita et al. 2003]. La trajectoire qui en sort est une trajectoire de mouvement corps complet qui prend en compte la stabilité du robot, la trajectoire de son centre de masse, ainsi que la synchronisation des bras et des jambes permettant la manipulation par pivotement de l'objet encombrant le long du chemin initial.

1.2.2.4 Validation dynamique

A ce stade, nous n'avons pris en compte que la cinématique du système. La trajectoire calculée est donc entrée dans le simulateur dynamique OpenHRP3 (chap. 1.2.3) pour contrôler que le robot exécute des mouvements dynamiquement stables. En cas d'échec, la planification est relancée en modifiant les paramètres d'exécution (paramètres de locomotion, vitesse d'exécution des trajectoires des mains ...) de manière heuristique. Si la trajectoire est validée nous pouvons exécuter le mouvement directement en boucle ouverte sur la plate-forme réelle et observer les résultats.

1.2.2.5 Planification de reprise d'objet à manipuler

En suivant la même approche, nous nous intéressons ensuite à la planification de reprise d'objet. Effectivement, dans un environnement très complexe, notre approche reste assez limitée. Le robot pourrait très vite rester bloqué dans un passage trop étroit à la manipulation, par exemple le passage d'une porte. Cette nouvelle étape donne au robot la possibilité de lâcher l'objet, contourner l'obstacle en se déplaçant seul, récupérer celui-ci de l'autre coté et continuer à manipuler afin d'atteindre son but. Cette étape ne sera traitée qu'en simulation.

Après une revue de l'état de l'art, les chapitres suivant s'arrêteront plus précisément sur chacun des points cités ci-dessus.

1.2.3 HRP-2 et son architecture logicielle

HRP-2 est une plate-forme de recherche de type robot humanoïde fabriqué par la société Kawada, en collaboration avec le groupe HRG de l'AIST. Il existe une vingtaine de robots de ce type à travers le monde. Nos travaux ont été intégrés sur la plate-forme *HRP-2 n°14* et réalisés dans l'environnement montré sur la figure 1.4b.

Ce robot fait $1m54$, pour $58kg$ et possède 40 degrés de liberté dont 30 commandables. Il peut marcher jusqu'à $2km/h$ et porter jusqu'à $2kg$ dans chaque main. La figure 1.3 expose les caractéristiques du robot HRP-2.

Dimensions		Total Height : 1,540mm, Total Width : 620mm
Mass (including batteries)		58kg
Degrees of Freedom		30 axes
Walking Speed		0~2km/h
Hand Grip		2kgf (per hand)
Sensors	Torso	3-axes vibration gyro 3-axes velocity sensor
	Arms	6-axes force sensors
	Legs	6-axes force sensors
Motor Drivers		48V 20A (I max) 2 axes / driver x 16
Power System		NiMH Battery DC 48V 14.8Ah

FIGURE 1.3 – HRP-2 et ses caractéristiques

Pour travailler avec le robot nous respectons le protocole suivant :

- Avant toute expérimentation, le robot doit être calibré et branché sur l'alimentation ou avec un niveau de batterie suffisant.
- Toute expérimentation exécutée avec le robot doit être filmée, pour qu'en cas d'accident nous puissions repasser le film et l'analyser si besoin.
- Toute expérimentation exécutée avec le robot doit se faire en présence d'au moins deux personnes, dont une personne qualifiée, formée tout spécialement à l'entretien et à la manipulation du robot HRP-2. Cette personne s'occupe aussi du bouton d'arrêt d'urgence. L'autre personne devant s'occuper du contrôle du robot depuis le logiciel spécifié.
- Lors des expérimentations, tous les ordres doivent être confirmés par les deux personnes présentes, avant toute exécution de mouvement.
- Le robot doit toujours être manipulé avec un système de sécurité, ici un lève-personne comme utilisé dans les hôpitaux. Le robot est entièrement autonome, mais si l'arrêt d'urgence est enclenché, le robot tombe comme une pierre. Ce système permet, en cas de coupure, de le retenir à une distance raisonnable du sol Fig. 1.4a.

(a) (b)

FIGURE 1.4 – (a) HRP-2 n°14 moteurs coupés et son système de sécurité. (b) Photo de l'espace de manipulation du robot au sein du LAAS-CNRS

KineoWorks : Les algorithmes développés dans cette thèse reposent sur les opérateurs géométriques du logiciel KineoWorks. KineoWorks est un logiciel développé par KineoCam [Laumond 2006], dédié à la planification de mouvements robotiques et basé sur des technologies développées au LAAS. Il regroupe un ensemble de programmes et de modules permettant de ne pas réécrire les algorithmes de planification déjà développés et validés tels que le moteur de planification, les méthodes locales de planification, les différentes stratégies de planification, les optimisateurs de chemin, le détecteur de collisions...

Human Path Planner - *HPP* est une plate-forme de développement logicielle dédiée aux recherches menées autour du robot humanoïde et des avatars virtuels : cinématique inverse généralisée, manipulation d'objet par pivotement, générateur de locomotion, générateur de marche pour humanoïde... Mes travaux sont implantés directement dans cette plate-forme.

Open Architecture Humanoid Robotics Platform - *OpenHRP3* est un logiciel [Kanehiro et al. 2004] pour le contrôle et la simulation de robots humanoïdes. Il intègre un contrôleur dédié à HRP-2 et est commercialisé par la startup GeneralRobotix. Il contient plusieurs modules correspondant à des programmes ou processus qui tournent indépendamment les uns des autres mais qui doivent communiquer. La structure logicielle qui permet ce type de développement parallèle et indépendant est le Common Object Request Broker Architecture - *CORBA*. Les principaux modules du logiciel sont utilisés pour la simulation (simulateur physique et dynamique, détecteur de collision...) et pour le contrôle du robot. En effet, comme le simulateur est conçu de manière très réaliste, le contrôleur du robot ne fait pas la distinction entre le robot simulé et le robot réel. Nous n'avons donc pas besoin de modifier le code ou les programmes que nous voulons tester sur le robot réel après les avoir vérifiés par la simulation : il y a

une compatibilité binaire entre les contrôleurs simulés et réels.

La figure 1.5 montre l'architecture logicielle dans son ensemble et sa connexion avec le robot HRP-2.

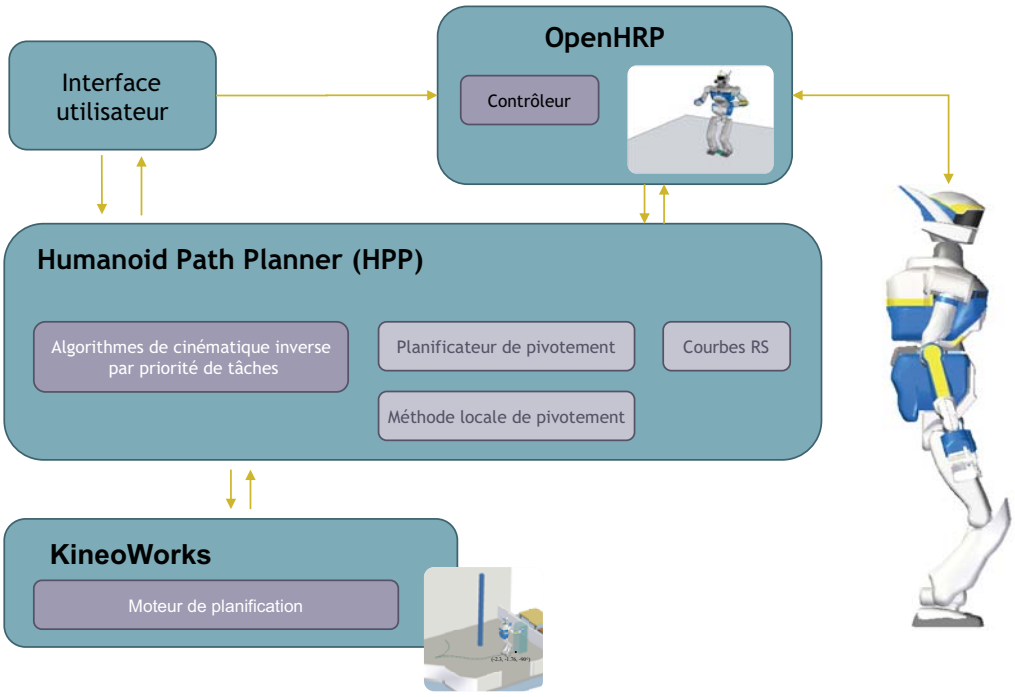


FIGURE 1.5 – L'architecture intégrant KineoWorks, le plate-forme HPP et OpenHRP

1.3 Organisation du Manuscrit

Après un état de l'art des grands principes et des bases sur lesquels nous allons nous appuyer pour réaliser ce travail (chap. 2), nous ferons une analyse de commandabilité pour un système se déplaçant par pivotement (chap. 3). Puis nous détaillerons l'élaboration de notre algorithme mis en place pour la manipulation d'objet encombrant par pivotement (chap. 4). Ensuite nous nous intéresserons à des problèmes de planification de manipulation par reprise (chap. 5) , avant de conclure sur quelques perspectives (chap. 6).

1.4 Publications associées à cette thèse

E.Yoshida and M. Poirier and J.-P. Laumond and R. Alami and K. Yokoi : "Pivoting Based Manipulation by Humanoids : a Controllability Analysis",' *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1130 - 1135, 2007.

E. Yoshida and M. Poirier and J.-P. Laumond and O. Kanoun and F. Lamiraux and R. Alami and K. Yokoi : "Whole-body motion planning for pivoting based manipulation by humanoids", *Proc. 2008 IEEE Int. Conf. on Robotics and Automation*, 1712 - 1717, 2008.

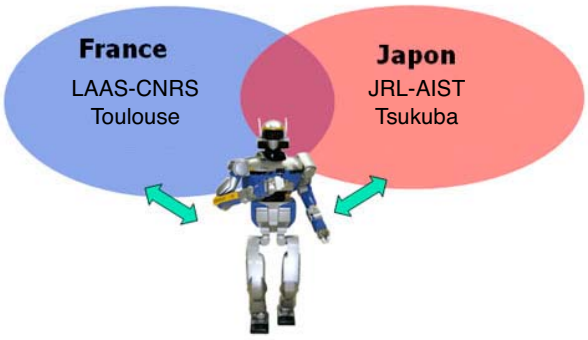
E. Yoshida and M. Poirier and J.-P. Laumond and O. Kanoun and F. Lamiraux and R. Alami and K. Yokoi : "Regrasp Planning for Pivoting Manipulation by a Humanoid Robot", *Proc. 2009 IEEE Int. Conf. on Robotics and Automation*, 2009.

E. Yoshida and M. Poirier and J.-P. Laumond and O. Kanoun and F. Lamiraux and R. Alami and K. Yokoi : "Pivoting based manipulation by humanoid robot", *Autonomous Robots 2009*.

1.5 Contexte de la thèse

Cette thèse a été réalisée au Laboratoire d'Analyse et d'Architecture des Systèmes (*LAAS*) mais aussi dans la structure du Laboratoire International Associé Franco-Japonais AIST-CNRS : *JRL-France* basé en partie à Toulouse et *JRL-Japan* basé a Tsukuba.

Aussi, grâce à un projet de coopération internationale entre le JST et le CNRS permettant l'échange de support technique et humain et l'organisation de colloques, pendant 3 ans (2008/2009 - 2010/2011), on m'a offert la possibilité de valider nos recherches et expérimentations à la fois sur le robot humanoïde HRP-2 n°14 en France mais aussi sur le robot HRP-2 n°10 au Japon dans des environnements d'expérimentation différents. Le budget du projet étant de 30*k* euros/an pour la France et 4*M* de yens/an pour le Japon.



2
Etat de l'art

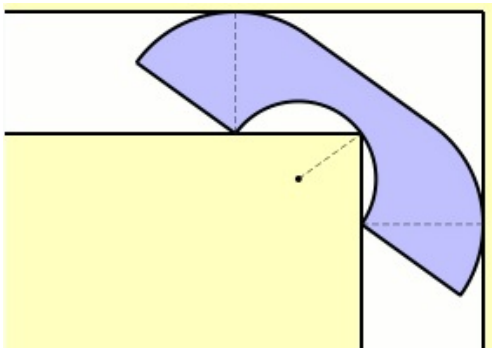
Dans ce chapitre, nous allons mettre en place les bases de ce qui va suivre. Nous aborderons des thèmes différents comme la planification et la manipulation en robotique ou encore la génération de mouvement pour des robots de type humanoïde. Le tout ayant pour but de comprendre ce qui a déjà été fait et ce sur quoi nous nous sommes basés pour approfondir, valider, consolider ou améliorer nos idées.

2.1 Planification de mouvement

2.1.1 Le problème de la planification de mouvement

Le problème de la planification de mouvement est un vieux problème. Que les gens en soient conscients ou non, nous nous posons tous fréquemment ce genre de questions. Bien sûr nous préférons nous poser le problème d'une manière tout à fait différente de ce qui va être montré dans cette section. Pourtant, qui n'a pas un jour réalisé un déménagement et été confronté aux problèmes du genre : comment vais-je pouvoir sortir mon canapé, mon lit ou encore cette commode de mon appartement et descendre les trois étages sans ascenseur ? La planification de mouvement répond à ce genre de problèmes qui ne sont pas forcément si simples à résoudre.

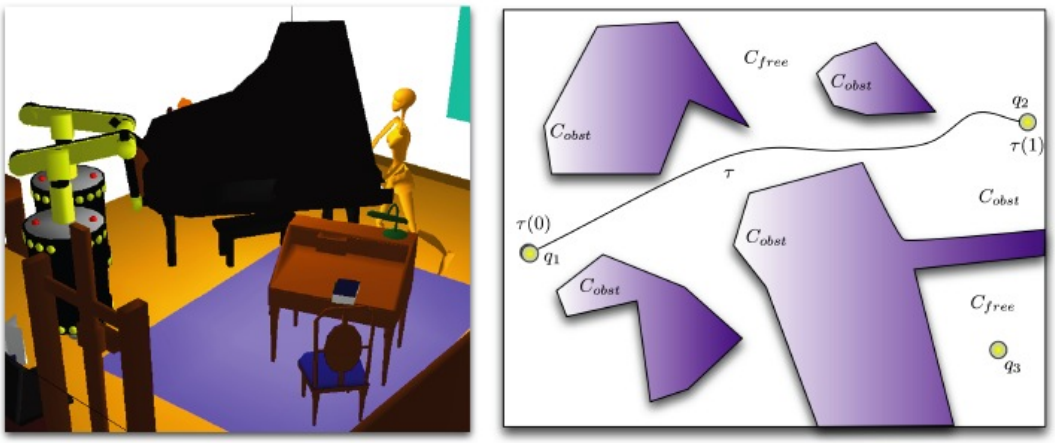
Ce problème a été formulé pour la première fois en 1966 par le mathématicien Leo Moser. Appelé *"le problème du sofa"*, il s'énonce de la manière suivante : Quelle est le plus large canapé (ou la plus grande surface) que l'on peut bouger dans un couloir de largeur unitaire ayant un angle droit ? (voir Fig. 2.1). Malgré un nombre conséquent de réponses [Wagner 1976; Gerver 1992; Croft et al. 1992; Stewart 2004], la valeur exacte de la constante reste un problème ouvert.



$A = \pi/2 + 2/\pi$ or 2.207416099 mais ce n'est pas la meilleure solution

FIGURE 2.1 – de [Stewart 2004] Exemple de solution proposée J. Hammersley au problème du sofa formulé par L. Moser

De ce problème initial découlent plusieurs variantes dont le problème dit "*du déménageur de piano*" posé par [Schwartz and Sharir 1987]. Ces premiers travaux mèneront ensuite [Latombe 1991] à développer ses recherches sur ce que nous appelons la *planification de mouvement*. Ce nouveau problème peut être énoncé de la manière suivante : Existe t-il un chemin libre de toute collision dans l'espace Euclidien ou espace de travail que l'on notera W, permettant de déplacer un piano d'une position et orientation initiale, appelée aussi configuration q_{init}, vers une position et orientation finale ou configuration q_{finale}, connaissant exactement l'environnement et la position des obstacles (Fig. 2.2a) ?



(a) (b)

FIGURE 2.2 – [de Esteves 2007]. (a) Exemple du problème du déménageur de piano. Deux robots et un personnage virtuel travaillent en collaboration pour déplacer un piano dans $W \subset \mathbb{R}^3$. (b) L'espace de configuration est divisé en C_{free} et C_{obst}. Un chemin entre q_1 et q_2 peut-être trouvé car ces configurations appartiennent à la même composante connexe ce qui n'est pas le cas de q_3

En 1980, [Lozano-Pérez 1980] introduit le concept de l'espace de configuration, ou espace C. Le

principe consiste à représenter dans cet espace un système donné par un point. Ce point représente la configuration du système. La dimension de la variété $C \subset \mathbb{R}^n$ est égale au nombre de variables représentant l'état d'un système ou son nombre de degrés de liberté : soit pour l'exemple du piano sa position dans l'espace x, y, z et son orientation θ, ϕ, ψ, d'ou $C_{piano} \subset \mathbb{R}^3 \times \mathbb{S}^3$. Ceci a permis d'avoir une nouvelle approche sur les problèmes de planification de mouvement car nous transformons le problème de faire bouger un corps de l'espace Euclidien $W \subset \mathbb{R}^3$ en un problème où il faut déplacer un point dans C Fig. 2.3 . Les obstacles dans l'espace W induisent des configurations en collision dans C. On notera l'ensemble de ces configurations C_{obst}. C_{free} est défini comme l'espace de configuration libre de l'espace de configuration C : $C_{free} = C \setminus C_{obst}$. Dans ce contexte, le problème initial est transformé en un nouveau problème consistant à calculer C_{obst} et à trouver un chemin $\tau : [0,1] \rightarrow C_{free}$ qui connecte une configuration initiale $\tau(0) = q_{init}$ à une configuration finale $\tau(1) = q_{finale}$. Ce chemin existe si et seulement si q_{init} et q_{finale} appartiennent à la même composante connexe de C_{free}, c'est-à-dire qu'elles appartiennent à un seul et même morceau de l'espace C_{free} Fig. 2.2b.

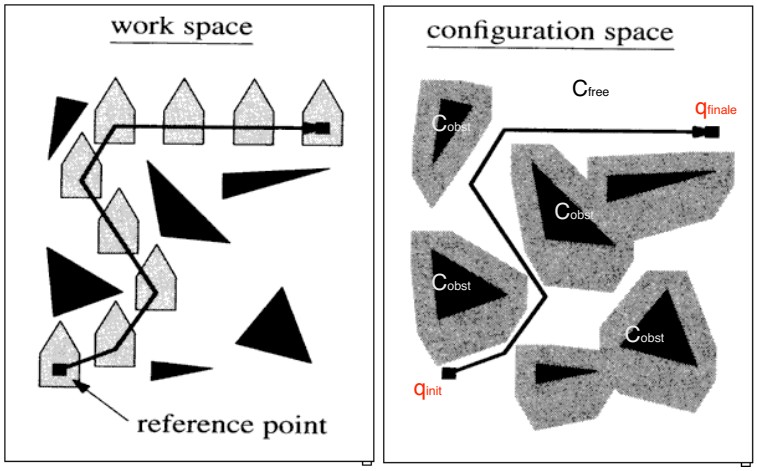
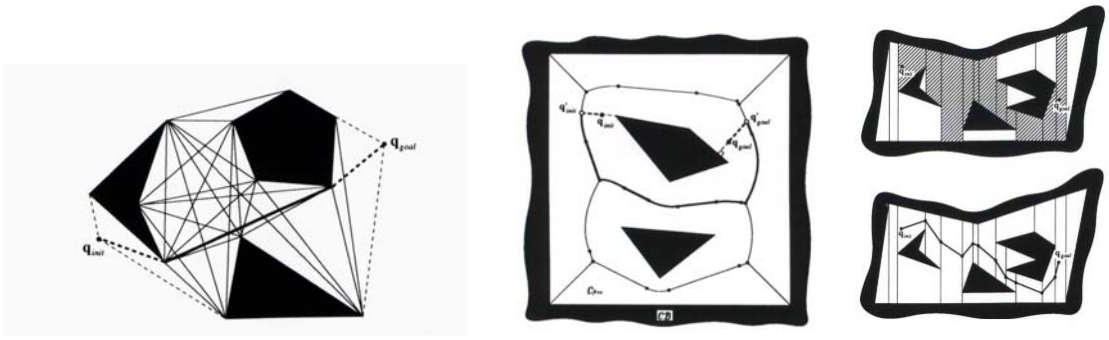


FIGURE 2.3 – Exemple de transformation de l'espace de travail W en espace de configuration C pour un mobile en translation dans le plan

Nous trouvons dans la littérature plusieurs solutions proposées par [Schwartz and Sharir 1987; Goodman and O'Rourke 2004; Canny 1988] sur la construction de la représentation de C_{obst}. Ces méthodes reposent sur la décomposition cellulaire, les graphes de visibilité ou encore les diagrammes de Voronoï [Latombe 1991] Fig. 2.4. Elles construisent un graphe dans C_{free}, permettant de trouver un chemin connectant q_{init} et q_{finale}, s'il en existe un. Bien que ces approches aient des propriétés telles que la complétude [1], la complexité combinatoire est telle qu'elle prend des temps de calcul conséquents, même pour les problèmes les plus simples.

1. propriété algorithmique disant que l'algorithme trouvera toujours la solution si celle-ci existe ou permettra de conclure si elle n'existe pas



(a) (b) (c)

FIGURE 2.4 – (a) graphe de visibilité, (b) diagramme de Voronoï, (c) décomposition cellulaire

2.1.2 Les méthodes de planification par échantillonnage aléatoire et diffusion

Depuis la fin des années 90, de nouvelles méthodes ont été développées pour réduire le temps de calcul à des temps raisonnables. Elles trouvent leurs origines dans les travaux de [Barraquand and Latombe 1991] et [Bessiere et al.] . Pour atteindre ces objectifs, ces méthodes affaiblissent un peu la propriété de complétude mais elles permettent de résoudre, avec efficacité, des problèmes dans des dimensions plus élevées.

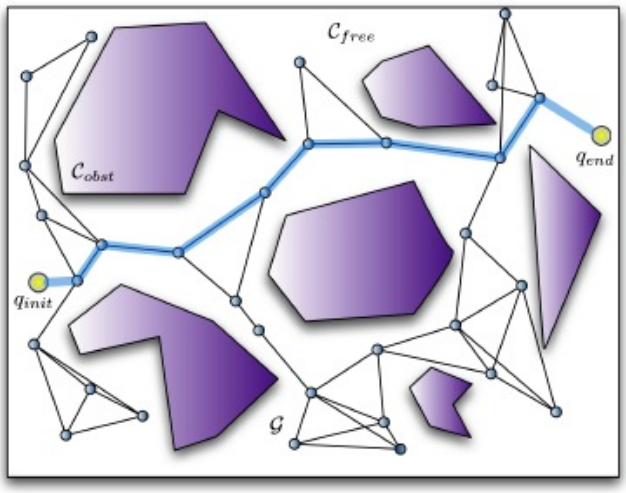
Le but de ces méthodes est de capturer la topologie de C_{free} dans un graphe (en anglais *Roadmap*) sans calculer explicitement C_{obst}. Une fois que nous disposons de ce graphe dans C_{free} il suffit de rechercher un chemin solution permettant de connecter q_{init} et q_{finale}.

Ces graphes sont construits de deux manières : par *échantillonnage aléatoire* ou par *diffusion*. Ces méthodes ont une propriété de *complétude probabiliste*, c'est-à-dire que la probabilité de trouver une solution si elle existe converge vers 1 si le temps de calcul tend vers l'infini. Elles ne permettront pas de décider si une solution n'existe pas.

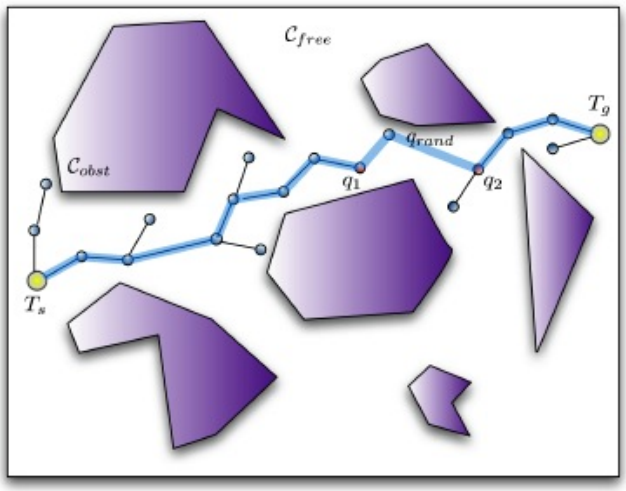
L'algorithme - PRM L'idée principale des méthodes d'échantillonnage aléatoire introduite par [Kavraki et al. 1996] et sa méthode de graphe probabiliste (en anglais *Probabilistic Roadmaps - PRM*) sont illustrées Fig. 2.5a. Elle consiste à tirer des configurations aléatoirement dans C. Si la nouvelle configuration q_{new} est en collision avec un obstacle (pour cela nous utilisons un test de collision), elle est rejetée. Au contraire si celle-ci est libre de collision, elle est rajoutée au graphe et reliée aux autres noeuds, si la méthode locale de planification (voir section 2.1.3) reliant les deux noeuds conduit à un chemin libre de collision.

Les algorithmes de diffusion - EST - RRT Lorsque nous parlons des algorithmes de diffusion, nous faisons souvent référence aux arbres expansifs dans l'espace (en anglais *Expansive-Space Trees - EST*) de [Hsu et al. 1998] ou aux arbres d'exploration rapide aléatoire (en anglais *Rapid-Random Trees - RRT*) de [LaValle 1998]. Ces techniques consistent à explorer C de manière aléatoire, en partant de quelques

configurations appelées *racines* et en explorant le voisinage dans une direction choisie arbitrairement, jusqu'à ce que la configuration finale q_{finale} soit atteinte Fig. 2.5b. De la même manière, une fois la direction choisie pour que le noeud soit validé, il faut que la configuration et le chemin reliant les deux noeuds soient libres de collision.



(a)



(b)

FIGURE 2.5 – [de Esteves 2007] (a) dans l'algorithme PRM le graphe est construit aléatoirement dans la zone libre d'obstacle, chaque configuration, tirée et libre de collision, est gardée et constitue un noeud du graphe. Ces noeuds sont ensuite reliés par une méthode de planification locale exécutable par le robot, les arêtes. Une fois le graphe constitué, nous recherchons le plus court chemin entre q_{init} et q_{finale} (b) Dans l'algorithme RRT deux arbres peuvent se développer simultanément, depuis $T_s = q_{init}$ et $T_g = q_{finale}$ respectivement. Chaque arbre se développe jusqu'à ce que deux configurations q_1 et q_2 appartenant à chacun des arbres puissent être connectées par une troisième configuration q_{rand} tirée aléatoirement.

L'algorithme - PRM de visibilité Pour les systèmes ayant un grand nombre de degrés de liberté, et dont l'espace de configuration C est hautement dimensionné, il existe une méthode, que nous utiliserons dans nos travaux, dérivée des algorithmes PRM appelée PRM de visibilité (en anglais *Visibility Probabilistic Roadmap - Visibility PRM*) proposé par [Simeon et al. 2000] et illustré Fig. 2.6.

Cette technique sera détaillée ultérieurement dans le manuscrit (voir section 5.1) mais l'idée principale, comme celle de l'algorithme PRM, est de capturer la topologie de C_{free} dans un graphe. Cependant, dans cette version, toutes les configurations tirées aléatoirement et libres de collision ne sont pas nécessairement incluses dans le graphe. Seuls les noeuds avec une certaine *visibilité* ou *connexion* sont retenus et ajoutés.

Le graphe obtenu par un PRM de visibilité est plus compact que celui obtenu par un PRM classique. Ceci permet, pour des environnements très complexes, de construire rapidement un graphe de base ou une vue d'ensemble de l'environnement qui pourra par la suite être affinée par d'autres techniques, comme celles citées au-dessus.

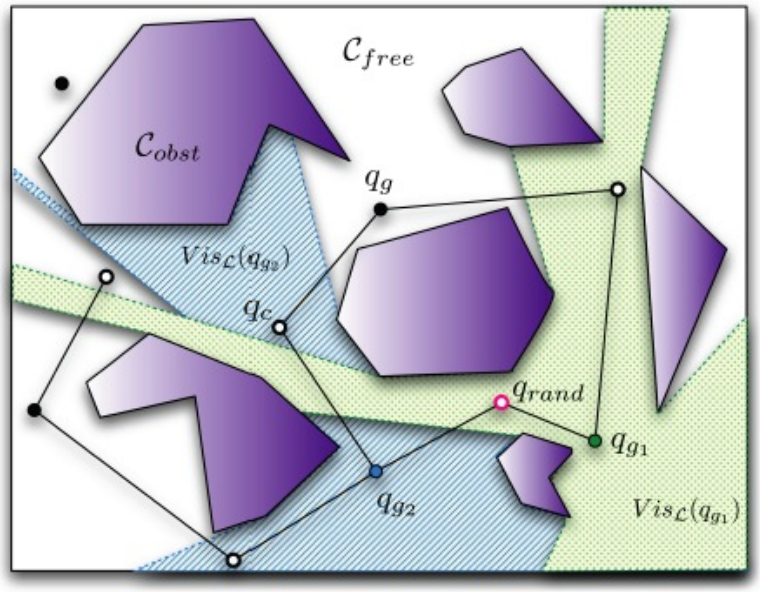


FIGURE 2.6 – [de Esteves 2007] L'algorithme de PRM Visibilité produit un graphe plus compact dans C_{free}, ce qui permet de capturer l'environnement plus rapidement mais avec moins de précision. Les noeuds en noir sont appelés des *gardes* q_g et les blancs sont les *connecteurs* q_c. Un nouveau noeud q_{rand} devient un garde si celui-ci ne peut être relié à aucun autre garde, il devient un connecteur s'il peut être relié à au moins deux gardes, sinon le noeud est rejeté.

Lorsque l'on utilise les algorithmes de planification par échantillonnage aléatoire, un chemin solution est le résultat de deux étapes bien distinctes : la phase d'apprentissage et la phase décisionnelle. Dans la phase d'apprentissage, des configurations aléatoires sont tirées selon la plage autorisée pour chaque degré de liberté du système en vue de construire un graphe probabiliste de l'environnement. Dans la

phase décisionnelle, les configurations initiales et finales sont ajoutées et connectées au graphe par des arêtes libre de collision.

Une recherche de graphe est ensuite lancée pour rechercher un chemin libre de collision entre les configurations initiales et finales. Nous noterons que la recherche d'un chemin solution dans un graphe peut se faire selon plusieurs approches, en utilisant différentes heuristiques. Chaque arête du graphe peut possèder un coût et suivant ce que l'on souhaite mettre en avant, l'algorithme de recherche prendra l'arête qui conviendra le mieux au type de solution recherchée. On peut, par exemple, rechercher le chemin le plus court mais aussi le chemin qui propose le moins de manoeuvres, ou le moins consommateur d'énergie.

Si un chemin est trouvé, il est ensuite lissé pour retirer les détours ou redondances inutiles [Hsu et al. 1999]. Puis le chemin est transformé en un chemin paramétré en temps, aussi appelé *trajectoire* $\tau(t)$ par des techniques classiques [Shin and McKay 1985; Bobrow et al. 1985; Slotine and Yang 1989; Lamiraux and Laumond 1999]

2.1.3 Méthode locale de planification et commandabilité locale en temps petit

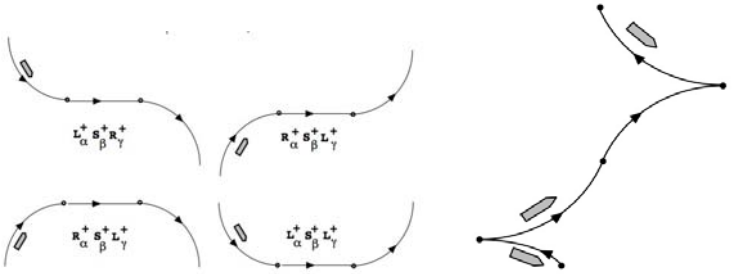
Méthode locale de planification Comme nous venons de le voir dans la section précédente, pour relier deux noeuds d'un graphe, nous utilisons une fonction appelée *méthode locale de planification*. Si cette fonction produit un chemin, entre deux configurations aléatoires, libre de collision, le chemin est ajouté au graphe et en constitue une arête. Si le système peut bouger librement dans toutes les directions (ex : le piano), n'importe quel chemin de C_{free} correspond à un chemin libre de collision et est admissible par le système dans W ; dans ces cas précis, la ligne droite est souvent considérée comme le chemin le plus court entre deux configurations.

Malheureusement, ceci ne fonctionne pas pour la plupart des systèmes, souvent du fait des contraintes mécaniques spécifiques. Prenons, par exemple, le cas d'une voiture, celle-ci peut avancer, reculer et tourner en avançant ou en reculant, mais en aucun cas se déplacer latéralement directement sans exécuter un certain nombre de manoeuvres (ex : garage en créneau). Ce type de contraintes cinématiques qui ne peuvent être intégrées et exprimées sous forme de variables de configuration, sont appelées *contraintes non-holonomes*. Elles ont été introduites dans le contexte de la planification de mouvement par [Laumond 1986] et largement discutées par la suite [Li and Canny 1992; Laumond 1998]. Si nous continuons sur l'exemple de la voiture, on ne dispose que de deux contrôles, sa vitesse linéaire et sa vitesse angulaire. Son espace de configuration est pourtant représenté en trois dimensions avec les coordonnées suivantes (x, y, θ) représentant respectivement la position et l'orientation de la voiture. Trouver un chemin solution admissible pour le système dans C_{free} consiste donc à résoudre le problème en deux parties.

Tout d'abord, un chemin admissible est calculé dans un espace C sans obstacles à l'aide de la méthode locale, puis dans un second temps celui-ci est testé pour savoir s'il est libre de collision dans C_{free} avant d'être ajouté au graphe. Beaucoup de recherches ont été menées sur ces méthodes locales utilisées sur des systèmes comme notre voiture. [Dubins 1957] démontra que le chemin le plus court pour relier en marche avant deux configurations d'une voiture, en absence d'obstacles, était une combinaison d'arcs de cercle et de lignes droites. En rajoutant, des points de rebroussement ou changement de sens, incluant la marche arrière [Reeds and Shepp 1990; Sussmann and Tang 1991] trouvèrent des chemins encore plus

courts.

Nous reviendrons plus précisément sur les courbes de Reeds et Shepp dans le chapitre 4.1.1.3.



(a) (b)

FIGURE 2.7 – de [Sussmann and Tang 1991] (a) Exemple de courbes de Dubins. (b) Exemple de courbes de Reeds et Shepp.

La commandabilité locale en temps petit La notion de commandabilité locale en temps petit est une propriété importante pour les planificateurs que nous utilisons. Nous reviendrons plus en détails sur cette notion dans le chapitre 3.2.

Ce qu'il faut savoir pour l'instant, c'est que si un système est prouvé localement commandable en temps petit, cette propriété permet de transformer n'importe quel chemin, admissible ou non, c'est-à-dire pas nécessairement exécutable par le système, mais libre de collision dans C_{free}, en un chemin admissible pour le système et toujours libre de collision.

Cette propriété est donc utilisée en deux étapes. On peut tout d'abord rechercher des chemins libres de collision par des méthodes rapides dans C_{free} et les approximer par la suite par des méthodes locales de planification respectant la propriété.

Une limitation de cette propriété pour un système donné est que la preuve de la commandabilité locale en temps petit montre simplement l'existence de celle-ci mais ne donne en aucun cas la fonction ou les commandes à appliquer au système pour que celui-ci soit localement commandable en temps petit. Il faut donc le prouver pour chaque méthode locale de planification appliquée au système.

2.2 Le mouvement humanoïde

Nous allons, dans cette section, voir les méthodes de génération de mouvements de locomotion pour un robot humanoïde. Même si celles-ci ne sont pas développées ou ne font pas partie intégrante des recherches menées ici, elles sont en revanche largement utilisées par nos algorithmes.

2.2.1 ZMP et modèles

La notion de *point de moment dynamique nul* (en anglais *Zero Moment Point - ZMP*) est introduite pour la première fois en 1968 par M. Vukobratovic et résumée avec ses travaux dans le livre [Vukobratovic et al. 1990]. Le ZMP est un critère d'équilibre couramment utilisé et essentiel pour la marche humanoïde.

Il est un concept mettant en relation le contrôle et la dynamique d'un système bipède, entre autre, celui des robots humanoïdes. Il représente le point où la force de réaction dynamique au contact du pied avec le sol ne produit pas de moment, c'est-à-dire le point où la force d'inertie est égale à zéro. En équilibre statique, il représente la projection du centre de masse du système au sol. Le concept du ZMP suppose que la zone de contact entre les pieds et le sol, soit plane et suffisamment adhérente pour maintenir les pieds au sol sans glissement Fig. 2.8.

On considère qu'un robot humanoïde est dynamiquement stable si le ZMP se situe dans le polygone convexe formé par les pieds : son polygone de sustentation Fig. 2.8, que ce soit en phase de simple support ou en phase de double support.

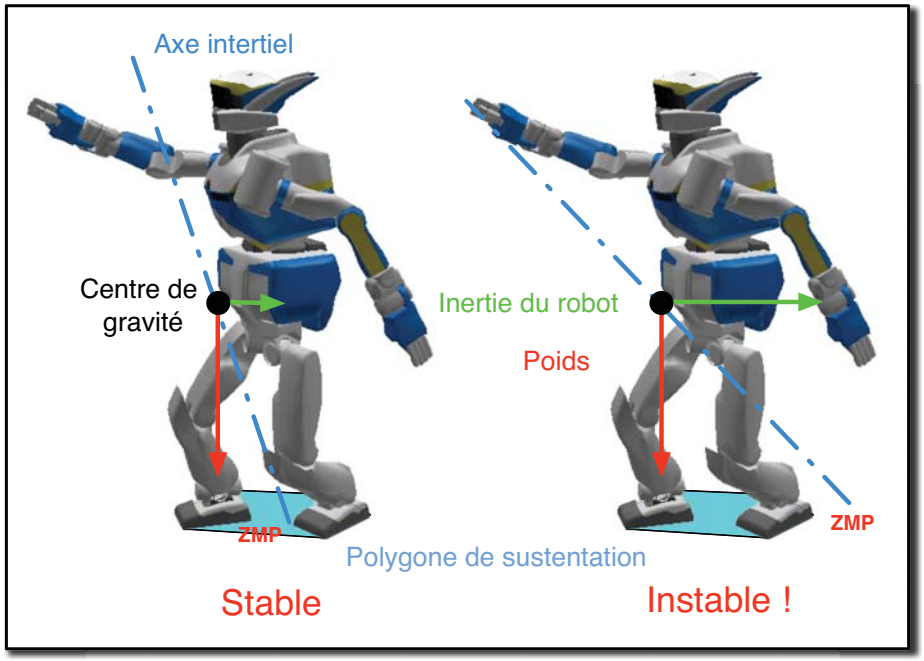
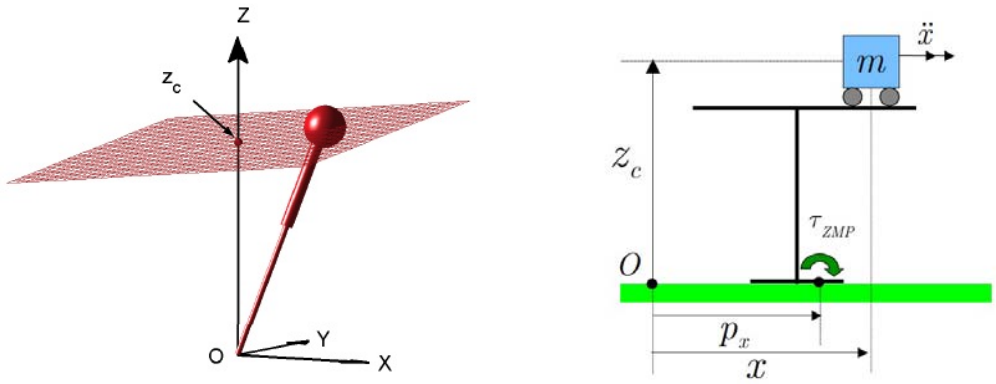


FIGURE 2.8 – HRP-2, ZMP, polygone de sustentation et stabilité

Le ZMP dépendant de la position, la vitesse et l'accélération de tous les corps du robot, son contrôle est très coûteux, notamment pour des robots à grand nombre de degrés de liberté. Pour surmonter ce problème, [Kajita et al. 1992; Yamaguchi et al. 1993] proposent de simplifier le modèle dynamique. Les deux représentations les plus utilisées sont le modèle du pendule inverse contraint sur un plan et le modèle de poids en équilibre sur une table (en anglais *Cart Table Model*) Fig.2.9. Ces modèles permettent, notamment, de linéariser les équations différentielles du ZMP et donc de simplifier les calculs pour la locomotion humanoïde basée sur cette approche [Kajita et al. 2003], sous les conditions suivantes :

- Le haut du corps ne bouge pas ou peu.
- Le poids des jambes est raisonnablement faible.
- Le mouvement de marche est régulier.



(a) Modèle pendule inverse

(b) Modèle de poids en équilibre sur une table (coupe 2D)

FIGURE 2.9 – images de [Kajita et al. 2003]

2.2.2 Générateur de marche

Un générateur de marche (en anglais *Pattern Generator - PG*) est un programme permettant de produire des mouvements de marche dynamiquement stable pour un robot bipède. Nous utilisons pour nos recherches le générateur de marche par contrôle prédictif, basé sur les simplifications citées ci-dessus et proposé par [Kajita et al. 2003].

L'utilisateur doit fournir en entrée du générateur une pile d'empreintes, utilisée pour planifier une trajectoire de ZMP de référence : $ZMP_{\text{référence}}$, à suivre pour que le robot reste dynamiquement stable. Cette trajectoire de référence est ensuite utilisée pour générer la trajectoire du centre masse du système, qui peut être vue, à un instant t, comme une tâche de cinématique inverse. [Kajita et al. 2003] utilisent dans leur générateur de marche des calculs de cinématique inverse exacte sur la partie basse du corps. Le mouvement dynamique engendré par le mouvement du haut du corps doit rester négligeable pour respecter la simplification.

Le ZMP est ensuite calculé de manière exacte grâce à la vitesse et à l'accélération du robot, à l'instant t. Ce ZMP présente un écart avec le $ZMP_{\text{référence}}$ dû au modèle de simplification utilisé. Le générateur de marche inclut donc un deuxième étage (Fig. 2.10) permettant de compenser cette erreur dans le contrôleur. Le contrôleur asservit le ZMP du robot en contrôlant le centre de masse. Pour cela, ils utilisent un contrôleur prédictif qui permet de calculer la valeur du centre de masse à un instant t en fonction de la trajectoire de ZMP future désirée.

Le mouvement est validé si le ZMP reste dans une zone de sécurité définie par le polygone de sustentation.

[Kajita et al. 2003] montrent que pour le robot HRP-2, une fenêtre de prédiction d'une durée de $t = 1.6s$ suffit à prédire correctement les futures références. La figure 2.11 présente le concept général de ce générateur de marche.

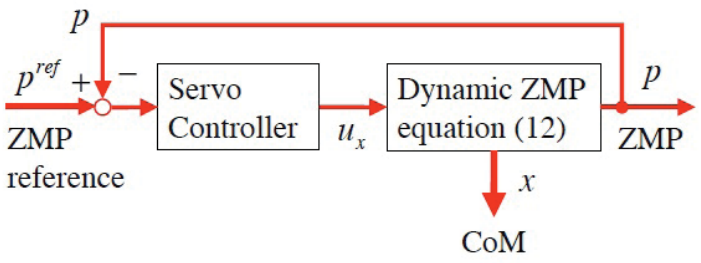


FIGURE 2.10 – de [Kajita et al. 2003] suivie du ZMP par le deuxième étage du générateur de marche

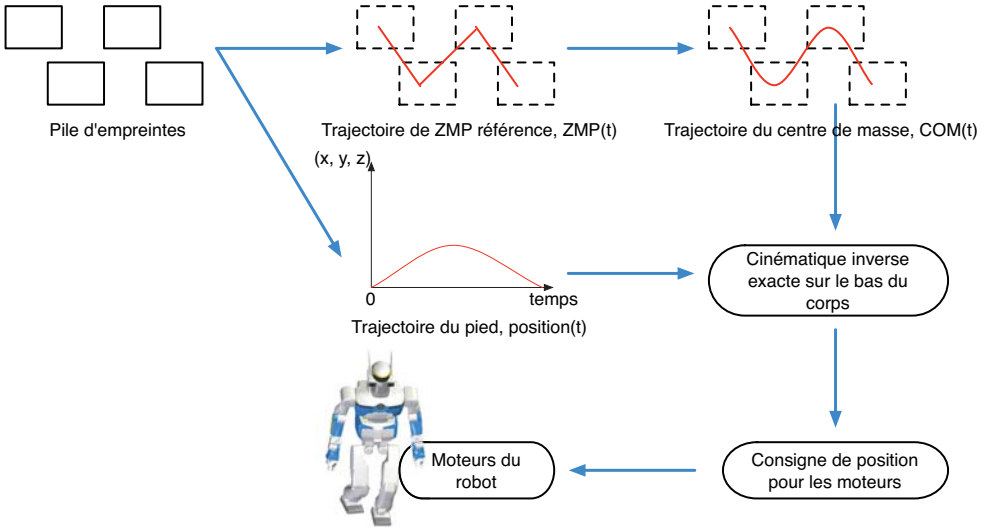


FIGURE 2.11 – Transformation de la pile d'empreintes en mouvement de marche pour le robot HRP-2

2.2.3 Cinématique inverse généralisée

Le problème de cinématique inverse est le problème du calcul de la configuration d'un robot satisfaisant une tâche donnée, fonction (souvent non linéaire) de la configuration. Par exemple, pour un robot de type bras manipulateur, nous cherchons à calculer la valeur de chaque degré de liberté du bras afin de positionner l'organe terminal, ou effecteur, dans une position et orientation souhaitées. Des techniques sont connues et utilisées depuis plusieurs décennies. Ici, nous pouvons considérer notre robot humanoïde, comme quatre bras manipulateurs attachés à une même base, elle-même mobile.

Contrairement à notre robot, sur un bras manipulateur classique, lorsque nous bougeons le bras, la plupart des degrés de liberté sont utilisés. Sur un robot humanoïde, nous pouvons tout à fait bouger un bras du robot sans utiliser le reste du corps, mais ce dernier est-il inutile pour autant ? On dit que le système est redondant.

[Liegeois 1977; Nakamura 1991; Siciliano and Slotine 1991] introduisent la cinématique inverse généralisée qui permet pour des systèmes hautement dimensionnés, comme notre robot anthropomorphe

à trente degrés, d'utiliser au mieux toutes les capacités du système, ou ici de son corps, pour positionner un ou plusieurs organes terminaux dans des positions et/ou orientations désirées. Les algorithmes de cinématique inverse généralisée utilisent pour cela la redondance ou les multiples possibilités qu'offrent de tels systèmes, comme ici un corps anthropomorphe. On parlera alors de posture pour un robot humanoïde et dans notre cas, de *mouvements corps complet* (voir définition au début du chapitre 1.2.1). Comme pour les êtres humains, toute posture n'est pas forcément admissible par le système. La plupart du temps, nous souhaitons réaliser une tâche principale et utiliser le reste de notre corps pour, soit s'équilibrer, soit accomplir une tâche secondaire ou moins importante.

[Nakamura 1991; Siciliano and Slotine 1991; Baerlocher and Boulic 2004] introduisent alors le concept de cinématique inverse généralisée avec priorité de tâche. Ce mécanisme de priorité sera expliqué plus en détail dans le chapitre 4.3. Les tâches à réaliser sont ordonnancées dans une pile de tâches. A un instant t, les algorithmes tentent de satisfaire tout d'abord la ou les tâches avec la priorité la plus importante et utilisent le sous-ensemble des degrés de liberté non ou peu utilisé pour réaliser au mieux les tâches secondaires qui doivent être exécutées au même instant.

L'idée de [Yoshida et al. 2006] est de combiner la cinématique inverse généralisée avec priorité de tâche [Nakamura 1991] avec le générateur de marche de [Kajita et al. 2003] Fig. 2.12.

Fig.2.13 montre que dans cette version du générateur de marche la cinématique inverse exacte est remplacée par la cinématique inverse généralisée. Ils prennent en compte directement les mouvements du haut du corps. De même, ils suppriment le deuxième étage du générateur de marche de [Kajita et al. 2003]. Le mouvement est validé de la même manière. Les conditions de simplification citées section 2.2.1 sont assurées en veillant à la continuité des tâches de cinématique inverse généralisée et à leur variation régulière.

Ces algorithmes sont les algorithmes que nous utiliserons. [Yoshida et al. 2006] proposent donc de ne plus limiter l'espace d'accessibilité du robot au simple espace qui l'entoure directement. En replanifiant son polygone de sustentation, ce qui revient à exécuter un pas, le robot peut atteindre une plus grande partie de l'espace Fig 2.14. De même, si l'on donne au robot la tâche de tourner la tête pour regarder derrière lui, arrivé en butée articulaire du cou ou un peu avant, le robot va être capable d'utiliser ses jambes pour se retourner et accomplir la tâche demandée. Si l'on exécute plusieurs fois cette replanification, le robot sera capable d'atteindre des buts ou d'exécuter des tâches qui se trouvent loin de lui sans pour autant avoir planifié ou calculé une pile d'empreintes au préalable [Sreenivasa et al. 2009].

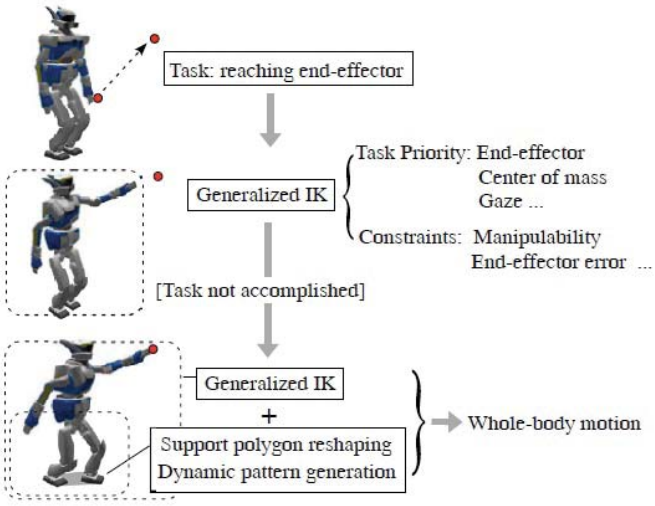


FIGURE 2.12 – de [Yoshida et al. 2006] cadre général d'utilisation des algorithmes de cinématique inverse incluant la replanification du polygone de sustentation

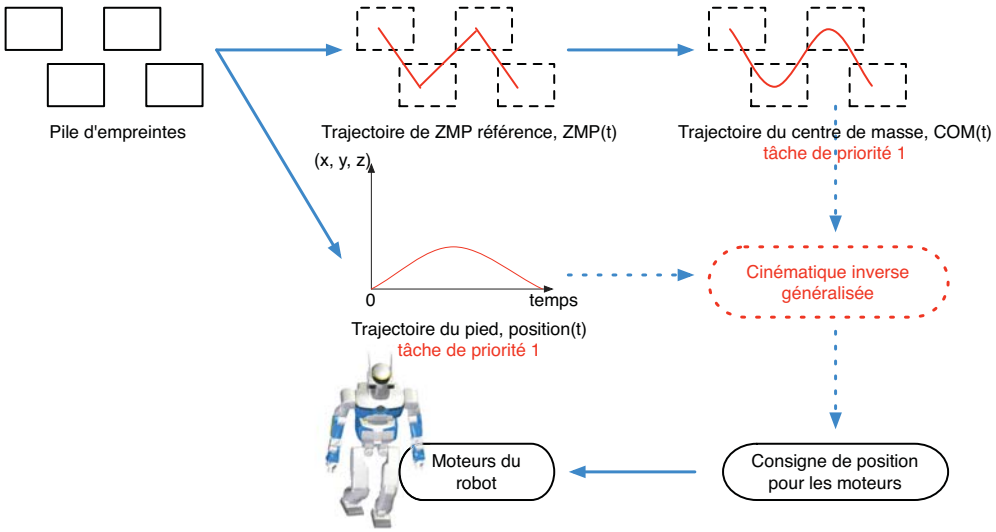


FIGURE 2.13 – Transformation de la pile d'empreintes en mouvement de marche corps complet par cinématique inverse généralisée et priorité de tâches, pour le robot HRP-2

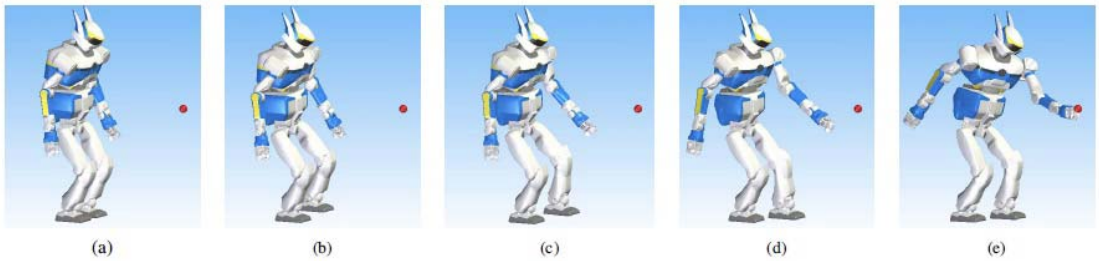


FIGURE 2.14 – de [Yoshida et al. 2006] Exemple de tâche réalisée par les algorithmes de cinématique inverse généralisée. Le Robot doit regarder et toucher la boule rouge avec sa main gauche. Celui-ci ne peut y arriver sans faire un pas. L'équilibre du robot étant la tâche prioritaire dans cet exemple, nous pouvons noter que pendant qu'il exécute la tâche avec son bras gauche, le robot utilise son bras droit pour maintenir l'équilibre en place.

2.2.4 Extension à la commande en couple

Dans le même esprit que les travaux ci-dessus pour la cinématique inverse généralisée [Sentis and Khatib 2004; Sentis and Khatib 2005] s'intéressent à la commande en couple pour le contrôle dynamique d'un robot humanoïde ou d'un avatar virtuel, en temps réel Fig. 2.15. Ces commandes peuvent aussi être utilisées pour réaliser des mouvements robotiques compliants ou faire du contrôle en force.

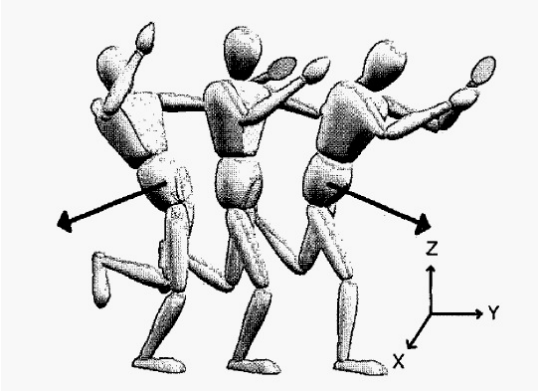


FIGURE 2.15 – de [Sentis and Khatib 2005] Exemple de simulation dynamique temps réel. L'avatar doit garder l'équilibre pendant qu'on lui applique des forces sur le bassin. On remarquera qu'il tente de s'équilibrer en utilisant ses bras et des mouvements de bassin opposés à la force.

Ce qui semble fonctionner en simulation avec des modèles de robots humanoïdes ou d'avatars virtuels que l'on peut contrôler directement en couple, ne fonctionne pas forcément en réel. Effectivement, la majorité des robots actuels, dont HRP-2, ne peuvent être contrôlés en couple. Ces robots sont contrôlés en position. Dans ses recherches, [Khatib 2008] tentent de transformer, en temps réel, ces commandes en couple, en des commandes en position exécutables sur les robots actuels. Malheureusement, les solutions

proposées sont encore coûteuses et nécessitent un modèle précis des moteurs utilisés sur les robots, choses dont nous ne disposons pas. La solution proposée par [Khatib 2008] impose de construire un "*transformateur*" pour chaque articulation commandée.

2.2.5 Prise en compte de la dynamique en planification

Dans leurs travaux, [Esteves 2007; Yoshida et al. 2008] s'intéressent à la prise en compte de la dynamique dès l'étape de planification. Leur méthode repose sur la *décomposition fonctionnelle*. Celle-ci consiste à diviser l'ensemble des articulations du système en un ensemble de sous-systèmes ayant des fonctions différentes Fig. 2.16.

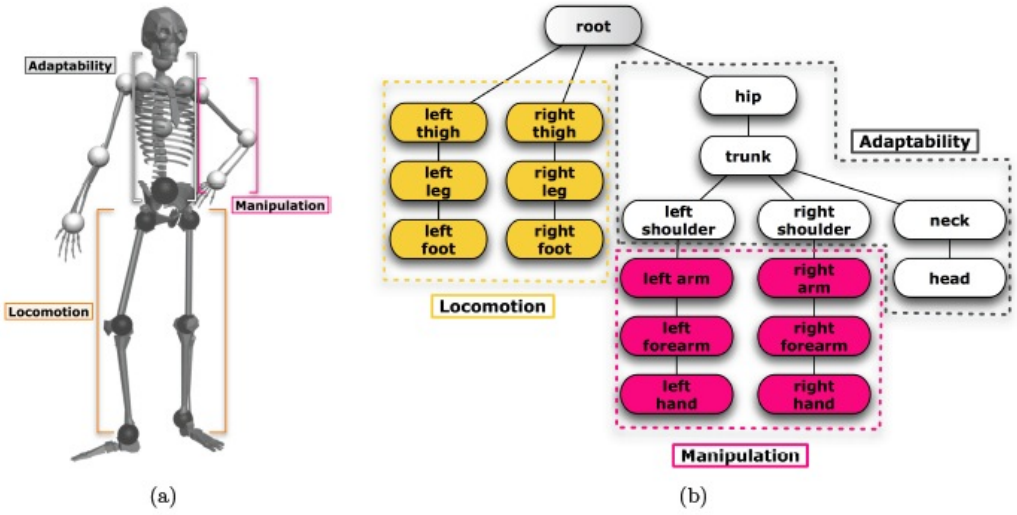


FIGURE 2.16 – de [Esteves 2007] (a) La décomposition fonctionnelle. Le système est décomposé en trois groupes : locomotion, manipulation et adaptabilité (b) Représentation de la décomposition sous forme d'arbre. Les arêtes représentent les articulations et les noeuds, les corps rigides attachés à ces articulations.

- Le groupe de locomotion réunit l'ensemble des degrés de liberté nécessaires au déplacement du système.
- Le groupe de manipulation regroupe l'ensemble des degrés de liberté impliqués dans la manipulation d'objets.
- Le groupe d'adaptation rassemble le reste des degrés de liberté non impliqués dans les deux premiers groupes et permettant d'exploiter la redondance du système anthropomorphe ou à des fins d'évitement d'obstacles.

L'avantage d'un tel découplage est que les sous-systèmes créés deviennent plus faciles à exploiter. Ils ont pu utiliser, par exemple, le générateur de marche de [Kajita et al. 2003] pour produire la marche

avec le groupe de locomotion sans s'occuper de la partie supérieur du corps du robot et inversement utiliser les algorithmes de manipulation par chaîne fermée de [Cortes and Simeon 2005] pour le groupe manipulation sans s'occuper de la marche.

Malheureusement, ce découplage présente plusieurs désavantages.

- Il réduit fortement les solutions de postures possibles en réduisant la redondance proposée par les systèmes anthropomorphes ce qui n'est pas le cas des algorithmes générant les *mouvements corps complets* que nous allons utiliser pour la manipulation.
- Une fois que la planification du haut et du bas du corps est faite, le résultat n'est pas forcement dynamiquement stable. Le groupe de manipulation n'étant pas pris en compte dans le calcul de la marche du robot. Le haut du corps peut perturber les trajectoires de marche. Il faut donc vérifier l'ensemble de la solution dans un simulateur dynamique et déformer légèrement les trajectoires s'il existe des collisions. Cette technique fait donc intervenir plusieurs boucles de planification-vérification ce qui coûte beaucoup de temps de calculs Fig. 2.17.

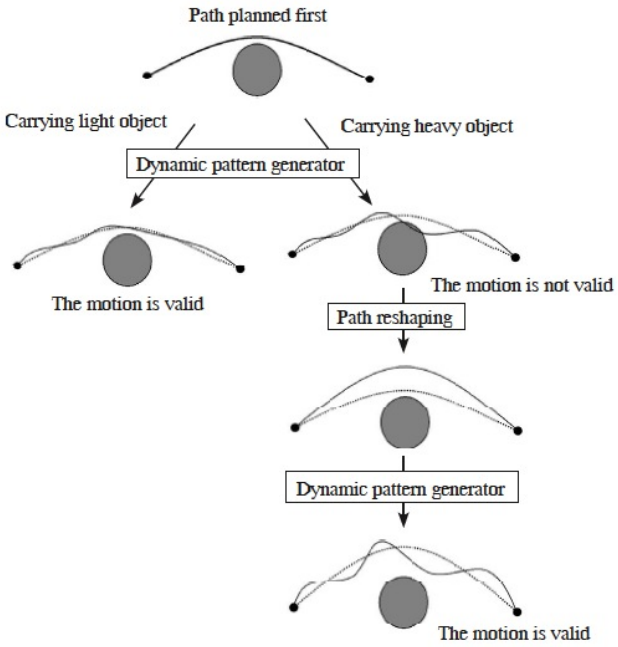


FIGURE 2.17 – de [Esteves 2007] Exemples de trajectoire planifiée en fonction de l'objet à manipuler

2.3 La manipulation en robotique

Nous allons voir dans cette section, l'état de l'art en rapport avec la manipulation d'objets dans la robotique puis dans la robotique humanoïde dans un second temps.

2.3.1 La manipulation de petits objets par des bras manipulateurs

Dans la littérature, nous trouvons des recherches menées sur la manipulation de petits objets par des effecteurs de bras manipulateurs. Ces effecteurs pouvant être des pinces, deux-trois doigts, des plaques mais aussi des mains. Beaucoup de techniques ont été étudiées. Dans ces travaux [Lynch 1992] s'intéresse à pousser ces petits objets avec plusieurs points de contact et sans préhension. [Sawasaki et al. 1989; Bicchi et al. 2004] étudient le déplacement de petits objets par renversement (en anglais *tumbling*). [Bicchi and Sorrentino 1995] s'intéressent la manipulation dextre d'objets sphériques par roulement. [Aiyama et al. 1993] font des recherches sur la manipulation de petits objets par pivotement. Leurs manipulations sont réalisées de manière très précise et en boucle fermée, grâce notamment à un retour vidéo et une analyse d'image. [Maeda et al. 2004] proposent d'utiliser l'ensemble des techniques citées ci-dessus dans un seul et même planifacteur de manipulation.

La figure 2.18 montre quelques unes des ces techniques. Celles-ci se focalisent surtout sur la dextérité que possède un effecteur de bras manipulateur pour réaliser la manipulation sans nécessairement attraper l'objet et le déposer à un endroit différent (en anglais *Pick and Place*).

Dans ce manuscrit, nous n'allons pas étudier la manipulation avec autant de précisions. Nous allons considérer la manipulation comme un ensemble de mouvements corps complet à réaliser pour déplacer un objet encombrant et rendre le mouvement visuellement naturel.

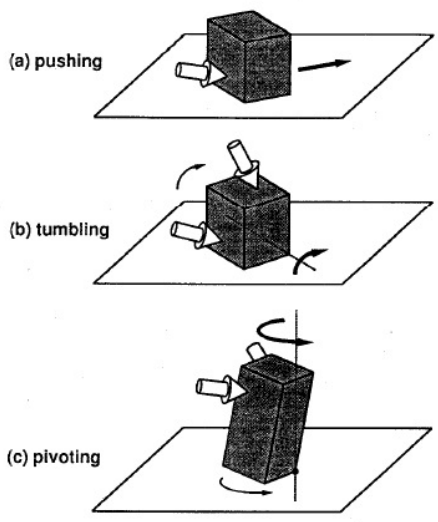


FIGURE 2.18 – de [Aiyama et al. 1993] Exemples des différentes techniques de manipulation sans préhension exécutées par un bras manipulateur muni de deux doigts

2.3.2 La manipulation humanoïde corps complet

Depuis la recherche intensive, ces dernières années, sur les mouvements corps complet, de nouvelles techniques de manipulation de larges objets par un robot humanoïde ont été développées. Plusieurs méthodes sont déjà proposées dans la littérature.

La manipulation par la levée Dans leurs travaux [Harada et al. 2005; Arisumi et al. 2007; Arisumi et al. 2008] s'intéressent aux techniques de manipulation d'objets par la levée (en anglais - *lifting*) et notamment aux techniques de levage ou porté dynamique à l'aide de mouvements corps complet. Effectivement, les robots ne sont pas forcément capables de soulever des masses importantes (voir informations sur HRP-2 chapitre 1.2.3). La figure 2.19 montre typiquement des mouvements d'arraché dangereux ou simplement impossibles à réaliser avec un robot humanoïde.

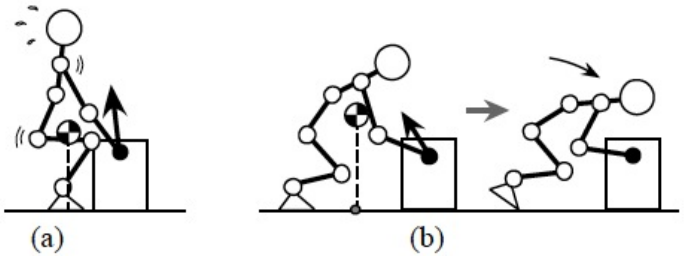


FIGURE 2.19 – de [Arisumi et al. 2007] Exemple de levée d'objets impossible à réaliser par force continue (a) la puissance pour l'arrachée est insuffisante (b) un mauvais équilibre et le robot chute vers l'avant.

[Arisumi et al. 2007] introduisent donc une phase de mouvement préliminaire pour utiliser au mieux la dynamique et notamment le moment créé par l'arrachée pour soulever l'objet du sol. Cette rotation est centrée autour des pieds. Ils utilisent un modèle de pendule inverse attaché à cette rotation pour replacer le centre de gravité du système dans une position dynamiquement stable Fig. 2.20 et 2.21.

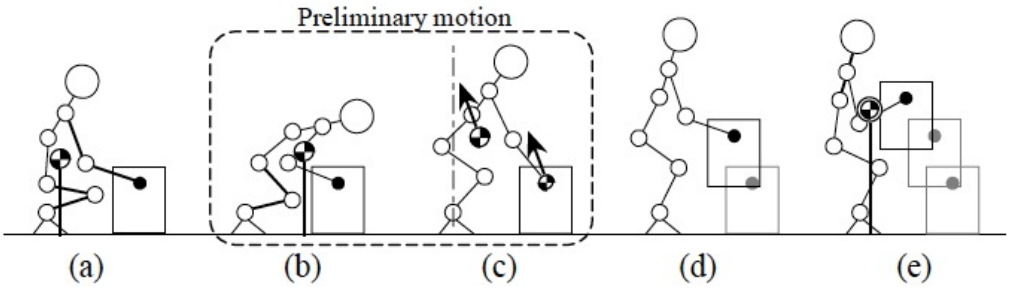


FIGURE 2.20 – de [Arisumi et al. 2007] Mouvement d'arrachée. (a) L'assise. (b) Il se penche vers l'avant. (c) L'arrachéc. (d) Tirer. (e) Tenir.

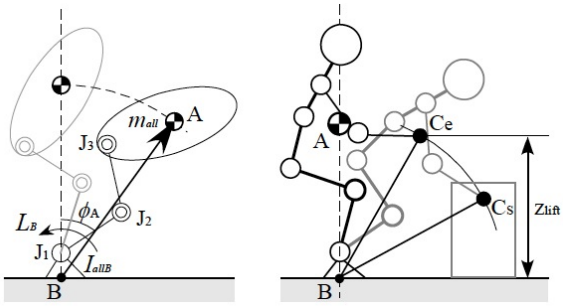


FIGURE 2.21 – de [Arisumi et al. 2007] Modèle de pendule inverse utilisé pendant la phase préliminaire

La manipulation par la poussée Dans les années 90, [Lynch and Mason 1996] ont démontré que l'on peut déplacer des objets de manière stable et contrôlée par des techniques de poussée (en anglais - *Pushing*) avec des robots mobiles à roues ayant une surface de contact plane à l'avant. En 2004, [Harada et al. 2004] proposent une technique de poussée pour les robots humanoïdes avec des mouvements corps complet, couplant le retour de force du capteur d'efforts situé dans les mains avec la trajectoire du ZMP. Comme nous l'avons vu précédemment dans les générateurs de marche, la trajectoire du ZMP influence la marche du robot pour garder son équilibre. Afin de garder une pression à peu près constante, [Harada et al. 2004] modifient l'allure (phase de simple et double support) du robot en temps réel Fig. 2.22

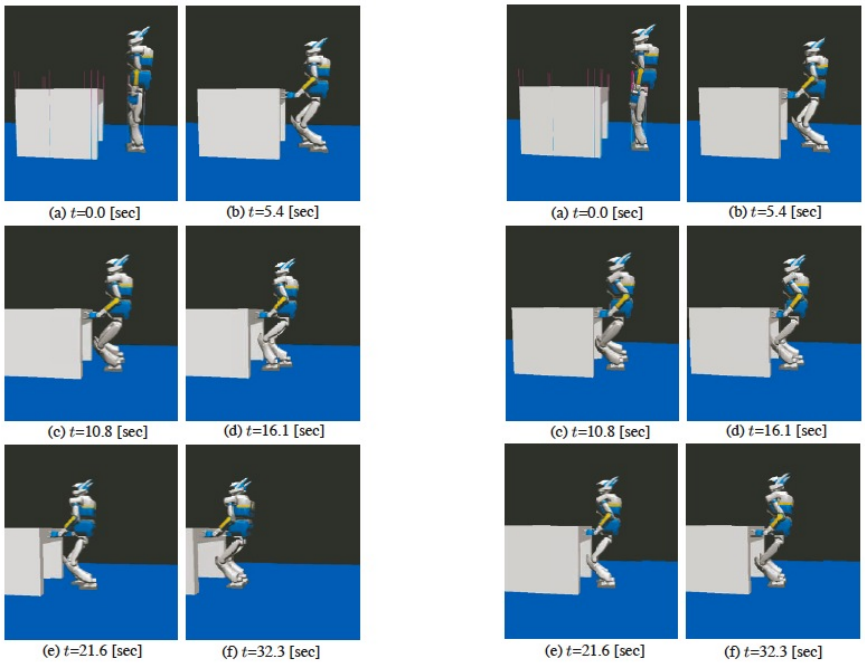


FIGURE 2.22 – de [Harada et al. 2004] sur les images de droite le robot pousse un objet de $5kg$ et sur les images de gauche un objet de $15kg$. On voit que le robot n'a pas parcouru la même distance.

La manipulation par le pivotement Inspirés des travaux de [Aiyama et al. 1993; Maeda et al. 2004] et toujours de l'être humain, [Yoshida et al. 2005; Yoshida et al. 2006] ont mené une étude sur la faisabilité de la manipulation d'un objet encombrant par pivotement. Leur méthode se déroule en deux étapes : tout d'abord, un contrôle de l'objet puis un contrôle du robot. Le robot doit faire un pas après avoir manipulé l'objet afin de suivre celui-ci.

Cette méthode offre plusieurs avantages, résumés ici et comparés aux autres méthodes sur le tableaux figure 2.23 :

- *Un positionnement précis* : Étant donné que l'objet n'est pas poussé mais maintenu entre les mains du robot, l'incertitude sur le déplacement est plus faible. Le robot peut déplacer l'objet dans la position désirée d'une manière assez simple.
- *Adaptabilité au terrain* : Effectivement les méthodes de poussée sont très sensibles à la composition et aux inégalités du terrain alors que la technique de pivotement peut s'adapter facilement. Les méthodes de levée et de renversement pourraient par contre, très bien fonctionner.
- *Stabilité* : La méthode de pivotement présente beaucoup moins de risques, pour le robot, que les méthodes de levée, lorsqu'il faut déplacer un objet lourd et encombrant.

	Precise positioning	Adaptability	Stability
Pushing	×	×	○
Lifting	○	△	×
Tumbling	△	△	○
Pivoting	○	○	○

○ : Suitable, △ : Can be used × : Not suitable

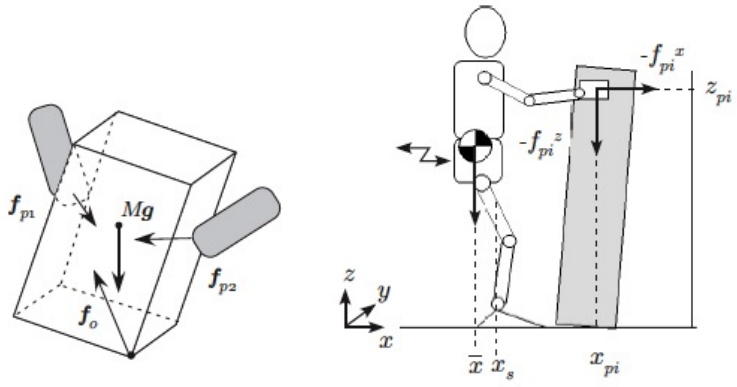
FIGURE 2.23 – de [Yoshida et al. 2006] Ce tableau montre les avantages à reprendre les mouvements par pivotement pour déplacer un objet encombrant

.

Dans ces travaux, plusieurs études sur les forces appliquées au système ont été faites (voir aussi Fig. 2.24) :

- Les forces à appliquer aux mains pour maintenir une pression suffisante sur l'objet.
- La force de contact sur le point de pivot.
- La force à appliquer au bassin pour garder l'équilibre du système.

Nous ne reviendrons pas sur l'étude de ces forces et nous expliquerons ce choix par le fait que celles-ci ont déjà été étudiées ici. Nous nous focaliserons donc sur la planification de la manipulation elle-même.



(a) (b)

FIGURE 2.24 – de [Yoshida et al. 2006] (a) Forces appliquées aux mains et au point de pivot, (b) Forces pour maintenir l'équilibre du système

Cette méthode montre une nouvelle alternative aux techniques précédemment citées, en ouvrant le champ à de nouvelles applications. Ces travaux ont permis de montrer que les mouvements étaient réalisables par un robot humanoïde. Mais ici, on ne parle pas vraiment de déplacement d'objets ou de planification de manipulation. Dans leurs travaux, [Yoshida et al. 2005] ne font que deux mouvements de pivotement. Les pas sont aussi réalisés après le déplacement de l'objet Fig. 2.25

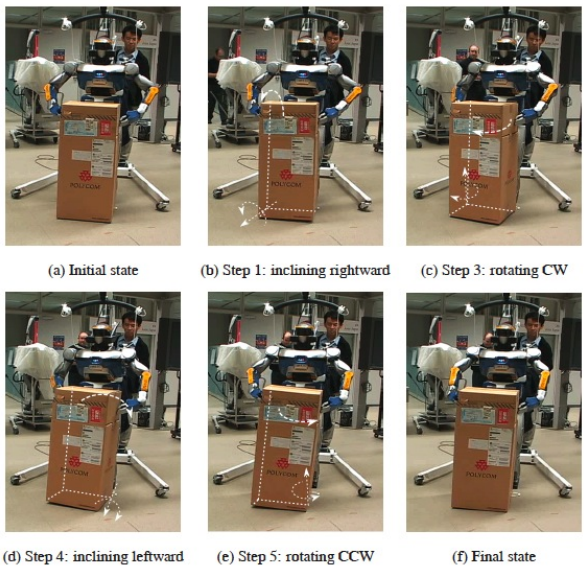


FIGURE 2.25 – de [Yoshida et al. 2006] expérimentation avec la plate-forme HRP-2n°10 au JRL-Japon validant la manipulation par pivotement

.

La manipulation corps complet télé-opérée En 2008, [Stilman et al. 2008] s'intéressent à la manipulation télé-opérée d'objets lourds et encombrants par mouvements corps complet. Le but étant de simplifier la commande à l'utilisateur en l'isolant des complexités de trajectoire dynamique, de placement de pieds pendant la marche et du contrôle de forces. A l'aide d'un simple joystick, l'utilisateur doit pouvoir contrôler le robot HRP-2 pour le faire attraper des objets, les manipuler et les relâcher.

Cela est réalisé en changeant le repère de travail et en le plaçant soit sur les mains, soit sur l'objet, soit sur le robot lui-même selon ce que l'utilisateur veut réaliser avec celui-ci. L'utilisateur peut donc bouger les mains, manipuler l'objet lorsque celui-ci est en main ou déplacer le robot. Pendant les phases de manipulation , les trajectoires des objets, les contraintes cinématiques et contraintes de stabilité dynamique sont automatiquement calculées par l'interface. Nous pouvons voir un exemple de manipulation télé-opérée figure 2.26.

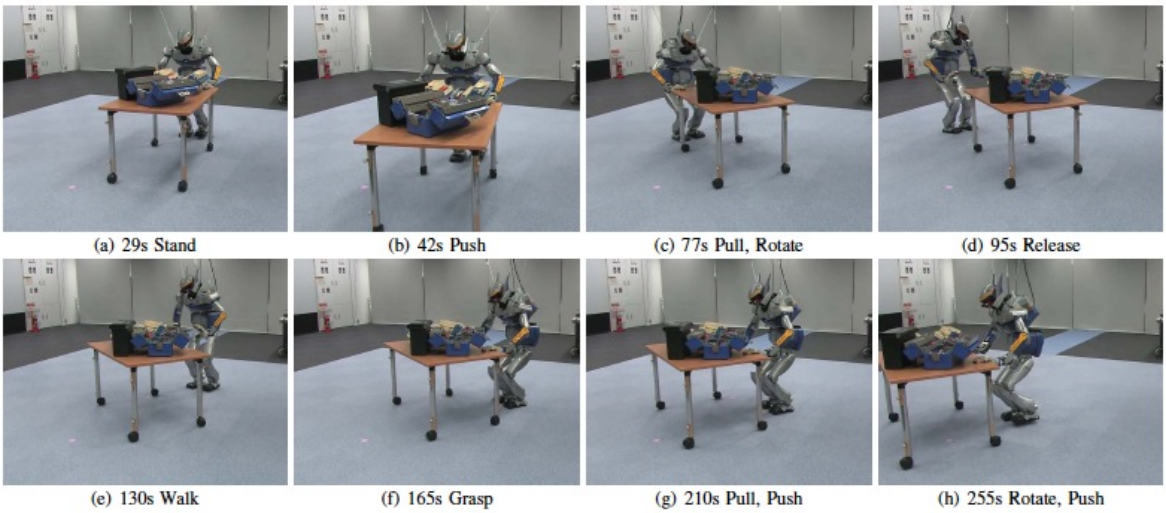


FIGURE 2.26 – de [Stilman et al. 2008] Images d'une seule et même expérimentation de télé-opération. Nous pouvons y voir diverses manipulations d'une table sur roulette recouverte de 55kg de matériel.

Remarque : Dans la plupart des travaux présentés ici, nous avons parlé de manipulations d'objet mais pas de planification. Le robot manipule des objets mais n'a pas de but précis à réaliser.

2.4 La planification de tâches de manipulation

La planification de mouvements en présence d'objets déplaçables a été introduite par [Wilfong 1988]. Cela a été, ensuite, amélioré en particulier dans [Chen and Hwang 1991; Latombe 1991] avant l'introduction des planificateurs de mouvements aléatoires où différents espaces de configurations sont traités en fonction des positions relatives du robots et des objets mobiles.

En 2004, [Maeda et al. 2004] proposent une analyse et un planificateur de manipulation, utilisant les diverses techniques de manipulation sans préhension citées précédemment. Pourtant, ces travaux

ne s'intègrent pas directement dans la manipulation corps complet pour des robots humanoïdes car ils s'intéressent surtout à la manipulation dextre utilisant des effecteurs munis de deux doigts.

En combinant cinématique inverse et une base de données de capture de mouvements Fig. 2.27, [Yamane et al. 2004] proposent un planificateur de manipulation pour personnages virtuels. Les avatars ayant pour but de réaliser la tâche de manipulation en respectant les contraintes cinématiques et dynamiques. Le tout devant être visuellement réaliste Fig. 2.28. Ils accordent notamment une attention particulière au regard durant la manipulation.

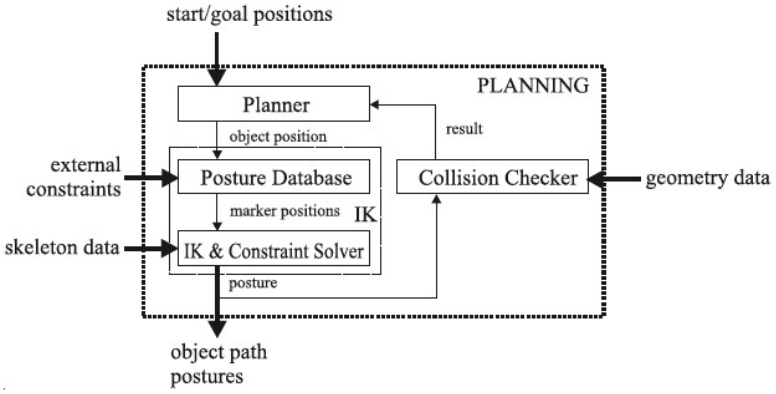


FIGURE 2.27 – de [Yamane et al. 2004] Présentation de la phase de planification .

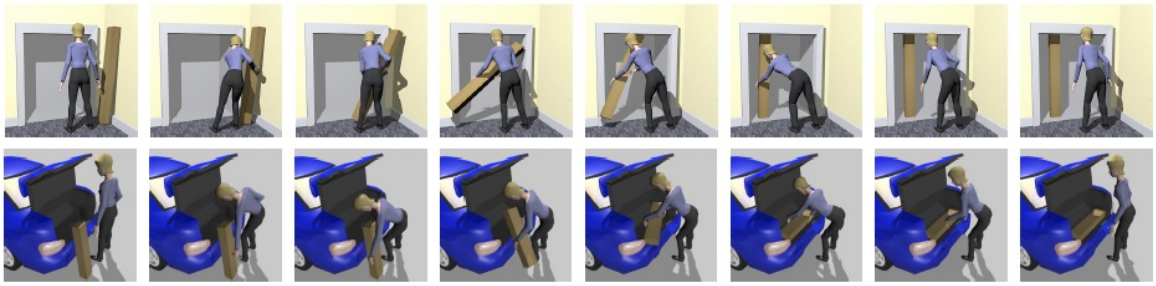


FIGURE 2.28 – de [Yamane et al. 2004] Deux exemples de planification de manipulation réalisés à partir de leur planificateur. Nous noterons que la tâche reste assez complexe car l'objet à manipuler ne peut pas rentrer directement dans l'emplacement désiré.

Dans la recherche en robotique humanoïde [Stilman and Kuffner 2004; Stilman et al. 2006a; Stilman et al. 2006b; Okada et al. 2004] s'intéressent au problème de la navigation en présence d'obstacles mobiles. Les deux méthodes mettent en avant une navigation basée sur une recherche dans un graphe représentant la structure de l'environnement Fig. 2.29. Le robot doit ensuite se frayer un chemin parmi les obstacles.

Nous noterons que dans l'article de [Stilman et al. 2006a] la manipulation se fait dans un milieu totalement connu où tous les objets, y compris le robot, sont traqués par un système de capture de mouvements Fig. 2.30. Les problèmes d'odométrie sont donc réduits. Les points de contact sont tout aussi connus et générés automatiquement. De plus, les objets utilisés sont montés sur des roulettes, ils ne sont donc soumis à aucune contrainte particulière de déplacement.

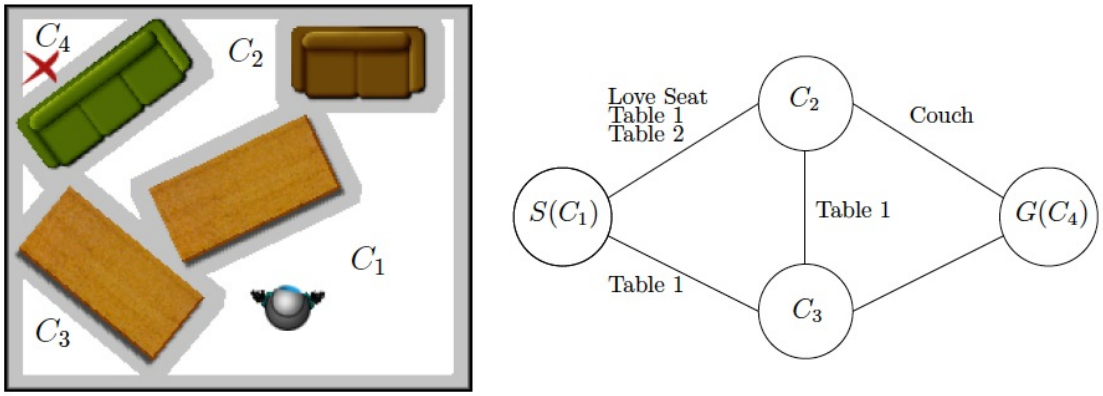


FIGURE 2.29 – de [Stilman and Kuffner 2004] Représentation de l'environnement sous forme de graphe. Les noeuds représentent les différentes composantes connexes de C_{free} et les arêtes les objets mobiles par lesquelles elles sont liées. Cette représentation est une représentation dynamique de l'environnement.

FIGURE 2.30 – de [Stilman et al. 2006a] Séquences d'une planification de manipulation exécutée par HRP-2 (temps = 0s, 36s et 59s). Le robot sélectionne la chaise pour libérer le chemin et atteindre son but

[Esteves 2007; Yoshida et al. 2008] proposent aussi dans leurs travaux sur la décomposition fonctionnelle déjà mentionnée précédemment, un planificateur pour la manipulation. La planification s'effectuant ici à deux niveaux, puisqu'il faut planifier le déplacement pour les jambes et les mouvements pour le haut du corps séparément. Ils proposent dans leur expérimentation de manipuler une perche dans un environnement complexe en évitant des obstacles en trois dimensions, le tout calculé et planifié automatiquement sans intervention humaine. Le mouvement généré est dynamiquement stable voir section 2.2.4. La figure 2.31 montre le résultat de cette planification de manipulation sur le robot HRP-2.

(a) Initial configuration (b) Locomotion begins

(c) Lifting the bar to avoid the box (d) The bar passes above the obstacle

(e) Smoothly lowering the bar (f) Goal configuration

FIGURE 2.31 – de [Esteves 2007] Résultat d'expérimentation de planification de mouvements par décomposition fonctionnelle sur la plate-forme réelle HRP-2 au JRL-Japon

[Alami et al. 1994; Sahbani 2003; Gravot 2004; Simeon et al. 2004] proposent une base théorique de la planification de manipulation incluant des phases ou chemins de transition sans l'objet et des phases de transfert avec celui-ci, permettant la planification d'objet avec re-saisie Fig. 2.32.

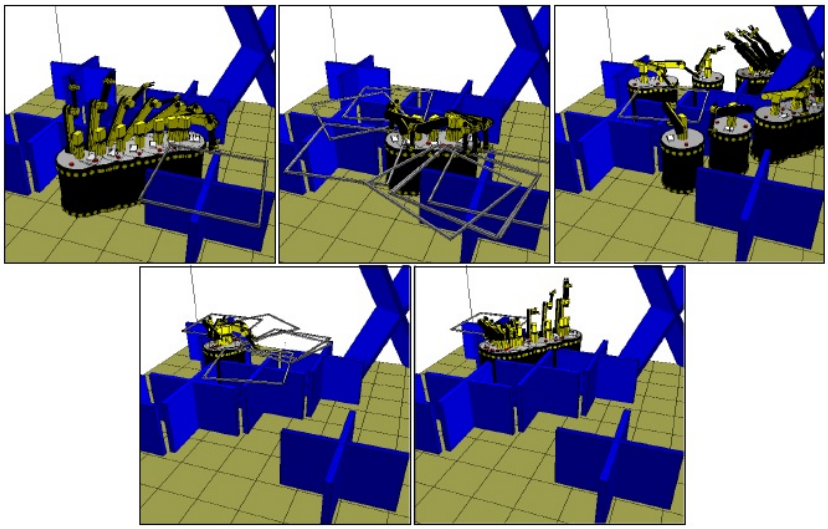


FIGURE 2.32 – de [Simeon et al. 2004] Exemple de planification de manipulation avec relâche et re-saisie de l'objet

2.5 Conclusion

Nous avons présenté dans cette partie quelques unes des techniques de planification de mouvements disponibles dans la littérature. Cet état de l'art n'est, bien sûr, pas exhaustif mais se focalise sur les fonctions et autres applications que nous utiliserons.

Les techniques de planification de mouvements dépendent souvent du type d'environnement dans lequel nous voulons évoluer et du type de système utilisé. Depuis plusieurs années, les méthodes par échantillonnage aléatoire présentent la meilleure alternative lorsque l'on veut optimiser les temps de calculs. Mais chacune des méthodes proposées possède son lot d'avantages et d'inconvénients. Ces méthodes sont rapides mais leurs solutions ne sont pas forcément les meilleures. Elles nécessitent des techniques supplémentaires comme les optimisateurs de chemin.

Nous avons ensuite expliqué les techniques actuelles pour produire des mouvements corps complet pour des systèmes anthropomorphes tels que les robots humanoïdes. Ces techniques seront largement utilisées dans nos travaux mais nous ne contribuerons pas à les améliorer.

Nous avons aussi présenté la manipulation sous différents aspects. Tout d'abord dans la robotique puis plus spécifiquement dans la robotique humanoïde. Nous avons pu voir que plusieurs techniques de manipulation d'objets existaient. Nous avons fait le choix, ici, de continuer les travaux commencés par [Yoshida et al. 2006] qui ont étudié la possibilité de réaliser des mouvements de pivotement par un robot humanoïde pour déplacer des objets encombrants.

Enfin, nous avons fait un état de l'art de la planification de tâches de manipulation. Nous avons pu voir plusieurs approches [Okada et al. 2004; Stilman et al. 2006a; Yoshida et al. 2008] mais aucune ne répond aux objectifs de notre travail. Par contre nous nous inspirerons des travaux de [Simeon et al. 2004] pour étudier une méthode de planification de reprise d'objet à manipuler par pivotement.

Pour finir, quelques remarques importantes sur la suite de notre travail :

- Les environnements sont statiques, ce qui veut dire qu'aucun obstacle mobile ne viendra perturber la scène.
- Les environnements sont complètement connus, aucune incertitude ne sera prise en compte quant au placement des objets dans la scène. Nous reproduirons autant que possible les environnements réels dans lesquels le robot est amené à évoluer.

3

Analyse de commandabilité

3.1 Analyse des propriétés géométriques de la méthode de pivotement

Nous introduisons ici notre méthode de manipulation par pivotement. Nous poserons le contexte et les simplifications que nous autorisent une telle méthode. Puis nous démontrerons, tout d'abord par une simple analyse géométrique, la commandabilité d'un tel système que nous prouverons, de façon plus formelle, localement commandable en temps petit, par la suite.

3.1.1 Définition d'un mouvement de pivot élémentaire

L'*objet* que nous souhaitons manipuler est un parallélépipède pouvant représenter un réfrigérateur, une armoire, etc. Cette hypothèse de forme n'est pas pour autant restrictive. Elle peut s'étendre à tout objet polyédrique pouvant pivoter sur deux de ses sommets. De même nous ne considérerons pas dans ce travail les problèmes induits par la masse ou l'inertie de l'objet. Nous utiliserons des objets suffisamment légers pour être manipulés par le robot. Nous nous intéressons, ici, à l'exploration de l'espace de manipulation.

Le fait est que, lorsqu'un humain souhaite déplacer de tels objets par ses propres moyens, il incline légèrement l'objet de manière à ne garder qu'une seule arête (ou si l'objet est convexe deux points) en contact avec le sol Fig. 3.1. Nous appellerons cette arête *l'arête support*.

FIGURE 3.1 – Exemple : un humain déplace une petite armoire par pivotement

Nous sélectionnons donc une des arêtes de l'objet comme arête support (Fig. 3.2). Tous les calculs effectués par la suite seront établis à partir de cette sélection. Le mouvement sera calculé à partir de celle-ci et se déroulera de la manière suivante :

Le robot commence par incliner l'objet afin de n'obtenir qu'une seule arête en contact entre l'objet et le sol. Puis il bascule une seconde fois celui-ci afin de n'obtenir qu'un seul point de contact avec le sol : le point de pivot. Ensuite, le robot effectue une rotation verticale de l'objet centrée sur ce point. A noter que, par la suite, cette rotation sera accompagnée d'un pas de marche exécuté par le robot, mais ceci ne nous intéresse pas pour l'instant car nous étudions le déplacement de l'objet lui-même. Puis le robot repose l'objet horizontalement le long de l'arrête afin de changer de point de pivot, et répéter l'opération de manière identique sur le point suivant. Ce mouvement est aussi appelé *mouvement élémentaire* de manipulation par pivotement. La figure Fig. 3.2 illustre ce mouvement.

Dans ce mouvement de pivotement les angles utilisés pour incliner l'objet peuvent être réduits autant que nécessaire. Nous ne nous intéresserons pas à des problèmes de franchissement d'obstacle pendant un mouvement de pivotement. Par conséquent, nous pouvons modéliser le problème du déplacement d'un objet pivotant autour de deux points de contact en un problème de déplacement d'un segment de droite pivotant sur ses extrémités. Cette modélisation ne change en rien la problématique générale.

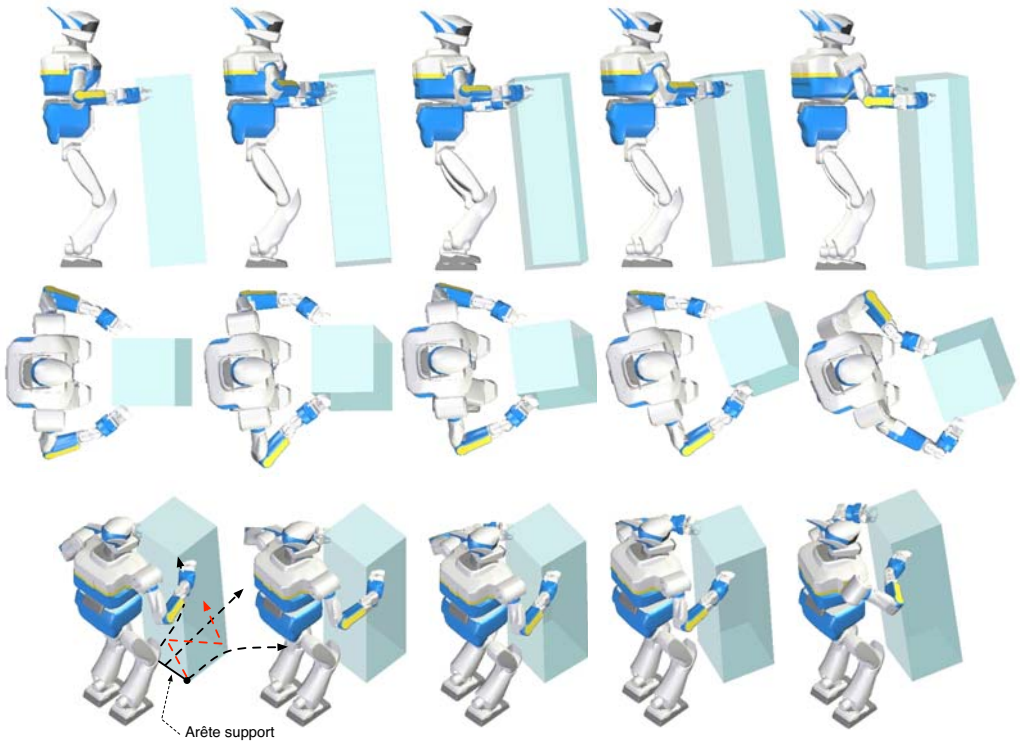


FIGURE 3.2 – Décomposition d'un mouvement de pivot élémentaire sous différentes vues.

3.1.2 Le problème du pivotement

Soit un segment AB de longueur l pouvant pivoter sur ses extrémités ou points de pivot A et B et de centre O. On appelle une configuration q de notre système, la position de son centre O : x, y et l'orientation du segment θ : $q(x,y,\theta)$ Fig.3.3 .

Le problème que nous abordons est le suivant. Il s'agit d'amener le segment d'une configuration initiale quelconque à une configuration finale quelconque en utilisant les deux seules opérations élémentaires autorisées : une rotation sur l'une ou l'autre des extrémités. Est ce possible ? Quelle stratégie utiliser ? Ces deux questions ont trait à la notion de commandabilité : le système considéré est de dimension trois et on ne dispose que de deux "commandes" pour se déplacer.

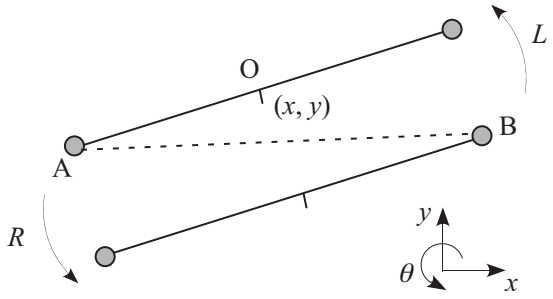


FIGURE 3.3 – Le problème du pivotement : déplacer le segment avec A ou B

Nous introduisons ici trois stratégies élémentaires montrant que l'on peut diriger le système dans les trois directions.

3.1.2.1 Définitions de trois déplacements élémentaires dans un plan

La translation horizontale Supposons que nous voulions nous déplacer horizontalement vers la gauche dans la direction du segment. Cette translation se décompose ainsi en quatres étapes (*le principe resterait le même pour une translation horizontale à droite*) :

- rotation (sens +) de ε autour de $A : B \rightarrow B'$
- rotation (-) de ε autour de $B' : A \rightarrow A'$
- rotation (-) de ε autour de $A' : B' \rightarrow B''$
- rotation (+) de ε autour de $B'' : A' \rightarrow A''$

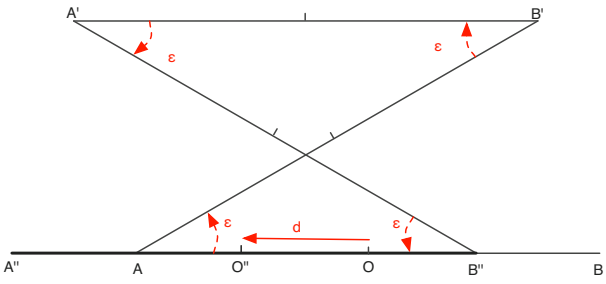


FIGURE 3.4 – Translation horizontale du système

Ceci engendre, comme nous pouvons le voir sur la figure 3.4, un décalage du centre de la barre sur la gauche d'une longueur d : $\boxed{d = 2l(1-\cos\varepsilon)}$

La translation verticale Supposons maintenant que nous voulions nous déplacer verticalement avec ce système. La translation se décompose ainsi en trois étapes (*le principe resterait le même pour la translation verticale vers l'arrière*) :

- rotation (+) de ε autour de $A : B \rightarrow B'$

– rotation (-) de ε autour de $B' : A \rightarrow A'$
– Ces deux premières étapes entraînent un décalage du centre du système vers la gauche de $d = l(1-\cos\varepsilon)$. Il faut donc ajouter à cela une translation horizontale, de longueur d, vers la droite, avec un angle de manoeuvre $\varepsilon_2 = \cos^{-1}(\frac{d}{l}+1)$

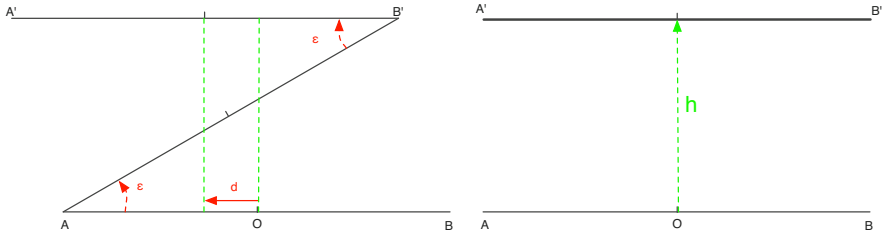


FIGURE 3.5 – Translation verticale du système : avant et après la 3ième étape

Ceci engendre, comme nous pouvons le voir sur la figure 3.5, un décalage du centre de la barre vers l'avant d'une longueur h : $\boxed{h = l\sin\varepsilon}$

La rotation ou changement d'orientation de la barre Finalement, supposons que nous voulions changer l'orientation du système. Cette rotation se décomposerait, dans le sens trigonométrique, en 3 étapes (*le principe resterait le même pour la rotation en sens inverse*) Fig. 3.6 :

– rotation (+) de $\varepsilon_1 = \theta_{final}$ autour de $A : B \rightarrow B'$. Il faut ensuite replacer le centre de la barre à ses coordonnées x et y initiales.
– une translation verticale vers l'arrière de longueur h : $\boxed{h = \frac{l}{2}\sin\varepsilon_1}$ que l'on peut réaliser avec un angle de manoeuvre $\varepsilon_2 = \sin^{-1}(\frac{h}{l})$.
– enfin une translation horizontale vers la gauche de longueur d : $\boxed{d = \frac{l}{2}(1-\cos\varepsilon_1)}$ que l'on peut réaliser avec un angle de manoeuvre $\varepsilon_3 = \cos^{-1}(\frac{d}{\frac{l}{2}}+1)$.

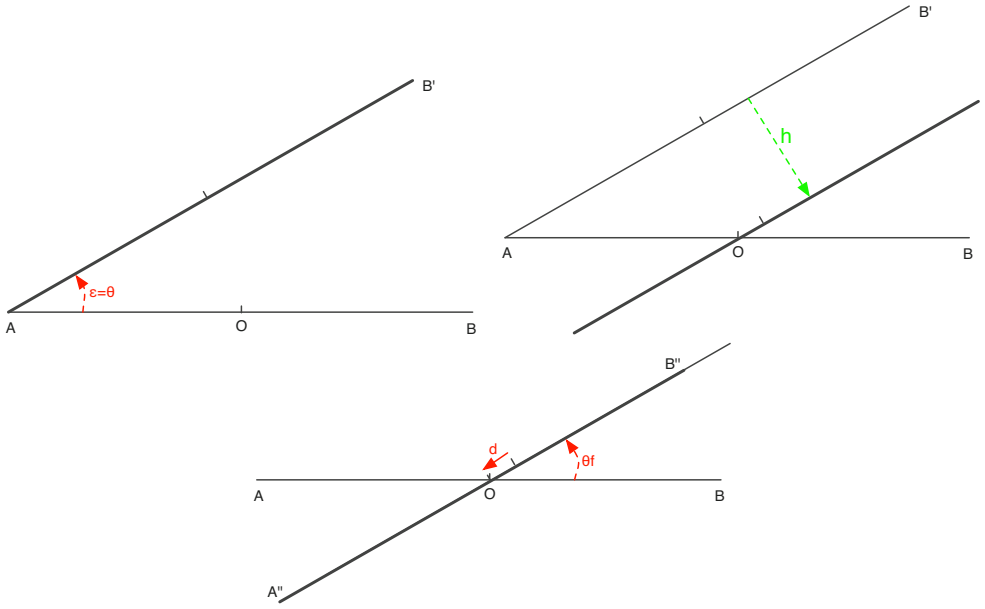


FIGURE 3.6 – rotation du système en 3 étapes

Conclusion Il est donc possible de déplacer un tel système en utilisant uniquement les deux rotations autorisées. Il reste que le nombre de commandes élémentaires est respectivement de 4, 6 et 11 pour les trois mouvements.

3.2 Pivotement et commandabilité locale en temps-petit : une preuve directe.

Un système est dit commandable, si celui-ci peut atteindre une configuration finale q_{finale} arbitraire à partir de n'importe quelle configuration initiale q_{init} [Sussmann 1982]. Il est dit *localement commandable en temps petit* si l'ensemble des configurations admissibles $Reach_q(T)$, qui peuvent être atteintes depuis une configuration q avant un temps $T(>0)$ donné, contient un voisinage de q. Cette propriété doit être vraie pour n'importe quelle configuration q et n'importe quel temps T. Cela veut dire que nous devons montrer l'existence d'un voisinage ouvert η de q dans lequel le système doit pouvoir se mouvoir sans sortir d'un plus grand voisinage V imposé, comme montré sur la partie gauche de la la Fig. 3.7

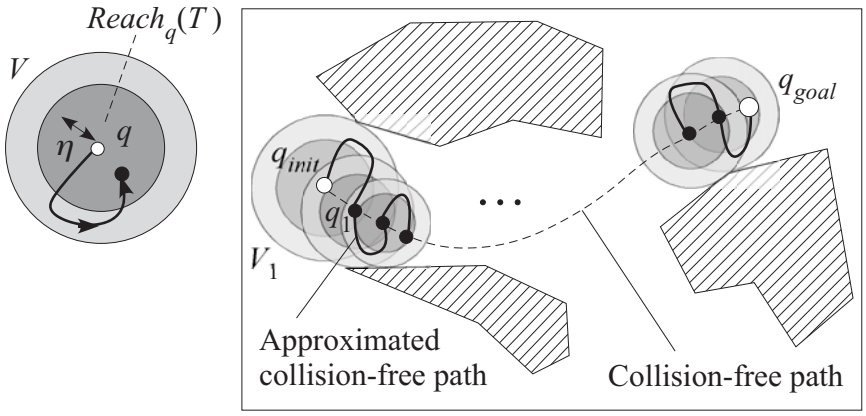


FIGURE 3.7 – un système est localement commandable en temps petit si l'on peut définir un voisinage ouvert de la configuration initiale $q_i(x,y,\theta)$ dans lequel on peut atteindre n'importe quelle configuration finale $q_f(x+\delta x, y+\delta y, \theta+\delta\theta)$ sans sortir d'un voisinage V en $T>0$.

La commandabilité locale en temps petit est une propriété très intéressante pour les algorithmes de planification de mouvements. Cette propriété permet d'approximer par des séquences de mouvements admissibles et sans collisions ou méthode locale de planification, une trajectoire initiale, elle aussi libre de collision mais pas forcément exécutable par le système (Fig. 3.7 droite). Ceci permet donc de réduire les contraintes sur la recherche de trajectoire initiale et d'avoir un temps de réponse beaucoup plus rapide de la part du moteur de planification de mouvements.

La commandabilité locale en temps petit peut-être prouvée en utilisant l'algèbre de Lie des champs de vecteur de la manière suivante.

Considérons notre système (Fig. 3.3) de longueur $2l$ avec $l = AO = OB$. La configuration du segment est exprimée par les coordonnées de son centre $O(x,y)$ et son orientation θ. A chacune des étapes d'une séquence de pivotement, seulement deux rotations élémentaires sont autorisées pour le segment : une rotation sur le point de pivot A ou une sur le point de pivot B.

Considérons tout d'abord la rotation autour du point A. Le champ de vecteurs $\mathscr{L}$ exprimé pour décrire ce mouvement est dérivé en calculant la vitesse tangente du point O à la configuration $q(x,y,\theta)$ comme :

$$\mathscr{L} = \begin{pmatrix} -l\sin\theta \\ l\cos\theta \\ 1 \end{pmatrix}. \tag{3.1}$$

Les mêmes calculs peuvent être appliqués à partir du point de rotation B. Le champ de vecteurs $\mathscr{R}$ est donc :

$$\mathscr{R} = \begin{pmatrix} l\sin\theta \\ -l\cos\theta \\ 1 \end{pmatrix}. \tag{3.2}$$

La loi de commande du système par pivotement peut donc s'écrire :

$$\dot{q} = \mathscr{L} u_1 + \mathscr{R} u_2 \tag{3.3}$$

Où u_1, u_2 sont les angles appliqués au système pour réaliser un mouvement de rotation. A noter, aussi, que les commande u_1 et u_2 ne peuvent pas être appliquées simultanément. Effectivement, nous ne pouvons pivoter que sur un seul point à la fois.

Pour prouver que notre système est localement commandable en temps petit, on applique la condition *Lie Algebra Rank Condition - LARC* [Sussmann 1982] en calculant le crochet de Lie correspondant $[\mathscr{L},\mathscr{R}]$. Rappelons que la k_{ieme} composante $[f,g]_k$ du crochet de Lie de champs de vecteurs f et g est :

$$[f,g]_k = \sum_{i=1}^{n} (g_i \frac{\partial f_k}{\partial q_i} - f_i \frac{\partial g_k}{\partial q_i}) \tag{3.4}$$

Où q_i est la i_{ieme} composante de configuration du système.
En appliquant cette formule à notre système de pivotement, nous obtenons

$$[\mathscr{L},\mathscr{R}] = \begin{pmatrix} -2l\cos\theta \\ -2l\sin\theta \\ 0 \end{pmatrix}. \tag{3.5}$$

On vérifie que $\mathscr{L}$, $\mathscr{R}$ et $[\mathscr{L},\mathscr{R}]$ sont linéairement indépendants. En effet, le déterminant de la matrice A (équation 3.6) est non nul.

$$\text{soit } A = \begin{pmatrix} -l\sin\theta & l\sin\theta & -2l\cos\theta \\ l\cos\theta & -l\cos\theta & -2l\sin\theta \\ 1 & 1 & 0 \end{pmatrix}, \quad \text{pour} \quad l=1 \quad \text{et} \quad \theta = 0 \quad det(A) = -4 \tag{3.6}$$

Les trois champs de vecteurs $\mathscr{L}$, $\mathscr{R}$ et $[\mathscr{L},\mathscr{R}]$ obtenus sont une base dans l'espace tangent à l'espace de configuration. La condition de *LARC* est donc tenue et nous pouvons conclure que le système est localement commandable en temps petit. L'espace d'accessibilité d'une configuration initiale q_i quelconque contiendra toujours un voisinage ouvert et accessible peu importe la taille de l'environnement.

3.3 Méthode locale de planification par pivotement en trois coups

Pour intégrer ce genre de système dans nos algorithmes de planification et de recherche de solution, une méthode locale de planification doit être définie. Cette méthode est une fonction permettant de calculer très rapidement un chemin admissible entre deux configurations du système. Même si le système est prouvé localement commandable en temps petit, ceci n'est qu'une preuve d'existence, cette propriété

ne donne pas la méthode à employer pour déplacer le système. Créer cette méthode locale de planification est donc un nouveau problème [Laumond 1998].

Nous introduisons ici *une* méthode locale de planification par pivotement en 3 coups, mais nous aurions pu en construire plusieurs autres. Nous aurions pu prendre une fonction en 2 coups mais son espace d'accessibilité aurait été trop réduit. Nous aurions pu prendre une fonction en 4 coups mais le gain d'espace d'accessibilité par rapport à la complexité de la solution n'était pas suffisamment intéressant. Nous avons choisi de privilégier les critères de complexité, de précision et d'accessibilité, le but étant de positionner le plus précisément le système à une configuration finale en réalisant le moins de mouvements de pivot possible.

Considérons la séquence de trois rotations illustrées Fig. 3.8. La configuration initiale est supposée $(0,0,0)$. Les angles pour les 3 rotations sont respectivement notés α_1, α_2 et α_3 en alternant à chaque fois les points de pivot. Comme ceci, nous pouvons calculer la configuration finale q_f telle que :

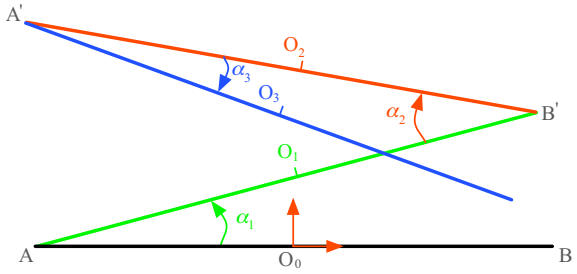


FIGURE 3.8 – Séquence d'une opération de trois pivotements pour atteindre une configuration voisine.

$$
\begin{aligned}
q_f &= f(\alpha_1,\alpha_2,\alpha_3) \qquad (3.7)\\
&= \begin{cases}
x = -l + 2l\cos\alpha_1 - 2l\cos(\alpha_1+\alpha_2) \\
\qquad +l\cos(\alpha_1+\alpha_2+\alpha_3), \\
y = 2l\sin\alpha_1 - 2l\sin(\alpha_1+\alpha_2) \\
\qquad +l\sin(\alpha_1+\alpha_2+\alpha_3), \\
\theta = \alpha_1+\alpha_2+\alpha_3.
\end{cases}
\end{aligned}
$$

En déterminant que chaque α_i soit compris entre $[-\pi/4,\pi/4]$, nous pouvons calculer l'inverse f^{-1} de la fonction f telle que :

$$
f^{-1}(x,y,\theta) = \begin{cases}
\alpha_2 = \pm\arccos(\dfrac{2-(X^2+Y^2)}{2}), \\
\alpha_1 = \phi \pm \arccos(\dfrac{X}{\rho}), \\
\alpha_3 = \theta - \alpha_1 - \alpha_2.
\end{cases}
$$

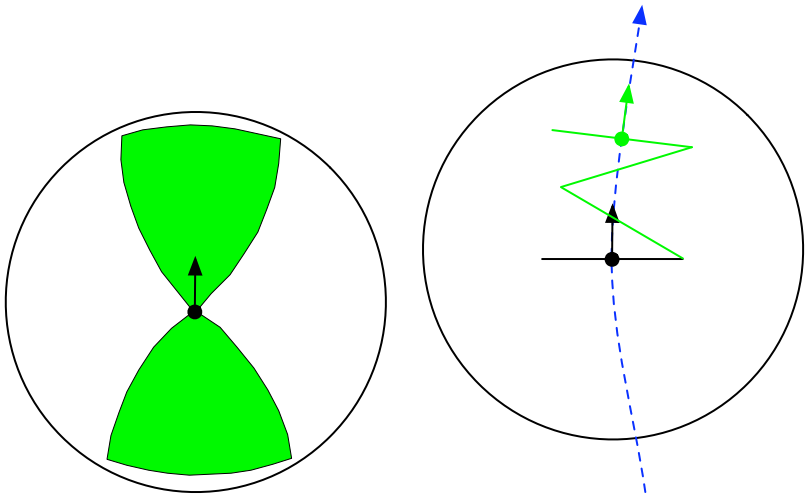
Où

$$
\begin{aligned}
X &= \frac{x + l(1-\cos\theta)}{2l}, \\
Y &= \frac{y - l\sin\theta}{2l}, \\
\rho &= \sqrt{(1-\cos\alpha_2)^2 + \sin^2\alpha_2}, \\
\phi &= \arctan\frac{\sin\alpha_2}{1-\cos\alpha_2}.
\end{aligned}
$$

Nous supposons que les calculs ci-dessus sont applicables si les arguments de la fonction *arccos* restent compris entre $[-1,1]$. Cette méthode pourrait être assimilée à trouver la cinématique inverse d'un bras manipulateur plan muni de trois articulations en rotation.

Cette méthode locale de planification ne rend pas compte de la commandabilité locale en temps petit. En effet, la configuration $(\varepsilon,0,0)$ n'est pas atteignable même avec des petits α_i. Nous l'introduisons néanmoins pour les raisons suivantes :

- Elle est faite de trois rotations élémentaires (à comparer aux 4, 6, et 11 rotations de la section 3.1.2).
- Si son espace d'accessibilité ne contient pas un ouvert de l'origine Fig. 3.9a, il permet néanmoins de progresser le long d'un chemin tangent à l'origine Fig. 3.9b.



(a) (b)

FIGURE 3.9 – (a) Espace d'accessibilité de la méthode en 3 coups pour une orientation donnée . (b) Cette méthode permet d'approximer un chemin tangent à l'orientation de la configuration d'origine.

3.4 Conclusion

Cette méthode locale de planification en trois pivotements peut être utilisée pour approximer la plupart des chemins. Malheureusement, lorsque l'on souhaite progresser rapidement le long d'un chemin, c'est-à-dire réaliser les plus grands mouvements de pivotement élémentaires possibles avec le robot, cette méthode ne produit pas nécessairement les solutions les plus optimisées. Nous l'utiliserons néanmoins pour des cas particuliers, que l'on appellera, dans le chapitre suivant, les *mouvements d'ajustement* entre configurations, car ces mouvements demandent une solution précise, ce que produit cette méthode.

Afin de suivre au mieux un chemin initial, nous allons utiliser nos connaissances sur le type de chemin produit et optimiser efficacement les pivotements de long de celui-ci. Nous allons donc planifier les mouvements d'un objet rigide soumis à des contraintes connues.

4

De la stratégie de pivotement à la manipulation

4.1 De la planification de chemin à la séquence de pivotement

Dans cette section, nous présentons une méthode de planification par pivotement décomposée en deux étapes. Nous verrons dans un premier temps comment trouver une trajectoire libre de collisions dans un environnement puis dans un second temps, nous verrons comment transformer de manière optimisée cette trajectoire en une séquence de pivotement.

Nous ne parlerons pas ici de manipulation mais de planification de chemin. Nous nous intéressons au déplacement du système robot/objet dans son ensemble. Cet ensemble robot/objet va être considéré comme un seul corps rigide. Deux choses sont donc à noter à ce stade :

- La première est qu'il faut garder en tête qu'au final, le robot devra se déplacer et manipuler l'objet en exécutant des mouvements visuellement réalistes.
- La deuxième est que nous n'avons pas besoin de prendre en compte les propriétés anthropomorphes du robot et donc ses 40 degrés de liberté.

4.1.1 Planification de mouvement pour un corps rigide soumis à des contraintes non-holonomes

4.1.1.1 La boîte englobante

N'ayant pas besoin de tous les degrés de liberté du système, la première chose que nous faisons est de le simplifier. Nous ramenons celui-ci à une boîte rigide dont le volume représente la boîte englobante du système. Nous avons rajouter à cette boîte englobante, l'espace de manipulation nécessaire au robot pour manipuler correctement l'objet, plus une marge de sécurité. Cela veut dire, que tout chemin solution

trouvé par nos algorithmes de planification de mouvement représentera un volume en trois dimensions dans l'espace de travail du robot dans lequel celui-ci peut manipuler l'objet sans entrer en collision avec son environnement. Pour notre travail, la taille de la boîte englobante est calculée proportionnellement à la taille de l'objet à manipuler pour que dans n'importe quelle situation les coudes du robot en train de manipuler la boîte ne dépassent pas de la boîte. La figure 4.1 illustre plusieurs vues de la boîte englobante incluant différentes configurations du robot et de la boîte.

La première étape de l'algorithme est donc dédiée à la recherche d'un chemin sans collision dans un environnement donné pour la boîte englobante. Cet objet rigide est aussi soumis à un certain nombre de contraintes.

- La racine de la boîte englobante, ou point utilisé pour générer la trajectoire, représente le centre de l'arrête support (voir figure 4.1).
- Sa configuration est représentée sous la même forme que l'objet à manipuler $q_{OBB}(x, y, \theta)$
- La boîte est soumise à des contraintes non holonomes. C'est-à-dire, un peu comme une voiture, elle peut glisser vers l'avant, l'arrière et tourner avec un rayon de braquage maximum donné. Mais elle ne peut pas directement se déplacer latéralement.

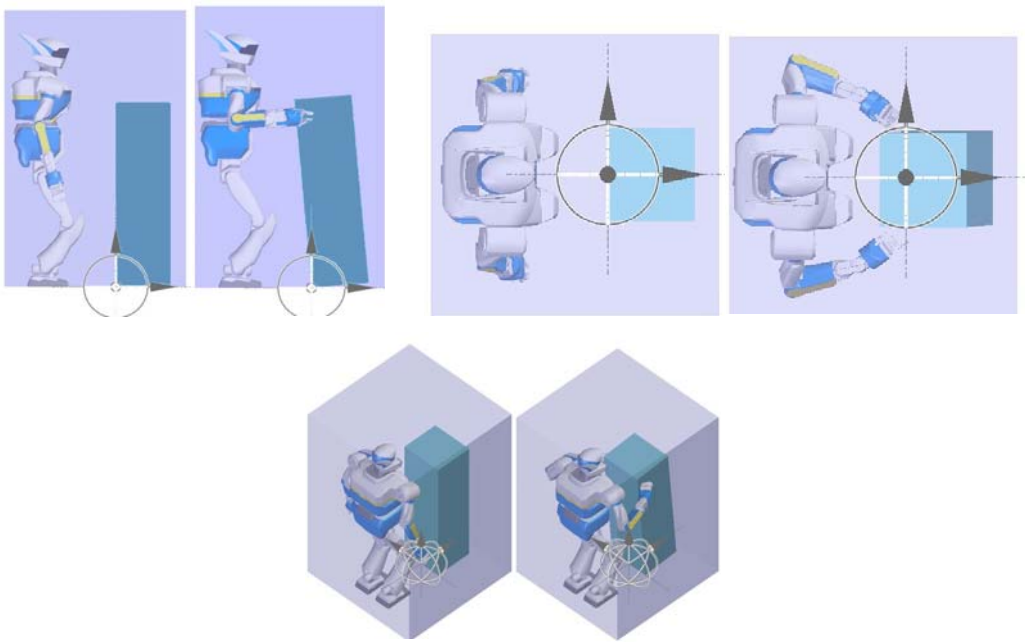


FIGURE 4.1 – Plusieurs vues de la boîte englobante. Le repère représente la racine de la boîte et le centre de l'arête support.

4.1.1.2 Contraintes non holonomes

Pourquoi soumettre la boîte à des contraintes non holonomes (voir section 2.1.3) ? Si nous analysons les trajectoires humaines de marche, pour essayer d'en construire des modèles, nous nous

apercevons que dans le cas de la locomotion naturelle (c'est-à-dire à l'exception des pas chassés), la locomotion humaine consiste à mettre un pied devant l'autre et à recommencer. La direction corporelle est tangente à la trajectoire suivie. Ce phénomène a été étudier dans [Arechavaleta et al. 2008]. Cette contrainte, de nature différentielle, est dite non holonome. Elle est bien connue en robotique puisque c'est à elle que sont soumis la plupart des robots mobiles.

Comment planifier un premier chemin qui rend compte de cette contrainte ? Nous reprenons ici un schéma de planification probabiliste basé sur le calcul des courbes de Reeds et Shepp [Reeds and Shepp 1990] Fig.4.2.

4.1.1.3 Méthode locale de planification : les courbes de Reeds et Shepp

Les courbes de Reeds et Shepp sont les chemins de longueur minimale pour une particule évoluant dans le plan et soumise à des contraintes de courbure minimale. Elles sont des courbes composées exclusivement de segments de droite et d'arcs de cercle à rayon constant. La figure 4.2 montre quelques représentations de courbe de Reeds et Shepp. Elles sont composées d'une combinaison d'au plus cinq motifs : quatre arcs de cercles et un segment de droite, auxquels il faut rajouter le sens avant ou arrière. En absence d'obstacle il n'existe entre deux configurations qu'au plus 48 possibilités dont au moins une est la solution. Nous utiliserons la connaissance de ces motifs, à la section 4.1.2, afin d'optimiser le déplacement par pivotement le long d'une trajectoire composée de courbes de Reeds et Shepp.

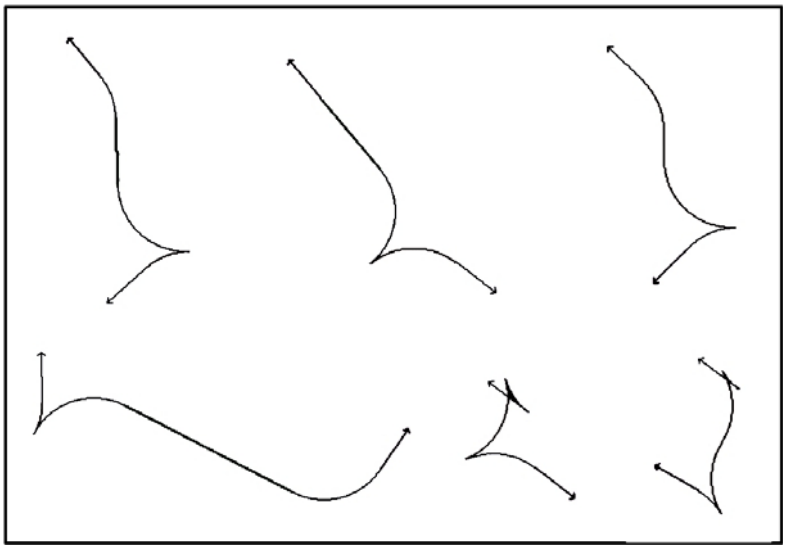


FIGURE 4.2 – Exemple de courbes de Reeds et Shepp

4.1.1.4 La planification de mouvement

Maintenant que nous avons simplifié notre système et défini une méthode locale de planification avec les courbes de Reeds et Shepp, il ne reste plus qu'à lancer la planification. Pour cela nous utilisons

les techniques standard de planification [Choset et al. 2006; LaValle 2006] déjà utilisées dans notre laboratoire, intégrées au logiciel de planification tel que KineoWorks et résumées ici.

Pour calculer la trajectoire sans collision entre les deux configurations initiale et finale, nous utilisons les méthodes de planification de mouvement par échantillonnage aléatoire qui consistent à construire un graphe dont les noeuds sont des configurations tirées aléatoirement dans l'espace de configuration C. Ces noeuds ou configurations sont ajoutés au graphe si et seulement si ils ne sont pas en collision avec l'environnement dans l'espace de travail W. Ici, pour conduire la recherche aléatoire nous utilisons les méthodes de cartes probabilistes *PRM* [Choset et al. 2006; LaValle 2006] (voir chapitre 2.1.2).

Ensuite, deux nœuds du graphe sont liés si et seulement si la courbe de Reeds et Shepp entre les deux configurations correspondantes est elle aussi sans collisions. Le graphe est donc construit progressivement par des tirs de configurations aléatoires. Cette construction du graphe est appelée la phase d'apprentissage.

Une fois le graphe construit, le planificateur y recherche un chemin. Du fait de l'échantillonnage aléatoire ce chemin inclut des détours inutiles. Pour que la manipulation soit plus efficace, les composantes redondantes sont supprimées à l'aide d'un optimisateur de chemin. Nous utilisons la méthode des raccourcis adaptatifs proposée par [Laumond et al. 1994] qui consiste à choisir plusieurs fois de suite (ce nombre est déterminé par l'utilisateur) une paires de noeuds aléatoirement sur le chemin solution et de la même manière, essayer de les relier entre eux directement par la méthode locale de planification. Si le raccourci est possible, tous les noeuds situés entre les deux choisis aléatoirement sont retirés du chemin solution.

4.1.2 Genération de séquence de pivotement

Dans cette partie, nous expliquons comment convertir un chemin solution, sans collisions en une séquence de pivotement. Celle-ci étant composée, d'une séquence de courbes de Reeds et Shepp, la génération de séquences de pivotement est alors créée à partir de deux opérateurs élémentaires optimisés pour les caractéristiques de ces courbes : les pivots le long d'un segment de droite et les pivots le long d'un arc de cercle. On rajoutera aussi un autre opérateur pour ajuster le mouvement dans sa phase terminale.

Une séquence de pivotement est une succession de mouvements de pivots élémentaires. Nous rappelons donc la définition d'un mouvement de pivots élémentaires comme illustrée sur la figure 3.2. Le robot exécute ce mouvement de la manière suivante :

- Le robot incline l'objet afin de n'obtenir qu'un seul point de contact entre celui-ci et le sol.
- Il effectue ensuite une rotation verticale de l'objet, centrée sur ce point. Pour garder une distance de manipulation convenable entre le robot et l'objet, le robot effectuera, en même temps, un pas en avant ou en arrière.
- Puis il repose l'objet horizontalement le long de l'arrête support.

Le robot peut ainsi répéter alternativement l'opération sur une extrémité puis l'autre de l' arrête support.

Il faudra cependant rajouter un mouvement au début et la fin de la séquence de pivotement pour initialiser et finaliser la manipulation. Au départ, l'objet étant posé à plat sur le sol, le robot doit incliner celui-ci afin de ne garder que l'arête support en contact avec le sol. Le robot peut ensuite enchaîner autant de mouvement de pivots élémentaires que l'on souhaite. A la fin le robot repose simplement l'objet au sol. La figure 4.3 illustre ce mouvement.

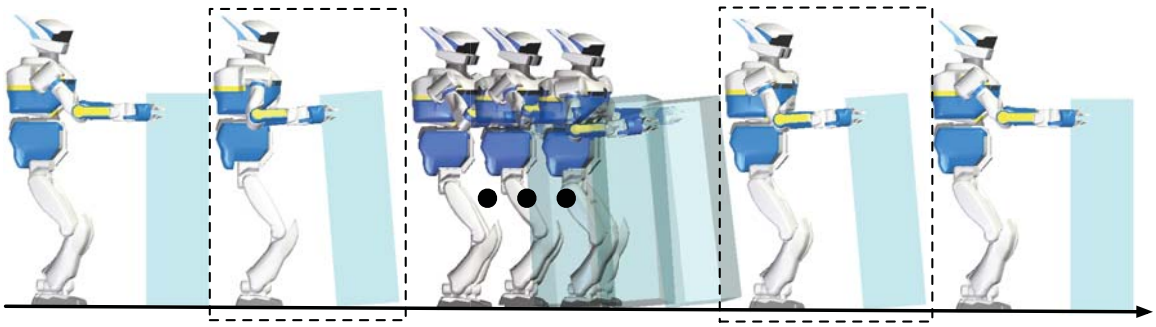


FIGURE 4.3 – Mouvement permettant d'initialiser et de finaliser une séquence de pivotement

4.1.2.1 Motifs optimisés

Le but de notre schéma d'approximation est de progresser le plus rapidement possible le long du chemin planifié. Sachant que l'espace de manipulation est limité par l'espace d'accessibilité des bras du robot. Nous introduisons un angle β de sorte que le robot soit en mesure d'effectuer un mouvement de pivotements élémentaires total maximum d'angle 2β.

Séquence le long d'une portion de ligne droite Soit L la longueur de la portion de courbe de Reeds et Shepp droite à suivre et $2l$ la longueur de l' arrête support sélectionnée. Après l'initialisation du processus par un pivot d'angle β, nous appliquons n fois le schéma de pivotement élémentaire d'angle 2β, n est défini comme le plus grand entier vérifiant $L > nl\sin(\beta)$. Le calcul de la séquence de pivotement le long d'un segment est illustré Fig. 4.4.

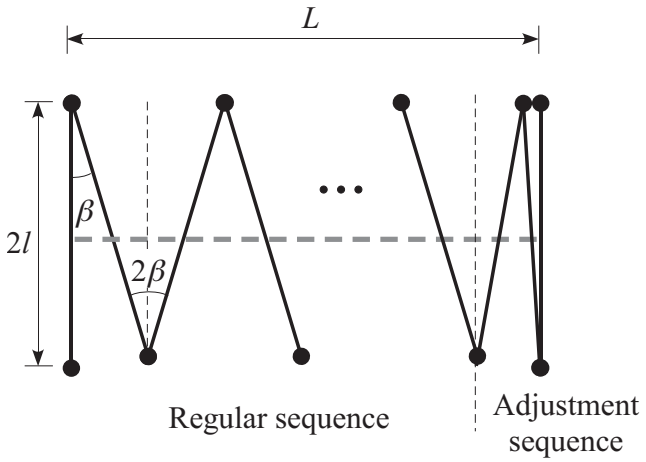


FIGURE 4.4 – Transformation d'un segment de droite du chemin solution en une séquence de pivotement.

Séquence le long d'une portion d'arc Le même principe s'applique si l'on veut suivre une portion d'arc de cercle. La figure 4.5 illustre le calcul de cette séquence. On note R et θ le rayon et l'angle décrit par l'arc. Nous appliquons ensuite un schéma de mouvement symétrique tel que l'objet soit perpendiculaire à l'arc après deux rotations β à gauche et une rotation de -2γ à droite (et inversement si l'arc tourne vers la droite). Les angles α et γ sont calculés en fonction de l, R et β tels que :

$$\begin{aligned} \alpha &= \arctan\left(\frac{2l\sin\beta}{R-l+2l\cos\beta}\right), \\ \gamma &= \beta-\alpha. \end{aligned}$$

Nous répétons m fois ce motif optimisé tant que $\theta > m\alpha$.

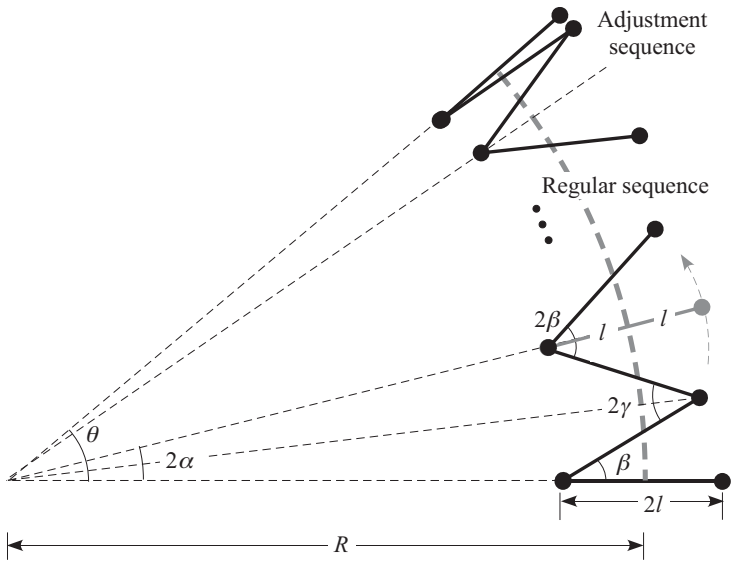


FIGURE 4.5 – Transformation d'un arc de cercle du chemin solution en une séquence de pivotement.

4.1.2.2 Les ajustements

A la fin de ces séquences de pivotement, l'objet n'est jamais exactement à la configuration désirée. Cela demande donc un ajustement. Cet ajustement est réalisé à partir de la méthode de déplacement en trois coups étudiée au chapitre 3.3. Si le déplacement en trois coups ne suffit pas, nous pouvons réitérer plusieurs fois l'opération jusqu'à atteindre la configuration finale désirée.

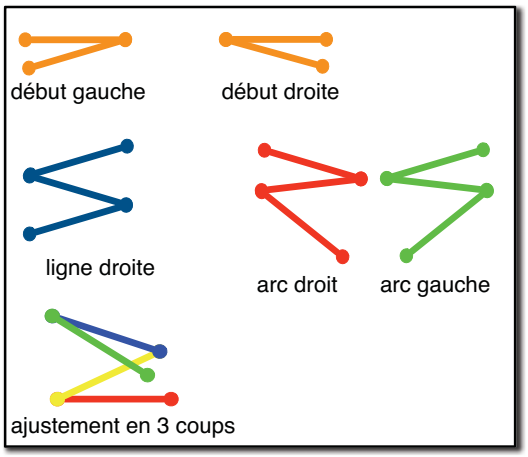
De la même manière, nous essayons de replacer l'objet directement en position optimisée entre deux portions de courbe Reeds et Shepp, afin de réutiliser tout de suite les motifs optimisés calculés précédemment.

Les ajustements sont recalculés à chaque fois, on ne peut pas les reproduire car ils sont tous différents.

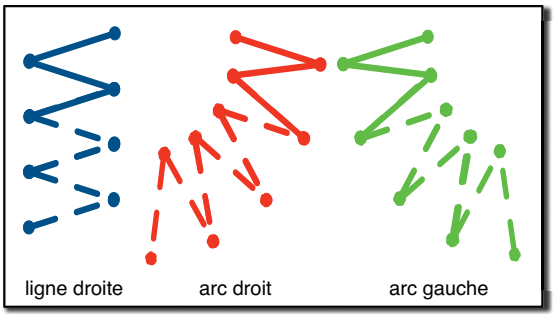
En conclusion, nous pouvons suivre n'importe quelle courbe de Reed et Shepp en combinant tous ces motifs optimisés et ajustements. La figure 4.6a montre l'ensemble des motifs utilisés, la 4.6b présente les formes que l'on peut réaliser en répétant ces motifs et la 4.6c illustre un exemple de courbe de Reeds et Shepp transformée en une séquence de pivotement. Nous créons ainsi une trajectoire d'animation de l'objet par pivotement le long du chemin solution. Si on anime cette trajectoire, l'objet part de sa configuration initiale jusqu'à sa configuration finale. Mais le robot ne manipule toujours pas l'objet !

4.2 Calcul de la trajectoire des mains et du positionnement des pieds

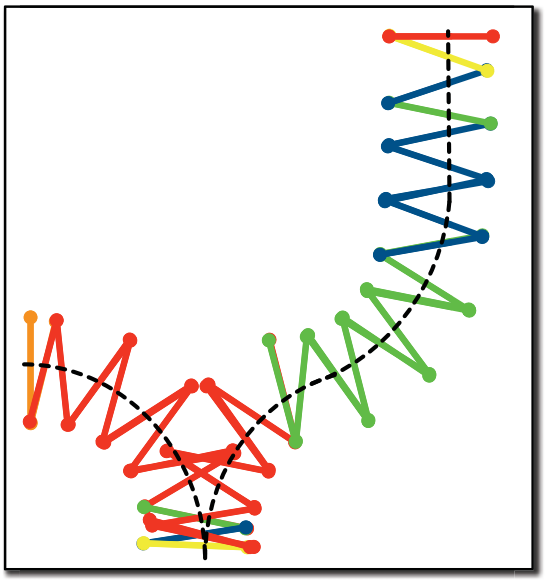
A partir de la séquence de pivotement nous allons maintenant pouvoir calculer le mouvement des mains du robot pour que celui-ci puisse manipuler l'objet conformément à l'animation mais aussi définir le positionnement de ses pieds afin qu'il puisse suivre ce dernier.



(a) Ensemble des motifs utilisés



(b) Motifs répétés



(c) Une trajectoire Reeds et Shepp transformée en séquence de pivotement

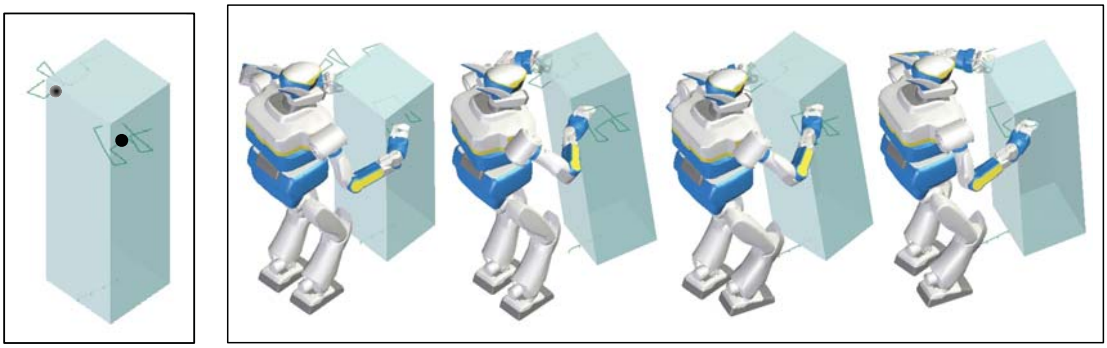
FIGURE 4.6 – Motifs optimisés et trajectoire transformée en séquence de pivotement

4.2.1 Calcul des trajectoires des mains

Le mouvement des mains du robot est directement extrait de la séquence de pivotement. Il correspond à la trajectoire des points de contact entre l'objet et le robot.

Ces points de contacts sont définis par l'utilisateur au moment de la construction de la représentation virtuelle de l'objet. La figure 4.7a illustre ces points. Ils doivent être choisis avec précaution car ils peuvent agir sur la stabilité du robot et le réalisme des mouvements exécutés. Malheureusement, nous n'avons pas de méthode précise pour les définir, nous avons choisi les nôtres après plusieurs essais.

Ces points de contact appartiennent à l'objet manipulé. Ils sont définis dans le repère de l'arête sélectionnée. Nous avons calculé l'animation de l'objet, nous connaissons donc la trajectoire de l'ensemble des points constituant ce dernier. Il nous suffit donc de reconstruire les trajectoires des points de contacts à partir de la séquence de pivotement. La figure 4.7b nous montre la reconstruction de cette trajectoire pour une séquence de deux pivotements élémentaires (voir chap. 3.1.1).



(a) (b)

FIGURE 4.7 – (a) Points de contact. (b) Exemple de trajectoire de mains pour deux pivotements élémentaires

La trajectoire que nous reconstruisons est une trajectoire paramétrée à vitesse constante. Si l'on exécutait directement cette trajectoire, chaque rotation, chaque étape du pivotement élémentaire (voir Fig. 3.2) irait à la même vitesse. Ceci pose un problème si nous voulons l'exécuter sur la plate-forme réelle. Les changements de directions pour chaque étape du mouvement de pivot créeraient une discontinuité de vitesse trop importante pour le robot et celui-ci risquerait la chute soit pour instabilité ou arrêt des moteurs.

Pour éviter cela, nous devons lisser les deux trajectoires reconstruites en minimisant les secousses (en anglais *minimum jerk*), ceci afin que les mains du robot puissent passer par la vitesse nulle à chaque changement de direction. Nous évitons de la même manière les accélérations trop brutales et gagnons de la souplesse dans le mouvement réalisé.

4.2.2 Calcul du positionnement des pieds

Pour manipuler correctement l'objet, nous devons faire en sorte que le robot garde ce dernier dans son espace de manipulation. Le robot doit donc avancer ou reculer en même temps que l'objet. Nous rappelons que pour plus de réalisme, nous avons choisi de réaliser ce pas au moment de la rotation verticale de l'objet (voir Fig. 3.2). Ceci demande une bonne coordination des bras et des jambes. Il s'agit donc maintenant de prévoir où et comment placer les pieds du robot pour que celui-ci garde une distance de manipulation adéquate et suive au mieux le chemin initialement calculé lors de la planification.

Le positionnement des pieds est calculé à partir du chemin de planification de la boîte englobante et de la séquence de pivotement.

Nous connaissons exactement la position de l'objet sur sa trajectoire de pivotement. Nous estimons, à partir de là, sa position sur chemin de planification non holonome. Ceci nous donne une position estimée, perpendiculaire au chemin. Nous essayons ici, de faire en sorte que le robot garde plus ou moins le chemin non-holonome entre ses jambes, pour rendre la marche visuellement plus naturelle. Une fois cette position calculée, nous traçons la tangente à la courbe en ce point et plaçons une empreinte à gauche ou à droite, alternativement, à des distances m et n. L'orientation des pieds est donnée par cette même tangente. La figure 4.8 illustre le positionnement de la nouvelle empreinte par rapport à la position estimée de l'objet sur le chemin initial. Les paramètres m et n sont quant à eux des paramètres par défaut et estimés par nos soins mais ils peuvent être modifiés par l'utilisateur. Il faut noter que ces paramètres ne correspondent qu'à des distances arbitrairement désirées mais d'autres informations sont à prendre en compte.

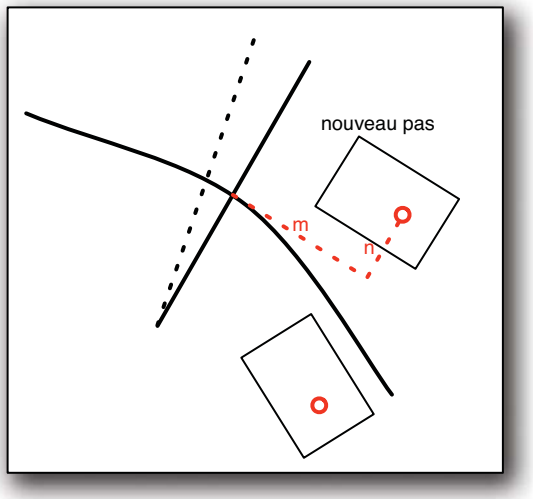


FIGURE 4.8 – Nous positionnons l'empreinte, alternativement à gauche ou a droite, à des distances m et n de la position estimée de l'objet autour du chemin solution initial.

Effectivement, la technique précédemment citée demande plusieurs vérifications. Le pied dont nous calculons la future empreinte ne doit en aucun cas écraser le pied resté au sol ou être trop éloigné de ce dernier. Cela pourrait arriver si le robot se déplace sur un arc de cercle dont le rayon de courbure est faible.

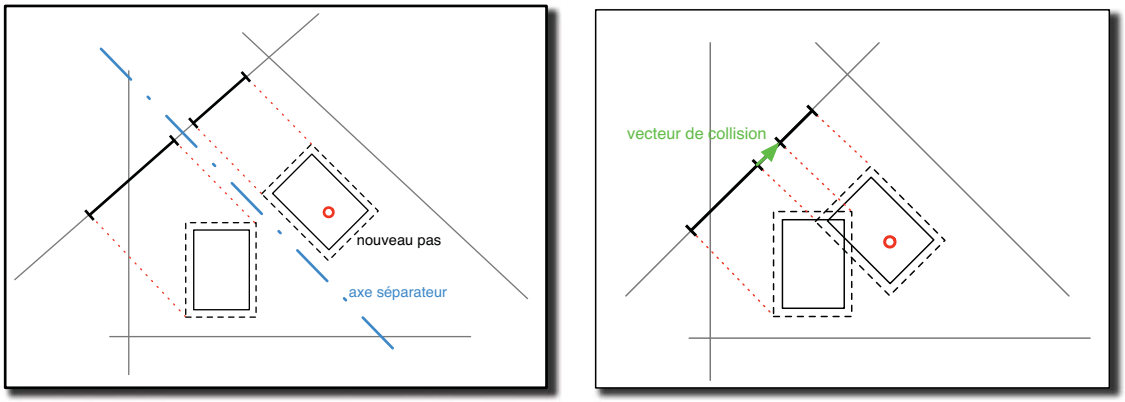
Pour vérifier la collision entre les pieds, nous calculons la future empreinte du pied devant se déplacer et la testons avec l'empreinte du pied resté au sol. Nous utilisons ici, la technique des axes séparateurs. cette technique consiste à vérifier qu'il existe un axe séparant les deux empreintes. Si cet axe existe les deux pieds ne sont pas en collision sinon ils le sont. De plus, pour plus de sécurité nous grossissons les empreintes de $1cm$ sur l'ensemble du contour. La figure 4.9 illustre différents cas de cette technique qui se déroule ainsi :

- Nous construisons, tout d'abord, tous les axes possibles. Pour des polygones, ces axes sont simplement les normales de chacune des arêtes du polygone, il y a donc autant d'axes que de arêtes. Nos empreintes étant rectangulaires, nous ne prendrons donc que deux axes par polygone.
- Nous projetons ensuite les sommets des deux polygones sur chacun des axes et nous en retenons, pour chaque polygone, les points extrêmes des projections sur les axes.
- Si le segment formé par les extrémités du 1er polygone chevauche le segment du 2eme polygone, il n'y a pas d'axe séparateur pour cette projection et inversement, il existe un axe séparateur si les deux segments sont disjoints.
- Si il existe un tel axe les polygones ne sont pas en collision. Si nous ne trouvons aucun axe séparateur, les polygones sont en collision.
- Si les polygones sont en collision, pour en sortir, il suffit de séparer les deux segments se chevauchant le moins sur l'axe correspondant. La longueur du chevauchement nous donne la norme d'un vecteur, l'axe, la direction de ce même vecteur et la future empreinte le sens du vecteur.

Nous construisons ainsi un vecteur de collision qu'il suffit d'ajouter à la position initialement calculée de la future empreinte pour sortir de la collision. En cas de non-collision nous ajoutons simplement le vecteur nul.

Vient ensuite une deuxième boucle de vérification pour contrôler que les deux pieds ne soient pas trop éloignés. Si c'est le cas il suffit de diminuer la norme du vecteur de collision qui avait été grossi artificiellement par sécurité et ré-appliquer la dernière opération.

La méthode développée ici, reste très heuristique mais donne de bon résultat, car nous nous sommes basées sur notre expériences des générateurs de marche pour robot humanoïde. Néanmoins ce problème de positionnement optimum du pieds en déplacement par rapport au pied fixe reste un problème difficile et toujours étudié [Sreenivasa et al. 2009; Kanoun et al. 2009] car il peut facilement induire des problèmes de stabilité du robot.



(a) (b)

FIGURE 4.9 – (a) Nous trouvons un axe séparateur sur au moins un des axes, il n'y a donc pas collision. (b) Aucun axe séparateur n'est trouvé, nous construisons donc un vecteur de collision permettant de sortir de celle-ci, que nous ajouterons à la position de la future empreinte déjà calculée.

4.3 Génération de mouvement corps complet

Nous disposons d'une séquence de pivotement pour l'objet à manipuler, des trajectoires des mains et d'une pile d'empreintes de pas correspondant au déplacement du robot. Tout ceci doit maintenant être exécuté par le robot humanoïde.

Nous rappelons que le mouvement doit être généré de manière à ce que les contraintes d'équilibre, de locomotion et de manipulation soient satisfaites par le robot. Pour cela nous avons utilisé les algorithmes de génération de mouvements corps complets proposés par [Yoshida et al. 2006] combinant le générateur de marche dynamique de [Kajita et al. 2003] et les méthodes de cinématique inverse généralisée par priorités de tâches de [Nakamura 1991; Siciliano and Slotine 1991]. En utilisant la génération de mouvements corps complets soit l'ensemble des articulations du système anthropomorphes qu'est le robot humanoïde, nous nous attendons à de meilleurs résultats, notamment en terme d'espace d'accessibilité que les fonctions de décomposition fonctionnelle utilisées par [Esteves 2007; Yoshida et al. 2008].

Nous allons tout d'abord décrire le mécanisme de cinématique inverse généralisée avec priorité de tâche, puis nous allons expliquer comment nous construisons nos contraintes, entrées de ces algorithmes, avant de lancer toute la mécanique permettant de générer le mouvement.

4.3.1 Algorithme de cinématique inverse généralisée avec priorité

Les algorithmes de cinématique inverse généralisée développés au sein de notre laboratoire s'articulent autour d'un mécanisme de priorité de tâches, de la manière suivante. Considérons une tâche $\dot{x}_j$ ayant pour

priorité j dans l'espace de travail, la relation entre celle-ci et la vitesse angulaire de l'articulation $\dot{q}$ peut s'écrire en utilisant la matrice jacobienne tel que :

$$\dot{x}_j = J_j \dot{q} \tag{4.1}$$

Pour la tâche avec la priorité la plus importante, en l'occurrence les tâches ayant pour priorité 1, la vitesse des articulations permettant de résoudre la tâche, en utilisant la matrice pseudo inverse $J_1^{\#}$, sont donnés par :

$$\dot{q}_1 = J_1^{\#}\dot{x}_1 + (I_n - J_1^{\#}J_1)y_1 \tag{4.2}$$

Où, n, y_1 et I_n sont respectivement le nombre d'articulations, un vecteur arbitraire de dimension n, et la matrice identité de dimension n.

Pour la tâche suivante $\dot{x}_2$ (priorité 2), la vitesse des articulations $\dot{q}_2$ est calculée comme suit [Nakamura 1991] :

$$\begin{aligned} \dot{q}_2 &= \dot{q}_1 + \hat{J}_2^{\#}(\dot{x}_2 - J_2\dot{q}_1) + (I_n - J_1^{\#}J_1)(I_n - \hat{J}_2^{\#}\hat{J}_2)y_2 \\ &\text{où} \quad \hat{J}_2 \equiv J_2(I_n - J_1^{\#}J_1) \end{aligned} \tag{4.3}$$

y_2 étant un vecteur arbitraire de dimension n. Ceci peut être ensuite étendu aux tâches de priorité suivantes j^{th} ($j \geq 2$) avec la formule suivante [Siciliano and Slotine 1991; Baerlocher and Boulic 2004] :

$$\begin{aligned} \dot{q}_j &= \dot{q}_{j-1} + \hat{J}_j^{\#}(\dot{x}_j - J_j\dot{q}_{j-1}) + N_j y_j \\ N_j &\equiv N_{j-1}(I_n - \hat{J}_j^{\#}\hat{J}_j),\ \hat{J}_j \equiv J_j(I_n - \hat{J}_{j-1}^{\#}\hat{J}_{j-1}) \end{aligned} \tag{4.4}$$

Dans ce processus, la priorité elle même n'est pas très important, c'est la différence de priorités entre les différentes tâches qui compte. Effectivement un résultat sera complètement différent du fait que toutes les tâches ont toute la même priorité ou qu'elles ont des priorités toutes différentes. Par contre deux tâches ayant pour priorité respective 1 et 2 , donnent le même résultat que deux tâches ayant les priorités 2 et 3, s'il n'existe pas d'autres tâches à l'instant t dans la pile de tâches à résoudre.

4.3.2 La constructions des contraintes

Toutes les contraintes qui vont suivre sont construites en amont de la génération de mouvements corps complet et empilées dans une pile de tâches. Elles disposent toutes d'au moins trois d'attributs communs :

- Un temps de début et un temps de fin, soit une durée.
- Une priorité, 1 étant la plus importante.
- Une période d'échantillonnage. Cette période nous est imposée à $5ms$ par la commande de la plate-forme réelle HRP-2.

Contraintes sur le bassin et le buste Afin de rendre le comportement du robot un peu plus réaliste et aider l'algorithme à générer des mouvements corps complet plus stables nous construisons trois contraintes sur le bassin et le buste de l'humanoïde qui sont les suivantes (voir figure 4.10) :

1. Une contrainte parallèle sur le bassin afin que celui-ci reste le plus droit possible. Cette contrainte tente de garder $\overrightarrow{Z_{bassin}}$ colinéaire à $\overrightarrow{Z_{monde}}$.
2. Une contrainte permettant de garder le bassin dans un plan. Les modèles de simplification dynamique utilisés par le générateur de marche de [Kajita et al. 2003] contraignant le bassin à se déplacer sur un plan (chap. 2.2).
3. Une contrainte parallèle sur le buste afin que celui reste le plus droit possible. Cette contrainte tente de garder $\overrightarrow{Z_{buste}}$ colinéaire à $\overrightarrow{Z_{monde}}$. Si l'on ne rajoute pas cette contrainte le robot effectue correctement la manipulation mais aura tendance à se baisser de plus en plus vers l'avant au fur et à mesure qu'il avance et inversement s'il recule.

Nous imposons ces contraintes par défaut à l'utilisateur. Elles ne peuvent donc être changées par celui-ci. Nous le verrons un peu plus tard mais ces contraintes ont des priorités moins importantes par rapport aux autres contraintes comme les contraintes sur le centre de masse, les pieds, etc. Mais elles restent néanmoins nécessaires à la qualité du résultat obtenu.

Contraintes sur les mains Nous construisons deux contraintes pour chacune des mains du robot, soit quatre contraintes au total qui seront empilées en parallèle. Pour chacune des mains nous fabriquons, toutes les $5ms$, une contrainte en position et une contrainte en orientation.

La contrainte en position est construite à partir des données des trajectoires des mains extraites des séquences de pivotement et lissées en minimum secousses. L'utilisateur peut changer la position des mains au tout début du processus de planification de mouvement en changeant la position des points de manipulation sur l'objet (voir figure 4.10).

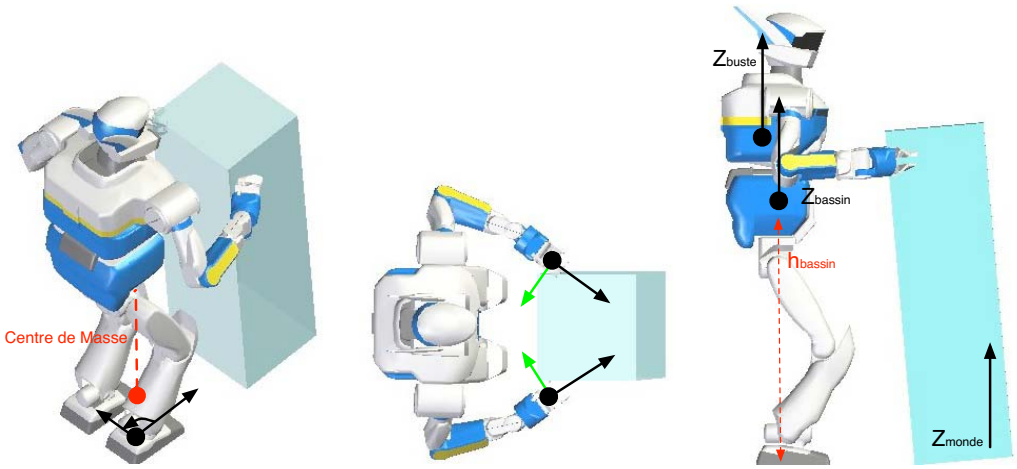
La contrainte en orientation est définie par l'utilisateur au moment de la création de l'objet à manipuler. Celui-ci peut préciser deux vecteurs par main. Bien sûr il existe des vecteurs proposés par défaut. Le premier indique la direction de la paume des mains du robot pendant la manipulation et l'autre la direction "de l'index" (voir figure 4.10). Ces deux vecteurs sont ensuite transformés en une seule et même matrice de rotation qui est appliquée à chacune des mains du robot pendant la manipulation. Cette contrainte permet d'orienter le générateur de mouvements corps complets pour avoir un comportement visuellement plus humain lors de la manipulation, comme par exemple avoir des coudes plus orientés vers l'extérieur.

Contraintes sur les pieds Nous construisons les contraintes sur les pieds comme des contraintes de positions et orientations. Ces positions et orientations sont directement tirées des données de la pile d'empreintes de pas construite dans la section précédente (voir figure4.10). Tous les problèmes de positionnement et d'orientation ayant été résolus en amont, il n'y a rien de plus à faire. Ces contraintes vont permettre de générer le mouvement de marche du robot, en interpolant la future trajectoire de $ZMP_{\text{référence}}$ (voir chap. 2.2.2).

Contraintes sur le centre de masse Cette contrainte est implicitement construite par le moteur de cinématique inverse généralisée incluant le générateur de marche prédictif. Elle permet principalement au robot de rester en équilibre. Pour cela l'algorithme a pour tâche de garder le centre de masse dans le polygone de sustentation du robot. Dans le cas de la marche, un mouvement de marche dynamiquement stable est généré à partir d'une trajectoire du centre masse, elle-même générée à partir de la pile d'empreintes de pas en utilisant les techniques de *ZMP* et les algorithmes du générateur de marche prédictif proposé par [Kajita et al. 2003] (voir section 2.2.2 et la figure 2.11). Si le robot reste immobile, comme pendant la phase de la première inclinaison (voir figure 4.3), le moteur produit tout de même une contrainte permettant de garder l'équilibre.

Les algorithmes calculent la jacobienne du centre de masse [Sugihara et al. 2002] pour construire la tâche correspondante.

La figure 4.10 montre l'ensemble des contraintes imposées à notre système à un instant t de la génération de mouvements corps complet.



(a) (b) (c)

FIGURE 4.10 – (a) Contraintes implicites sur centre de masse et contraintes sur le pied se déplaçant. (b) Contraintes en position et orientation sur les mains. (c) Contraintes sur le buste et contraintes en hauteur et en orientation sur le bassin.

Chaque contrainte possède une priorité différente. Le tableau suivant nous montre la liste des tâches par ordre de priorité.

Priorité 1 :

- Contraintes sur centre de masse - trajectoire (2 degrés de liberté)
- Contraintes sur les pieds (6 degrés de liberté, Faire un pas)

Priorité 2 : Contraintes des mains (6 degrés de liberté, $\times$ 2)

Priorité 3 :

– Contraintes sur le bassin (3 degrés de liberté, horizontale et hauteur)
– Contraintes sur le buste (2 degrés de liberté, horizontale)

4.3.3 Le mécanisme général : des contraintes au mouvement

L'algorithme commence par construire toutes les contraintes et à les empiler en les ordonnançant. Comme nous venons de la voir, ces contraintes sont pour la plupart construites à partir de la pile d'empreintes et des trajectoires de mains du robot. L'algorithme charge ensuite la configuration courante du robot et tente de résoudre la pile de tâche à l'instant t, échantillonné toutes les $5ms$ en activant les contraintes les unes après les autres par ordre chronologique et ordre de priorité (voir tableau ci-dessus). Celui-ci calcule alors la nouvelle configuration du robot permettant de répondre au mieux aux tâches de la pile, à l'instant t. Cette nouvelle configuration deviendra par la suite la nouvelle configuration courante du robot, pour résoudre les tâches à l'instant $t+1$.

La figue 4.11 illustre la génération de mouvements corps complets à partir de d'une séquence de pivotements.

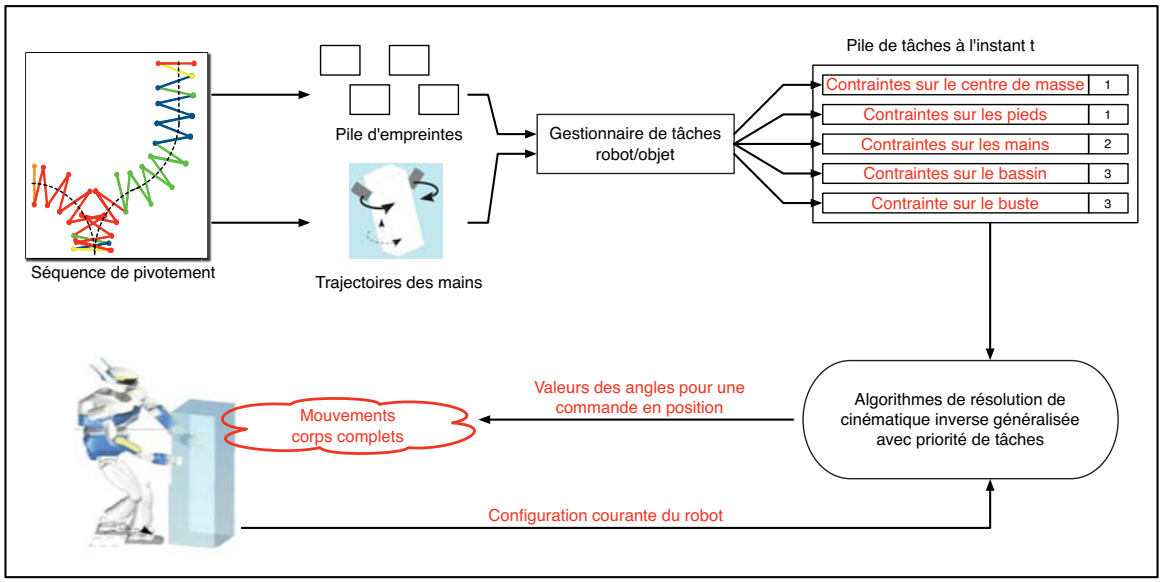


FIGURE 4.11 – De la séquence de pivotement à la génération de mouvements corps complets

Quand une séquence de pivotement vertical de l'objet entre dans la pile de tâche, les fonctions de marche s'activent et se planifient simultanément aux mouvements de manipulation. Pendant cette phase, nous rappelons que le pied sélectionné en fonction du point de rotation utilisé, est positionné et orienté de manière à garder au mieux le chemin initialement calculé lors de la planification, entre les jambes du robot.

Comme nous avons pu le constater dans le tableau ci-dessus, les contraintes sur le centre de masse et les pieds sont les contraintes ayant les priorités les plus hautes. Ceci pour donner avantage à la contrainte

matérielle la plus importante : l'équilibre du robot. L'algorithme utilise le reste des articulations pour réaliser au mieux les autres tâches. Les mouvements corps complets générés impliquent donc à la fois la locomotion, la stabilité et la manipulation en un seul mouvement. Ceci met donc en évidence une différence importante avec les méthodes de décompositions fonctionnelles.

A la fin de ce processus nous récupérons une trajectoire complète sous la forme d'une suite de configurations permettant au robot de manipuler l'objet de sa configuration initiale jusqu'à sa configuration finale.

Nous pouvons ensuite visualiser directement les résultats sur notre simulateur cinématique, KineoWorks (voir section 1.2.3) afin de vérifier une dernière fois qu'il n'y ait pas de problème de collisions et notamment d'auto-collision sur le robot. En cas d'échec, nous devons analyser d'où vient le problème (par exemple un coude qui passe trop près du bassin), modifier un peu certaines contraintes et relancer toute la planification.

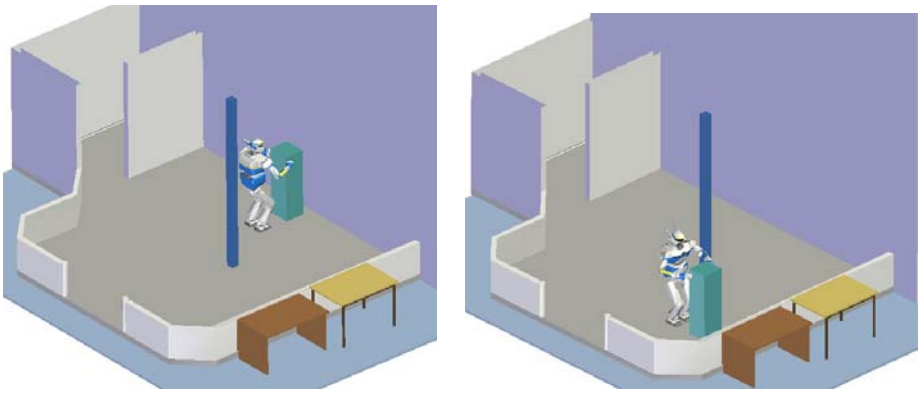
Cette trajectoire est aussi vérifiée sur le simulateur dynamique OpenHRP3 (voir section 1.2.3) afin de confirmer que le mouvement est bien dynamiquement stable.

Si les résultats des deux simulateurs répondent correctement à notre attente, nous pouvons directement utiliser la trajectoire solution sur la plate-forme réelle.

4.4 Résultats expérimentaux

Nous allons présenter dans cette section, nos résultats expérimentaux. ces résultats vont être mis en scène sous forme d'un exemple qui se construit pas à pas. Cet exemple illustrera l'approche énoncée dans les précédentes sections.

Le scénario que nous avons imaginé se déroulera ainsi : le robot doit manipuler un objet supposé représenter l'armoire de la figure 3.1 dans la salle de manipulation de HRP-2 (cet environnement a été modélisé en trois dimensions pour la simulation). L'objet est initialement posé assez près du mur. Le robot est placé à 40 cm de l'objet, face au mur. Celui-ci doit donc dans un premier temps dégager l'objet du mur en le manipulant à reculons. Puis avancer tout en contournant l'obstacle posé au milieu de l'environnement, il doit continuer à manipuler l'objet jusqu'à sa position finale désignée par l'utilisateur. La figure 4.12 illustre ce scénario.



(a) (a)

FIGURE 4.12 – Scénario que nous allons réaliser dans cet exemple, le robot doit déplacer l'objet de sa position initiale (a) à sa position finale (b).

4.4.1 Résultats expérimentaux de simulation

4.4.1.1 Les données expérimentales

Nous notons ici toutes les données nécessaires à la production de notre exemple. Toutes ces données sont en mètre et degré.

Données modifiables par l'utilisateur :

- L'objet à manipuler virtuel : un parallélépipède de 0.32 x 0.35 ($= 2l$) x 1.10
- Configuration initiale : $q_i(0,0,0^\circ)$ et configuration finale de l'objet : $q_f(-2.3,-1.76,90^\circ)$
- Distance de départ entre l'objet et le robot : $0.4m$
- Positions des points de contact entre les mains du robot et l'objet dans le repère de l'objet : $PC_d(0.05,-0.170,0.8)$ et $PC_g(0.05,0.170,0.8)$
- Les vecteurs de manipulation pour les mains du robots sont défini à partir des points de contact, dans le repère de l'objet.
- La position de l'obstacle : $P_{obst}(-1.15,-0.8)$.
- Le rayon utilisé pour les courbes de Reeds et Shepp : $1m$.
- La méthode générale de planification : PRM
- L'angle de rotation horizontale β utilisé est de 15°, soit $\beta_{max} = 2\beta = 30^\circ$.
- L'angle d'inclinaison utilisé est de 5°.

Données non modifiables par l'utilisateur :

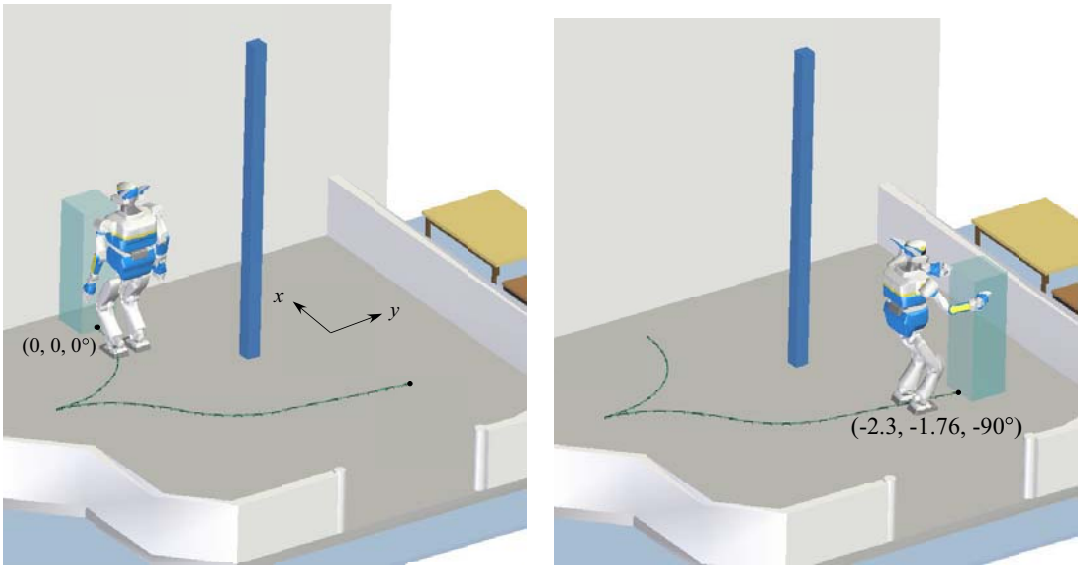
- Le modèle 3D de HRP-2 fourni par Kawada.
- La méthode locale de planification : Reeds et shepp

4.4.1.2 Simulation cinématique

La première partie de la simulation est tout d'abord réalisée sur le logiciel de planification cinématique KineoWorks incluant nos algorithmes de recherche sur le mouvement humanoïde - HPP (voir section

1.2.3). Ce simulateur ne prend en compte que la cinématique du système. Néanmoins les calculs de mouvements corps complets du robot incluant la marche, par les algorithmes de cinématique inverse généralisée utilisant le générateur de marche prédictif reste des calculs dynamiques. La marche calculée en sortie de cette simulation est donc supposée être dynamiquement stable mais elle est tout de même vérifiée dans un simulateur dynamique dans la section suivante. Ce simulateur cinématique permet aussi de vérifier les collisions et auto-collisions à différentes étapes de la planification.

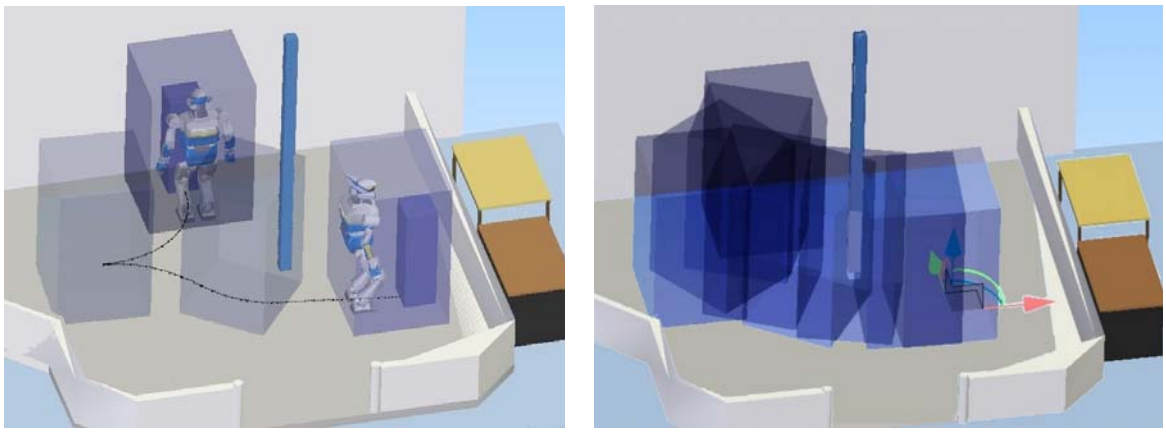
Voici les images commentées, obtenues après chacune des étapes



(a) Etat initial du système

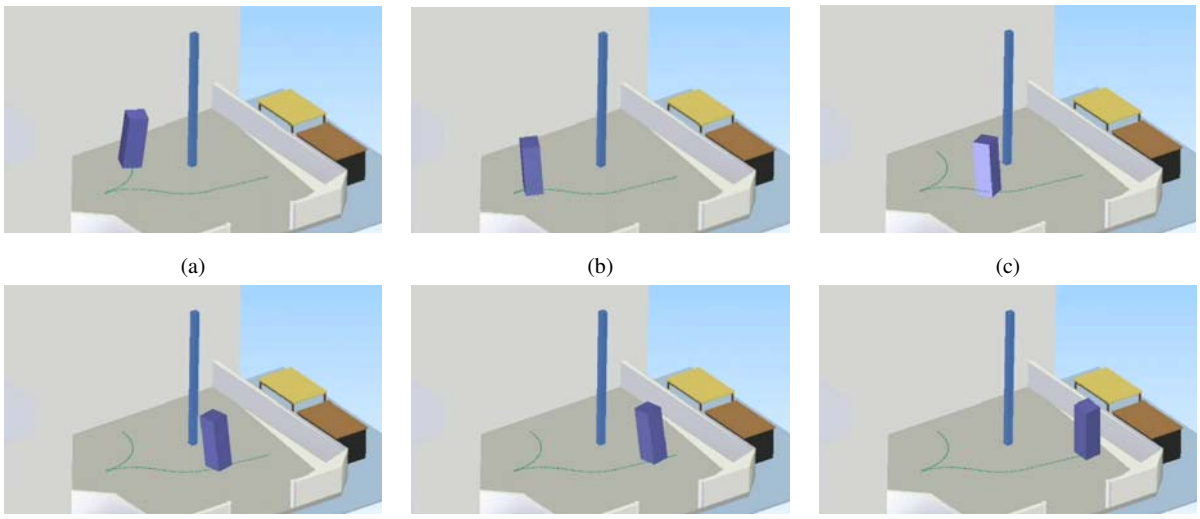
(b) Etat final du système

FIGURE 4.13 – **Définition du problème**. On pose l'objet à manipuler près du mur, à sa configuration initiale (a) et le robot est placé 0.4m derrière. Le robot doit manipuler l'objet jusqu'à sa position finale (b) en suivant une courbe optimisée de Reeds et Shepp lui permettant de dégager l'objet du mur, de contourner l'obstacle et d'atteindre le son but. (La courbe représentée ici est le résultat des prochaines étapes)



(a) (b)

FIGURE 4.14 – 1^{ere} **étape** : (a) les algorithmes de planification de mouvement trouvent la trajectoire non-holonome pour la boîte englobante du système, utilisant les courbes de Reeds et Shepp. (b) On obtient ainsi un volume dans lequel nous savons que le robot peut manipuler l'objet sans entrer en collision avec son environnement, si celui-ci et l'objet suivent le chemin initialement trouvé.



(a) (b) (c)

(d) (e) (f)

FIGURE 4.15 – 2^{ieme} **étape** : On construit la séquence de pivotement autour de la trajectoire initiale. L'objet à manipuler est animé par un mouvement de pivotement autour de celle-ci.

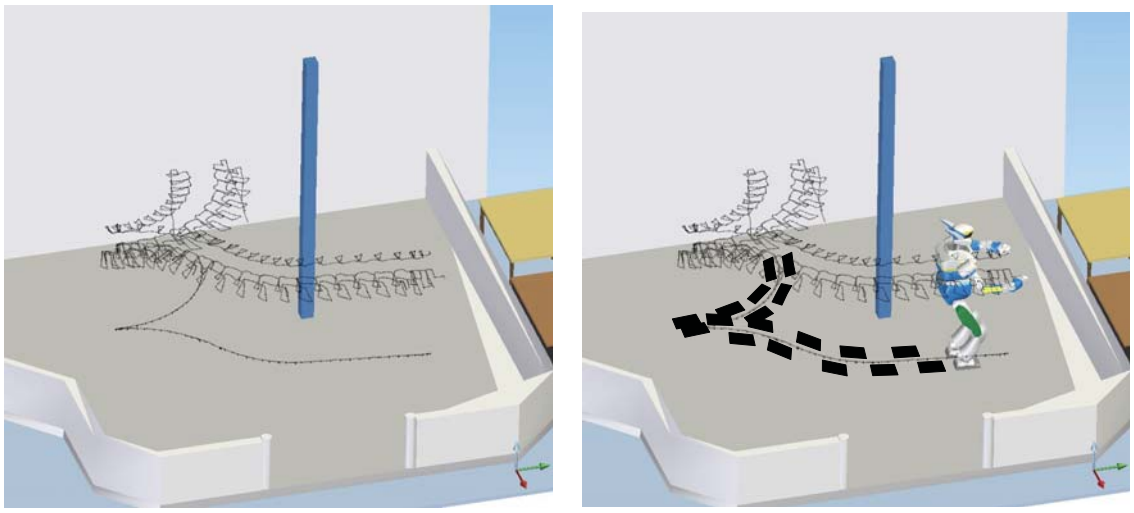


FIGURE 4.16 – 3^{ieme} **étape** : Nous reconstruisons et lissons en minimisant les secousses, les trajectoires que vont devoir suivre les mains du robot pour manipuler l'objet correctement à partir de la séquence de pivotement. Nous construisons aussi la pile d'empreintes de pas que le robot va devoir suivre afin d'accompagner l'objet pendant la manipulation.

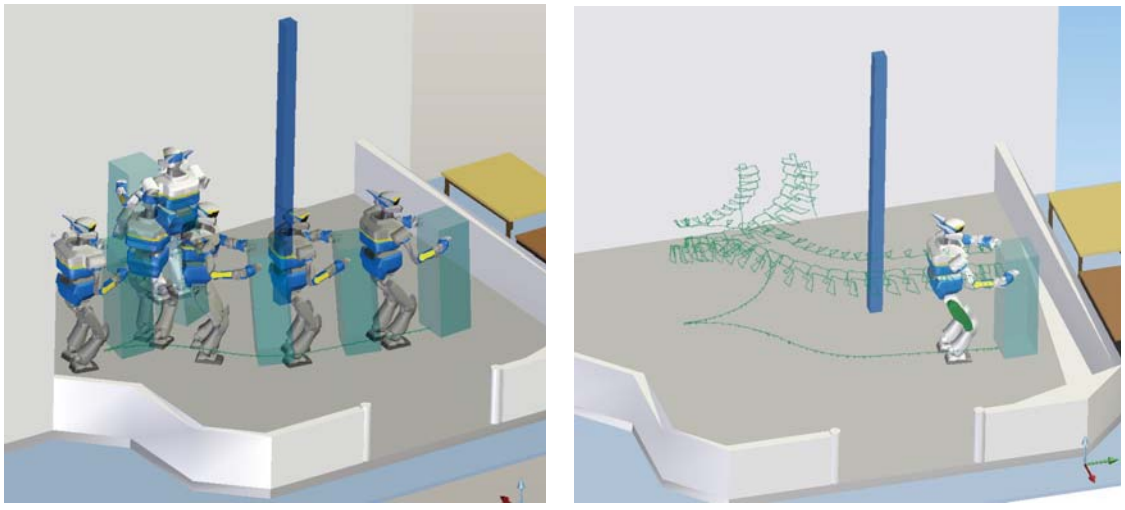


FIGURE 4.17 – 4^{ieme} **étape** : Le mouvement corps complet est ensuite généré pour le robot par nos algorithme de cinématique inverse généralisée. Nous voyons ici que le robot dégage bien l'objet du mur en exécutant un quart de cercle à reculons pour s'éloigner de l'obstacle. Puis celui-ci repasse en marche avant, fini de contourner l'obstacle et manipule l'objet jusqu'à sa configuration finale

4.4.1.3 Simulation dynamique

La deuxième partie de la simulation consiste à vérifier à l'aide du simulateur dynamique OpenHRP3 (voir section 1.2.3), si le mouvement calculé par KineoWorks et nos algorithmes fonctionne dans des conditions un peu plus réalistes.

Dans cette simulation nous ne regardons que ce qui se passe sur le robot, nous n'y chargeons pas l'environnement ni l'objet à manipuler. Nous disposons simplement d'un sol rigide, du poids du robot $P = 549.36N$ et de la force de gravité $g = 9.81m.s^{-2}$ permettant de mettre en évidence un risque de perte d'équilibre du robot.

Nous pouvons aussi surveiller plusieurs paramètres pendant l'exécution du mouvement comme :

- Les consignes de commande envoyées aux moteurs.
- Les forces perçues par les capteurs de forces des mains et des pieds.
- Les accélérations linéaires du robot.
- Les accélérations angulaires du robot.

Avec ces paramètres, nous avons la possibilité de détecter plus facilement les possibles discontinuités durant le mouvement ou encore contrôler la force d'impact des pieds sur le sol Fig. 4.18a.

Effectivement, en simulation la force d'impact ne doit pas dépasser 600-700N ; une précaution, car les capteurs de pressions situés sous les pieds (un des composants les plus chers et les plus importants du robot HRP-2) ne peuvent supporter des impacts supérieurs à 1100N sans subir de dégâts. Aussi, la structure mécanique du robot risque de pas pouvoir absorber cet impact et de s'abîmer.

La figure 4.18 montre l'exécution de la trajectoire solution sur le logiciel OpenHRP3.

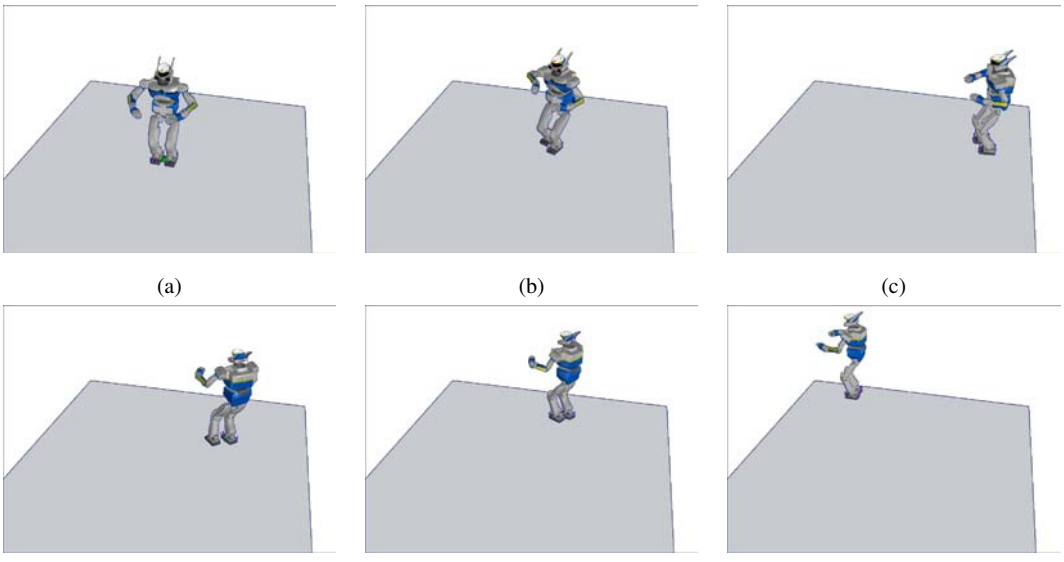
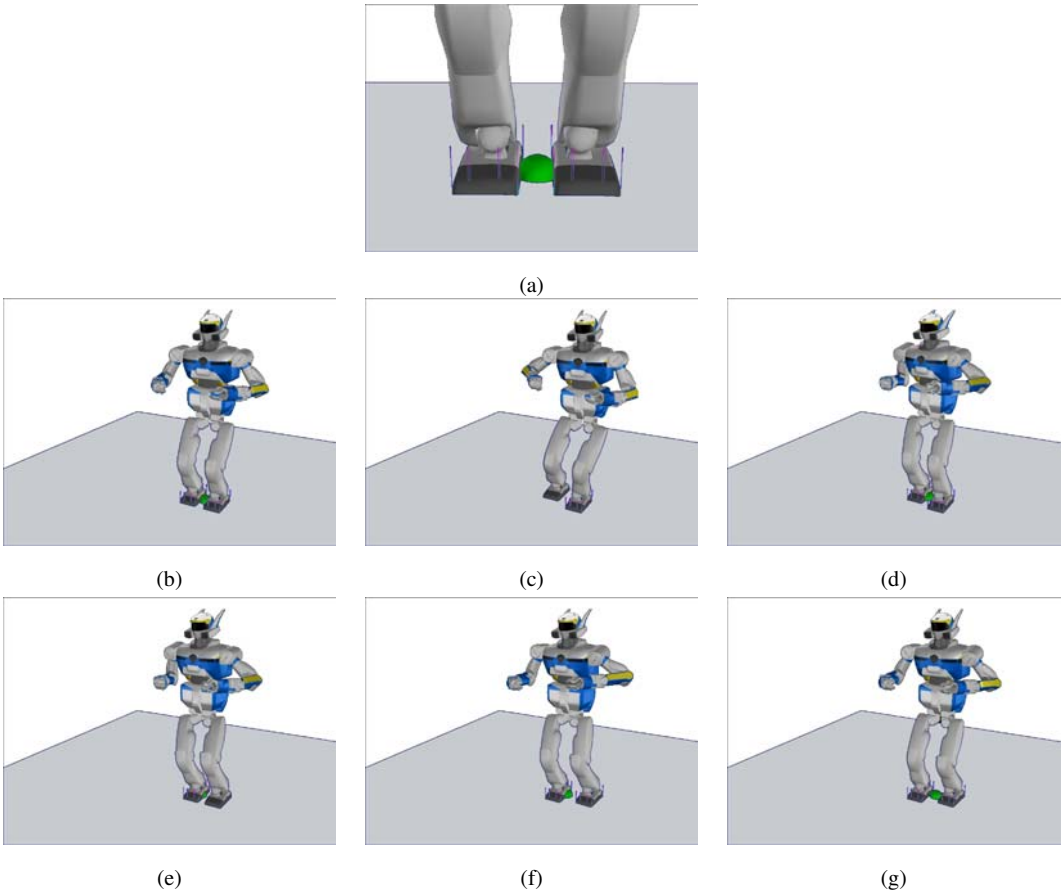


FIGURE 4.18 – Exécution de la trajectoire solution sur le simulateur dynamique OpenHRP3. (a) Position initiale du robot. (b-e) Différentes étapes du mouvement. (f) Position finale du robot. Le temps total du calcul de la simulation dynamique est égale à 1740s (soit 29min), pour un temps d'exécution de 300s

Sur la figure 4.19 nous nous intéressons au mouvement du centre de masse lors de deux mouvements de pivotement en marche arrière.



(a) (b) (c) (d) (e) (f) (g)

FIGURE 4.19 – (a) Zoom sur les pieds pendant la simulation. Les traits verticaux représentent les forces de réaction normales avec le sol, la boule verte indique la projection du centre de masse au sol. (b-g) Exécution de deux mouvements de pivotement en marche arrière. Nous pouvons voir la projection du centre de masse se déplacer d'un pied à l'autre lors des changements d'appuis.

4.4.1.4 Temps de calcul des différentes simulations

Tous les calculs de simulation ont été réalisés sur un PC Intel Core2 Duo à 2.13 GHz.

- La 1^{ere} étape de ce problème, soit trouver un chemin libre de collision pour la boîte englobante prend environ 4 secondes.
- La 2^{ieme} étape, soit la transformation du chemin en une séquence de pivotements ne prend que quelques millisecondes.
- De même la 3^{ieme} étape, incluant la reconstruction des trajectoires de mains et de pieds ne prend que quelques millisecondes.
- Le temps de calculs de la 4^{ieme} étape, soit la génération de mouvement corps complet avec HPP

est de 290 secondes pour générer une trajectoire de 56200 configurations de 40 degrés de liberté, échantillonnée à 5 ms. Cette trajectoire représente donc une trajectoire d'exécution réelle de 281 secondes.

– Le temps de calculs pour la validation dynamique prend quant à lui 1740 secondes, soit 29 minutes pour reproduire dynamiquement la trajectoire trouvée. Par contre, une fois la trajectoire dynamique validée, le temps d'exécution réel reste approximativement le même, soit environ 290 secondes car le simulateur OpenHRP3 se comporte comme la plate-forme HRP-2. Le contrôleur de commande exécute une configuration toutes les $5ms$.

Nous pouvons donc voir que les calculs de génération de mouvement corps complet sont pratiquement temps réel. Cela veut dire qu'à terme, nous pourrions produire directement les calculs sur les machines embarquées par le robot et exécuter les mouvements à la volée.

Malheureusement par souci de sûreté nous préférons passer par notre boucle de validation dynamique et le temps de calcul nécessaire pour cette simulation est tout de suite très important.

4.4.2 Expérimentation sur la plate-forme réelle HRP-2

4.4.2.1 Les données expérimentales

De la même manière nous notons ici toutes les données nécessaires à la production de notre exemple. Toutes les données sont en mètre et en degré. Pour la plupart des paramètres nous utilisons les données utilisées en simulation. Ensuite nous exécutons le mouvement en boucle ouverte, c'est-à-dire que nous faisons les calculs en simulation et, après vérification, nous les exécutons directement sur la plate-forme.

Données modifiables par l'utilisateur :

– L'objet à manipuler : un parallélépipède en bois léger de 0.32 x 0.35 ($= 2l$) x 1.10 pour un poids de 7kg
– Les configurations initiales et finales de l'objet sont les mêmes que celles utilisées en simulation mais deviennent relatives à la position initiale de l'objet.
– Distance de départ entre l'objet et le robot : 0.4m
– Positions des points de contact entre les mains du robot et l'objet dans le repère de l'objet : $PC_d(0.05, -0.170, 0.8)$ et $PC_g(0.05, 0.170, 0.8)$
– La position de l'obstacle : $P_{obst}(-1.15, -0.8)$ relativement à la position initiale de l'objet dans l'environnement réel.

Données non modifiables par l'utilisateur :

– Le temps d'échantillonnage des configurations envoyées au robot : $5ms$
– Le stabilisateur embarqué sur le robot et construit par Kawada. Celui-ci fonctionne de manière autonome et tente de garder le robot à la configuration consigne envoyée, mais il peut parfois osciller autour de cette consigne, ajoutant des perturbations au système. Ici, le robot fait beaucoup de mouvements en même temps mais la vitesse de déplacement reste relativement lente, le stabilisateur n'engendre donc que peu de perturbations.

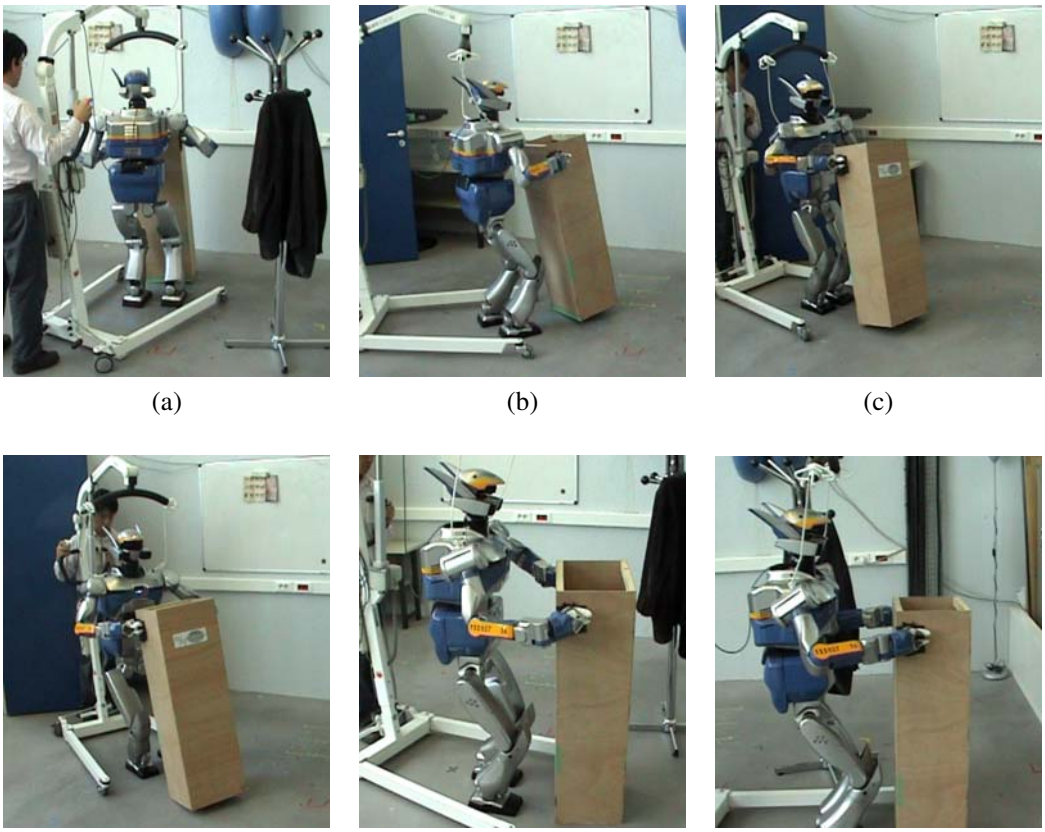
Nous noterons également que durant les expérimentations réelles, nous ne nous occupons pas du tout des forces de contact, ou cône de friction entre le robot et l'objet. Pour contourner le problème nous plaçons les points de contact un peu à l'intérieur de l'objet. Si $2l = 0.35m$ les points devraient être placés à $0.175m$ de part et d'autre de la boîte, sur la surface. Mais nous les avons placés à $0.170m$ soit à l'intérieur pour créer une légère pression entre les mains et maintenir l'objet en place. Nous avons aussi rajouté pour ne pas abîmer le robot et donner encore plus d'adhérence, une bande adhésive antidérapante autour des points de contact de la boîte Fig 4.20

Aussi, par souci de sécurité le robot n'est jamais amené à se déplacer seul. Même si celui-ci est complètement autonome pendant la phase d'exécution, il est toujours accompagné, selon le protocole, par une personne qui le suit avec un lève-personne. Ceci permettant en cas de danger potentiel de déclencher l'arrêt d'urgence et de couper tous les moteurs du robot. Celui-ci devient alors une marionnette complètement désarticulée pendue par deux cordes (voir figure 1.4a).

FIGURE 4.20 – Le point de contact avec l'objet est légèrement décalé à l'intérieur de celui-ci pour exercer une pression suffisante au maintien de la boîte. Nous rajoutons aussi de la bande adhésive antidérapante.

4.4.2.2 Exécution de la trajectoire solution en boucle ouverte.

Après étalonnage du robot, nous installons l'objet près du mur et plaçons le robot $0.4m$ derrière celui-ci. Nous positionnons également l'obstacle, ici représenté sur la figure 4.21 par un porte manteau. Comme pour le simulateur dynamique, la trajectoire calculée par nos algorithmes de planification est envoyée directement depuis le logiciel OpenHRP3 vers les ordinateurs du robot afin d'être exécutée en boucle ouverte. La figure illustre l'exécution de la manipulation en boucle ouverte par le robot HRP-2. La figure 4.22 s'intéresse plus en détail à un mouvement de manipulation.



(a) (b) (c)

(d) (e) (f)

FIGURE 4.21 – Résultats d'expérimentations. départ de la position initiale de l'objet (a) avec l'obstacle sur le côté droit. le robot manipule l'objet à reculons depuis le mur (b). Après un changement de direction vers l'avant (c), le robot continue de manipuler l'objet vers sa position finale tout en évitant l'obstacle (d-f).

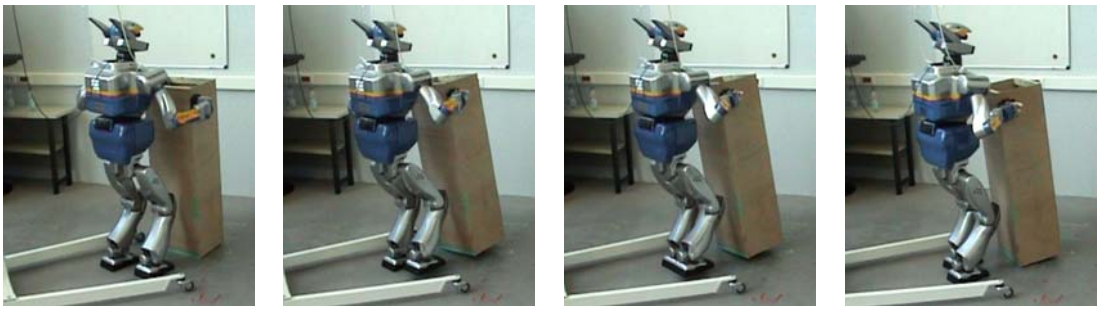


FIGURE 4.22 – Mouvement corps complet détaillé en direction de l'arrière. démarrant depuis la configuration initiale, le robot humanoïde coordonne ses mouvements de bras et jambes. celui-ci fait tourner l'objet dans le sens des aiguilles d'une montre en pivotant sur le coin gauche de l'arrête support et en faisant un pas vers l'arrière avec la jambe droite.

Comme nous pouvons le constater, le robot réussit à exécuter la trajectoire jusqu'au bout. Si nous comparons la trajectoire exécutée par la plate-forme réelle et la trajectoire attendue par les résultats de simulation, nous pouvons calculer que l'erreur moyenne sur la position finale de l'objet est de $0.46m$ (soit $0.31m$ en x et $0.35m$ en y) et que l'erreur d'orientation est en moyenne de $5°$. L'erreur en position de $0.46m$ représente 9% de la longueur totale de la trajectoire qui fait $5.1m$. Si nous prenons en considération que cette trajectoire a été exécutée en boucle ouverte, c'est-à-dire sans retour de capteurs, ou autre moyen de localisation pendant toute l'exécution du mouvement, nous pouvons dire que les résultats de la manipulation sont relativement satisfaisants.

Depuis, cette démonstration a été exécutée un certain nombre de fois, notamment au *Salon Européen de la Recherche et de l'Innovation 2008 - SERI 08*. Elle s'est toujours déroulée sans problème.

Aussi, nous avons pu exécuter les mêmes mouvements sur un deuxième robot : HRP-2 n°10, lors de nos déplacements au JRL-Japon à Tsukuba. Nous avons tout de même constaté quelques différences d'exécution notamment dues : au poids de l'objet utilisé, à la distance de départ entre le robot et la boîte et à la matière dont est composé le sol. Effectivement au JRL-France à Toulouse le sol est en béton lisse, alors qu'au JRL-Japon à Tsukuba le sol est en gomme, les frottements et les impacts des pieds du robot et de l'objet sur le sol sont donc différents.

4.5 Conclusion

Après une analyse formelle d'un système se déplaçant par pivotement, nous avons construit une application permettant le déplacement d'objets encombrants par un robot humanoïde.

Pour cela, nous avons dans un premier temps, réduit l'ensemble du système à une boîte englobante que nous avons dotée de propriétés spécifiquement choisies, connues et issues de la robotique mobile. Ceci nous a permis de trouver rapidement un chemin mais aussi un volume de manipulation libre de collisions dans des environnements complexes.

Grâce à nos connaissances sur les courbes de Reeds et Shepp, nous avons été capable d'optimiser le déplacement par pivotement le long de celles-ci, en n'évaluant qu'une seule fois des motifs optimisés qui vont ensuite servir à approximer le chemin initial de planification.

Une fois l'animation de l'objet créée, nous avons pu reconstruire les différentes contraintes : mains, pieds, buste, bassin, centre de masse que le robot va devoir respecter. Nous donnons ensuite ces contraintes sous forme d'une pile de tâches ordonnancées en entrée de nos algorithmes de cinématique inverse généralisée. Ceux-ci produisent les mouvements corps complet du robot correspondant à la manipulation de l'objet par pivotement.

Apres avoir vérifié nos résultats sur des simulateurs cinématique et dynamique, nous avons pu exécuter la manipulation sur la plate-forme réelle HRP-2. Bien que les expérimentations soient réalisées en boucle ouverte nous obtenons des résultats satisfaisants.

Le scénario choisit pour ce manuscrit à été réalisé afin de démontrer une certain complexité, comme la manipulation en marche arrière, en marche avant et l'évitement d'obstacle. L'environnement de manipulation du robot HRP-2 au LAAS-CNRS n'étant pas très grand, il nous à sembler que ce scénario était le plus pertinent car nous pouvions aller jusqu'à la manipulation réelle. Néanmoins d'autres scénarios de simulation auraient pu être envisagés.

Nous avons choisit de développer la manipulation par pivotement mais avec quelques modifications et les études préalables de commandabilité adéquates, nous pourrions très bien imaginer de manipuler d'autres objets : comme un diable (utilisé pour les déménagements). Pour cela il faudrait créer l'animation correspondant au nouvel objet le long du chemin solution et vérifier que celui-ci est bien exécutable par le robot.

Nous venons donc de déplacer un objet encombrant d'une position initiale à une position finale par une technique de pivotement et à l'aide d'un robot humanoïde. Mais pouvons nous pousser le concept un peu plus loin, en donnant la possibilité au robot, si celui-ci se retrouve dans une situation difficile, de lâcher l'objet, de se déplacer et de le reprendre ?

5

Planification de reprise pour de la manipulation par pivotement

Nous venons de démontrer que nous pouvons, grâce de notre approche déplacer un objet encombrant, par une méthode de pivotement, dans un environnement, en contournant des obstacles. Parfois ces obstacles peuvent devenir très contraignants, empêchant, par exemple, le passage du robot manipulant l'objet. Dans ce chapitre, nous voulons donc offrir au robot la possibilité de relâcher l'objet à une configuration provisoire le temps que celui-ci se déplace à une meilleure position et puisse continuer la manipulation dans de meilleurs conditions. Ce comportement est typiquement adopté par un humain lors d'un déménagement.

Tous les calculs de l'approche précédemment détaillée sont basés sur la désignation d'une arête support et des points de contact sur l'objet. Nous avons donc la possibilité de les changer et de créer une nouvelle position de saisie. De plus, nous avons choisi d'utiliser un objet parallélépipède permettant de simplifier le changement de prises. Nous pouvons donc utiliser les résultats précédents pour résoudre ce nouveau problème.

Nous proposons de résoudre ce problème de planification de reprise pour la manipulation en considérant différents sortes graphes :

- Un graphe $\mathscr{G}$ construit par une méthode de planification par PRM de visibilité [Simeon et al. 2000]. Ce graphe permet de très vite capturer la topologie de l'environnement.
- Un graphe $\mathscr{G}_{obj}$ construit à partir de $\mathscr{G}$. Ce graphe permet d'identifié ce que l'on appellera dans notre méthode des configurations critiques. Ces configurations serviront de points de reprise pour l'objet à manipuler.

– Des graphes de manipulations $\mathscr{G}^i_{manip}$, i étant le numéro de la prise correspondante. Ces graphes sont construits sur les bases de $\mathscr{G}_{obj}$ mais ils peuvent toutefois être compléter en utilisant une méthode de planification par RRT (voir chapitre 2.1.2) si nous ne disposons pas d'assez de noeuds ou d'arête pour résoudre le problème.
– Un graphe de locomotion $\mathscr{G}_{regrasp}$ pour les déplacements du robot uniquement. Ce graphe est construit à partir de $\mathscr{G}$ et représente la carte de déplacement du robot dans son environnement.

La difficulté de cette nouvelle planification résidera donc dans le fait d'identifier ces configurations critiques et de construire la série de graphes pour les différentes positions de manipulation et la locomotion du robot.

Une fois ces graphes obtenus, un chemin de manipulation libre de collisions incluant la reprise d'objet est recherché en appliquant les méthodes de planification par échantillonnage aléatoire sur l'ensemble des graphes. Finalement, comme pour la méthode précédente, les mouvements corps complets du robot sont générés sur cette base.

Cette nouvelle approche ne sera étudiée qu'en simulation. Les environnement utilisés devenant plus complexes, les contraintes réelles d'environnement deviennent trop importantes pour nous. La manipulation de la plate-forme HRP-2 est limitée à la salle de manipulation vue au chapitre 1.2.3.

La figure 5.1 illustre un exemple de scénario que nous souhaitons résoudre, sachant que le robot ne peut traverser aucun des passages étroits (a, b, c, d, e et f), tout en manipulant l'objet.

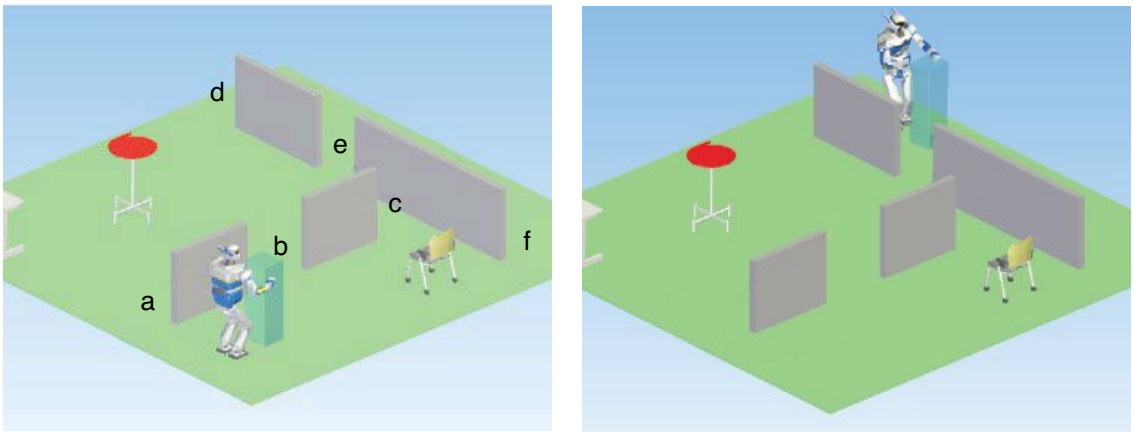
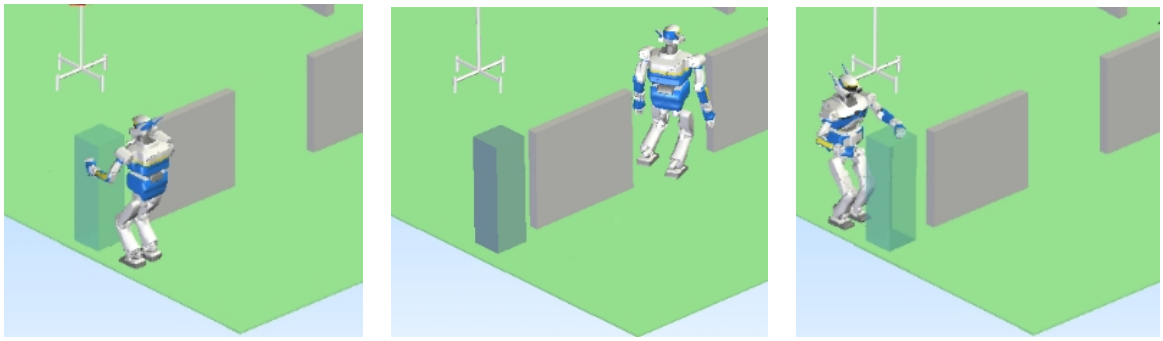


FIGURE 5.1 – Exemple de scénarios proposés pour la planification de reprise. Le robot doit déplacer l'objet de sa position initiale à sa position finale. Le robot ne peut pas traverser les passages noté a, b, c, d, e et f si celui-ci manipule l'objet.

5.1 Identification des configurations critiques

Nous appelons *configurations critiques* ou point de reprise, une configuration de l'objet ne permettant plus au robot de manipuler celui-ci s'il garde la même prise sur l'objet.

Ces configurations critiques, une fois identifiées, sont utilisées pour la planification de tâche de manipulation comme configurations finales d'une première partie de la solution mais aussi comme configuration initiale de la portion suivante si celle-ci n'est pas la configuration finale à atteindre. La figure 5.2 illustre ce changement de prise, en laissant l'objet à une position provisoire.



(a) (b) (c)

FIGURE 5.2 – (a) Fin d'une première portion de manipulation. (b) Transition. (c) début de la portion suivante

Pour identifier ces configurations, nous construisons dans un premier temps, un graphe $\mathscr{G}$ en utilisant les méthodes d'échantillonnage aléatoire de PRM de visibilité [Simeon et al. 2000; LaValle 2006]. Ce graphe est ensuite reconstruit en identifiant les passages trop étroits pour le système robot/objet et en réduisant longueur des arêtes inutiles. Ce nouveau graphe est appelé $\mathscr{G}_{obj}$.

Nous rappelons ici comment est construit le graphe PRM de visibilité. Dans cette méthode, deux type de noeuds sont à distinguer :

1. Pour devenir un noeud **garde**, un noeud q_{new} du graphe ne doit pas être en mesure de voir d'autres noeuds de garde : la région visibilité, $V(q_{new})$, doit être vide de tout autre garde.
2. Pour devenir un noeud **connecteur**, un noeud q_{new} doit être vu par au moins deux noeuds gardes. Il doit exister des gardes q_1 et q_2 , tel que $q_{new} \in V(q_1) \cap V(q_2)$.

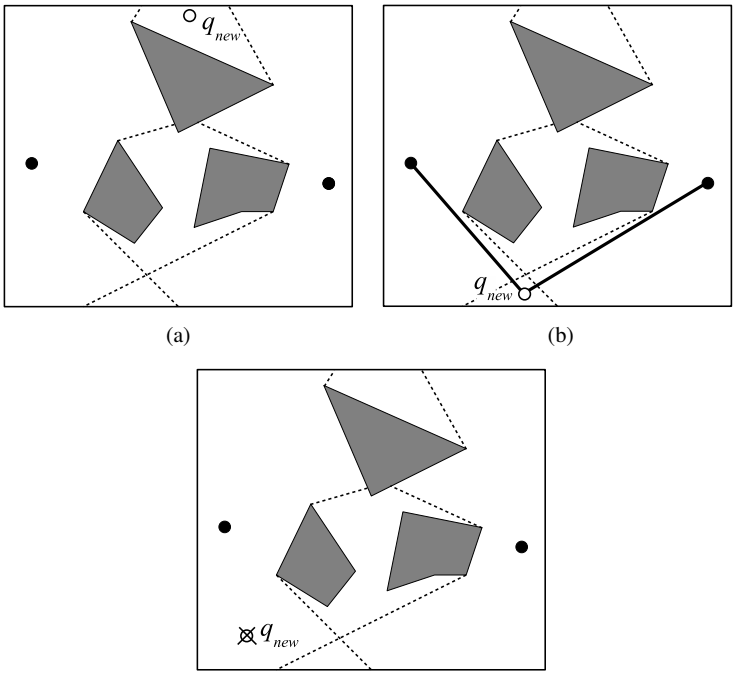


FIGURE 5.3 – Trois cas de la construction d'un graphe de PRM de visibilité.

La figure 5.3 illustre la construction d'un graphe PRM de visibilité. Les points noirs sont les gardes déjà inclus dans le graphe $\mathscr{G}$. Lorsque l'on tire une nouvelle configuration aléatoire q_{new}, il existe trois cas. Tout d'abord, si q_{new} ne peut être connecté à aucun autre garde, q_{new} devient alors un garde et, est inséré dans le graphe $\mathscr{G}$ (Fig. 5.3a). Si q_{new} peut être connecté à au moins deux composantes différentes de $\mathscr{G}$, soit deux gardes, q_{new} devient alors un connecteur et l'on rajoute deux arêtes à $\mathscr{G}$ pour relier les deux composantes (Fig. 5.3b). Enfin, si q_{new} ne satisfait aucune des deux conditions précédemment citées, il est simplement éliminé (Fig. 5.3c).

Les graphes de PRM de visibilité ont la caractéristique de rester compacts, car ils ne font que relier les différentes composantes connexes de l'environnement entre elles. Ceci permet donc d'identifier les passages étroits avec plus d'efficacité que les graphes de planification par échantillonnage aléatoire PRM.

Pour ces raisons nous avons adopté les graphes de PRM de visibilité afin de capturer une vue générale de l'environnement. Le graphe de PRM de visibilité sert alors de pré-processus pour la construction des différents autres graphes, aidant le planificateur à identifier les configurations critiques pour la manipulation.

Le graphe de PRM de visibilité est alors parcouru, on y cherche des chemins en utilisant seulement l'objet à manipuler. Supposons qu'un nouveau noeud q_{new}^{obj}, choisit dans $\mathscr{G}_{obj}$, soit jugé bon pour être ajouté sur le graphe $\mathscr{G}_{obj}$ (Fig. 5.4a). Avant d'ajouter définitivement le noeud et l'arête au graphe $\mathscr{G}_{obj}$, un test de collision est réalisé en attachant à l'objet une boite englobante représentant approximativement le volume du robot. La nouvelle arête e_{new}^{obj} est ajoutée au graphe $\mathscr{G}_{obj}$ pour connecter les deux composantes uniquement si e_{new}^{obj} est libre de toute collision. Si e_{new}^{obj} induit de nouvelles collisions entre le robot et les

obstacles, la portion libre de collision e^{rob}_{free} est extraite (Fig. 5.4b) et e^{obj}_{new} est remplacée par e^{rob}_{free} dans $\mathscr{G}_{obj}$. Un nouveau noeud q^{rob}_{free} est inséré dans $\mathscr{G}_{obj}$ et est considéré comme une configuration critique, une configuration candidate pour un point de reprise de l'objet à manipuler.

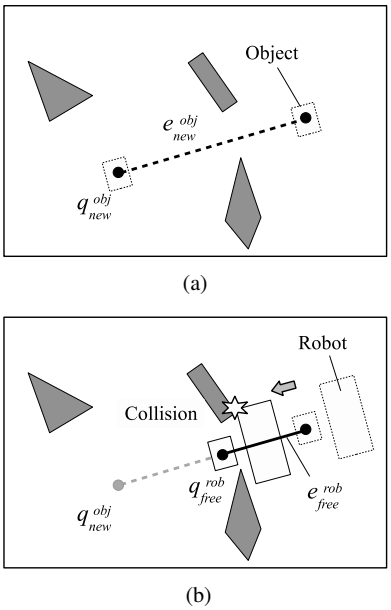


FIGURE 5.4 – (a) Exemple d'arête du graphe $\mathscr{G}$ servant à identifier les configurations critique. (b) En pointillé l'arête du graphe $\mathscr{G}$, en gras la nouvelle arête ajoutée au graphe $\mathscr{G}_{obj}$, q^{rob}_{free} désigne la configuration critique identifiée.

La figure 5.5 représente un exemple de graphe de PRM de visibilité $\mathscr{G}$ créé pour l'objet seul. Nous pouvons observer que tous les passages étroits sont pratiquement trouvés par la méthode. Les arêtes du graphe sont symboliques, elles ne représentent que la connexion possible entre deux noeuds par une courbe de Reeds et Shepp (voir section 4.1.1.3).

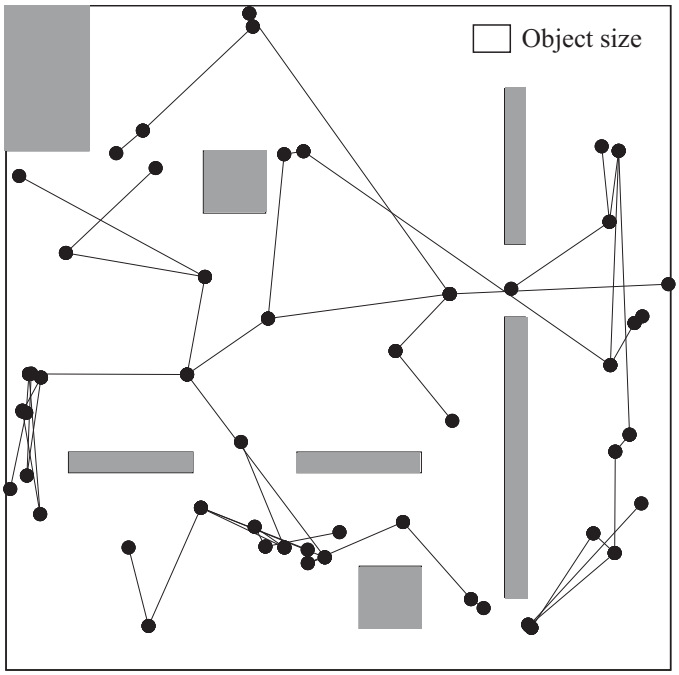


FIGURE 5.5 – Un exemple d'un graphe $\mathcal{G}$ résultat pour l'objet à manipuler. Les arêtes entre les noeuds et illustrées sur la figure par des liens symboliques représentent des courbes de Reeds et Shepp.

5.2 Planification de reprise

Dans cette section, nous décrivons la méthode de planification de reprise d'objet à manipuler par combinaison de graphes de manipulation et de locomotion.

Ces deux types de graphe sont décrit de la manière suivante :

- Le premier appelé *graphe de manipulation*, $\mathcal{G}^i_{manip}$ représentant le graphe de la i_{ieme} position de prise r^i_{manip} relative à la position du robot par rapport à l'objet. Dans ce type de graphe, le robot et l'objet bougent ensemble.
- Le second appelé *graphe de locomotion* pour la reprise, $\mathcal{G}_{loco}$. Dans ce type de graphe le robot se déplace seul entre deux configurations critiques.

La figure 5.6 illustre une vue d'ensemble du schéma de planification. Plusieurs positions de prises sont possible sur l'objet pour une même configuration de celui-ci. Les graphes $\mathcal{G}^1_{manip}$ et $\mathcal{G}^2_{manip}$ correspondent à deux prises différentes de l'objet par le robot. Ces graphes sont interconnectés entre eux par le graphe de locomotion $\mathcal{G}_{loco}$. si l'on recherche un chemin entre A et B celui-ci n'est possible qu'en alternant les différents graphes de manipulation et de locomotion.

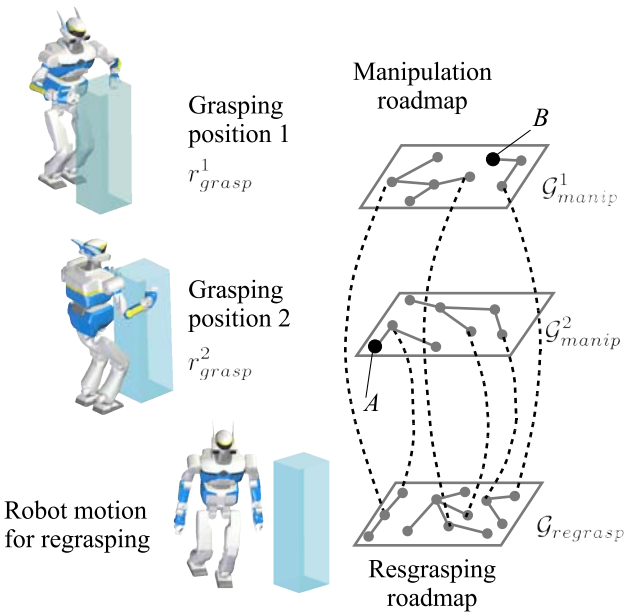


FIGURE 5.6 – Schéma combinant différents graphes de manipulation $\mathcal{G}^1_{manip}$ et $\mathcal{G}^2_{manip}$ connectés entre eux à l'aide d'un graphe de locomotion $\mathcal{G}_{loco}$ pour la reprise de l'objet. Ceci est le seul moyen de trouver un chemin entre les configurations A et B.

Nous allons utiliser le graphe de PRM de visibilité $\mathcal{G}_{obj}$ introduit dans la section 5.1 pour construire les nouveaux graphes de manipulation et de locomotion. L'objectif de cette opération étant de permettre aux différents graphes de couvrir au mieux les zones situées autour des configurations critiques. Le graphe $\mathcal{G}_{obj}$ depuis le graphe de PRM de visibilité est suffisant pour dégrossir la carte et trouver les passages étroits mais il est parfois nécessaire d'affiner la recherche pour trouver précisément les points de reprise. Effectivement, si aucune solution n'est trouvée après l'opération de construction, nous utilisons une méthode d'exploration par diffusion RRT (voir chapitre 2.1.2) afin d'élargir le graphe original autour des endroits étroits.

La procédure de construction des graphes de manipulation et de locomotion est détaillée par l'algorithme de la figure 5.7. Cet algorithme permet de rechercher, en même temps, un chemin dans l'ensemble de ces graphes.

Etant donné qu'il existe plusieurs façons de tenir l'objet à manipuler, il existe donc un ensemble de positions initiales $S_{initiales}$ et de positions finales $S_{finales}$ pour la planification. La procédure PlanWithObjectConfig(q_{obj}, $S_{initiales}$, $S_{finales}$) consiste à faire une recherche de chemin entre l'ensemble des noeuds initiaux et finaux dans l'ensemble des graphes de manipulation et de locomotion, en ajoutant une nouvelle configuration pour le système robot/objet, si celle-ci est libre de collision, dans chacun des graphes de manipulation correspondant ainsi que dans le graphe de locomotion. Cette nouvelle configuration est générée à partir de la configuration de l'objet q_{obj} entrée.

PlanManipulation(q_{obj}^{start}, q_{obj}^{goal})

$\mathcal{G}_{manip}^{i} \leftarrow \emptyset$, $\mathcal{G}_{loco} \leftarrow \emptyset$ n
// Ensemble des noeuds initiaux et finaux : $S_{initiales}$ and $S_{finales}$
$S_{initiales} \leftarrow \emptyset$, $S_{finales} \leftarrow \emptyset$
// Ajoute les noeuds initiaux et de finaux aux graphes
for i-th grasping position r_{grasp}^{i} **do**
    $q_{rob}^{start} \leftarrow$ CalcRobConfigFromObj(q_{obj}^{start}, r_{grasp}^{i})
    $q_{rob}^{goal} \leftarrow$ CalcRobConfigFromObj(q_{obj}^{goal}, r_{grasp}^{i})
    AddNode($\mathcal{G}_{manip}^{i}$, q_{rob}^{start}), AddNode($\mathcal{G}_{manip}^{i}$, q_{rob}^{goal})
    AddNode($\mathcal{G}_{loco}$, q_{rob}^{start}), AddNode($\mathcal{G}_{loco}$, q_{rob}^{goal})
    AddNodeSet($S_{initiales}$, q_{rob}^{goal}), AddNodeSet($S_{finales}$, q_{rob}^{goal})
end for
// Recherche dans tout le graphe $\mathcal{G}_{obj}$
Path $P \leftarrow \emptyset$
for all node q_{obj} in $\mathcal{G}_{obj}$ **do**
    $P \leftarrow$ PlanWithObjectConfig(q_{obj}, $S_{initiales}$, $S_{finales}$)
    if $P \neq \emptyset$ **then**
        Return solution path P
    end if
end for
// Si besoin on élargie $\mathcal{G}_{obj}$ par diffusion RRT : *cf.* [Hsu et al. 1999; LaValle and Kuffner 2001]
for $n < Ntry$ **do**
    $q_{obj} \leftarrow$ RandomObjConfig()
    $q_{obj}^{near} \leftarrow$ NearestNode(q_{obj}^{random}, $\mathcal{G}_{obj}$)
    $q_{obj}^{new} \leftarrow$ Extend(q_{obj}^{near},ε)
    $P \leftarrow$ PlanWithObjectConfig(q_{obj}^{new}, $S_{initiales}$, $S_{finales}$)
    if $P \neq \emptyset$ **then**
        Return solution path P
    end if
end for

PlanWithObjectConfig(q_{obj}, $S_{initiales}$, $S_{finales}$)

for i-th grasping position r_{grasp}^{i} **do**
    $q_{rob}^{i} \leftarrow$ CalcRobConfigFromObj(q_{obj}, r_{grasp}^{i})
    if q_{rob}^{i} is collision-free **then**
        AddNodeAndLink($\mathcal{G}_{manip}^{i}$, q_{rob}^{i})
        AddNodeAndLink($\mathcal{G}_{loco}$, q_{rob}^{i})
        return FindPath($S_{initiales}$, $S_{finales}$, $(\{\bigcap_i \mathcal{G}_{manip}^{i}\} \cap \mathcal{G}_{loco})$)
    end if
end for

FIGURE 5.7 – Algorithmes pour combiner les différents graphes.

La Fig. 5.8 illustre un exemple de graphe résultat généré à partir de la méthode par combinaison de graphes. Les fines lignes avec des noeuds en forme de cercles noirs sont la partie du graphe construite à partir du graphe $\mathscr{G}_{obj}$ initiale de l'objet. Les lignes avec des noeuds en forme de carré sont le résultat de la recherche par diffusion RRT nécessaire à la complétion du graphe initial. Nous avons choisi une méthode de diffusion avec des paramètres auto-réglables [Ferre and Laumond 2004]. Sur cette figure, les différents graphes de manipulation, correspondant aux différentes prises, ont été fusionnés. Nous pouvons observer que la couverture approximative du graphe $\mathscr{G}_{obj}$ sert de guide pour le processus de diffusion et permet de trouver un chemin incluant la reprise.

Dans cet environnement, n'importe quelle méthode locale de planification qui connecte deux noeuds entre eux aurait pu être utilisée. Comme mentionné plus haut, nous avons choisi d'utiliser les courbes de Reeds et Shepp pour les trajectoires de l'objet à manipuler et reproduire ainsi la méthode de manipulation par pivotement vue au chapitre 4. Par contre, la méthode utilisée pour permettre au robot de se déplacer librement dans l'environnement et changer de prises, est une combinaison de courbe de Reeds et Shepp et d'une interpolation linéaire pour réaliser des pas-chassés ou de la marche arrière sur de courtes distances.

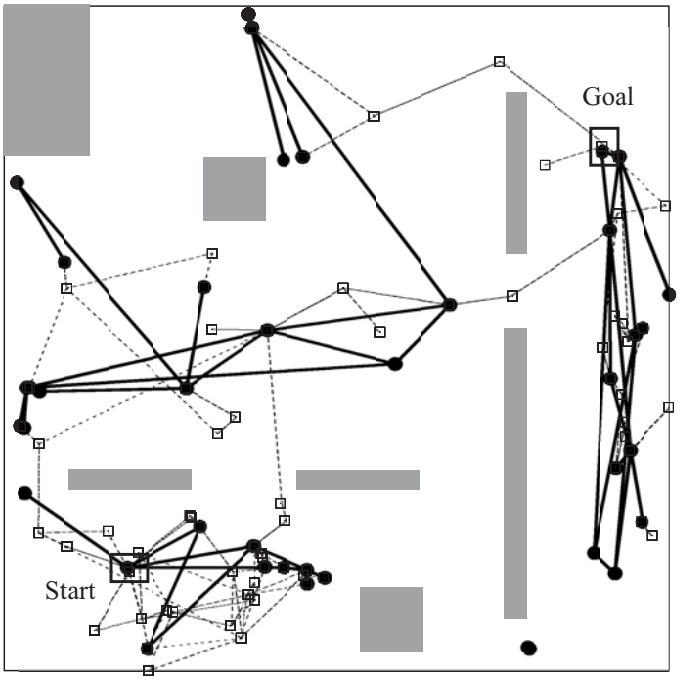


FIGURE 5.8 – Graphe de manipulation généré par la méthode proposée. Les fines lignes avec des cercles dérivent du graphe $\mathscr{G}_{obj}$ initial de l'objet. Les fines lignes avec des carrés ont été additionnées par la méthode de planification par diffusion RRT.

5.3 Problèmes d'implémentation

Nous avons implémenté la méthode proposée dans HPP greffé au logiciel de planification de mouvement cinématique KineoWorks. Grâce à la modularité de l'architecture orientée objet et aux définitions d'interfaces communes employées, le programmeur peut implémenter les composants nécessaires à ses propres problèmes de planification en utilisant l'hérédité des classes mères et les bases de la planification de mouvement déjà proposées par le logiciel de planification cinématique.

Dans la section précédente, nous avons implémenté un système de planification pour la méthode de manipulation d'objet par pivotement utilisant des mouvements corps complets. Ce planificateur interagit avec les fonctions de base du logiciel tel que le moteur de planification, les optimisateurs de chemin, les méthodes locales de planification ou le détecteur de collision . Nous avions ainsi déjà développé la méthode locale utilisant les courbes de Reeds et Shepp pour la manipulation d'objets par pivotement, la génération de séquences de pivotement et la génération de mouvements corps complets.

Dans cette nouvelle partie nous y ajoutons un planificateur de marche pour le robot, mais aussi, la construction des graphes de PRM de visibilité par échantillonnage aléatoire, le constructeur de graphes résultats combinant les deux type de graphes : manipulation et locomotion. De la même manière que dans le chapitre 4, les algorithmes de cinématique inverse généralisée permettent de générer les mouvements corps complets dynamiquement stable incluant la synchronisation des bras et des jambes pour la manipulation mais aussi pour la marche du robot.

La figure 5.9 illustre un schéma de l'implémentation des algorithmes dans HPP et de leur interaction avec KineoWorks.

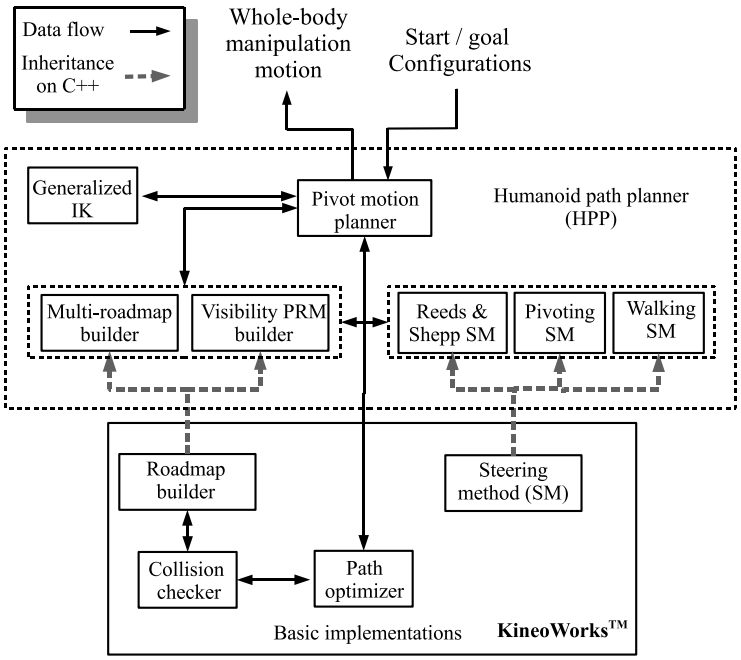


FIGURE 5.9 – Implémentation sur la plate-forme HPP et interaction avec KineoWorks

5.4 Résultats de simulation

La méthode décrite dans ce chapitre a été vérifiée, en simulation, dans un environnement 3D incluant des obstacles et des zones étroites. Le robot humanoïde HRP-2 doit amener un objet virtuel parallélépipède de sa configuration initiale (Fig. 5.10a) à sa configuration finale (Fig. 5.10h).

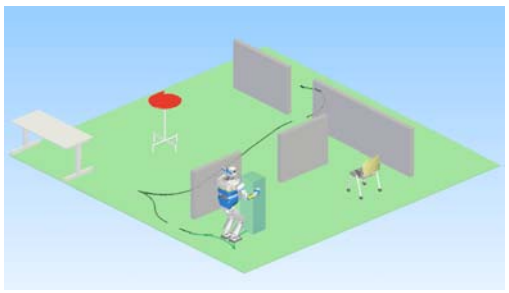
Tout d'abord, une trajectoire de pivotement est planifiée en utilisant la nouvelle méthode proposée. Etant donné que le résultat de la planification de reprise n'est jamais optimale, nous appliquons un optimisateur de chemin pour enlever les possibles redondantes. Cette nouvelle trajectoire est ensuite transformée en une séquence de pivotement pour l'objet puis en un ensemble de contraintes et enfin en mouvements corps complets.

Le robot humanoïde déplace l'objet à l'entrée d'un passage étroit (Fig. 5.10b, c). Il relâche ensuite celui-ci et marche en direction de l'autre coté du mur (Fig. 5.10d). En combinant manipulation vers l'arrière et vers l'avant, le robot dégage l'objet du premier passage pour l'amener à un second passage étroit (Fig. 5.10e, f). Après une autre manoeuvre de déplacement pour la reprise, le robot déplace enfin ce dernier jusqu'à sa configuration finale (Fig. 5.10g, h).

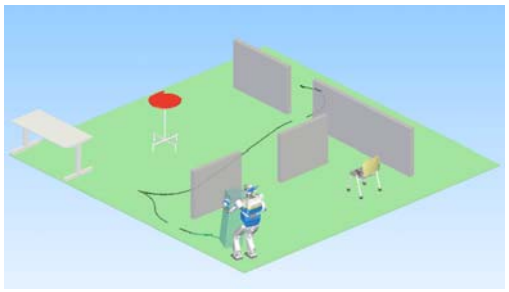
La méthode proposée aurait pourtant besoin d'être améliorée pour trouver un bon compromis entre les graphes de PRM de visibilité et les algorithmes de recherche par diffusion RRT. Effectivement quand l'exploration par visibilité est insuffisante, le planificateur de recherche par diffusion prend plus de temps à explorer l'environnement. De plus, dans notre exemple avec plusieurs passages exigus, les contraintes non-holonomes engendrées par les courbes de Reeds et Shepp rendent la connexion entre les configurations valides assez difficile.

Le temps de calcul moyen pour cet exemple est de $263.7s$, réalisé sur un PC équipé d'un processeur Intel Centrino Duo à $2.16GHz$ et $2GB$ de mémoire. Le ratio d'échec est de 10%, principalement dû à la difficulté d'identifier correctement les configurations critiques de l'objet et aux contraintes non-holonomes posées sur le système robot/objet ou boîte englobante, comme dans le chapitre 4.

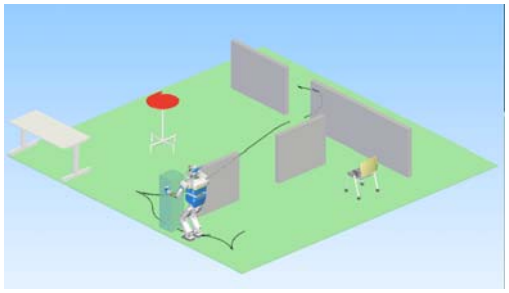
D'un point de vue théorique, la méthode par combinaison de graphes hérite de la complétude probabiliste des algorithmes de recherches par échantillonnage aléatoire PRM et de diffusion RRT, comme dans [Simeon et al. 2004].



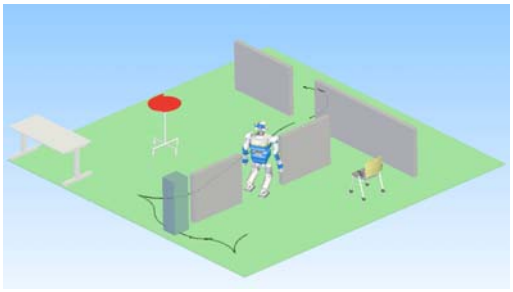
(a) Position initiale



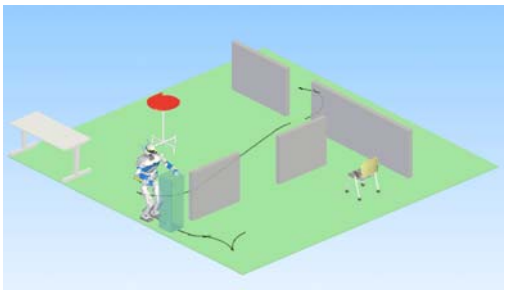
(b)



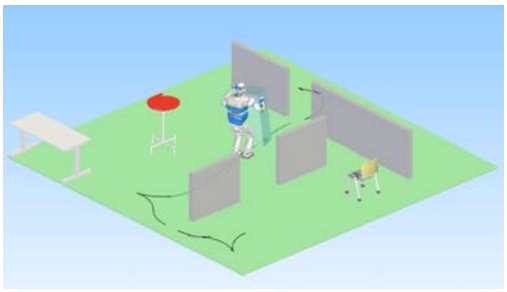
(c)



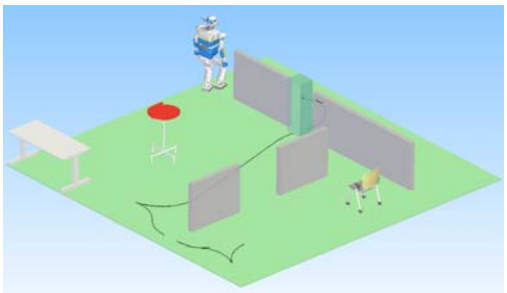
(d)



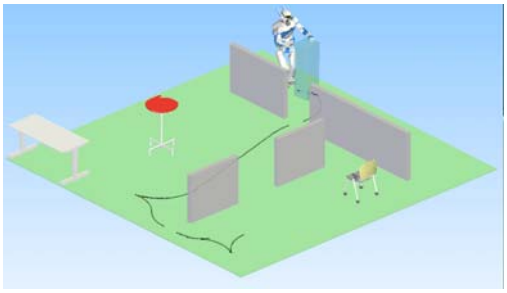
(e)



(f)



(g)



(h) Position finale

FIGURE 5.10 – Résultats de la planification de reprise en simulation. Le robot part de la position initiale (a) et fait pivoter l'objet (b) jusqu'à le placer devant l'entrée d'un premier passage (c). Il laisse ensuite l'objet et marche librement en combinant marche et pas-chassés (d) jusqu'à la nouvelle position de saisie de l'autre coté du mur (e). Le robot avance ensuite vers un second passage étroit (f) et recommence l'opération précédente (g) pour finalement atteindre la configuration finale (h) de l'objet par pivotement.

5.5 Conclusion

Nous avons présenté une méthode de planification de reprise pour de la manipulation d'objet par un robot humanoïde. Pour résoudre ce problème complexe incluant manipulation et changement de prises, nous avons bénéficié d'un bon développement des méthodes de planification par échantillonnage aléatoire. Tout d'abord, les graphes de PRM de visibilité sont utilisés comme pré-processus de la méthode afin d'avoir, rapidement, une vision globale de l'environnement et de capturer les configurations critiques. Il est ensuite utilisé comme base pour combiner différents graphes dont la méthode est composée. Les graphes de manipulation sont construits pour chacune des différentes positions de prises de l'objet à manipuler et interconnectés entre eux par un graphe de locomotion pour la reprise qui permet au robot de se déplacer d'une position de saisie à une autre. L'ensemble de ces graphes est finalement exploré afin de trouver un chemin incluant des changements de prise. Le tout est transformé en contraintes d'entrées pour les algorithmes de cinématique inverse généralisée qui produiront les mouvements corps complets utiles à l'exécution du mouvement par le robot.

6 Conclusion

6.1 Conclusion générale

Le but de cette thèse était de répondre à la question suivante : comment utiliser un robot humanoïde pour déplacer des objets encombrants dans un environnement donné ?

Pour répondre à cela, nous avons étudié la commandabilité d'un système se déplaçant par pivotement et proposé une méthode de planification de tâches de manipulation par pivotement, décomposée en quatre étapes :

1. Planification de mouvements pour une boîte englobante soumise à des contraintes non holonomes connues.
2. Transformation du chemin solution en une séquence de pivotements élémentaires et construction des trajectoires des mains et des pieds du robot.
3. Transformation de ces trajectoires en contraintes ordonnancées dans une pile de tâches et génération de mouvements corps complets pour la manipulation.
4. Validation dynamique et expérimentation réelle.

Nous avons ensuite étendu la méthode à de la planification de reprise d'objet.

Analyse de commandabilité Nous avons commencé notre étude par une analyse de commandabilité d'un système se déplaçant par pivotement.

Dans un premier temps, nous avons réduit le système à une barre ne pouvant pivoter alternativement que sur ses extrémités. Ceci ne change en rien la problématique générale.

Nous avons ensuite étudié, de manière géométrique, comment produire des mouvements simples de translation verticale, translation horizontale et de rotation. Nous avons montré que le système pouvait se déplacer mais que ces mouvements devenaient très vite fastidieux à réaliser.

Puis, nous avons démontré, de manière plus formelle, que le système était localement commandable en temps petit.

Enfin, nous avons construit une méthode locale de planification en trois pivotements permettant au système de se déplacer plus rapidement et de manière très précise. Même si cette fonction ne rend pas compte de la commandabilité locale en temps petit, car il n'existe pas d'ouvert à l'origine, elle permet néanmoins de progresser le long d'un chemin tangent à l'origine.

Planification de mouvement pour la boite englobante Nous démarrons le méthode en réduisant notre système robot/objet à une boîte englobante. Nous lançons ensuite la planification de mouvements pour cette boîte soumise à des contraintes non holonomes connues en utilisant les courbes de Reeds and Shepp et l'algorithme de planification par échantillonnage aléatoire PRM, entre la configuration initiale de l'objet à manipuler et sa configuration finale. Nous obtenons ainsi un chemin solution et donc un volume de travail dans lequel le robot peut évoluer en manipulant l'objet par pivotement sans entrer en collisions avec son environnement.

Séquence de pivotements et trajectoires des mains et des pieds Nous transformons ensuite ce chemin solution en une séquence de pivotements optimisés en utilisant au mieux nos connaissances sur les courbes de Reeds and Shepp. Nous optimisons ainsi les déplacements le long d'un arc de cercle et d'un segment de droite. Nous reconstruisons ensuite les trajectoires que vont devoir suivre les mains du robot pour manipuler correctement l'objet, ainsi que la pile d'empreintes de pas pour que celui-ci puisse accompagner l'objet dans son déplacement vers sa configuration finale.

Pile de tâches et mouvements corps complets Ces séquences de pivotements, trajectoires et piles d'empreintes de pas, sont à nouveaux transformés sous forme de contraintes et ordonnancées dans une pile de tâches. Cette pile est résolue et exécutée, toutes les $5ms$, par nos algorithmes de cinématique inverse généralisée à partir de la configuration courante du robot afin de produire une trajectoire de mouvements corps complets permettant à celui-ci d'avoir une marche dynamiquement stable tout en manipulant l'objet de sa configuration initiale à sa configuration finale.

Validation dynamique et expérimentation réelle Après avoir été validée par un premier simulateur cinématique, la trajectoire corps complets produite est validée dans un second simulateur dynamique pour vérifier que les mouvements exécutés par le robot sont bien des mouvements dynamiquement stables. Une fois validée, la trajectoire est exécutée telle quelle, en boucle ouverte, sur la plate-forme HRP-2. Les résultats obtenus ont ensuite été comparés aux résultats attendus par la simulation cinématique.

Planification de tâche de manipulation avec reprise Nous étendons ensuite la méthode en offrant au robot la possibilité de laisser l'objet à une configuration provisoire le temps de changer de position

et de reprendre l'objet pour continuer à manipuler. Cette nouvelle méthode permet de résoudre des cas de manipulation dans des environnements plus complexe, avec par exemple des passages étroits que le robot ne peut traverser s'il manipule l'objet.

Pour cela nous créons un graphe de planification de mouvement pour l'objet seul, en utilisant une méthode de PRM de visibilité, permettant d'obtenir rapidement la topologie de l'environnement. Nous réduisons ensuite ce premier graphe, en rajoutant le volume du robot et en retestant les arêtes une à une. Nous détectons ainsi les configurations critiques, configurations candidates pour devenir des points de changement de prise ou configurations provisoires.

A partir de ce deuxième graphe, nous construisons une séries de graphes de manipulations et de locomotion. Nous créons un graphe de manipulation par type de prises proposées sur l'objet. Le graphe de locomotion permet au robot de se déplacer seul dans l'environnement.

Enfin, nous cherchons un chemin dans un graphe résultat créé à partir d'une combinaison des graphes de manipulations et de locomotion, le graphe de locomotion servant de lien entre les graphes de manipulations.

Cette méthode a été validée sur simulateur seulement.

6.2 Perspectives

Nous avons réussi, dans une certaine mesure, à répondre à notre question. Mais certains points restent à améliorer sur la méthode de planification de tâche de manipulation comme sur la planification de reprise.

La méthode de planification de tâche de manipulation par pivotement

Notre méthode fonctionne en boucle ouverte, nous n'avons donc pas de retour de la part du robot ni de correction si celui-ci est perturbé pendant l'expérimentation réelle. Nous ne pouvons que mesurer l'erreur une fois l'exécution terminée.

Nous avons commencé à introduire dans nos derniers travaux, la notion de réactivité. Cependant pour que celle-ci soit efficace il faut exécuter les mouvements du robot en boucle fermée. Ce qui sous-entend d'exécuter, à partir des positions courantes de l'objet et du robot, juste une partie du mouvement, ici une séquence de rotations élémentaires accompagnée d'un pas et recommencer l'opération jusqu'à atteindre la configuration finale.

L'idée est donc la suivante :

1. Planifier le mouvement de la boite englobante comme dans la précédente méthode afin d'obtenir un chemin solution à suivre.
2. Récupérer à l'aide d'un système externe (stéréo vision, système de capture de mouvements, etc.) et du robot lui-même en lisant la valeur des encodeurs des moteurs, la position courante du robot et de l'objet dans le monde et les replacer dans le repère du chemin solution.
3. Calculer, à partir de la position courante de l'objet et du robot, la meilleure valeur de l'angle à exécuter pour la prochaine rotation. Nous entendons par meilleure valeur, la valeur de l'angle permettant au système de progresser un maximum le long du chemin tout en s'en écartant le moins

possible. Ceci est réalisé avec une fonction d'optimisation, donnant plus ou moins d'importance à la progression et à l'écart. Pour ne pas diverger et donner de meilleurs résultats cette fonction calcule, en prédiction, plusieurs rotations successives du système.

4. Une fois la valeur de l'angle obtenue, nous construisons une séquence de rotations élémentaires. Puis comme pour la méthode précédente, nous reconstruisons la trajectoire des mains et la position du pas à effectuer.
5. Nous transformons le tout en contraintes ordonnancées dans une pile de tâches et générons le mouvement corps complet du robot seulement pour la séquence de rotation élémentaire, impliquant une seule rotation verticale de l'objet.
6. Si l'objet n'a pas atteint sa configuration finale, nous recommençons à partir de l'étape 2.

De cette manière, si quelque chose vient perturber le robot, ou que celui-ci, du fait des frottements avec le sol, ne peut exécuter correctement une rotation, l'information est directement prise en compte et corrigée à la rotation suivante afin de rester le plus près possible du chemin solution trouvé.

La figure 6.1 illustre par un schéma cette nouvelle méthode.

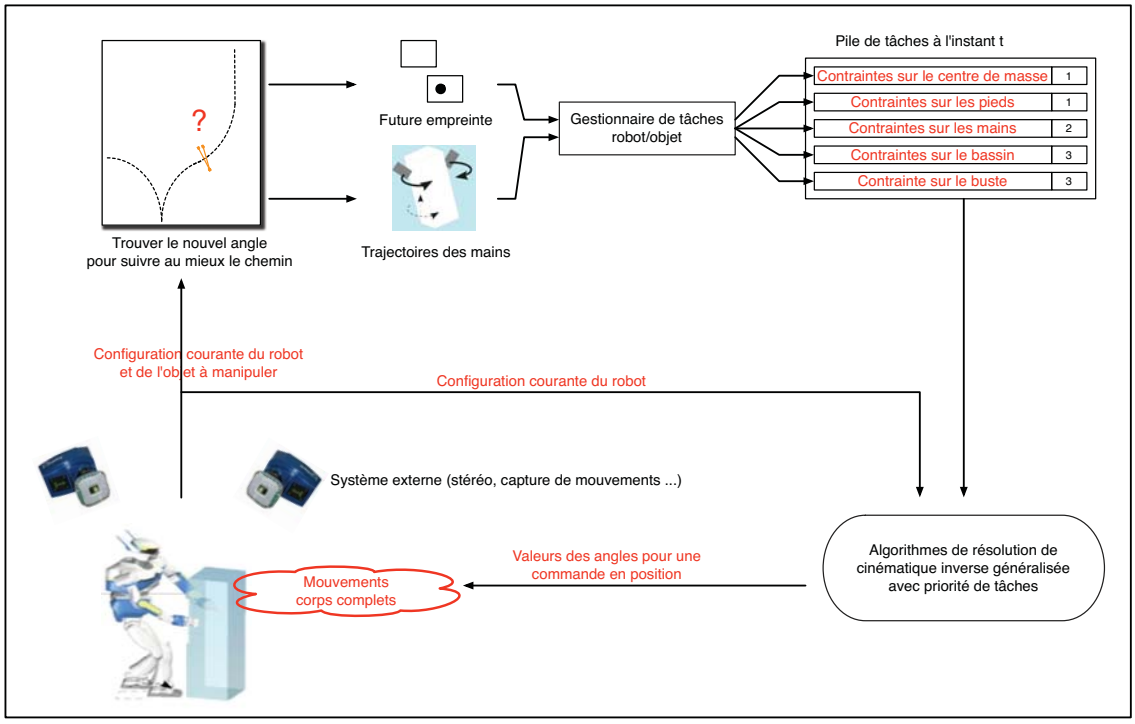


FIGURE 6.1 – Méthode planification de tâche de manipulation par pivotement intégrant la réactivité.

De même, grâce à cette méthode nous ne serions plus limités par l'utilisation des courbes de Reeds and Shepp. Le chemin pourrait être de forme quelconque, puisque l'on ne calcule que la valeur de l'angle pour la rotation suivante.

La méthode de planification de reprise d'objet

Pour la méthode de planification de reprise d'objet, nous voyons déjà deux améliorations à apporter.

1. La première idée serait en partie résolue grâce à la méthode proposée ci-dessus. Effectivement l'utilisation des courbes de Reeds et Shepp dans la construction de nos graphes pouvait poser quelques problèmes car elles invalidaient certaines arêtes du graphe à cause des manœuvres à effectuer dans un environnement restreint pour relier deux nœuds. L'idée serait de relâcher les contraintes de non holonomie.
2. La deuxième idée serait d'intégrer en plus des graphes de manipulation pour le pivotement, d'autres graphes assimilées à d'autres techniques de manipulation comme la levée [Harada et al. 2005; Arisumi et al. 2007; Arisumi et al. 2008], la poussée [Lynch and Mason 1996; Harada et al. 2004] etc.

Références

AIYAMA, Y., INABA, M., AND INOUE, H. 1993. Pivoting : A new method of graspless manipulation of object by robot fingers. In *Proc. 1993 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. 136–143. 27, 30

ALAMI, R., LAUMOND, J., AND SIMEON, T. 1994. Two manipulation planning algorithms. 17 pages. 36

ARECHAVALETA, G., LAUMOND, J., HICHEUR, H., AND BERTHOZ, A. 2008. On the nonholonomic nature of human locomotion. *Autonomous Robots 25,* 1, 25–35. 53

ARISUMI, H., CHARDONNET, J.-R., KHEDDAR, A., AND YOKOI, K. 2007. Dynamic lifting motion of humanoid robots. In *Proc. of IEEE Int. Conf. on Robotics and Automation*. 2661 – 2667. 28, 97

ARISUMI, H., MIOSSEC, S., CHARDONNET, J.-R., AND YOKOI, K. 2008. Dynamic lifting by whole body motion of humanoïde robots. In *Proc. of IEEE Int. Conf. on Int. Robots and Systems*. 668 – 675. 28, 97

BAERLOCHER, P. AND BOULIC, R. 2004. An inverse kinematics architecture enforcing and arbitrary number of strict priority levels. *The Visual Computer 20*, 402–417. 22, 63

BARRAQUAND, J. AND LATOMBE, J. 1991. Robot motion planning : A distributed representation approach. *The International Journal of Robotics Research 10,* 6, 628. 14

BESSIERE, P., AHUACTZIN, J., TALBI, E., AND MAZER, E. The ariadne's clew algorithm : Global planning with local methods. *93*, 1373–1380. 14

BICCHI, A., CHITOUR, Y., AND MARIGO, A. 2004. Reachability and steering of rolling polyhedra : a case study in discrete nonholonomy. *IEEE Trans. on Automatic Control 49,* 5, 710–726. 27

BICCHI, A. AND SORRENTINO, R. 1995. Dexterous manipulation throught rolling. In *Proc. 1995 IEEEInt. Conf. on Robotics and Automation*. 452–457. 27

BOBROW, J. E., DUBOWSKY, S., AND GIBSON, J. S. 1985. Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotics Research 4,* 3, 3–17. 17

CANNY, J. F. 1988. *The complexity of robot motion planning*. MIT Press. 13

CHEN, P. AND HWANG, Y. 1991. Pratcical path planning among movable obstacles. *Proc. IEEE Int. Conf. on Robotics and Automat.*, 444–449. 32

CHOSET, H., LYNCH, K., HUTCHINSON, S., KANTOR, G., BURGARD, W., KAVRAKI, L., AND THRUN, S. 2006. *Principles of Robot Motion : Theory, Algorithms, and Implementation*. MIT Press. 54

CORTES, J. AND SIMEON, T. 2005. Sampling-based motion planning under kinematic loopclosure constraints. *In Algorithmic Foundations of Robotics VI*, 75–90. 26

CROFT, H., FALCONER, K., AND GUY, R. 1992. *Unsolved Problems in Geometry*. Springer-Verlag. 11

DUBINS, L. E. 1957. On curves of minimal length with a constraint on average curvature and prescribed initial and terminal positions and tangents. *American Journal of Mathematics 79*, 497–516. 17

ESTEVES, C. 2007. Motion planning : from digital actors to humanoid robots. Ph.D. thesis, Institut National Polytechnique, Toulouse. Doctorat. 25, 34, 62

FERRE, E. AND LAUMOND, J.-P. 2004. An iterative diffusion algorithm for part disassembly. In *Proc. IEEE Int. Conf. on Robotics and Automat.* 3149–3154. 87

GERVER, J. 1992. On moving a sofa around a corner. *Geometriae Dedicata 42*, 267–283. 11

GOODMAN, J. AND O'ROURKE, J. 2004. *Handbook of Discrete and Computational Geometry*. CRC Press, Boca Raton, FL, USA. 13

GRAVOT, F. 2004. Fondation d'un planificateur robotique intÈgrant le symbolique et le gÈomÈtrique. Ph.D. thesis, UniversitÈ Paul Sabatier, Toulouse, 156p. Doctorat. 36

HARADA, H., KAJITA, S., KANEHIRO, F., FUJIWARA, K., KANEKO, K., YOKOI, K., AND HIRUKAWA, H. 2004. Real-time planning of humanoid robot's gait for force controlled manipulation. In *Proc. 2004 IEEE Int. Conf. on Robotics and Automation*. 616–622. 29, 97

HARADA, H., KAJITA, S., SAITO, H., MORISAWA, M., KANEHIRO, F., FUJIWARA, K., KANEKO, K., AND HIRUKAWA, H. 2005. A humanoid robot carrying a heavy object. In *Proc. 2005 IEEE Int. Conf. on Robotics and Automation*. 1712–1717. 28, 97

HSU, D., LATOMBE, J., AND MOTWANI, R. 1998. Path planning in expansive configuration spaces. *Path planning in expansive configurationspaces. 9,* 4/5, 495–512. 14

HSU, D., LATOMBE, J.-C., AND SORKIN, S. 1999. Placing a robot manipulator amid obstacles for optimized execution. In *Proc. 1999 Int. Symp. on Assembly and Task Planning*. 280–285. 17, 86

KAJITA, S., KANEHIRO, F., KANEKO, K., FUJIWARA, K., HARADA, K., YOKOI, K., AND HIRUKAWA, H. 2003. Biped walking pattern generation by using preview control of zero-moment point. In *Proc. 2003 IEEE Int. Conf. on Robotics and Automation*. 1620–1626. 6, 19, 20, 22, 25, 62, 64, 65

KAJITA, S., YAMAURA, T., AND KOBAYASHI, A. 1992. Dynamic walking control of a biped robot along a potential energy conserving orbit. *IEEE Trans. on Robotics and Automation 8*, 431–438. 19

KANEHIRO, F., HIRUKAWA, H., AND KAJITA, S. 2004. OpenHRP : Open architecture humanoid robotics platform. *Int. J. of Robotics Research 23,* 2, 155–165. 8

KANOUN, O., YOSHIDA, E., AND LAUMOND, J.-P. 2009. An optimization formulation for footsteps planning. In *IEEE-RAS International Conference on Humanoid Robots*. 61

KAVRAKI, L., SVESTKA, P., LATOMBE, J.-C., AND OVERMARS, M. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. on Robotics and Automation 12,* 4, 566–580. 14

KHATIB, O. 2008. Torque-position transformer for task control of position controlled robots. In *Proc. 2008 IEEE Int. Conf. on Robotics and Automation*. 1729–1734. 24, 25

LAMIRAUX, F. AND LAUMOND, J.-P. 1999. Feasible trajectories for mobile robots with kinematic and environment constraints. In *In Proc. of the Fifth International Symposium on Experimental Robotics*. 301–309. 17

LATOMBE, J.-C. 1991. *Robot motion planning*. Kluwer Academic Publishers, Boston-Dordrecht-London. 12, 13, 32

LAUMOND, J., JACOBS, P., TAIX, M., AND MURRAY, R. 1994. A motion planner for nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation 10,* 5, 577–593. 54

LAUMOND, J.-P. 1986. From paths to trajectories for multi-body mobile robots. In *In Intelligent Autonomous Systems*. 346–354. 17

LAUMOND, J.-P. 1998. *Robot Motion Planning and Control*. Lectures Notes in Control and Information Sciences, vol. 229. Springer. 17, 47

LAUMOND, J.-P. 2006. Kineo cam : a success story of motion planning algorithms. *IEEE Robotics & Automation Magazine 13,* 2, 90–93. 8

LAVALLE, S. 2006. *Planning Algorithm*. Cambridge University Press. 54, 81

LAVALLE, S. AND KUFFNER, J. 2001. Rapidly-exploring random trees : Progress and prospects. In *Algorithmic and Computational Robotics : New Directions*, K. M. Lynch and D. Rus, Eds. A K Peters, 293–308. 86

LAVALLE, S. M. 1998. *Rapidly-exploring random trees : A new tool for path planning.* Tech. Rep. TR98-11, Computer Science Department, Iowa State University. October. 14

LI, Z. AND CANNY, J. F. 1992. *Nonholonomic Motion Planning.* Kluwer Academic Publishers. 17

LIEGEOIS, A. 1977. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *Systems, Man and Cybernetics, IEEE Transactions on 7,* 12 (Dec.), 868–871. 21

LOZANO-PÉREZ, T. 1980. Spatial planning : A configuration space approach. In *A.i. memo no.605*. Massachusetts Institute of Technology. 12

LYNCH, K. 1992. The mechanics of fine manipulation by pushing. In *Proc. IEEE R. A.* 2269–2276. 27

LYNCH, K. AND MASON, M. 1996. Stable pushing : Mechanics, controllability, and planning. *Int. J. Robotics Research 15,* 6, 533–556. 29, 97

MAEDA, Y., NAKAMURA, T., AND ARAI, T. 2004. Motion planning of robot fingertips for graspless manipulation. In *Proc. 2004 IEEE Int. Conf. on Robotics and Automation*. 2951–2956. 27, 30, 32

NAKAMURA, Y. 1991. *Advanced Robotics : Redundancy and Optimization*. Addison-Wesley Longman Publishing, Boston. 21, 22, 62, 63

OKADA, K., HANEDA, A., NAKAI, N., INABA, M., AND INOUE, H. 2004. Environnement manipulation planner for humanoid robots using task grpah that generates action sequence. In *Proc. IEEE/RSJ Int. Conf. on Intel. Robots ans Syst.* 1174–1179. 33, 37

REEDS, J. A. AND SHEPP, R. A. 1990. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics 145,* 2, 367–393. 17, 53

SAHBANI, A. 2003. Planification de t,ches de manipulation en robotique par des approches probabilistes. Ph.D. thesis, Universite Paul Sabatier, Toulouse, 137p. Doctorat. 36

SAWASAKI, N., INABA, M., AND INOUE, H. 1989. Tumbling objects using a multi-fingered robot. In *Proc. 20th ISIR*. 609–616. 27

SCHWARTZ, J. T. AND SHARIR, M. 1987. On the piano movers' problem : Ii. In *General Techniques for Computing Topological Properties of Real Algebraic Manifolds. Ablex Series in Artificial Intelligence.* Ablex Publishing Corporation, Norwood, NJ, USA, Chapter 2 51–96. 12, 13

SENTIS, L. AND KHATIB, O. 2004. Prioritized multi-objective dynamics and control of robots in human environnements. In *Proc. 2004 IEEE-RAS Int. Conf. on Humanoid Robots*. 764–780. 24

SENTIS, L. AND KHATIB, O. 2005. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *Int. J. of Humanoid Robotics 2,* 4, 505–518. 24

SHIN, K. AND MCKAY, N. 1985. Minimum-time control of robotic manipulators with geometric path. *IEEE Transactions on Automatic Control 30,* 6, 531–541. 17

SICILIANO, B. AND SLOTINE, J.-J. E. 1991. A general framework for managing multiple tasks in highly redundant robotic systems. In *Proc. IEEE Int. Conf. on Advanced Robotics*. 1211–1216. 21, 22, 62, 63

SIMEON, T., LAUMOND, J.-P., CORTES, J., AND SAHBANI, A. 2004. Manipulation planning with probabilistic roadmaps. *Int. J. of Robotics Research 23,* 7-8, 729–746. 36, 37, 89

SIMEON, T., LAUMOND, J.-P., AND NISSOUX, C. 2000. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics 14,* 6, 477–494. 16, 79, 81

SLOTINE, J.-J. E. AND YANG, H. S. 1989. Improving the efficiency of time-optimal pathfollowing algorithms. *IEEE Transactions on Robotics and Automation 5,* 1, 118–124. 17

SREENIVASA, M., SOUERES, P., LAUMOND, J.-P., AND BERTHOZ, A. 2009. Steering a humanoid robot by its head. *in Proc. of IEEE Int. Conf. on Intelligent Robots and Systems*. 22, 61

STEWART, I. 2004. *Another fine Math You've Got Me Into*. Dover Publications. 11

STILMAN, M. AND KUFFNER, J. 2004. Navigation among movable obstacles : Real-time reasoning in complex environments. In *Proc. 2004 IEEE-RAS Int. Conf. on Humanoid Robotics*. 322–341. 33

STILMAN, M., NISHIWAKI, K., AND KAGAMI, S. 2008. Humanoid teleoperation for whole body manipulation. *IEEE Int. Conf. on Robotics and Automation*. 32

STILMAN, M., NISHIWAKI, K., KAGAMI, S., AND KUFFNER, J. 2006a. Planning and executing navigation among movable obstacles. In *Proc. IEEE/RSJ Int. Conf. on Intel. Robots ans Syst.* 820–826. 33, 37

STILMAN, M., NISHIWAKI, K., KAGAMI, S., AND KUFFNER, J. 2006b. Planning and executing navigation among movable obstacles. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS '06)*. 33

SUGIHARA, T., NAKAMURA, Y., AND INOUE, H. 2002. Realtime humanoid motion generation through zmp manipulation based on inverted pendulum control. In *Proc. 2002 IEEE Int. Conf. on Robotics and Automation*. 1404–1409. 65

SUSSMANN, H. 1982. Lie brackets, real analyticity and geometric control. In *Differential Geometric Control Theory*, R. Brockett, R. Millman, and H. Sussmann, Eds. Progress in Mathematics, vol. 27. Michigan Technological University, Birkhauser, 1–116. 44, 46

SUSSMANN, H. AND TANG, G. 1991. Sortest paths for the reeds-shepp car : A worked out example of the use of geometric techniques in nonlinear optimal control. *Rutgers Center for Systems and Control Technical Report*. 17

VUKOBRATOVIC, M., BOROVAC, B., AND D. SURLA, D. S. 1990. *Biped Locomotion : Dynamics Stability Control and Application*. Vol. 7 of Scientific Fundamentals of Robotics. Springer-Verlag. 18

WAGNER, N. 1976. The sofa problem. *The American Mathematical Monthly 83*, 188–189. 11

WILFONG, G. 1988. Motion planning in the presence of movable obstacles. *Proc. ACM Symp. Computa. Goemetry*, 279–288. 32

YAMAGUCHI, J., TAKANISHI, A., AND KATO, I. 1993. Development of a biped walking robot compensating for three-axis moment by trunk motion. *in Proc. of IEEE Int. Conf. on Intelligent Robots and Systems*. 19

YAMANE, K., KUFFNER, J., AND HODGINS, J. 2004. Synthesizing animations of human manipulation tasks. *ACM Trans. on Graphics 23,* 3, 532 – 539. 33

YOSHIDA, E., BLAZEVIC, P., AND HUGEL, V. 2005. Pivoting manipulation of a large object : A study of applcation using humanoid platform. In *Proc. 2005 IEEE Int. Conf. on Robotics and Automation*. 1052–1057. 30, 31

YOSHIDA, E., BLAZEVIC, P., HUGEL, V., YOKOI, K., AND HARADA, K. 2006. Pivoting a large object : whole-body manipulation by a humanoid robot. *J. of Applied Bionics and Biomechanics 3,* 3, 227–235. 30, 36

YOSHIDA, E., ESTEVES, C., BELOUSOV, I., LAUMOND, J.-P., SAKAGUCHI, T., AND YOKOI, K. 2008. Planning 3D collision-free dynamic robotic motion through iterative reshaping. *IEEE Trans. on Robotics 24,* 5, 1186–1198. 4, 25, 34, 37, 62

YOSHIDA, E., KANOUN, O., ESTEVES, C., LAUMOND, J.-P., AND YOKOI, K. 2006. Task-driven support polygon reshaping for humanoids. In *Proc. 2006 IEEE-RAS Int. Conf. on Humanoid Robots*. 827–832. 6, 22, 62

YOSHIDA, E., POIRIER, M., LAUMOND, J.-P., ALAMI, R., AND YOKOI, K. 2007. Pivoting based manipulation by humanoids : a controllability analysis. *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1130–1135.

YOSHIDA, E., POIRIER, M., LAUMOND, J.-P., KANOUN, O., LAMIRAUX, F., ALAMI, R., AND YOKOI, K. 2008. Whole-body motion planning for pivoting based manipulation by humanoids. *Proc. 2008 IEEE Int. Conf. on Robotics and Automation*, 1712–1717.

YOSHIDA, E., POIRIER, M., LAUMOND, J.-P., KANOUN, O., LAMIRAUX, F., ALAMI, R., AND YOKOI, K. 2009a. Pivoting based manipulation by humanoid robot. *Autonomous Robots*.

YOSHIDA, E., POIRIER, M., LAUMOND, J.-P., KANOUN, O., LAMIRAUX, F., ALAMI, R., AND YOKOI, K. 2009b. Regrasp planning for pivoting manipulation by a humanoid robot. *Proc. 2009 IEEE Int. Conf. on Robotics and Automation*.