ACADÉMIE DE MONTPELLIER

UNIVERSITÉ MONTPELLIER II

- SCIENCES ET TECHNIQUES DU LANGUEDOC -

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ MONTPELLIER II

Discipline : Génie Informatique, Automatique et Traitement du Signal
Formation Doctorale : Systèmes Automatiques et Microélectronique
École Doctorale : Information, Structures et Systèmes

présentée et soutenue publiquement

par

Sébastien LENGAGNE

le 21 octobre 2009

préparée dans l'équipe projet INRIA - DEMAR, hébergée au LIRMM :

Planification et re-planification de mouvements sûrs pour les robots humanoïdes

JURY :

Fethi Ben Ouezdou	Professeur, Université Versailles Saint-Quentin	Président
Luc Jaulin	Professeur ENSIETA	Rapporteur
Philippe Soueres	Directeur de recherche, CNRS/LAAS	Rapporteur
David Guiraud	Directeur de recherche, INRIA/Sophia-Antipolis	Examinateur
Philippe Fraisse	Professeur, Université Montpellier 2	Directeur de thèse
Nacim Ramdani	Maitre de Conférences HDR, Université Paris 12.	Co-directeur de thèse

Préface

J'ai rencontré Sébastien tout à fait fortuitement lorsque fin 2007, pris par le désir de faire du sport, je me suis inscrit dans un club dont il était membre. J'ignorais alors que ce garçon sympathique qui m'apprenait avec patience les rudiments du tennis de table était passionné par la robotique et qu'il travaillait d'arrache-pied dans ce domaine.

J'ai progressivement découvert qu'il avait sacrifié beaucoup pour sa passion et ses études, allant jusqu'à quitter sa région, sa famille, sa fiancée et ses amis.

Sébastien est un jeune homme courageux, généreux et dévoué. Ce qui le caractérise et le différencie du commun des mortels, c'est sa capacité d'écoute, sa gentillesse et surtout sa manière de vous venir en aide et de vous soutenir dans des moments plus difficiles que d'habitude en agissant efficacement et en restant humble.

Bien que béotien en matière de robotique, j'ai lu son ouvrage entièrement et j'ai compris beaucoup de choses. Bien entendu, je n'ai pas déchiffré le quart de ce qu'il a présenté par des formules riches de connaissances scientifiques, mais j'ai compris pourquoi il avait choisi ce métier pour lequel il soutient sa thèse aujourd'hui.

S'il avait été moins sensible, il aurait fait un excellent médecin, néanmoins, son esprit charitable et son amour des êtres l'ont amené à choisir une voie dans laquelle son besoin d'aider ceux qui l'entourent et sa passion pour la technologie, pouvaient s'exprimer de concert.

En ce début de 21ème siècle, les progrès scientifiques sont tels que nous pouvons désormais utiliser des technologies qui étaient inaccessibles il y a encore une dizaine d'années. Que nous réserve le futur ? Sébastien fait partie de ceux qui apporteront les réponses.

Puisse l'avenir démontrer que les réussites des travaux que tu entreprendras récompenseront tes qualités à leurs justes valeurs et feront progresser les hommes dans leur quête d'amélioration constante.

Honoré de réaliser la préface de cette thèse, je n'ajouterai que quelques mots :

"Félicitations pour votre travail Docteur Sébastien Lengagne."

Jean-Charles BECKMANN

Remerciements

Ce mémoire porte sur les travaux de recherche effectués au Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM). Je remercie Monsieur Michel Robert, son directeur, de m'avoir accueilli au sein du laboratoire. Je remercie également la Région Languedoc Roussillon et l'Institut National de Recherche en Informatique et Automatique (INRIA), d'avoir financé mes travaux de recherche.

Je tiens particulièrement à remercier tous les membres l'équipe DEMAR (DEambulation et Mouvement ARtificiel), et notamment David Guiraud, à la tête du projet, pour m'avoir permis de diriger ma thèse dans la direction que je souhaitais, ainsi que Annie Aliaga pour son aide administrative.

Bien sûr, je n'oublie pas mes deux encadrants Nacim Ramdani et Philippe Fraisse, pour la confiance qu'ils m'ont accordé, mais aussi pour les conseils tant au niveau technique que méthodologique. Leur rigueur, leur sens critique ainsi que leur expérience compose une part importante dans la progression des travaux présentés.

Je voudrais, également, témoigner toute ma reconnaissance envers les membres du jury, et particulièrement les rapporteurs, pour le temps et la confiance qu'ils m'ont accordés, et pour la qualité de leurs commentaires.

Sur un plan, plus personnel j'aimerai dire merci à Baptiste, Michel, Mourad, Lotfi, Lei, Jeremy, Guillaume ainsi qu'à tous les thésards que j'ai pu croiser, pour leur bonne humeur et surtout d'avoir rit, ou tout du moins d'avoir fait semblant de rire à mes blagues pas toujours très drôles je dois le reconnaître, me permettant ainsi de rester dans un état d'esprit positif, nécessaire à un travail de thèse.

Merci aux quelques chti'mi que j'ai retrouvé dans le sud, je pense notamment à Cécile et Fanny qui en leur disant bonjour, m'ont permis de me croire l'espace d'un instant dans ma région d'origine.

Je ne vais pas oublier mes amis pongistes que j'ai pu rencontrer durant ces trois années. J'ai une petite pensée pour les joueurs du club de Clermont l'Herault à qui je souhaite de continuer dans leur lancée. Bien sûr je ne vais pas oublié Jean-Charles et Marie-France qui m'ont apporté énormément sur le plan humain et que je vais beaucoup regretter une fois parti.

J'ai gardé le meilleur pour la fin, un grand merci et un grand bisou à Audrey, que j'aime de tout mon coeur, dont j'ai été séparé pendant quatre ans et que je vais enfin pouvoir retrouver.

Table des matières

Chapitre 1

Introduction

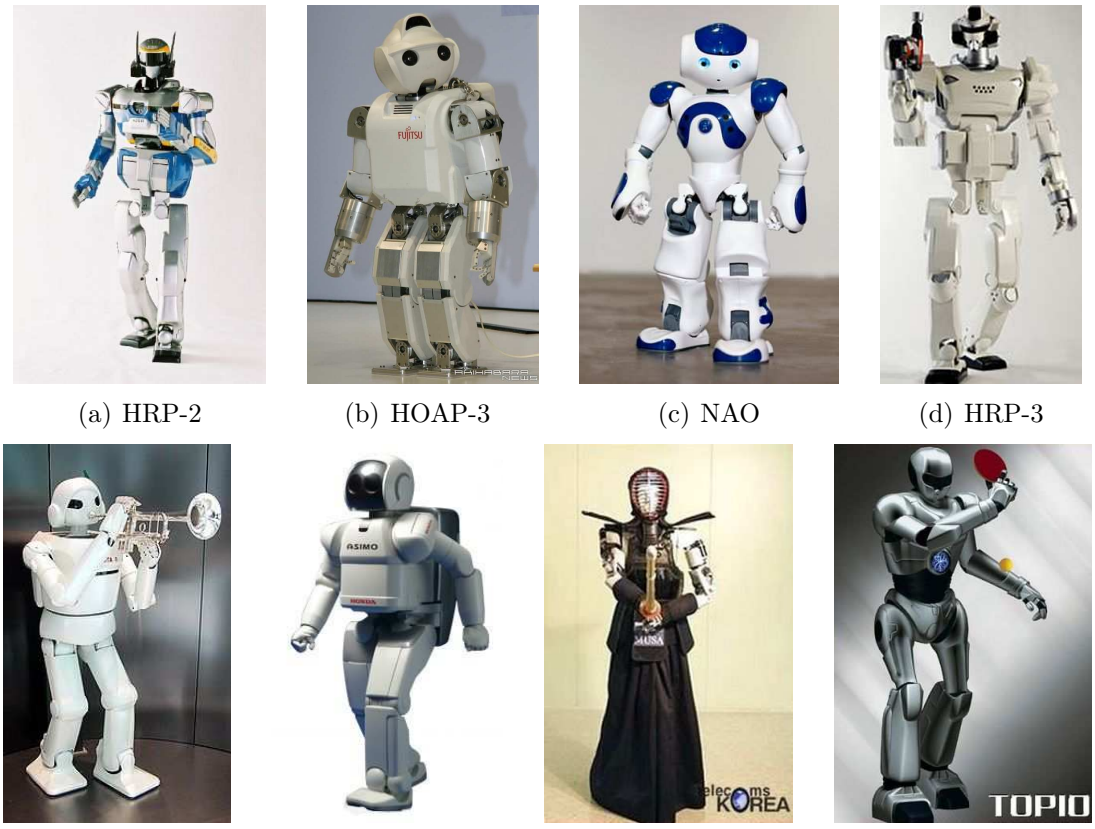
Sommaire

1.1 Pourquoi des robots ? et des robots pour quoi ?

De nos jours, la population mondiale est en train de vieillir, 7.6% des habitants de notre planète ont plus de 65 ans, en 2020 ils seront 9.3% et 16.2% en 2050 (chiffres de l'Institut National des Études Démographiques). Ce phénomène est très présent dans les pays développés. Les habitants de France et des USA sont respectivement 17% et 13% à être âgés de plus de 65 ans aujourd'hui, ils seront 26.9% et 21.6% en 2050. En dehors des aspects démographiques et économiques, le principal problème des prochaines décennies sera de maintenir la qualité de vie et d'assurer l'autonomie des personnes appartenant à cette tranche de la population.

Le Japon, le plus vieux pays du monde avec 22.6% aujourd'hui et qui, en 2050, aura près de 37.8% de ses habitants âgés de plus de 65 ans, a misé sur le développement des robots humanoïdes pour résoudre ce problème en assistant les seniors par la réalisation de tâches simples (aller chercher un objet, aider une personne à se lever, ...). Leur présence permanente dans les logements sera plus rassurante que le passage d'un auxiliaire de vie quelques heures par jour, sans pour autant le rendre inutile. En effet, certaines actions trop complexes ne pourront pas être confiées aux robots et devront être laissées à des êtres humains compétents. Les robots ne constitueront donc pas une alternative gratuite à la compétence humaine, mais une aide permettant d'améliorer la vie quotidienne.

A cause du vieillissement de sa population, le Japon a été le premier pays à créer un grand nombre de robots humanoïdes, le plus célèbre étant le robot ASIMO (cf. Figure



(a) HRP-2 (b) HOAP-3 (c) NAO (d) HRP-3

(e) PARTNER (f) ASIMO (g) MUSA (h) TOPIO

FIG. 1.1 – Photographies de quelques robots humanoïdes.

1.1(f)) de Honda. L'avancée technologique des japonais dans la construction des robots humanoïdes fût telle que certains de leurs robots furent importés pour permettre aux chercheurs français de tester et de valider leurs méthodes. De cette façon, le robot HRP-2 (Figure 1.1(a)), de Kawada, a élu domicile au LAAS à Toulouse, tandis que le robot HOAP-3 (Figure 1.1(b)), de Fujitsu, a emménagé au LIRMM à Montpellier. De nos jours, la France a rattrapé son retard dans la construction de robots grâce, notamment, à Aldebaran Robotics et leur robot Nao (Figure 1.1(c)), qui a été choisi comme plate-forme standard pour la Robocup.

La Robocup est une compétition annuelle qui regroupe plusieurs équipes pour des matchs de football entre robots, avec l'objectif affiché d'avoir une équipe de robots capable de battre l'équipe championne du monde d'ici 2050. En dehors de l'aspect ludique et attractif, la pratique du sport par des robots humanoïdes constitue un véritable enjeu pour la recherche, certains robots sont même dédiés à la pratique d'un sport : le robot Musa (cf. Figure 1.1(g)) est un robot d'entraînement au Kendo, alors que Topio représenté Figure 1.1(h) est un robot pongiste. Le sport nécessite certaines aptitudes, comme la gestion de l'équilibre, la rapidité de réaction ou la stratégie de jeu, qui sont également importantes pour la réalisation de tâches quotidiennes.

1.2 Sécurité pour protéger et servir

L'avenir des robots est donc d'assister les humains dans leurs activités quotidiennes. Il est évident que ce futur n'aura lieu que si le robot ne présente aucun risque pour l'être humain. Les trois lois de la robotique, écrites par l'écrivain de science-fiction Isaac Asimov, sont des règles auxquelles tous les robots qui apparaissent dans sa fiction doivent obéir afin d'assurer la sécurité des hommes. Exposées pour la première fois dans sa nouvelle *Cercle vicieux* (Runaround, 1942) mais annoncées dans quelques histoires plus anciennes, ces lois sont :

1. Un robot ne peut blesser un être humain ni permettre qu'un être humain soit exposé au danger que ce soit par action ou inaction.
2. Un robot doit obéir aux ordres que lui donne un être humain, sauf si de tels ordres entrent en conflit avec la première loi.
3. Un robot doit protéger son existence tant que cette protection n'entre pas en conflit avec la première ou la seconde loi.

Contrairement à Jeff Vintar et Akiva Goldsman, scénaristes du film *I-Robot*, nous pouvons considérer que ces trois lois sont parfaites d'un point de vue cognitif. Cependant, elles ne peuvent être mises en pratique que si le robot a la conscience et la pleine maîtrise de tous ses mouvements. Apporter, au robot, la conscience des effets de ses mouvements est un problème d'intelligence artificielle qui nécessite des capacités d'apprentissage et d'anticipation. Permettre aux robots de maîtriser leurs mouvements est le but du travail de recherche mené par les automaticiens et roboticiens et auquel nous contribuons par cette thèse.

A l'heure actuelle, les lois d'Isaac Asimov ne sont, donc, toujours pas applicables aux robots humanoïdes. Historiquement, la première étape fut d'assurer la sécurité du robot lui-même afin d'en garder le contrôle par le développement de méthodes de commande et de planification de mouvements. Par exemple, s'assurer de l'équilibre et du bon fonctionnement du robot permet de le maintenir sous contrôle et d'éviter d'éventuelles chutes qui peuvent s'avérer dangereuses pour les personnes occupant le même environnement. De nos jours cette étape n'est pas encore totalement achevée, et certains chercheurs s'orientent déjà vers une autre étape qui est l'interaction avec l'environnement ou avec l'homme.

Dans cette thèse, nous avons choisi de contribuer à la première étape en développant des méthodes de planification et de re-planification de mouvements sûrs pour les robots humanoïdes.

1.3 Plan du rapport

1.3.1 État de l'art

Dans le chapitre 2, nous présentons les principales méthodes de planification de chemins et de mouvements, ainsi que les méthodes de séquencement d'actions élémentaires pour la navigation du robot, puis nous mettons en avant l'importance de garantir la sécurité de fonctionnement des robots, pour finalement montrer les enjeux et les techniques employés pour l'adaptation de mouvements.

La planification de mouvements pour les robots humanoïdes nécessite de savoir les modéliser. Dans le chapitre 3, nous détaillons comment calculer l'évolution des variables

articulaires à partir d'un ensemble de paramètres, puis nous présentons le fonctionnement et les limites des robots humanoïdes en partant des moteurs et des couples qu'ils doivent générer pour réaliser un mouvement jusqu'à un ensemble de grandeurs physiques qui doivent être bornées pour garantir la sûreté de fonctionnement du robot, comme le glissement du pied sur le sol, l'équilibre du robot, ou encore l'évitement d'auto-collision. Finalement, nous introduisons la plate-forme de test que nous utilisons (le robot HOAP-3) ainsi que l'ensemble des contraintes à respecter pour assurer l'intégrité du robot.

1.3.2 Planification de mouvements sûrs

Le chapitre 4 présente la planification de mouvements basée sur la résolution d'un problème d'optimisation afin de trouver l'évolution des variables articulaires qui minimise un critère et valide des ensembles de contraintes de type inégalité et égalité. Ce problème est transformé en un problème de programmation semi-infinie (*Semi infinite Programming* SIP) grâce à une paramétrisation des trajectoires articulaires. Afin de pouvoir utiliser des logiciels d'optimisation, il faut obtenir un problème de dimensions finies et donc nous devons discrétiser les contraintes 'inégalité' continues qui traduisent les limites physiques du robot. Dans la plupart des cas, cette discrétisation se base sur une grille temporelle qui va assimiler les contraintes à un ensemble de valeurs correspondant aux instants de la grille. Nous montrons que cette méthode est dangereuse car elle n'assure pas le respect de ces contraintes sur toute la durée du mouvement.

Afin de résoudre ce problème, nous développons une méthode de discrétisation basée sur un outil mathématique : L'Analyse par Intervalles. Cette méthode de discrétisation garantie décompose la durée du mouvement en plusieurs intervalles et calcule les extrema des contraintes 'inégalité' continues pour chaque intervalle. De cette manière, l'algorithme d'optimisation peut trouver une solution qui respectera les limites physiques du robot et en assurera la sécurité et l'intégrité.

Nous avons développé une librairie utilisable en C++, la GDL (*Guaranteed Discretization Library*), disponible sur le site `http://www.lirmm.fr/~lengagne/GDL/` que nous avons utilisée pour la génération de mouvements sûrs.

La méthode de planification de mouvements sûrs que nous proposons permet donc d'assurer la validité des contraintes. Cependant le temps de calcul trop important de cette méthode nous a incité à développer une méthode de planification hybride qui assure la validité des contraintes dans un laps de temps CPU comparable à celui des méthodes classiques. La méthode hybride alterne une phase d'optimisation basée sur la méthode de discrétisation classique et une phase de vérification utilisant la discrétisation garantie qui va modifier les contraintes en fonction des violations calculées jusqu'à ce que le mouvement produit respecte toutes les limites physiques du robot.

1.3.3 Re-planification de mouvements

Nous sommes, donc, en mesure de générer des mouvements optimaux qui sont parfaitement adaptés à un environnement donné. Mais que se passe-t-il si l'environnement est différent, ne serait-ce que légèrement ? Dans le chapitre 5, nous mettons en évidence que l'utilisation de n'importe quelles méthodes d'optimisation pour la re-planification de mouvement n'est pas une solution acceptable à cause de son temps de calcul trop important. Pour pouvoir générer un mouvement qui soit adapté à l'environnement courant, nous présentons une méthode de re-planification rapide de mouvements. Cette méthode

se base sur le calcul d'un sous-ensemble représentant une approximation intérieure de l'ensemble de tous les paramètres du mouvement qui satisfont les contraintes 'inégalité' liées aux limites physiques. Ce sous-ensemble, obtenu grâce à l'analyse par intervalles ne contiendra que des mouvements qui assurent la sûreté de fonctionnement du robot. Le processus de re-planification consistera à rechercher dans ce sous-ensemble un mouvement satisfaisant les nouvelles contraintes 'égalité' qui dépendent de l'environnement courant.

Nous appliquons cette méthode de re-planification à un mouvement de coup de pied dans une balle, nous calculons un mouvement optimal pour une position donnée de la balle, puis re-planifions le mouvement pour qu'il soit adapté à une position différente de la balle. Nous montrons que le processus de re-planification est suffisamment rapide pour pouvoir être utilisé lors d'une application en temps réel.

1.3.4 Expérimentation

Le chapitre 6 présente une expérimentation consistant à effectuer la navigation globale du robot à partir d'une base de données de mouvements de pas obtenus grâce à la méthode de planification hybride de mouvements sûrs.

Tout d'abord, nous définissons les caractéristiques de la base de données : nombre de pas, type de pas, etc ... Puis, nous présentons la génération d'une séquence de pas dans le but de rejoindre une cible fixe dans un environnement comportant des obstacles. Nous montrons que cette solution n'est pas adaptée à une utilisation en ligne pour de grands déplacements. Finalement, nous développons un programme qui permet au robot de rejoindre une cible mobile en calculant la direction dans laquelle se trouve la cible pour ensuite choisir le mouvement adéquat.

1.4 Publications de l'auteur

1.4.1 Publications internationales

- **S. Lengagne**, N. Ramdani, P. Fraisse : *"Guaranteed computation of constraints for safe path planning"*, 2007 IEEE International Conference on Humanoid Robots, Humanoids 2007, Pittsburg USA, 29 nov- 1 dec pp. 312-317.
- **S. Lengagne**, N. Ramdani, P. Fraisse : *"A new method for generating safe motions for humanoid robots "*, 2008 IEEE International Conference on Humanoid Robots, Humanoids 2008, Daefon Corée du Sud, 1-3 dec pp. 105-110.
- **S. Lengagne**, N. Ramdani, P. Fraisse : *"Safe motion planning computation for databasing balanced movement of Humanoid robot "*, 2009 IEEE International Conference on Robotics and Automation, ICRA 2009, Kobe Japon 12-17 mai pp. 1669-1674
- **S. Lengagne**, N. Ramdani, P. Fraisse : *"Planning and Fast Re-Planning of Safe Motions for Humanoid Robots : Application to a Kicking Motion"*, 2009 IEEE International Conference on Robots and Systems, IROS 2009, Saint-Louis USA, 11-15 oct. *nominé pour le Best Paper Award catégorie Robocup*

1.4.2 Publications nationales

- **S. Lengagne**, N. Ramdani, P. Fraisse : *"Méthode pour la planification de Trajectoires Garanties"*, 2008 Journées Francophones de Planification Décision et Apprentissage, JFPDA 08, Metz, 19-20 juin.

– S. Miossec, **S. Lengagne**, A. Kheddar, K. Yokoi : *"Motion optimization of robotic systems and validation on HRP-2 robot "*, 2008 RSJ National Conference, Kobe Japon, 9-11 sept

1.5 Contexte de la thèse

Ces travaux de thèse, co-financés par l'INRIA (*Institut National de Recherche en Informatique et en Automatique*) et la Région Languedoc-Rousillon, ont été menés au sein de l'équipe-projet DEMAR (*DÉambulation et Mouvement ARtificiel*) qui se situe au LIRMM (*Laboratoire D'Informatique de Robotique et de Microélectronique de Montpellier)* et qui travaille en commun avec le CNRS (*Centre National de la Recherche Scientifique*) et les universités de Montpellier 2 et 1. Les validations expérimentales ont été réalisées sur le robot HOAP-3 (*Humanoïd for Open Architecture Platform 3*) de l'entreprise Fujitsu que nous présentons plus en détail dans le chapitre 3.8.

Chapitre 2

Génération de mouvements

Sommaire

2.1 Planification

Qu'ils soient industriels ou d'aide à la personne, les robots humanoïdes sont conçus de manière à avoir des capacités proches de celles des êtres humains qui leur permettent d'évoluer dans des environnements dans lesquels peuvent se trouver des objets fixes ou mobiles et dont le sol peut être irrégulier (escalier, pente, accidenté, ...). Afin d'accomplir leurs tâches, ils doivent être capables de planifier leurs déplacements et leurs mouvements dans cet environnement tout en prenant en compte les obstacles et en conservant leur équilibre.

Les robots mobiles ou manipulateurs considèrent les obstacles comme des régions interdites de l'environnement, grâce à leurs capacités motrices, les robots humanoïdes peuvent les appréhender différemment, ils peuvent les enjamber ou s'en servir comme point d'appui.

La planification des déplacements est un axe de recherche où l'on peut retrouver différents thèmes : la planification de chemins, la planification de mouvements et la planification de séquences (de mouvements ou de points d'appuis).

Nous définissons la *planification de chemin* comme le processus qui permet à un robot humanoïde de se déplacer d'un point de départ vers un point d'arrivée, tout en évitant les obstacles, qu'ils soient mobiles ou fixes. Le chemin calculé définira le déplacement global du robot comme le déplacement d'un seul corps, il sera continu et défini dans un espace de dimension 3 (deux positions, une orientation) pour un environnement plan, ou dans un espace de dimension 6 (3 positions, 3 orientations) pour un environnement 3D.

La *planification de mouvement* est définie comme le processus qui détermine, de manière continue, les grandeurs articulaires qui permettent au robot d'effectuer un mouvement qui sera défini dans un espace à n dimensions, (où n est le nombre d'articulations actives du robot). La planification de mouvement doit prendre en compte les mêmes contraintes que la planification de chemin, telles que la non-collision avec les obstacles, mais aussi d'autres types de contraintes qui traduisent les limites physiques du robot, comme les couples maximums, les butées articulaires, les auto-collisions, l'équilibre, ...

La *planification de séquences* n'entre pas dans ces deux catégories. Contrairement à la planification de chemins ou de mouvements, la planification de séquences ne définit pas le déplacement de manière continue mais plutôt par des points de passage du robot qui devront être reliés par des mouvements.

Pour pouvoir naviguer dans son environnement, le robot a besoin de planifier le chemin sous la forme d'une séquence de postures (ou de points d'appuis) entre sa position courante et sa position finale ainsi que les mouvements qui lui permettront de rallier ces postures. Dans cette section nous allons donc présenter les principales méthodes qui existent pour la planification de chemins, de mouvements ou de séquences.

2.1.1 Planification de chemin

Définition On peut définir un chemin comme étant l'évolution dans l'espace (2D ou 3D), d'un point (ou de la projection d'un point) du robot en fonction du temps. Ce point (ou cette projection) est définie comme étant la configuration du robot à un instant t. Lavalle définit le problème de planification de chemin de la manière suivante [LaValle 06] soit :

- un environnement $W \subseteq \mathbb{R}^3$
- des régions d'obstacle $O \subset W$
- un robot R défini dans W avec m degrés de libertés
- l'espace des configurations C (ou *C-space* en anglais) défini comme l'ensemble des configurations possibles applicables au robot.
- L'image du robot R dans une configuration q se note $R(q)$.
- A partir de C on obtient :
  - $C_{free} = \{q \in C | R(q) \cap O = \varnothing\}$
  - $C_{obs} = C \setminus C_{free}$
- une configuration initiale et finale : $(q_i, q_f) \in C^2_{free}$
- un algorithme de planification de chemin doit calculer un chemin continu $\tau : [0, 1] \rightarrow C_{free}$ tel que $\tau(0) = q_i$ et $\tau(1) = q_f$

Les principaux travaux sur la planification de chemin sont destinés aux robots mobiles dont la lus grande partie (robots mobiles à roues du type voiture) a la particularité d'être des systèmes non holonomes, ce qui leur impose de se déplacer toujours selon la tangente à leur axe principal. Les robots humanoïdes sont des systèmes holonomes car ils ont la possibilité d'effectuer des mouvements dans des directions autres que celles définies par le torse (par exemple : des pas de côté). Cependant, Arechavaleta montre que la marche humaine reste non holonome car on peut observer que la direction instantanée du corps est tangente à la trajectoire réalisée (ce qui est dû à certaines restrictions mécaniques, anatomiques, etc. du corps au moment de la marche) [Arechavaleta-Servin 07].

Dans la plupart des cas, pour relier une position initiale à une position finale, il existe un grand nombre de chemins possibles. Arechavaleta [Arechavaleta 08] montre que l'être humain aurait tendance à minimiser la variation d'orientation de son torse durant ses

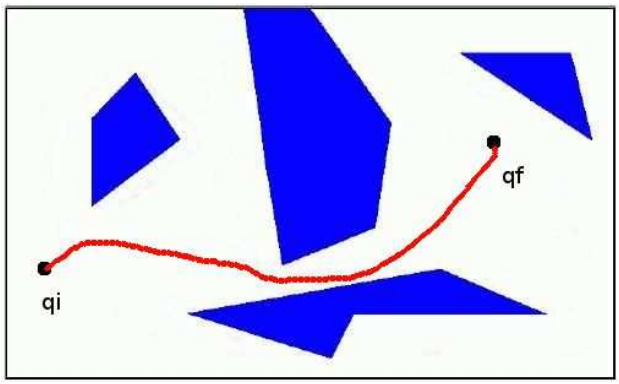


FIG. 2.1 – Exemple d'un espace des configurations avec la position initiale q_i et la position finale q_f, les zones d'obstacles C_{obs} en bleu et le chemin τ en rouge

déplacements. Mettler met en avant le fait que l'homme essaierait de diminuer la durée du déplacement lorsqu'il gère les déplacements d'un véhicule tel qu'un hélicoptère miniature [Mettler 08].

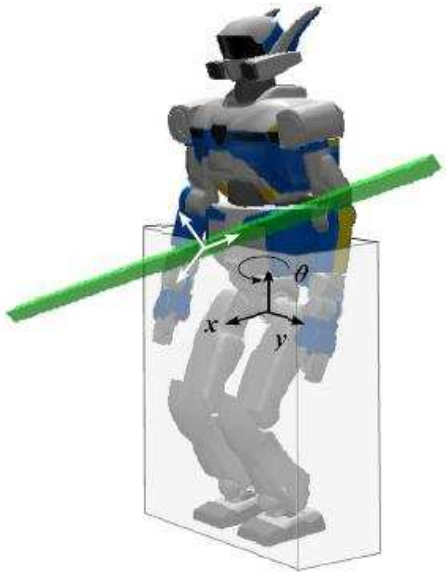


FIG. 2.2 – Représentation du Robot HRP-2 avec les membres inférieurs englobés dans un volume [Yoshida 05].

Pour considérer le robot humanoïde comme un robot mobile, Pettré [Pettré 03] contraint les membres inférieurs d'un personnage numérique dans un cylindre et planifie le chemin de ce cylindre dans l'environnement et déplace les membres supérieurs de façon à éviter la collision avec les obstacles sans se soucier de l'équilibre qui est confié à un autre processus : le *Pattern Generator* de Kajita [Kajita 02] . Yoshida reprend cette idée pour la génération des déplacements du robot HRP-2 [Kaneko 04] portant une barre entre ses mains : il englobe les membres inférieurs dans un parallélépipède rectangle (cf. Figure 2.2) dont il calcule la trajectoire, puis s'assure que les membres supérieurs et la barre n'entrent pas en collision avec l'environnement [Yoshida 05].

En résumé, la planification de chemin d'un système anthropomorphe (être humain, robot humanoïde, avatar virtuel) est semblable à celle d'un robot mobile, elle peut donc

utiliser toutes méthodes dédiées aux déplacements des robots mobiles. Cependant, le critère qui définit le meilleur chemin peut être différent et une prise en compte de l'équilibre peut s'ajouter, sauf si elle est gérée par un processus annexe. Nous détaillerons, donc, les deux principaux types de méthodes utilisées pour la planification de chemin des robots mobiles à savoir les méthodes combinatoires qui décomposent l'espace libre et les méthodes d'échantillonnage qui le discrétisent.

Méthodes combinatoires Les méthodes combinatoires décomposent l'espace libre C_{free} en plusieurs sous-espaces convexes (toutes les configurations d'un même sous-espace peuvent être reliées entre elles). La planification de chemin cherche des configurations à la frontière des sous-espaces, la convexité des sous-espaces permet de relier les configurations qui se trouvent sur différentes frontières (cf. Figure 2.3). Dans son livre, Latombe présente différentes méthodes pour la décomposition en sous-espaces. La Figure 2.3 montre une décomposition verticale de l'espace des configurations [Jean-Claude Latombe 91].

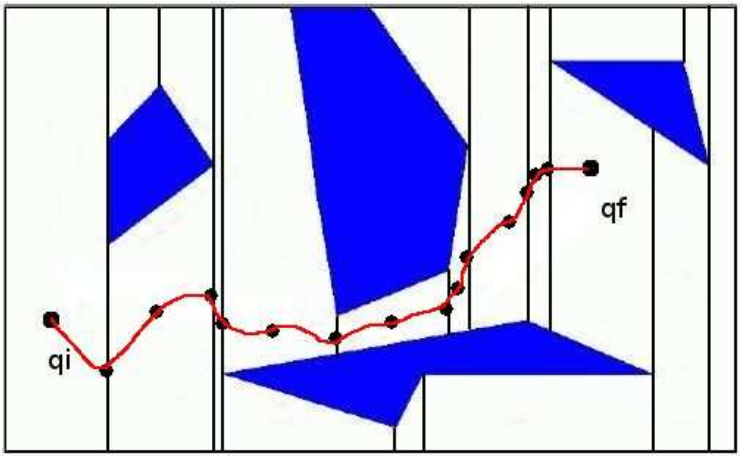


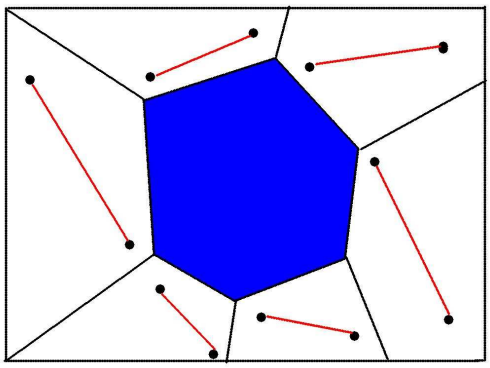
FIG. 2.3 – Décomposition de l'espace libre en sous-espaces

Elles ont l'avantage de pouvoir trouver toute solution existante, cependant elles ne peuvent s'appliquer qu'à des environnements structurés avec des obstacles de forme polyédrique dans l'espace des configurations. Les obstacles comportant des parties circulaires ne permettent pas une décomposition de tout l'espace en sous-espaces convexes (cf. Figure 2.4), pour utiliser les méthodes combinatoires il faut négliger certaines parties de l'espace libre. Ces parties peuvent contenir le ou les seuls chemins possibles, dans ce cas l'algorithme ne trouvera pas de solution malgré leur existence.

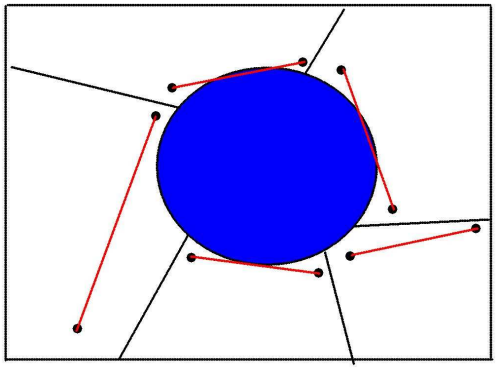
Méthodes d'échantillonnage Les méthodes d'échantillonnage ne calculent pas de manière explicite l'espace libre, mais le discrétisent de manière à obtenir un ensemble de configurations appartenant à l'espace libre. Cet ensemble constitue une carte (*roadmap*) où les configurations sont représentées par des nœuds qui sont reliés par des chemins locaux : les arêtes.

Les principales méthodes de cette classe sont la PRM *(Probabilistic Roadmap)* et le RRT (*Rapidly exploring Random Tree*).

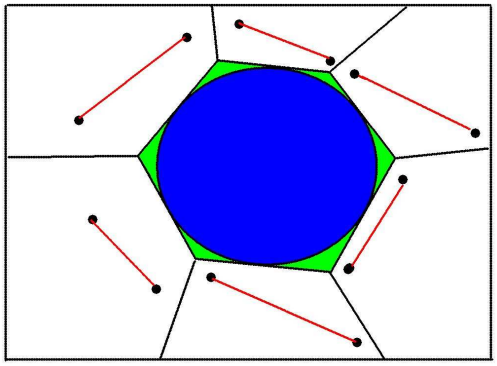
***Probabilistic Roadmap Method* - PRM** Les Méthodes de Cartographie Probabilistes (*Probabilistic Roadmap Method* : PRM) [Kavraki 96] sont souvent utilisées



(a) obstacle polyédrique



(b) obstacle circulaire : toutes les configurations d'un sous-espace ne peuvent pas être reliées



(c) obstacle circulaire : les parties de l'espace libre en vert sont négligées

FIG. 2.4 – Exemple d'obstacles et décomposition de l'espace

pour la planification de chemin pour les robots mobiles et se décomposent en 4 phases [Sanchez-Lopez 03] :

- la phase d'apprentissage : L'algorithme discrétise l'espace libre C_{free} afin d'obtenir une carte. Différentes méthodes existent en fonction de la nature du problème et de la forme de l'espace libre [Sanchez-Lopez 03].
- la phase de recherche locale : un planificateur local essaie de relier ces nœuds par des arêtes le long desquelles les contraintes sont respectées.
- la phase de recherche globale : un planificateur global essaie de trouver un chemin composé d'arêtes reliant la position initiale à la position finale.
- la phase de lissage : le caractère probabiliste de l'algorithme produit des chemins longs et irréguliers, la phase de lissage permet de les réduire et de les lisser.

Lozano applique la méthode PRM aux robots manipulateurs en calculant la carte dans le domaine articulaire [Lozano-Perez 83]. Il existe plusieurs variantes de cette méthode, ainsi Bohlin diminue le nombre d'appel au détecteur de collision en utilisant la méthode *Lazy PRM* [Bohlin 00].

***Rapidly-Exploring Random Trees* - RRT** Les méthodes rapides d'exploration d'arbre aléatoire (*Rapidly-Explornig Random Trees* - RRT) ont été présentées en 1998 par LaValle [Lavalle 98] pour traiter les problèmes ayant des contraintes non holonomes, mais

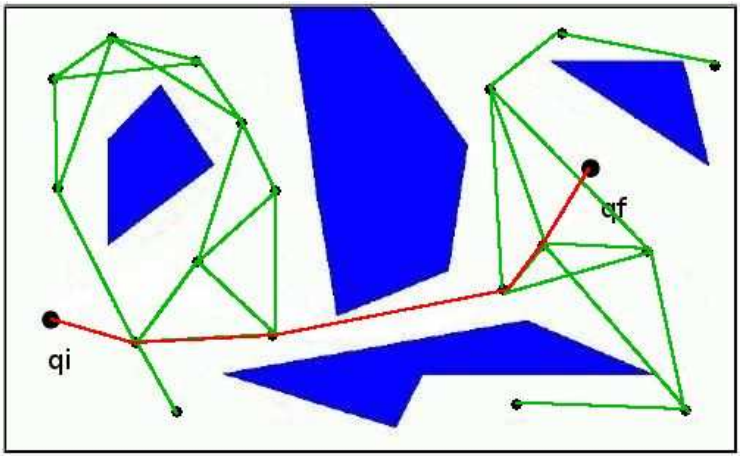


FIG. 2.5 – Discrétisation de l'espace libre
Les nœuds sont représentés par des points, les arêtes par des segments verts et le chemin en rouge

se révèlent également efficaces pour les problèmes holonomes. Ces méthodes sont basées sur une construction incrémentale d'arbre de recherche afin de couvrir uniformément et rapidement l'espace de recherche [LaValle 00] (cf. Figure 2.6). La position initiale du robot constitue la base de l'arbre de recherche, l'algorithme y ajoute des branches en reliant des configurations, tirées au hasard, au point de l'arbre le plus proche. De cette manière l'arbre se développe jusqu'à ce que la position finale puisse y être ajoutée.

Dans le cas de systèmes avec beaucoup de degrés de liberté, il est possible de construire deux arbres, le premier ayant comme base la position initiale et le second la position finale, de les développer jusqu'à ce qu'un arbre soit visible par l'autre ce qui permet de les connecter [Hsu 97, Kuffner 00].

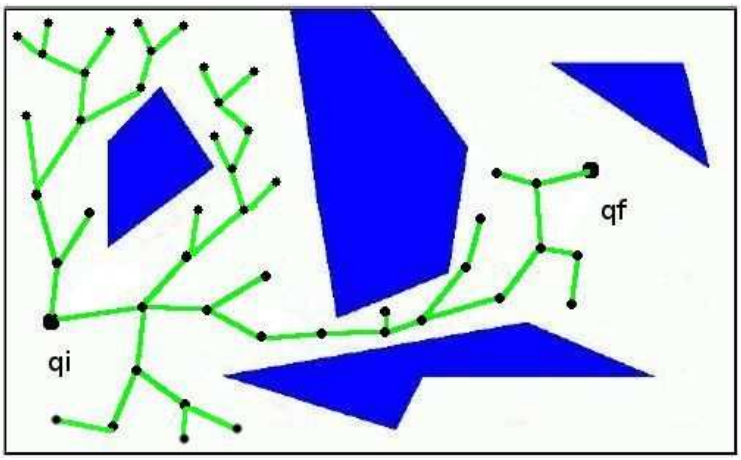


FIG. 2.6 – Rapidly-Explornig Random Trees

Une version de l'algorithme RRT a également été développé pour traiter les cas où l'état du système n'est pas connu de façon précise [Pepy 09].

2.1.2 Planification de mouvement

Il est donc possible de déterminer le chemin que doit suivre le robot, mais comment trouver l'évolution des valeurs angulaires des articulations de telle manière à générer des mouvements qui vont permettre d'effectuer le déplacement ? Les méthodes précédentes n'établissent un chemin qu'en prenant en compte des contraintes géométriques. Or, un mouvement doit également respecter un certain nombre de contraintes dynamiques qui ne sont pas facilement prises en compte par les méthodes de planification de chemin.

Ces mouvements peuvent être générés par des processus d'optimisation hors-ligne ce qui permet de considérer des modèles complets ou complexes où l'environnement est supposé connu et fixe à une application. Dans ce cas, on utilise un algorithme d'optimisation afin de trouver l'évolution des trajectoires articulaires optimales qui va satisfaire un ensemble de contraintes et minimiser un critère (cf. Équation 2.1). Ces trajectoires sont généralement suivies grâce à un simple contrôleur local (PID, etc.).

$$\begin{array}{rl} \text{trouver} & q(t) \\ \text{qui minimise} & F(q(t)) \\ \text{tel que} & \forall i, \forall t \in [0,T] \quad g_i(q(t)) \leq 0 \\ \text{et} & \forall j \quad h_j(q(t)) = 0 \end{array} \tag{2.1}$$

Le problème 2.1 est un problème de programmation infinie car la continuité des valeurs articulaires à trouver et des contraintes à respecter peut être assimilée à une infinité de valeurs. En calculant l'évolution articulaire $q(t) = f(X,t)$, à partir d'un ensemble fini de paramètres $X \in \mathbb{R}^n$ on transforme le problème 2.1 en un problème de programmation semi infinie (SIP) [Hettich 93], présenté comme ceci :

$$\begin{array}{rl} \text{trouver} & X \\ \text{qui minimise} & F(X,t) \\ \text{tel que} & \forall i, \forall t \in [0,T] \quad g_i(X,t) \leq 0 \\ \text{et} & \forall j \quad h_j(X) = 0 \end{array} \tag{2.2}$$

Le problème est dit semi infini car seules les contraintes restent continues et assimilables à une infinité de valeurs alors que la paramétrisation restreint la recherche à un espace à n dimensions.

Nous présentons plus en détail cette formulation dans la section 4.2.1.

Les méthodes basées sur un processus d'optimisation sont principalement utilisées dans deux cas :

- pour des mouvements répétitifs, la production d'un mouvement optimal minimisant un critère comme l'énergie permet d'augmenter l'autonomie du robot.
- pour des mouvements complexes, l'optimisation permet de gérer des systèmes ayant un grand nombre de degrés de liberté avec un grand nombre de contraintes (non-collision, équilibre, couples maximum,...).

Sur la base d'un processus d'optimisation, Miossec a implémenté un algorithme d'optimisation de mouvement qui calcule le modèle dynamique et sa dérivée par rapport aux paramètres d'optimisation de façon analytique, afin d'obtenir des mouvements en simple support, comme un mouvement de coup de pied [Miossec 06] et en double support tels que le lancer d'une balle ou le lever d'une barre [Miossec 08] (les notions de simple et double support sont définies dans la section 3.4.4). De la même façon, Suleiman utilise les groupes de Lie pour le calcul du modèle et de sa dérivée par rapport aux paramètres d'optimisation afin d'augmenter la fluidité et la stabilité de mouvements pré-calculés [Suleiman 08, Suleiman 07]. Kanehiro obtient un mouvement le plus rapide

possible dans un environnement très contraint en utilisant une méthode en deux étapes. La première étape consiste à trouver un mouvement statique qui respecte les contraintes de non collision (avec l'environnement et entre les corps du robot), de butée articulaire et d'équilibre statique. La deuxième étape conserve l'évolution des trajectoires des articulations et optimise la durée du mouvement pour satisfaire les limites concernant les vitesses articulaires et l'équilibre dynamique [Kanehiro 08].

Les méthodes d'optimisation sont aussi utilisées pour les problèmes comportant des discontinuités du modèle dynamique tels que les mouvements avec impacts pour lesquels le mouvement généré est décomposé en trois phases : la posture et la vitesse instantanées au moment de l'impact, le mouvement avant impact et le mouvement après impact. De cette manière il est possible d'obtenir un mouvement de planter de clous [Tsujita 08a, Tsujita 08b] ou de casser de planche à la manière d'un karatéka [Konno 08]. Dans la catégorie des mouvements avec impacts nous pouvons inclure des mouvements où le robot va appliquer une force impulsionnelle sur un objet comme le lever d'objet lourd à la manière d'un haltérophile [Arisumi 07, Arisumi 08].

2.1.3 Planification de séquences

La planification de séquence est différente de la planification de chemin et de la planification de mouvements. Elle ne génère pas le déplacement de manière continue mais spécifie des points de passage (point d'appui ou posture) pour rejoindre la position finale. Elle est généralement employée pour des déplacements plus complexes. La séquence peut être composée de postures spécifiées par le planificateur ou de postures prédéfinies appartenant à une base de données.

Postures prédéfinies : À partir d'un ensemble de mouvements de pas de base (cf. Figure 2.7), calculés hors-ligne et assurant l'intégrité et l'équilibre du robot, il est possible de planifier le déplacement du robot entre sa position courante et une position finale [Kuffner 01]. Pour trouver l'enchaînement de pas qui permettra de rejoindre l'objectif, la plupart des algorithmes se basent sur la construction d'un arbre où chaque nœud représente une configuration (position et orientation) du robot et chaque branche correspond à un pas qui amène le robot à une nouvelle configuration (cf. Figure 2.8). Les configurations en collision avec l'environnement sont rejetées de l'arbre (en rouge sur la Figure 2.8) et la progression de la branche en aval n'est pas étudiée. L'algorithme doit donc déployer cet arbre jusqu'à ce qu'une branche mène à la posture finale.

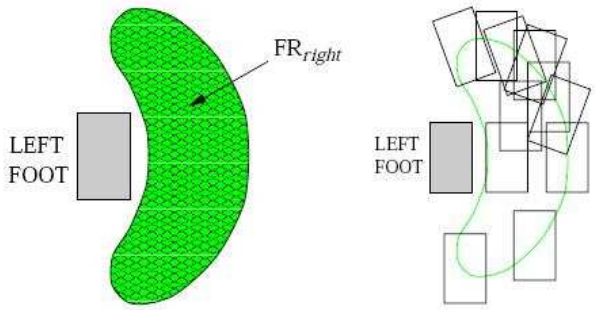


FIG. 2.7 – Représentation des pas possibles utilisés dans [Kuffner 01]

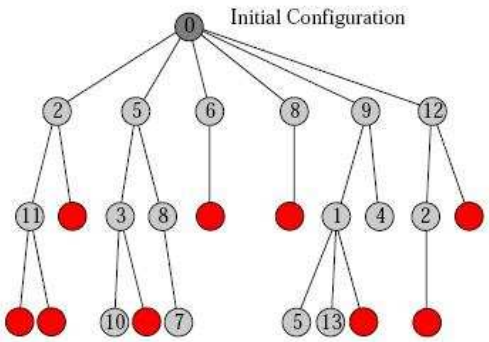


FIG. 2.8 – Représentation de l'arbre (image venant de [Kuffner 01])

Les travaux de Yagi [Yagi 99] prévoient un ensemble de pas possibles avec différentes hauteurs et longueurs, ceux de Chestnutt [Chestnutt 05] prennent en compte l'état (positions et vitesses articulaires) courant pour déterminer les futurs placements du pied afin d'obtenir un déplacement fluide.

Cependant, la navigation du robot à partir d'un ensemble de pas prédéfinis génère un arbre de très grande dimension et donc long à calculer. Pour obtenir une solution rapidement, Chestnutt prend en compte un critère heuristique afin de favoriser les chemins en ligne droite avec un minimum de pas [Chestnutt 03] alors que Ayaz propose d'adopter une méthode comportementale [Ayaz 06], en limitant à trois le nombre de choix face à un obstacle : passer à droite, passer à gauche, passer par dessus, ce qui permet de ne considérer que certaines branches de l'arbre et donc de diminuer le temps de calcul.

Ces méthodes permettent donc de planifier rapidement un déplacement à partir de quelques mouvements de base dans un environnement peu encombré. Pour un environnement très encombré, il faudrait disposer d'un grand nombre de mouvements, ce qui augmenterait la taille de l'arbre et donc le temps de calcul, dans ces conditions il faut pouvoir déterminer une séquence de postures adaptées à chaque étape du déplacement.

Postures non définies : Dans des environnements très encombrés où un ensemble de pas prédéfinis pourrait ne pas convenir, le robot ne doit plus considérer les obstacles comme des zones interdites, mais plutôt comme des zones de points d'appuis potentiels, la Figure 2.9 montre le robot HRP-2 [Kaneko 04] qui prend appui sur la table devant lui pour se lever de la chaise dans le but d'atteindre un objet placé sur la table.

Pour résoudre ce problème Hauser propose une méthode de planification de mouvements où le robot prend appui sur son environnement avec certaines parties de son corps définies à l'avance [Hauser 05]. En revanche, les travaux d'Escande considèrent des points d'appuis entre toutes les parties du robot et son environnement pour générer une succession de postures [Escande 06]. Les travaux de Bouyarmane génèrent un mouvement statique entre deux de ces postures en contraignant la recherche de la solution optimale dans un espace guide qui est proche de l'espace de contact [Bouyarmane 09].

En résumé, dans cette section nous avons vu qu'il existe différents types de planifications (chemin, mouvement, séquence) adaptés à différents type d'applications. Toutes ces méthodes ont pour but d'amener le robot d'un point (ou d'une configuration) initial à un point (ou configuration) final, en s'assurant de la sécurité du robot, et notamment de son équilibre.

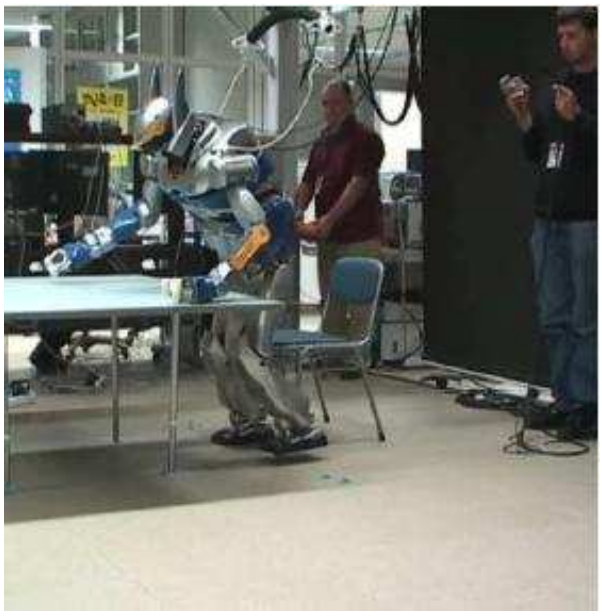


FIG. 2.9 – Représentation d'un mouvement de lever de chaise avec appui sur la table [Escande 08].

2.2 Sûreté de fonctionnement

Garantir la sûreté de fonctionnement du robot permet de garantir la sécurité des personnes l'entourant, en effet un dépassement des limites du robot peut provoquer sa chute et entraîner la chute ou la blessure de personnes à proximité.

La sûreté de fonctionnement est une notion importante qui peut se retrouver à différents niveaux de la conception à l'utilisation en passant par la fabrication. La partie commande est basée sur des systèmes électroniques qui doivent être robustes face aux perturbations qu'elles soient de source électromagnétique ou autre. Le logiciel embarqué doit également être fiable pour pouvoir assumer ses fonctions convenablement. De plus, l'utilisateur doit disposer d'un modèle du robot qui doit être suffisamment précis et correct, afin de pouvoir anticiper toute mauvaise utilisation. Enfin, l'utilisateur doit être en mesure de générer des mouvements qui ne nuisent pas au robot. Dans ce but, les travaux présentés dans cette thèse supposent que tous les concepts précédents sont maîtrisés et proposent des méthodes pour la génération de mouvements assurant l'intégrité et l'équilibre des robots humanoïdes.

Les trajectoires de référence sont censées garantir la sécurité du robot, les méthodes d'optimisation présentées précédemment supposent que la sécurité du robot sera assurée si un certain nombre de contraintes sur le mouvement sont garanties. Par exemple, Bekris présente une méthode de planification de chemin où la trajectoire de référence est recalculée à chaque pas d'échantillonnage en faisant l'hypothèse qu'entre deux échantillons un obstacle ne peut pas entrer en collision puis se trouver à bonne distance à l'instant d'échantillonnage suivant [Bekris 07]. Certains auteurs vérifient a posteriori que le mouvement généré assure la sécurité du robot comme Merlet qui valide le chemin d'un robot parallèle à l'aide de méthodes basées sur l'Analyse par Intervalle [Merlet 01].

Dans la section 4.6, nous nous attarderons sur notre méthode, également basée sur l'analyse par intervalle qui s'assure, lors de la planification, de la validité de ces contraintes sur toute la durée du mouvement et donc de l'intégrité du robot.

2.3 Adaptation

Dans la section 2.2, nous avons exposé quelques méthodes qui permettent d'assurer la sécurité des robots pour des mouvements devant se dérouler dans un cadre d'utilisation défini. Cependant, il est impossible de prévoir toutes les situations auxquelles le robot sera confronté. Il faut donc le doter de la capacité d'adaptation afin qu'il puisse calculer rapidement des mouvements assurant sa sécurité et qui soient appropriés à la situation.

Nous définissons l'adaptation comme la génération (ou l'exécution) d'un nouveau mouvement en se basant sur un mouvement déjà existant. Elle peut se faire par replanification d'un nouveau chemin ou mouvement ou alors en effectuant un mouvement de type réflexe.

2.3.1 Chemin

La replanification en temps réel est un prérequis dans l'exécution de mouvement dans un environnement changeant. Dans le cadre des robots mobiles, Fraichard calcule un chemin optimal sans obstacles mouvants puis introduit l'espace des états-temps, qui est l'espace des configurations auquel on ajoute le temps. En définissant la position des obstacles à chaque instant, il peut définir l'espace des états-temps libre et y trouver un chemin [Fraichard 99].

Pour la replanification de chemin, on peut utiliser les chemins homotopes qui sont des chemins que l'on peut déformer de manière continue sans changer leur situation finale, on peut donc facilement changer de chemin pour un chemin homotope qui serait plus adéquat à la situation. Pour les robots manipulateurs, Choi propose de déterminer si les chemins sont homotopes de manière récursive en partant de la première articulation jusqu' à la dernière en tenant compte des obstacles [Choi 07]. Brock utilise également les trajectoires homotopes pour un robot manipulateur mobile, il calcule une sphère autour de certains points du robot, s'assure qu'elles n'entrent pas en collisions avec les obstacles afin d'établir un tunnel autour de la trajectoire optimale. La replanification se fait par un algorithme *potential field based* qui va déformer la trajectoire optimale dans ce tunnel en fonction des collisions possibles [Brock 00].

2.3.2 Mouvements

L'adaptation de mouvements est aussi un axe de recherche important. Dans la plupart des temps, il a pour but d'exécuter sur un robot humanoïde des mouvements venant de la capture de mouvements humains. La principale difficulté est dûe au fait que la mécanique du robot est différente de celle de l'homme, ce qui ne lui permet pas de reproduire certains mouvements humains (par exemple les mouvements faisant intervenir un appui sur la pointe des pieds). Nakaoko propose donc de partir de mouvements qui forment les primitives des futurs mouvements (cf Figure 2.10) [Nakaoka 03, Nakaoka 04, Nakaoka 07].

L'adaptation de mouvement peut aussi s'utiliser pour améliorer un mouvement déjà existant. Yanase, part d'un mouvement de coup de pied et l'améliore grâce à un algorithme génétique [Yanase 06]. Il optimise des postures intermédiaires, puis simule le mouvement avec le logiciel openHRP, l'utilisateur donne une note qui sera prise en compte pour proposer un autre mouvement jusqu'à obtenir le mouvement optimal souhaité.

FIG. 2.10 – Le robot HRP-2 danse le 'Jongara-bushi' [Nakaoka 07]

2.3.3 Réflexes

Dans certains cas, il n'est pas possible d'adapter le mouvement, il faut pouvoir effectuer un autre mouvement de type réflexe, qui va permettre de conserver l'équilibre du robot. Renner propose une méthode pour détecter les instabilités lors d'une marche omnidirectionnelle [Renner 06]. En fonction de l'amplitude de l'instabilité ainsi détectée, il choisit un des deux réflexes afin de stabiliser le robot :

- pour des instabilités faibles, la marche est ralentie
- pour des instabilités plus importantes, il provoque un arrêt réflexe de la marche.

De la même manière, pour des perturbations importantes, Zaier interpole la position courante du robot HOAP-3 jusqu'à une posture d'équilibre pré-calculée [Zaier 08, Zaier 06].

2.4 Conclusion

Dans ce chapitre, nous avons montré différentes méthodes qui existent pour la planification de chemins, de mouvements ou de séquences. Les méthodes de planification de chemins sont souvent basées sur des méthodes existantes pour les robots mobiles. La planification de mouvements repose sur des processus d'optimisation. Aussi utilisée pour des robots manipulateurs, cette méthode nécessite une prise en compte de contraintes supplémentaires liées à l'équilibre, pour les robots humanoïdes. La planification de séquences de mouvements pré-calculés repose sur la construction d'un arbre de recherche. Cependant, l'ajout de certaines heuristiques permettent de diminuer la taille de l'arbre ainsi que le temps de calcul.

Toutes ces méthodes de planification doivent générer des déplacements qui assurent la sûreté de fonctionnement du robot. Dans cette thèse, nous supposons que le modèle fourni par le constructeur ne comporte pas d'erreur et que la partie logicielle et matérielle ne présentent aucun risque. Pour obtenir un mouvement sûr, le mouvement devra donc vérifier un certain nombre de contraintes que nous détaillons dans le chapitre 3.

La planification de mouvements est souvent utilisée hors-ligne pour générer des mouvements optimaux qui sont adaptés à un environnement précis. De tels mouvements, mis

en œuvre dans un environnement différent de celui prévu, peuvent apparaître comme sous-optimaux ou même dangereux. Il est donc nécessaire de générer rapidement un mouvement adapté à l'environnement courant avant qu'il n'évolue. Dans le chapitre 5, nous présentons notre méthode de replanification rapide.

Chapitre 3

Modélisation

Sommaire

3.1 Introduction

Dans cette thèse, nous traitons la génération de mouvement pour les robots humanoïdes. Dans le chapitre 4, nous mettons en évidence la nécessité de paramétrer le mouvement afin de faciliter le processus d'optimisation nécessaire à la planification de mouvements. Nous allons donc, tout d'abord, présenter différentes méthodes de paramétrisation des trajectoires articulaires.

Afin de considérer une stratégie pour le mouvement (le plus rapide, qui consomme le moins, ...) et d'assurer la sûreté de fonctionnement du robot, les processus de génération de mouvements ont besoin d'un modèle précis du robot. Ce modèle permet de calculer un ensemble de grandeurs liées aux actionneurs ou aux caractéristiques du mouvement souhaité, comme les couples articulaires nécessaires, le non-glissement du pied de support, la conservation de l'équilibre ou l'évitement d'auto-collision.

3.2 Trajectoires Articulaires

Dans nos travaux, nous cherchons à générer des mouvements optimaux par le calcul de trajectoires articulaires optimales. Les trajectoires articulaires peuvent se décomposer en un ensemble de valeurs angulaires à chaque instant du mouvement qu'il nous faut déterminer. Nous verrons dans le chapitre 4 que ceci constitue un problème de programmation infinie. Une solution consiste donc à établir des équations paramétrées pour le calcul de ces trajectoires, il suffira donc de trouver quels sont les paramètres optimaux. Nous présentons différentes manière d'obtenir des trajectoires paramétrisées.

3.2.1 Oscillateurs

Certains chercheurs se sont penchés sur la locomotion des êtres vivants (salamandre, chats, ...), pour en comprendre les mécanismes et essayer de les retrouver chez l'humain [Duysens 98]. Il en résulte que la locomotion peut être modélisée comme une tâche rythmique. Certains chercheurs ont développé des outils mathématiques pouvant traduire ces comportements cycliques : les *Central Pattern Generators* (CPG) qui sont assimilés à des circuits neuronaux capables de produire des mouvements coordonnées pour plusieurs articulations [Ijspeert 08]. En ce qui concerne le mouvement humain, et spécifiquement le mouvement des doigts, tâche essentiellement neuronale sans couplage mécanique, l'ensemble d'oscillateurs non linéaires couplés HKB (Haken, Kelso, Bunz) est le plus adapté [Haken 85].

Ces CPG sont des oscillateurs qui peuvent être couplés ou non et qui vont générer localement les trajectoires articulaires. Le plus simple et le plus connu est l'oscillateur de Van der Pol [Dutra 03] qui modélise la coordination de mouvements rythmiques entre plusieurs articulations et dont l'équation est donnée par :

$$\ddot{q}_i - \epsilon_i \left[1 - p_i(q_i - q_{i0})^2\right] \dot{q}_i + \Omega_i(q_i - q_{i0}) - \sum_{j=1}^{n} c_{i,j}(\dot{q}_i - \dot{q}_j) = 0 \tag{3.1}$$

avec ϵ_i, p_i, Ω_i des paramètres de l'oscillateur et $c_{i,j}$ des coefficients de couplage entre les oscillateurs.

Les oscillateurs (ou CPG dans le cas de la locomotion) sont capables de reproduire les trajectoires articulaires observées, même s'ils n'ont pas de lien avec les éléments physiologiques ou biomécaniques. Cependant l'obtention de l'expression analytique des variables

articulaires à chaque instant $q(t)$, en fonction des paramètres du mouvement n'est pas triviale.

3.2.2 Splines cubiques

Les splines cubiques permettent de générer une trajectoire passant par un ensemble de valeurs [Khalil 02]. Le principe est d'interpoler les points en calculant de façon globale les accélérations au niveau de ces points de passage afin d'assurer la continuité en vitesse et position. En connaissant la valeur des accélérations il est possible de les intégrer pour obtenir la vitesse et la position.

Les accélérations sont définies à partir de l'équation :

$$M\ddot{q}_j = N_j \tag{3.2}$$

Avec la matrice M et le vecteur N_j qui sont donnés page 358 du livre de Khalil et Dombre [Khalil 02] et $\ddot{q}_j$ le vecteur qui contient les accélérations aux temps de passage sauf pour le premier et le dernier instant. La matrice M est composée de fonction ne dépendant que de la valeur des instants de passage, elle est donc commune pour toutes les articulations alors que le vecteur N_j dépend de la valeur angulaire souhaitée aux instants de passage, il est donc différent pour chaque articulation. En résolvant le système linéaire 3.2, il est possible de trouver toutes les accélérations, ce qui permettra d'avoir les trajectoires articulaires.

Cette méthode permet de relier rapidement un ensemble de configurations, malheureusement l'obtention d'une équation analytique de la trajectoire en fonction des points de passage n'est pas une chose facile. Nous présentons donc la méthode des B-splines.

3.2.3 B-splines

Les fonctions de type B-splines, comme représentées sur la Figure 3.1, constituent un ensemble de fonctions de base $B_j(t)$ qui seront combinées pour obtenir les trajectoires articulaires. A partir d'un même ensemble de B-splines nous pouvons calculer les différentes trajectoires articulaires à travers un ensemble de paramètres $p_{i,j}$ de la manière suivante :

$$q_i(t) = \sum_{j=0}^{N_s} p_{i,j} \times B_j(t) \tag{3.3}$$

Les vitesses et accélérations articulaires étant obtenues par dérivation de l'équation 3.3.

$$\begin{aligned} \dot{q}_i(t) &= \textstyle\sum_{j=0}^{N_s} p_{i,j} \times \dot{B}_j(t) \\ \ddot{q}_i(t) &= \textstyle\sum_{j=0}^{N_s} p_{i,j} \times \ddot{B}_j(t) \end{aligned} \tag{3.4}$$

Les B-splines se calculent de manière récursive [de Boor 78] comme indiqué dans l'annexe B. Dans notre cas nous utilisons des fonctions B-splines normalisées qui vérifient l'équation suivante :

$$\sum_{j=0}^{N_s} B_j(t) = 1 \tag{3.5}$$

Comme nous pouvons le voir dans l'équation 3.3, l'expression des variables articulaires est linéaire par rapport aux paramètres, ce qui donne l'avantage d'être facilement dérivable.

$$\frac{\partial q_i(t)}{\partial p_{i,j}} = B_j(t) \tag{3.6}$$

Dans nos travaux, nous souhaitons des mouvements qui commencent et finissent avec une vitesse et une accélération nulles (voir Annexe B), nous considérons donc les courbes représentées par la Figure 3.1.

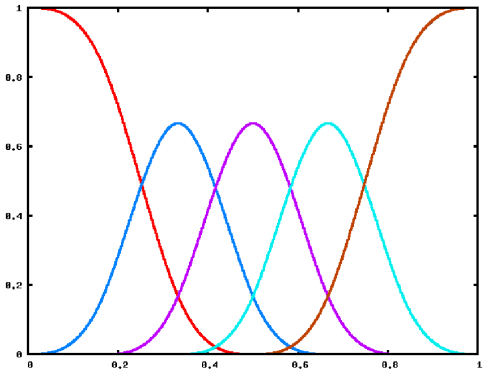


FIG. 3.1 – Représentation des fonctions de base

Les B-splines représentent donc une très bonne solution pour l'optimisation de mouvements, dans la suite de ce document nous les utiliserons pour caractériser les trajectoires articulaires.

3.3 Actionneurs

3.3.1 Utilisation pour les robots humanoïdes

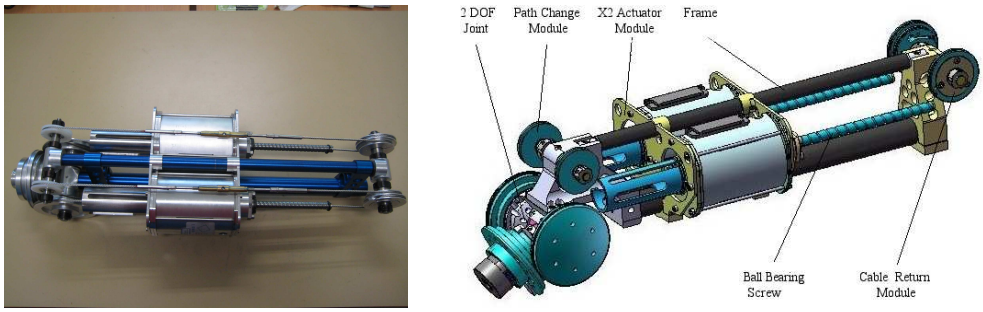
Un robot humanoïde est un système polyarticulé anthropomorphe constitué d'un ensemble de corps reliés les uns aux autres à travers des articulations qui doivent être actionnées pour générer un mouvement. Le choix des actionneurs a un effet sur le coût et les performances du robot. De cette manière, les robots les plus abordables sont souvent équipés de servomoteurs pour des applications grand public comme les jouets et les prototypes destinés à la recherche sont souvent équipés de moteurs développés spécifiquement.

Les servomoteurs sont des systèmes tout-en-un qui comportent une partie mécanique, un moteur à courant continu et un système de commande, ils sont capables de rallier et de maintenir un angle donné. Même s'ils représentent une solution peu onéreuse et simple à mettre en œuvre, les servomoteurs utilisés sont souvent des produits déjà existants et qui n'ont pas été développés spécifiquement pour le robot auxquels ils sont intégrés, ce qui fait que leurs performances ne seront pas optimales.

Pour obtenir un robot ayant de bonnes performances l'utilisation de servomoteurs n'est pas la solution à retenir. La plupart des robots humanoïdes présents dans les laboratoires de recherche (HRP-2 [Kaneko 04], HRP-3 [Kaneko 08], Nao [Gouaillier 08], HOAP-3, Asimo, ...) sont actionnés par des moteurs à courant continu associés à un réducteur et à différents capteurs (position, vitesse, courant, etc.). Cet ensemble de composants donne

accès à toutes les grandeurs utiles pour l'utilisateur. Ils sont souvent développés spécifiquement pour le robot auxquels ils vont appartenir. Les moteurs choisis sont souvent des moteurs à aimants permanents, ce qui permet de limiter la taille et de faciliter la commande des moteurs.

Le choix du type d'actionneur peut aussi avoir un impact sur la sécurité. En effet, lors d'un impact avec l'environnement, Khatib [Khatib 95] et Zinn [Zinn 04] ont relié la force de collision d'un robot avec son inertie. Il apparaît qu'une inertie plus petite entraînera une force plus petite en cas de collision. Pour assurer la sécurité des personnes se trouvant dans l'environnement des robots, notamment ceux de grandes tailles, de récents travaux se sont intéressés à réduire l'inertie des corps du robot et de leurs actionneurs. Olaru présente un nouveau système d'actionneur par câble [Olaru 09] (voir Figure 3.2) alors que Shin [Shin 08] propose un actionneur hybride, où un actionneur pneumatique prend en charge la composante basse fréquence du mouvement et un actionneur électrique (moteur à courant continu) la composante haute fréquence, de la même manière que chez l'homme, le muscle contient des fibres rapides et des fibres lentes qui se répartissent l'effort à produire.



(a) premier prototype (b) modélisation sous Solid Works

FIG. 3.2 – Représentation de l'actionneur par câble présenté dans [Olaru 09]

Dans cette thèse, nous nous intéresserons uniquement aux robots actionnés par des moteurs à courant continu, nous détaillerons donc leur fonctionnement et leurs limites.

3.3.2 Modélisation

La motorisation est une part essentielle des robots humanoïdes. C'est elle qui permet au robot d'exécuter des mouvements. Il faut donc tenir compte de ces propriétés pour la génération de mouvements optimaux. Dans l'annexe A, nous détaillons le modèle électrique ainsi que le bilan de puissance des moteurs à courant continu pour définir la puissance électrique consommée par un moteur par :

$$P_{elec} = \frac{R}{(K_{em}K_r)^2}(\Gamma + K_v \times \Omega + K_s)^2 + \Gamma \times \Omega \tag{3.7}$$

Où, K_{em}, K_r, K_v et K_s sont des paramètres liés à l'ensemble moteur et réducteur, tandis que Γ et Ω sont, respectivement, le couple et la vitesse de rotation à la sortie du réducteur. N'ayant pas la valeur de ces caractéristiques pour le robot HOAP-3, dans l'Annexe A, nous considérons le rapport : $\frac{R}{(K_{em}K_r)^2}$ égal à 0.1. Dans le chapitre sur la planification (Chap. 4), quand nous choisirons le critère $F(t)$ comme étant l'énergie, nous

essaierons de minimiser la grandeur suivante :

$$F(t) = \int_0^T P_{elec}.dt = \int_0^T 0.1\Gamma^2 + \Gamma \times \Omega.dt \tag{3.8}$$

L'énergie consommée pendant un mouvement s'exprime en Joules.

3.3.3 Caractéristiques des moteurs

Lors de la conception d'un robot, le choix des moteurs et de leur commande est une étape importante. Les performances seront limitées par la commande électrique (tension et courant d'alimentation) et aussi par les limites intrinsèques du moteur. Le choix du moteur se fait à partir des valeurs nominales et maximales fournies par le constructeur. Les valeurs nominales sont définies à partir de l'énergie thermique qu'il est capable de dissiper en fonctionnement nominal, pour lequel le moteur est conçu. Cette énergie provient des différentes pertes (Joule et mécanique). Les valeurs nominales ne doivent pas être confondues avec les valeurs maximales qui sont fixées par la tension d'alimentation maximale admissible en ce qui concerne la vitesse maximale de rotation du moteur (cf. Équation A.1), et le courant maximum admissible pour les couples (cf. Équation A.3).

Les valeurs nominales, sont toujours inférieures aux valeurs maximales. Un dépassement des valeurs maximales provoque la destruction quasi instantanée du moteur alors qu'un dépassement des valeurs nominales provoquera une augmentation de la température interne du moteur et une diminution de sa durée de vie mais peut être tolérée si un temps de repos permet de ramener la température à une valeur acceptable.

En résumé les moteurs à courant continu vont nous imposer une limite sur les couples articulaires ainsi que sur la vitesse de rotation. Les butées articulaires ne sont pas liées aux capacités du moteur, elles peuvent être liées au réducteur mais permettent surtout d'éviter la collision entre les corps reliés à une même articulation.

3.4 Modèle dynamique inverse

A partir d'un ensemble de paramètres, nous sommes donc en mesure de calculer les trajectoires articulaires $q(t)$ ainsi que leurs dérivées première et seconde $\dot{q}(t), \ddot{q}(t)$ ce qui nous permet de générer un mouvement pour le robot. Mais quel est l'effet de ce mouvement sur les couples articulaires ? Va-t-il pouvoir être réalisé en respectant les limites électriques et mécaniques des moteurs. Pour répondre à cette question il faut pouvoir calculer les couples en fonction des trajectoires articulaires : c'est le but du modèle dynamique inverse.

Le modèle dynamique inverse (ou modèle dynamique) est la relation qui lie les couples Γ aux efforts extérieurs f_e et aux positions q, vitesses $\dot{q}$ et accélérations $\ddot{q}$ articulaires par l'équation :

$$\Gamma = f(q, \dot{q}, \ddot{q}, f_e) \tag{3.9}$$

Le modèle dynamique inverse est présenté dans le livre de Khalil et Dombre [Khalil 02], dans la suite de cette section, nous reprendrons brièvement comment définir la structure mécanique du robot, afin de pouvoir expliquer le calcul du modèle dynamique inverse à travers les deux méthodes les plus connues : le formalisme de Lagrange et l'algorithme de Newton-Euler. Puis nous présenterons les spécificités d'un mouvement en simple support et en double support.

3.4.1 Structure du robot

Du point de vue de la modélisation, les robots humanoïdes peuvent être vus comme une chaîne arborescente dont l'origine est un corps de référence qui peut être fixe ou mobile (cf .Figure 3.3). Dans notre cas, nous faisons l'hypothèse que le pied droit est fixe au sol avec une position et une orientation connue, ce qui nous permet de le considérer comme corps de référence. Évidement lors de la génération de mouvements nous devrons imposer certaines contraintes qui permettront de valider cette hypothèse, comme le non-glissement et le non-basculement du pied droit.

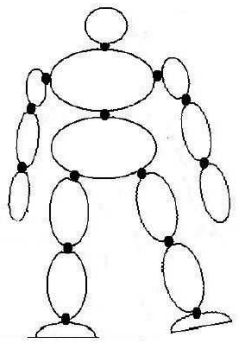


FIG. 3.3 – Le robot humanoïde est une chaîne arborescente

Le robot est donc vu comme une succession de corps rigides, reliés entre eux par des articulations. Chaque corps comporte un repère, pour définir la structure du robot il faut définir comment passer d'un corps à l'autre. Pour cela nous utilisons les paramètres de Denavit-Hartenberg qui permettront de définir l'équation qui va relier un repère au repère suivant.

Définition des repères

Pour pouvoir utiliser les paramètres de Denavit-Hartenberg, il faut placer les repères en prenant certaines précautions. On définit un repère R_j, fixé au corps C_j de telle façon que :

- l'axe z_j est porté par l'axe de l'articulation j,
- l'axe x_j est porté par la perpendiculaire commune aux axes z_j et z_{j+1}.

Si les repères ne peuvent pas être définis directement de cette façon, il est possible d'ajouter un corps virtuel (de masse et inertie nulles) dont le repère permettra le passage entre ces deux corps. Quant à lui, Khalil considère deux paramètres supplémentaires pour définir le changement de repère [Khalil 02].

Paramètres de Denavit-Hartenberg

Le passage du repère R_{j-1} au repère R_j s'exprime à partir des quatre paramètres suivants représentés sur la Figure 3.4 :

- α_j : angle entre les axes z_{j-1} et z_j, correspondant à une rotation autour de l'axe x_{j-1}.
- d_j : distance entre z_{j-1} et z_j selon l'axe x_{j-1}.
- θ_j : angle entre les axes x_{j-1} et x_j, correspondant à une rotation autour de l'axe x_j. Si l'articulation est motorisée ce paramètre est directement lié à l'angle du moteur.

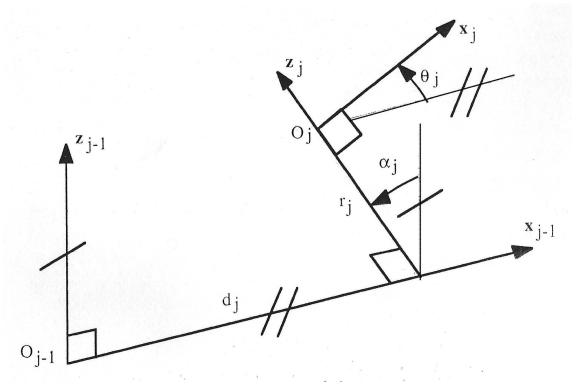


FIG. 3.4 – Représentation des paramètres géométriques de Denavit-Hartenberg [Khalil 02]

– r_j : distance entre x_{j-1} et x_j selon l'axe z_j.

La matrice de transformation définissant le repère R_j dans le repère R_{j-1} est l'équation :

$$^{j-1}T_j = Rot(x,\alpha_j)Trans(x,d_j)Rot(z,\theta_j)Trans(z,r_j) = \begin{bmatrix} \cos\theta_j & -\sin\theta_j & 0 & d_j \\ \cos\alpha_j \sin\theta_j & \cos\alpha_j \cos\theta_j & -\sin\alpha_j & -r_j \sin\alpha_j \\ \sin\alpha_j \sin\theta_j & \sin\alpha_j \cos\theta_j & \cos\alpha_j & r_j \cos\alpha_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

3.4.2 Lagrange

Le formalisme de Lagrange permet d'obtenir le modèle dynamique inverse d'un robot à n degrés de liberté sous la forme de l'équation suivante :

$$\Gamma = A(q)\ddot{q} + C(q,\dot{q})\dot{q} + Q(q) \tag{3.11}$$

Où :

– $A(q)$ de taille $n \times n$ est la matrice inertie du robot, elle est symétrique, définie positive et dépend des positions articulaires et des paramètres mécaniques du robot,
– $C(q,\dot{q})$ de taille $n \times n$ est le vecteur qui représente l'effet des efforts de Coriolis et des forces centrifuges,
– $Q(q)$ de taille n est le vecteur qui traduit l'effet de la gravité.

En définissant le vecteur x tel que

$$x = [q_1, q_2, \ldots, q_n, x_{ref}, y_{ref}, z_{ref}, \theta_{ref}, \alpha_{ref}, \gamma_{ref}]^T \tag{3.12}$$

Il est possible d'obtenir une expression étendue du modèle dynamique :

$$\begin{bmatrix} \Gamma \\ F_{ref} \\ M_{ref} \end{bmatrix} = M(x)\ddot{x} + H(x,\dot{x}) \tag{3.13}$$

Avec

– $x_{ref}, y_{ref}, z_{ref}, \theta_{ref}, \alpha_{ref}, \gamma_{ref}$ la position et orientation du corps de référence exprimé dans un repère absolu,
– F_{ref} de taille 3 est la force exercée par le corps de référence sur l'environnement,
– M_{ref} de taille 3 est le moment exercé par le corps de référence sur l'environnement,
– $M(x)$ de taille $(n+6) \times (n+6)$ la matrice d'inertie,

– $H(x, \dot{x})$ de taille $(n+6)$ le vecteur représentant l'effet de Coriolis, des forces centrifuges et de la gravité.

Le formalisme de l'équation 3.13 permet d'obtenir les efforts au niveau du corps de référence.

Dans la suite de cette thèse, nous considérons comme corps de référence, un corps virtuel situé au niveau du sol auquel est relié l'articulation du pied droit, cette équation nous permettra donc de connaître les couples articulaires ainsi que les efforts exercés par le robot au niveau du sol.

3.4.3 Newton-Euler

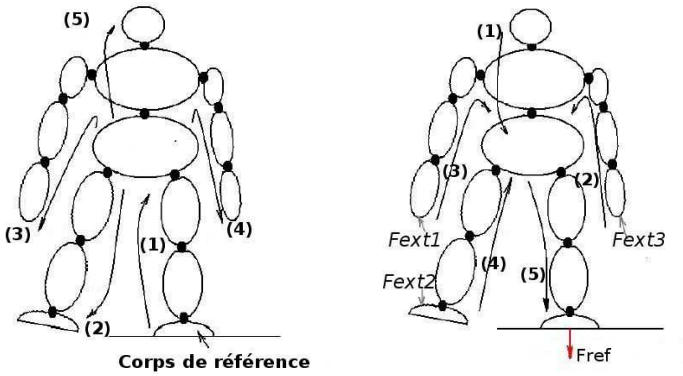
Contrairement au formalisme de Lagrange, le formalisme de Newton-Euler ne nécessite pas le calcul de matrices ou de vecteurs de grandes dimensions. Il permet donc de diminuer le nombre de calcul pour des robots ayant un grand nombre de degrés de liberté [Khalil 02]. Ce modèle dynamique inverse est obtenu grâce à un calcul récursif en deux étapes.

Récurrence avant

La première récurrence consiste à partir du corps de référence, pour calculer la position, vitesse et accélération du corps suivant (cf. Figure 3.5(a)). Ainsi, de proche en proche il est possible de connaître la position, vitesse et accélération de tous les corps du robot en fonction des données articulaires $q, \dot{q}, \ddot{q}$ ainsi que les efforts provoqués par le mouvement de chaque corps.

Récurrence arrière

La seconde récurrence consiste à répercuter les efforts dûs aux accélérations des extrémités vers le corps de référence (cf. Figure 3.5(b)), ce qui permet d'obtenir les efforts (forces et moments) qui s'appliquent sur chaque articulation, donc, les couples articulaires (projection des efforts sur l'axe de rotation) ainsi que l'effort exercé par le robot sur le corps de référence.



(a) récurrence avant (b) récurrence arrière

FIG. 3.5 – Principe de l'algorithme de Newton-Euler pour le calcul du modèle dynamique

3.4.4 Simple support

Nous définissons un mouvement en simple support comme un mouvement pendant lequel une seule partie du robot est en contact permanent avec l'environnement, par exemple pendant un mouvement de coup de pied (voir Figure 3.6).

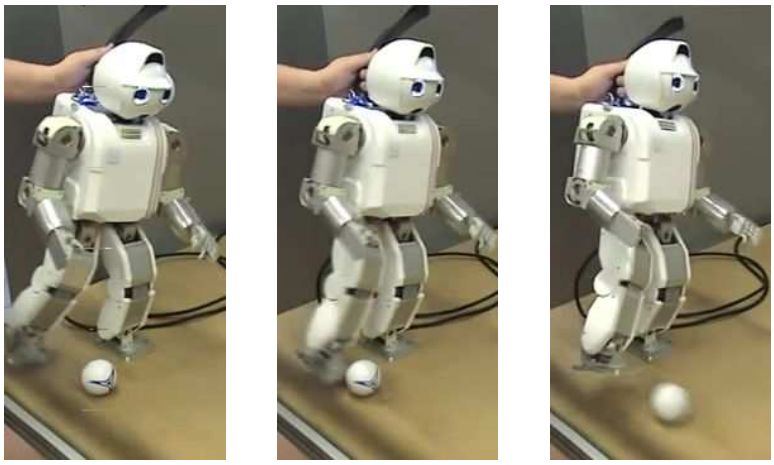


FIG. 3.6 – Représentation d'un mouvement en simple support. Seul le pied gauche est en contact avec le sol.

Le modèle dynamique d'un mouvement en simple support est obtenu par l'équation 3.13, en connaissant les trajectoires articulaires $(q(t), \dot{q}(t), \ddot{q}(t))$ il est possible de calculer l'évolution unique des couples et des efforts au niveau du sol.

3.4.5 Double support

Nous définissons un mouvement en double support, comme étant un mouvement pendant lequel deux parties du robot sont en contact permanent avec l'environnement, par exemple pendant le transfert de poids d'un pied vers l'autre pendant un mouvement de marche (voir Figure 3.7).

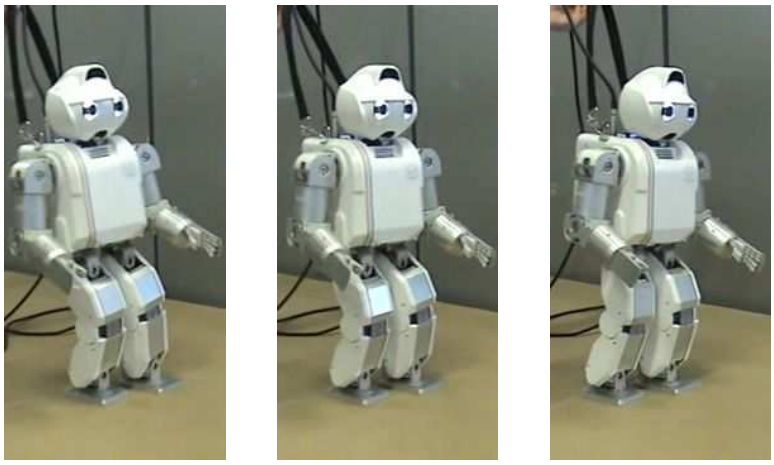


FIG. 3.7 – Représentation d'un mouvement en double support. Le poids du corps est transféré d'un pied vers l'autre.

Le modèle dynamique d'un mouvement en double support est donné par l'équation suivante :

$$\begin{bmatrix} \Gamma \\ F_{ref} \\ M_{ref} \end{bmatrix} = M(x)\ddot{x} + H(x,\dot{x}) + J^T \begin{bmatrix} F_c \\ M_c \end{bmatrix} \tag{3.14}$$

Où F_c et M_c représentent l'effort de contact au niveau du pied gauche, si on considère le pied droit comme étant le corps de référence, et J la matrice jacobienne du robot.

Cette équation contient une indétermination, en effet les couples articulaires et les efforts au niveau du corps de référence dépendent des efforts de contact. Pour un même mouvement les couples et efforts peuvent être différents en fonction des efforts internes liés aux forces de contact.

Pour lever l'indétermination, il faut poser des hypothèses sur les efforts de contacts. Dans le simulateur HuManS [Wieber 06], les efforts de contact sont souhaités les plus proches de la normale à la surface de contact. Dans [Miossec 08], Miossec préfère avoir des efforts de contact qui minimisent l'énergie électrique consommée. Dans ces deux cas, les forces de contact sont obtenues par la résolution d'un problème d'optimisation sous contrainte. Cependant cette solution est lourde en temps de calcul car elle nécessite la résolution d'un problème d'optimisation à chaque pas d'échantillonnage [Lengagne 06].

Dans notre cas, nous allons ajouter quatre hypothèses qui vont permettre d'avoir une solution analytique des forces de contacts.

- les forces de contact s'exercent uniquement selon l'axe normal au sol (de haut en bas suivant l'axe vertical),
- les forces de contact en double support sont proportionnelles aux forces de contact en simple support,
- Le ZMP (voir section 3.6.2 calculé en simple support est identique à celui calculé en double support,
- les forces de contact sur le pied de support sont pondérées par la distance les séparant du ZMP.

Ces hypothèses ainsi que les calculs des efforts sont présentés plus en détail dans l'annexe C, ce qui nous permet d'obtenir le modèle suivant :

$$\begin{bmatrix} \Gamma \\ F_{ref} \\ M_{ref} \\ F_c \\ M_c \end{bmatrix} = M'(x)\ddot{x} + H'(x,\dot{x}) \tag{3.15}$$

3.5 Glissement

Le glissement est aussi un élément important pour le robot. Comment garantir que le mouvement ne provoquera pas un déplacement non désiré dû au glissement du robot sur le sol ? Les lois de Coulomb assurent que le glissement a lieu quand les forces de contacts sortent de leur cône de frottement (voir Figure 3.8).

Par conséquent pour assurer le non-glissement, les efforts de contact doivent respecter l'équation suivante :

$$\sqrt{F_X^2 + F_Y^2} - \sigma F_Z \leq 0 \tag{3.16}$$

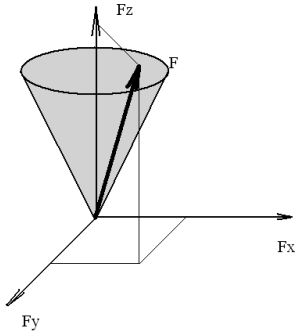


FIG. 3.8 – Représentation du cône de frottement.

Ce cône de frottement est défini par le paramètre σ qui dépend des matériaux en contact. L'équation 3.16 définit donc le non-glissement du robot, afin de s'affranchir des erreurs numériques pour le calcul de la dérivée ($(\sqrt{u})' = \frac{u'}{2\sqrt{u}}$) quand la force tangentielle est petite $F_x^2 + F_y^2 \simeq 0$ nous préférons exprimer la contrainte de glissement sous la forme équivalente suivante :

$$frott(t) = F_X^2 + F_Y^2 - \sigma^2 F_Z^2 \leq 0 \tag{3.17}$$

3.6 Équilibre

La plupart des robots industriels ont leur base solidement fixée au sol, contrairement aux robots humanoïdes qui ne sont fixés à aucune partie de l'environnement. Les contacts entre le robot et l'environnement se produisent généralement entre la semelle des pieds et le sol. Une détérioration totale ou partielle de ces contacts rendrait le robot incontrôlable, il est donc nécessaire de pouvoir maintenir le contact entre les semelles et le sol. Nous présentons différentes méthodes utilisées dans le cas de mouvements quasi-statiques, dynamiques ou discontinus (présence d'impact).

Une notion importante pour la gestion de l'équilibre est le polygone de sustentation qui correspond à l'enveloppe convexe de l'ensemble des points de contact entre le robot et le sol comme le montre la Figure 3.9.

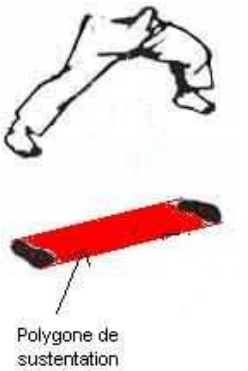


FIG. 3.9 – Représentation du polygone de sustentation

3.6.1 Quasi-statique

Les premiers mouvements effectués par des systèmes anthropomorphes étaient des mouvements quasi-statiques, c'est à dire suffisamment lent pour pouvoir négliger les effets cinématiques et dynamiques du mouvement. Dans ces conditions, le robot conservera son équilibre si la projection, suivant la gravité, de son centre de masse se trouve dans le polygone de sustentation : (cf Figure 3.10)

FIG. 3.10 – En posture statique l'équilibre est conservé si la projection du centre de masse est dans le polygone de sustentation

Le centre de masse peut se calculer directement en fonction des positions, masses et centres de gravité de tous les corps du robot, cependant il existe une méthode qui permet de calculer une chaîne série équivalente dont l'effecteur est le centre de masse du robot humanoïde [Cotton 08], ceci afin d'appliquer toutes les méthodes de commande dédiées aux robots séries à la commande de l'équilibre statique.

3.6.2 Dynamique

Pour des mouvements plus rapides, la projection du centre de masse n'est plus adaptée. La Figure 3.11 montre bien la posture d'un mouvement dynamique où la projection du centre de masse est en dehors du polygone de sustentation sans provoquer le déséquilibre. En 1972 [Vukobratović 72], Vukobratovic caractérise l'équilibre sur un sol plan en définissant le *Zero Moment Point* (ZMP) qui assure l'équilibre du robot s'il se trouve dans le polygone de sustentation.

Le ZMP est défini comme le point du plan où le moment résultant des forces de contact s'annule. Une question divise les roboticiens : Le ZMP peut-il sortir du polygone de sustentation ? Nous montrons que la réponse à cette question dépend de la définition choisie d'un point de vue théorique et du système de mesure des efforts de contact d'un point de vue expérimental, si le robot est en équilibre les deux définitions donnent le même résultat et se confondent avec la projection du centre de masse en quasi-statique.

Le ZMP ne peut pas sortir du polygone de sustentation

En reprenant la définition mathématique originale, Kajita [Kajita 09] assimile le ZMP au centre de pression (cf. Figure 3.12) et prouve qu'il ne peut pas sortir de la zone de sustentation. Le déséquilibre se produit quand le ZMP (ou centre de pression) se trouve à la limite du polygone, quand le ZMP se trouve à l'intérieur du polygone de sustentation le robot conservera son équilibre. Le calcul du ZMP (centre de pression) peut se faire à

FIG. 3.11 – Pour un mouvement dynamique la projection du centre de masse ne caractérise pas l'équilibre

partir de la mesure de N capteurs, mesurant l'effort suivant un seul axe, situés au niveau de la semelle du robot (Figure 3.13). Dans ce cas la position (p_x, p_y) du ZMP se calcule de la manière suivante :

$$p_x = \frac{\sum_{j=1}^{N} p_{jx} f_{jz}}{\sum_{j=1}^{N} f_{jz}} \tag{3.18}$$

$$p_y = \frac{\sum_{j=1}^{N} p_{jy} f_{jz}}{\sum_{j=1}^{N} f_{jz}} \tag{3.19}$$

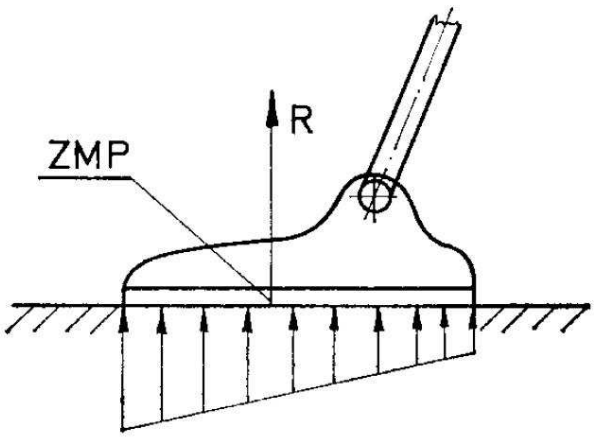


FIG. 3.12 – Définition du Zero Moment Point (ZMP) [Vukobratović 72]

Le ZMP peut sortir du polygone de sustentation

En ne retenant que la définition littérale du ZMP : le point du plan où le moment résultant des forces de contact s'annule, il n'est pas possible d'affirmer que le ZMP doit rester dans le polygone de sustentation. Afin de garder l'association du ZMP comme le

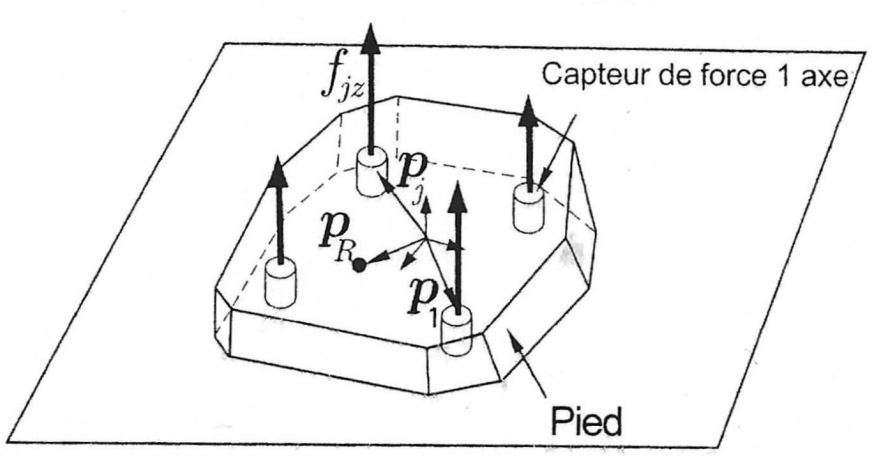


FIG. 3.13 – Définition des variables pour le calcul du ZMP (centre de pression) à partir de plusieurs capteurs de forces un axe

centre de pression, Yin définit la notion de ZMP fictif (*Fictitious Zero Moment Point* : FZMP) qui provient de l'équation traduisant la définition littérale :

$$\vec{M}_{R(p_x,p_y)} = \vec{0} \tag{3.20}$$

Si on dispose des six composantes d'effort en un point du pied (généralement la cheville) situé à une hauteur d au dessus du repère du contact comme présenté sur la Figure 3.14, il est alors possible de calculer la position (p_x, p_y)du ZMP (FZMP) en résolvant l'équation suivante :

$$\vec{M}_{R(p_x,p_y)} = \begin{bmatrix} M_X \\ M_Y \\ M_Z \end{bmatrix} + \begin{bmatrix} F_X \\ F_Y \\ F_Z \end{bmatrix} \wedge \begin{bmatrix} p_x \\ p_y \\ d \end{bmatrix} = \vec{0} \tag{3.21}$$

Où $[M_X, M_Y, M_Z]^T$ et $[F_X, F_Y, F_z Z]^T$ sont les moments et les forces mesurés, ce qui nous permet de calculer la position du ZMP :

$$p_x = \frac{-M_Y + F_X d}{F_Z} \tag{3.22}$$

$$p_y = \frac{M_X + F_Y d}{F_Z} \tag{3.23}$$

Le ZMP (FZMP) peut donc sortir du polygone de sustentation, ce qui peut provoquer le déséquilibre du robot.

Dans les deux cas, pour conserver l'équilibre la position (p_x, p_y) du ZMP doit vérifier ces contraintes :

$$\underline{p_x} \leq p_x(t) \leq \overline{p_x} \tag{3.24}$$

$$\underline{p_y} \leq p_y(t) \leq \overline{p_y} \tag{3.25}$$

$$\tag{3.26}$$

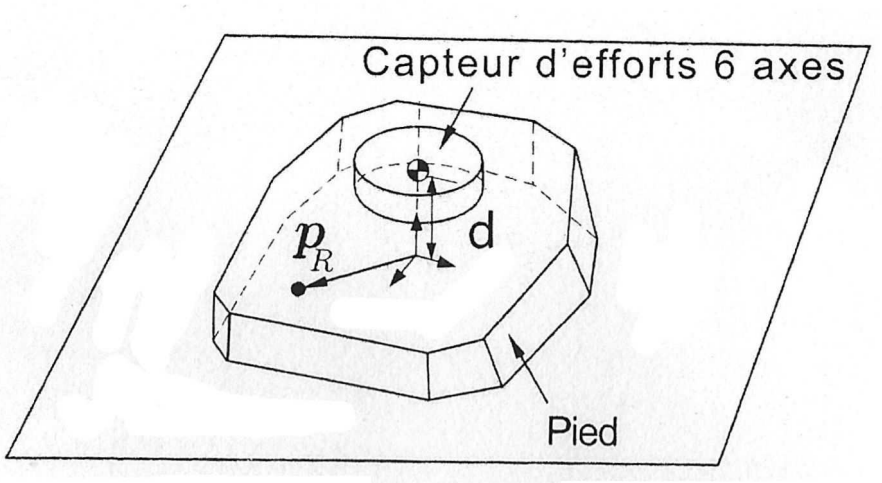


FIG. 3.14 – Définition des variables pour le calcul du ZMP à partir d'un capteur d'effort six axes

3.6.3 Discontinuité

Certaines tâches peuvent amener le robot à effectuer un mouvement au cours duquel une discontinuité se produit. Cette discontinuité est due à l'apparition ou à un changement brutal des efforts extérieurs s'exerçant sur le robot, comme durant un impact avec l'environnement ou pendant le lever d'un objet.

Dans ce dernier cas, Arisumi définit le centre de percussion comme le point de la surface de contact où l'impact produit une force et un moment qui s'annulent mutuellement [Arisumi 07]. Pour assurer l'équilibre du robot, il propose de fixer le centre de percussion au milieu du polygone de sustentation.

3.6.4 Contacts non-coplanaires

Dans le cas où le robot est en contact avec plusieurs objets qui sont sur des plans différents, comme sur la Figure 3.15), l'équilibre ne peut plus être défini par les mêmes méthodes.

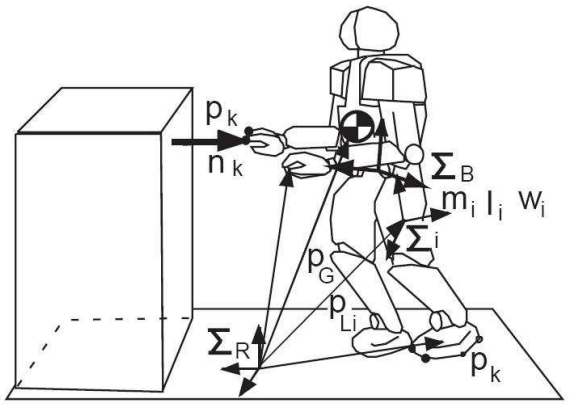


FIG. 3.15 – Exemple de posture où le robot a des contacts sur différents plans [Hirukawa 06]

Harada propose une généralisation du ZMP : le *Generalized Zero Moment Point* (GZMP) [Harada 03], alors que Hirukawa ou Bretl propose un critère pour les mouvements quasi-statiques qui consiste à vérifier si il existe des efforts de contact restant à l'intérieur de leurs cônes de frottements qui permettent de compenser l'effet de la gravité [Hirukawa 06],[Bretl 08].

3.7 Auto-collision

Dans cette thèse, nous ne traitons pas de la génération de mouvement évitant les collisions avec des objets. Cependant vu leur structure et leurs limites articulaires les robots humanoïdes sont sujets à des phénomènes d'auto-collision non souhaités, c'est-à-dire la collision entre deux parties du robot.

Afin d'éviter ce phénomène, il faut donc pouvoir assurer une distance minimale entre les corps du robot. Rusaw utilise des volumes de forme polyédrique pour obtenir une expression de la distance dont le gradient n'était pas continu [Shawn 01]. Pour remédier à cela, Escande propose une méthode de calcul basée sur des volumes englobant qui définissent un corps par un ensemble de sphères et de tores [Escande 08]. La Figure 3.16 montre les volumes englobant les corps du robot HRP-2.

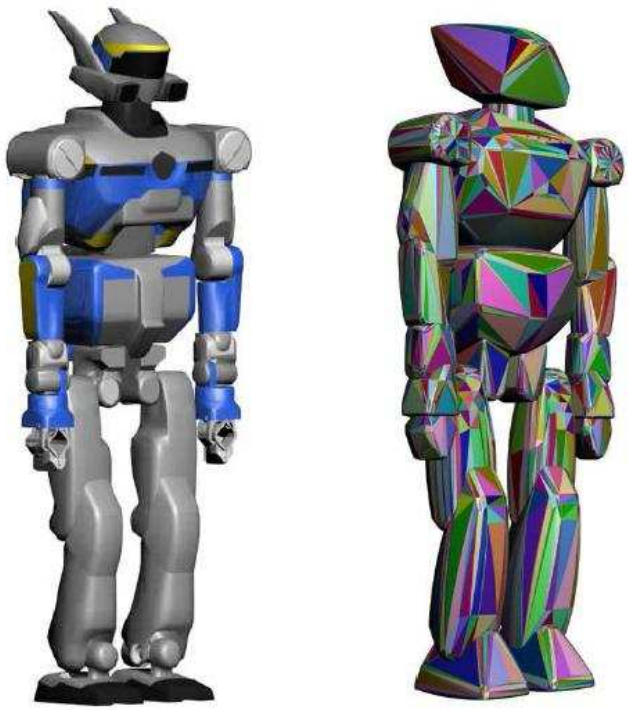


FIG. 3.16 – Le robot HRP-2 et ses volumes englobants [Escande 08]

Dans ce rapport, nous ne nous intéressons qu'à la planification des mouvements pour les membres inférieurs, nous faisons donc l'hypothèse que la seule source d'auto-collision est le pied en mouvement qui pourrait heurter la jambe du pied de support. Nous modélisons donc la distance entre ces corps comme la distance entre les deux cylindres représentés sur la Figure 3.17.

Les centres des cylindres sont fixes par rapport au pied, en connaissant la position du pied on peut donc avoir la position du centre des cylindres O_1 et O_2. La distance d entre

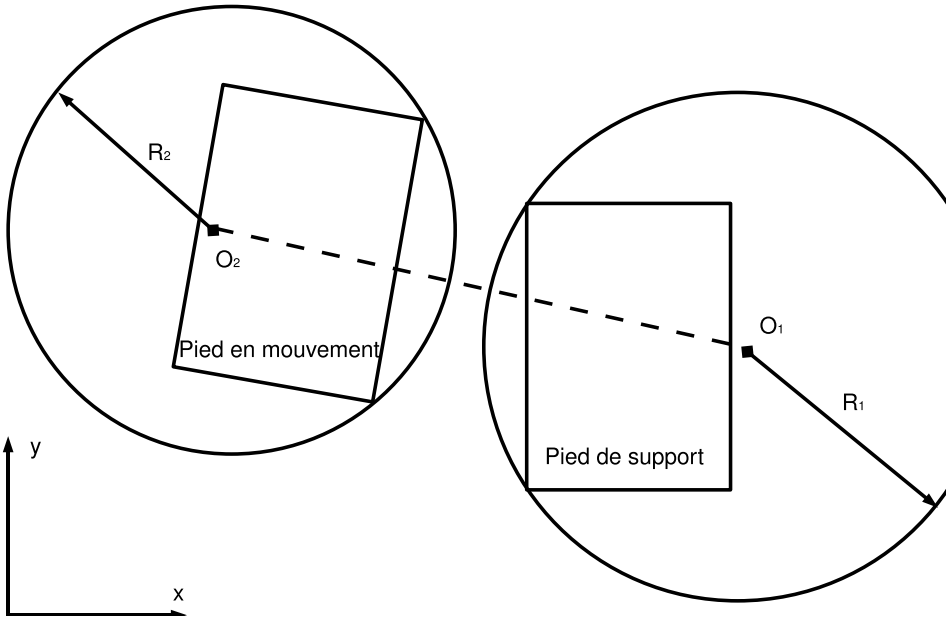


FIG. 3.17 – Les cylindres utilisés pour le calcul de distance afin d'éviter les auto-collisons des membres inférieurs du robot HOAP-3

les deux corps peut donc être calculée comme la distance entre les deux centres moins les rayons :

$$d(t) = \sqrt{(O_{1x} - O_{2x})^2 + (O_{1y} - O_{2y})^2} - R_1 - R_2 \tag{3.27}$$

Pour éviter les collisions il faut que $d(t)$ reste positif. Évidemment ce calcul induit une erreur sur la distance entre les deux corps. En prenant en compte les butées articulaires on peut en déduire que la valeur maximale de cette erreur se produit quand les deux pieds sont côte à côte et parallèles, dans ce cas elle vaut :

$$\epsilon_{max} = R_1 - \sqrt{R_1^2 - \left(\frac{L}{2}\right)^2} + R_2 - \sqrt{R_2^2 - \left(\frac{L}{2}\right)^2} \tag{3.28}$$

Avec L la longueur du pied.

Pour réduire cette erreur il serait intéressant d'augmenter la valeur des deux rayons, cependant cela peut provoquer une intersection entre ces deux cylindres à un endroit éloigné des pieds ce qui produirait une fausse détection de la collision (cf. Figure 3.18).

Pour les mouvements que nous devons générer, nous avons choisi d'imposer $R_1 = R_2 = 0.2m$ ce qui nous donne une erreur maximale de $\epsilon_{max} = 7mm$, et rend impossible la fausse détection de collision pour les mouvements que nous avons à générer.

3.8 Fonctionnement normal du robot

3.8.1 Le robot HOAP-3

Nous avons testé les méthodes présentées dans cette thèse à l'aide du robot HOAP-3 (*Humanoid for Open Architecture Platform 3*) représenté en Figure 3.19, un petit robot

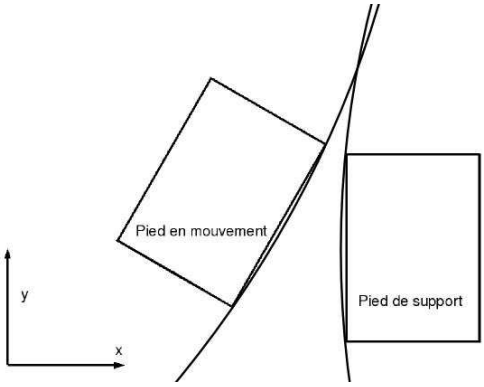


FIG. 3.18 – Les cylindres ayant des rayons importants peuvent s'intersecter alors que les pieds sont suffisamment éloignés

humanoïde fabriqué par l'entreprise Fujitsu et qui a élu domicile au LIRMM depuis janvier 2007.

FIG. 3.19 – Le robot humanoïde HOAP-3 a servi de plate-forme de test et de validation.

Le robot HOAP-3 pèse 8,8 kg et mesure 60 cm de haut, il dispose de 28 articulations motorisées réparties de la manière suivante :

- 6 articulations pour chaque jambe,
- 6 articulations pour chaque bras,
- 1 articulation au niveau du bassin qui lui permet de se pencher en avant,
- 3 articulations au niveau de la tête.

Pour mieux appréhender son environnement le robot possède différent capteurs :

- des capteurs de positions sur les articulations,
- un système de vision composé de deux caméras,
- des capteurs d'effort au niveau de la semelle des pieds (4 par pied),
- un télémètre laser,
- un accéléromètre et un gyromètre qui permettent de connaître l'orientation du bassin.

Il est contrôlé par un PC embarqué avec un système d'exploitation linux temps réel.

3.8.2 Ensemble de contraintes

A partir des propriétés du robot et de la modélisation présentée dans ce chapitre nous pouvons détailler les limites du robot qui devront être respectées pendant l'exécution de tout mouvement. Ces limites constitueront une partie des contraintes du problème d'optimisation présenté dans le chapitre suivant.

Les mouvements devront donc respecter les butées mécaniques, les limites des vitesses articulaires, les couples maximums ainsi que garantir l'équilibre, le non-glissement du pied de support et éviter les auto-collisions, nous obtenons donc l'ensemble suivante :

$$\underline{q} \leq q(t) \leq \overline{q} \tag{3.29}$$
$$\underline{\dot{q}} \leq \dot{q}(t) \leq \overline{\dot{q}} \tag{3.30}$$
$$\underline{\Gamma} \leq \Gamma(t) \leq \overline{\Gamma} \tag{3.31}$$
$$\underline{p_x} \leq p_x(t) \leq \overline{p_x} \tag{3.32}$$
$$\underline{p_y} \leq p_y(t) \leq \overline{p_y} \tag{3.33}$$
$$frott \leq 0 \tag{3.34}$$
$$-d(t) \leq 0 \tag{3.35}$$

L'ensemble de ces contraintes inégalités traduisant les limites du systèmes peuvent être mis sous la forme :

$$g(q, \dot{q}, \ddot{q}) \leq 0 \tag{3.36}$$

3.9 Calcul du modèle en langage C

Nous utilisons le logiciel HuMaNS [Wieber 06] pour générer le modèle dynamique inverse ($M(q)$,$H(q,\dot{q})$,$J(q)$) sous forme de fichiers C/C++, à partir d'une description des paramètres de Denavit-Hartenberg (voir section 3.4.1), des centres de masse et des matrices d'inertie des corps. Nous reprenons la notation des repères donnée par le constructeur (Figure 3.20). Par la suite, nous planifierons des mouvements pour les membres inférieurs, soit 12 articulations. Nous supposons donc les membres supérieurs fixes pendant toute la durée du mouvement.

3.10 Conclusion

Dans ce chapitre, nous avons détaillé comment calculer les limites du robot pour un mouvement défini par un ensemble de paramètres. Ce modèle nous permet d'obtenir une évaluation de l'ensemble des grandeurs qui doivent être contraintes pour garantir la sûreté de fonctionnement et l'équilibre du robot (angles, vitesses couples articulaires, *Zero Moment Point*, cône de frottement et auto-collisions).

Dans les chapitres 4 et 5, nous présentons des méthodes de planification et de replanification rapides qui se basent sur les équations présentées ici pour générer des mouvements sûrs.

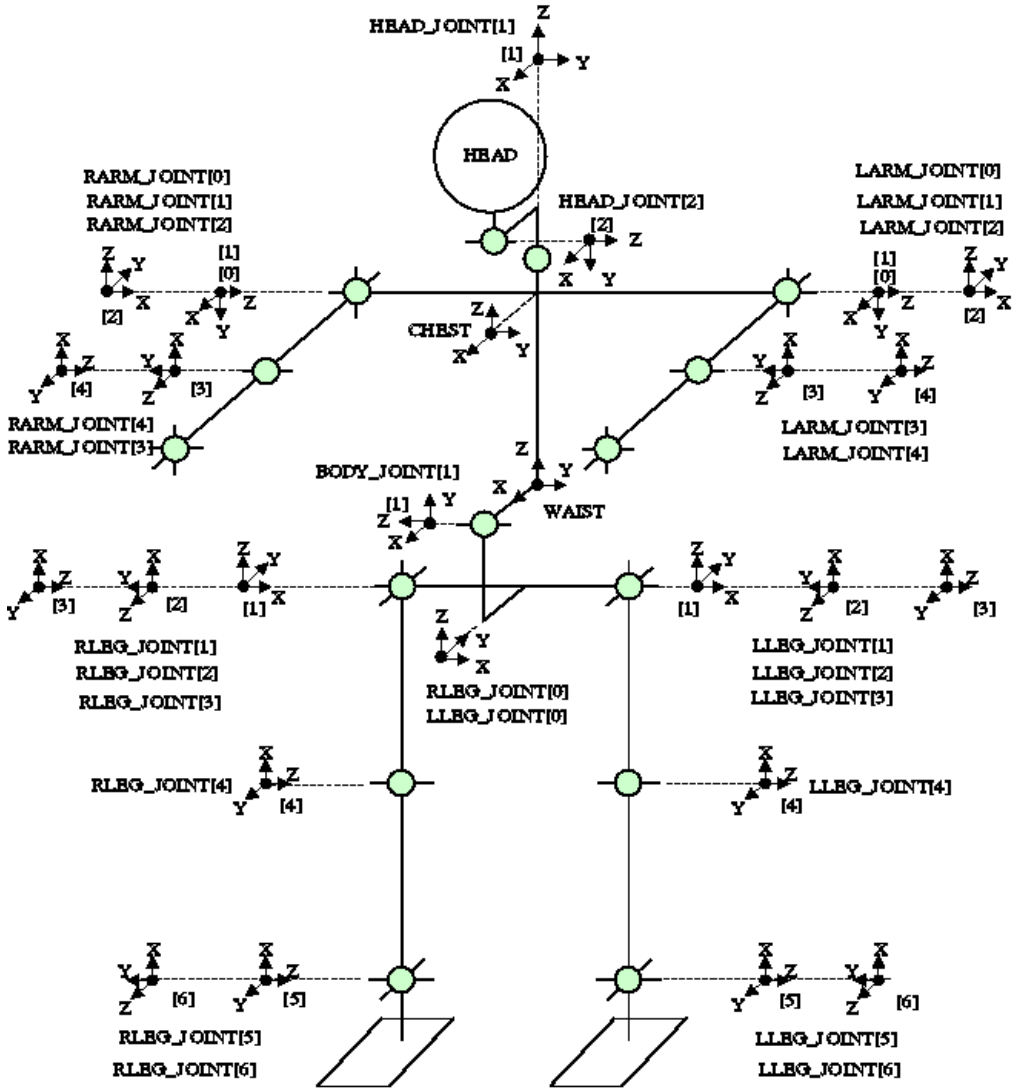


FIG. 3.20 – Représentation des degrés de liberté du robot HOAP-3.

Chapitre 4

Planification de mouvements sûrs

Sommaire

4.1 Introduction

Dans le chapitre 3, nous avons présenté la modélisation des robots humanoïdes. Les modèles obtenus sont non-linéaires et complexes et leur prise en compte pour la génération de mouvements optimaux ne peut se faire de manière triviale sans un algorithme d'optimisation sous contraintes.

Dans ce chapitre, nous présentons le problème d'optimisation (sa fonction de coût ainsi que ses contraintes 'égalité' et 'inégalité' ponctuelle et continue) qui peut se mettre sous la forme d'un problème de programmation semi-infinie (SIP). Ce problème est souvent résolu à travers un processus de discrétisation temporelle qui va considérer les contraintes 'inégalité' continues pour des instants appartenant à une grille temporelle. Bien que cette méthode soit souvent utilisée, nous montrons qu'elle peut s'avérer dangereuse pour le robot ou son équilibre.

Nous proposons donc une méthode de planification qui se base sur une discrétisation garantie utilisant l'Analyse par Intervalles, pour calculer les contraintes 'inégalité' continues sur des intervalles de temps qui couvrent toute la durée du mouvement. De cette façon, nous nous assurons que le mouvement généré ne met pas en danger le robot ou son équilibre.

La planification garantie génère des mouvements sûrs mais est très coûteuse en temps de calcul. Nous avons donc utilisé le concept de la discrétisation garantie pour vérifier les résultats d'une planification utilisant une discrétisation classique dans une méthode itérative d'optimisation hybride.

4.2 Problème d'optimisation

4.2.1 Définition

Dans le chapitre 3, nous avons vu que le robot a des limites qu'il faut prendre en compte pour générer un mouvement. Il doit également établir une stratégie pour son déplacement : Est ce qu'il faut aller le plus vite possible, consommer le moins d'énergie, ou minimiser d'autres critères ? Par conséquent, la planification de mouvements revient à résoudre un problème d'optimisation sous contraintes pour trouver le meilleur ensemble de trajectoires articulaires $q(t)$ qui résout :

$$\begin{array}{rl} \text{minimiser} & F(q(t)) \\ \text{tel que} & \forall i, \forall t \in [0,T] \quad g_i(q(t)) \leq 0 \\ \text{et} & \forall j \quad h_j(q(t_j)) = 0 \end{array} \tag{4.1}$$

Où T est la durée du mouvement, F est la fonction de coût (aussi appelée fonction objectif) qui traduit la stratégie mise en œuvre, g_i l'ensemble de contraintes 'inégalité' et h_i l'ensemble des contraintes 'égalité' qui vont garantir le respect des limites et aussi déterminer les caractéristiques du mouvement.

Il est à noter que les contraintes du type $\underline{\xi} \leq \xi(t) \leq \overline{\xi}$ peuvent être décomposées de la manière suivante :

$$\begin{array}{l} \underline{\xi} - \xi(t) \leq 0 \\ \xi(t) - \overline{\xi} \leq 0 \end{array} \tag{4.2}$$

4.2.2 Fonction de coût

Le choix de la fonction de coût $F(q(t))$, qui doit être minimisée, permet de définir la stratégie du mouvement (le plus rapide possible, consommant le moins d'énergie, ...), il doit tenir compte des caractéristiques du robot et de l'application visée.

Dans le cas de robots manipulateurs industriels, comme par exemple durant l'application de *pick and place* (cf. Figure 4.1(a)), qui consiste à transporter des objets d'un endroit à un autre, ou encore pendant les tâches d'une chaîne de montage, comme la peinture ou la soudure (cf. Figure 4.1(b)), les mouvements doivent s'effectuer en un temps minimum afin de maximiser la production. Piazzi s'est donc intéressé à la génération de mouvement les plus rapides possibles [Piazzi 98], mais aussi à des mouvements minimisant les jerks (dérivée de l'accélération par rapport au temps) [Piazzi 00], ce critère pouvant s'appliquer si les objets à transporter sont fragiles et risquent d'être détériorés par des mouvements trop violents.

(a) processus de *pick and place* (b) chaîne de montage

FIG. 4.1 – Exemple d'application où les mouvements doivent s'exécuter en temps minimum

Pour les robots humanoïdes, on préfère généralement utiliser des critères basés sur une interprétation des mouvements humains. Ainsi, en considérant que l'homme minimise une expression de l'énergie (expression encore non identifiée), Miossec prend la consommation d'énergie définie par l'équation 3.7 comme critère pour obtenir des mouvements semblables avec un robot humanoïde [Miossec 06], alors que Uno considère plutôt que l'homme a tendance à minimiser la dérivée des couples par rapport au temps et l'utilise comme critère pour reproduire une tâche de dessin avec un robot manipulateur [Uno 89].

Dans notre cas, nous pourrons considérer deux types de critère :

- la durée du mouvement : $F(\mathbf{q(t)}) = \int_0^T 1dt$ qui nous permettra, dans la section 4.3.1, de mettre en avant le fait que les méthodes classiques de résolution du problème d'optimisation, ne garantissent pas toujours la sécurité du robot.
- l'énergie du mouvement : $F(\mathbf{q(t)}) = \int_0^T (\sum_{i=0}^N 0.1\Gamma_i^2 + \Gamma_i \times \dot{q}_i)\ dt$. Même si, à ce jour, personne n'a prouvé que l'homme minimise une quelconque forme d'énergie pendant ses mouvements, le critère énergétique permet la génération de mouvements similaires à celui de l'homme et surtout augmente l'autonomie du robot en diminuant sa consommation d'énergie.

4.2.3 Contraintes ‘égalité’

L'ensemble des contraintes de type égalité $h_j(q(t))$ (appelées contraintes ‘égalité’) permet de définir le mouvement en imposant la valeur de certaines grandeurs à des instants définis. Par exemple, en attachant un repère au pied mobile, nous pouvons définir le mouvement de coup de pied comme représenté dans la figure 4.3, par des contraintes ‘égalité’ qui vont concerner la position et la vitesse de ce repère (voir Figure 4.2).

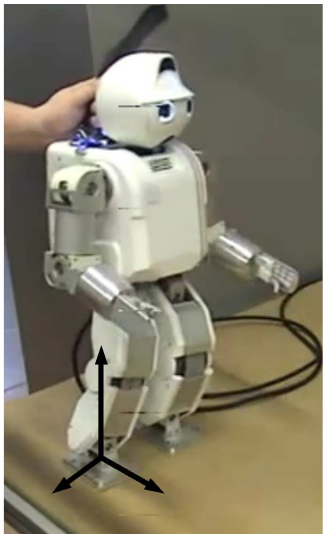
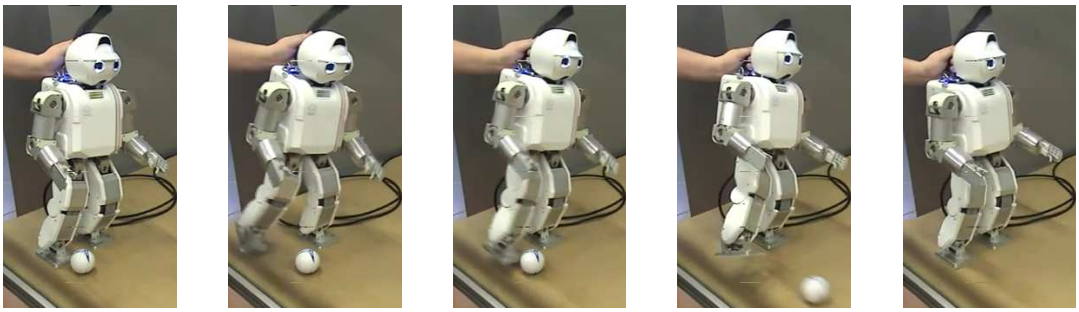


FIG. 4.2 – Représentation du repère attaché au pied en vol

Les contraintes ‘égalité’ pour un mouvement de coup de pied vont donc rassembler :
- La position et orientation du repère du pied au début du mouvement (Figure 4.3(a))
- La position et orientation du repère du pied à la fin du mouvement (Figure 4.3(e))
- La position et orientation du repère du pied au moment de l'impact (Figure 4.3(c))
- La vitesse du repère du pied au moment de l'impact (Figure 4.3(c))



(a) t = 0 (b) t = T/4 (c) t = T/2 (d) t = 3T/4 (e) t = T

FIG. 4.3 – Exemple d'un mouvement de coup de pied

4.2.4 Contraintes ‘inégalité’

Les contraintes de type inégalité $g_i(q(t))$ (appelées contraintes ‘inégalité’) permettent de définir les valeurs maximales et/ou minimales. Nous les répartissons en deux catégories : les contraintes ‘inégalité’ ponctuelles $g_{i_p}(q(t))$ et les contraintes ‘inégalité’ continues $g_{i_c}(q(t))$.

Contraintes ponctuelles

De la même manière que les contraintes 'égalité', les contraintes 'inégalité' ponctuelles permettent de définir une particularité du mouvement. Elles vont définir une valeur minimale ou maximale pour une grandeur du mouvement à un instant. Par exemple, pendant un mouvement de marche, il est intéressant d'ajouter une contrainte 'inégalité' ponctuelle qui va forcer le pied à se lever au dessus d'une hauteur définie h, comme présenté sur la Figure 4.4.

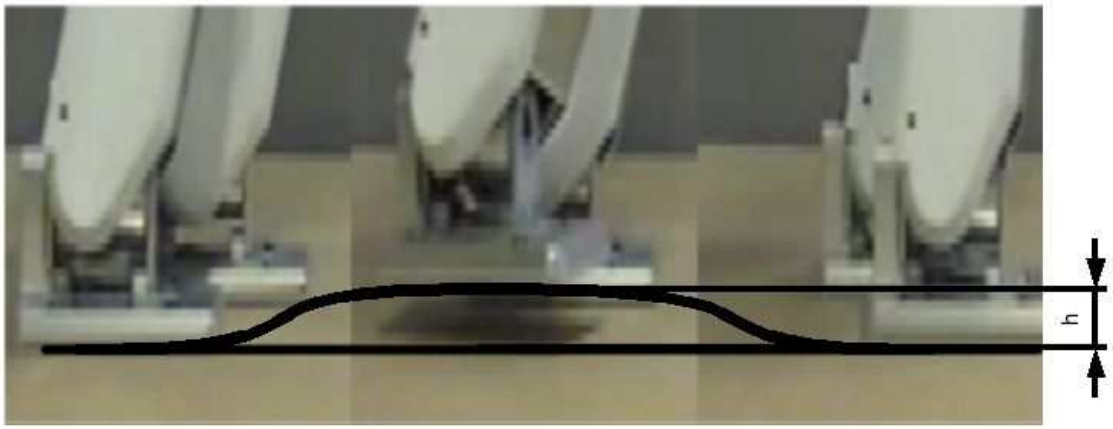


FIG. 4.4 – Représentation de la hauteur du pied pendant la phase de vol.

Contraintes continues

L'ensemble des contraintes 'inégalité' continues permet de déterminer les limites de fonctionnement que devront satisfaire tous les mouvements générés. Nous rappelons que dans notre cas, ces contraintes devront traduire les limites concernant :

- les butées mécaniques,
- les limites des vitesses articulaires,
- les couples maximums,
- le non-glissement du pied de support,
- les auto-collisions,
- et l'équilibre.

Par conséquent, l'intégrité du robot est liée au bon respect de ces contraintes. La résolution du problème d'optimisation (Équation 4.1) devra donc générer un mouvement qui satisfait ces contraintes sur toute la durée d'un mouvement.

Ensemble des contraintes 'inégalité'

Les contraintes inégalités peuvent donc être décomposées de la sorte :

$$\forall i, \forall t \in [0,T] \quad g_i(q(t)) \leq 0 \equiv \begin{cases} \forall i_p, \forall t \in \mathbb{T}_p & g_{i_p}(q(t)) \leq 0 \\ \forall i_c, \forall t \in [0,T] & g_{i_c}(q(t)) \leq 0 \end{cases} \tag{4.3}$$

Où $\mathbb{T}_p = \{t_{p_1}, t_{p_2}, t_{p_3}...\}$ représente l'ensemble des instants auxquels doivent être vérifiées les contraintes 'inégalité' ponctuelles.

4.3 *Semi Infinite Programming* (SIP)

4.3.1 Définition

La résolution du problème présenté par l'équation 4.1 permet d'obtenir l'évolution continue des trajectoires articulaires $q(t)$ qui satisfait un ensemble de contraintes dont une partie des contraintes 'inégalité' est continue $g_{i_c}(q(t))$. L'aspect continu des contraintes ou des trajectoires peut être assimilé à une décomposition en une infinité d'éléments. Nous avons donc un espace de recherche (des paramètres de Bsplines) et un espace libre de dimensions infinies. Par conséquent le problème défini par l'équation 4.1 est un problème de programmation infinie.

Un calcul des trajectoires articulaires $q(t)$ à partir d'un ensemble de n paramètres $X \in \mathbb{R}^n$ permet d'obtenir un nombre fini (n) de paramètres d'optimisation, et de passer d'un espace de configuration infini à un espace de dimension finie. Dans la section 3.2.3, nous avons vu que les B-splines offrent une bonne solution pour la paramétrisation des trajectoires articulaires. Par conséquent, en considérant N_s coefficients de B-splines pour les N_j articulations on transforme le problème infini en un problème semi infini (*Semi Infinite Programming* - SIP) [Hettich 93] ayant $n = (N_s \times N_j + 1)$ paramètres (en incluant aussi la durée du mouvement).

La planification de mouvement revient à trouver l'ensemble des paramètres $\tilde{X}$ qui :

$$\begin{array}{rl} \text{minimiser} & F(\tilde{X}) \\ \text{tel que} & \forall i, \forall t \in [0,T] \quad g_i(\tilde{X}, t) \leq 0 \\ \text{et} & \forall j \quad h_j(\tilde{X}) = 0 \end{array} \tag{4.4}$$

Nous avons trouvé différents logiciels d'optimisation (ALIAS [ali], FSQP [Lawrence], IPOPT [IPO 06], ...) qui permettent de résoudre des problèmes d'optimisation. Cependant ils ne traitent que les problèmes dont l'espace des paramètres ainsi que l'espace libre sont de dimensions finies. Nous allons présenter la méthode classique d'implémentation de ces logiciels qui consiste à effectuer une discrétisation de l'espace libre correspondant à une discrétisation du temps dans notre cas. Ce qui va nous permettre de mettre en lumière le fait que cette méthode peut provoquer la violation de certaines contraintes, puis nous présenterons notre méthode de discrétisation basée sur les intervalles.

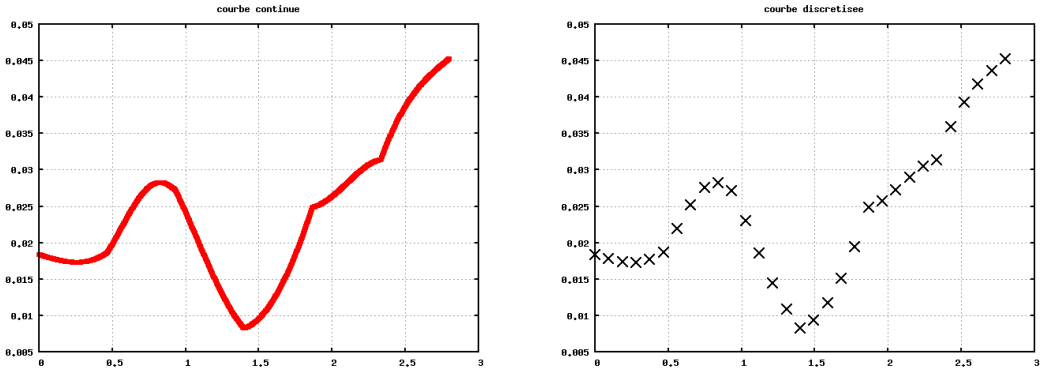
4.3.2 Discrétisation par grille temporelle

La discrétisation par grille est le processus qui assimile une fonction continue à sa valeur pour un ensemble d'instants appartenant à une grille temporelle [Hettich 93, von Stryk 93]. Pour la résolution du problème d'optimisation, les contraintes ne seront donc prises en compte que pour ces instants de la grille (cf. Figure 4.5) et les contraintes 'inégalité' continues présentées dans l'équation (4.4) sont remplacées par :

$$\forall i, \forall t_k \in \mathbb{T} \quad g_{i_c}(\mathbf{X}, t_k) \leq 0 \tag{4.5}$$

Où $\mathbb{T} = \{t_0 = 0, t_1, ..., t_{N-1}, t_N = T\}$.

Par conséquent le problème continu (4.4) où apparaît $\forall t \in [0,T]$ devient un problème de dimension finie : $\forall t_k \in \{0, t_1, ..., t_{N-1}, T\}$, il peut être résolu par les logiciels d'optimisation précités.



(a) Courbe continue (b) Valeurs discrètes

FIG. 4.5 – Discrétisation : la courbe continue est considérée comme un ensemble de points.

4.3.3 Violation de contraintes

En utilisant la méthode de discrétisation classique, l'optimisation fournira une solution où les contraintes seront satisfaites pour les instants appartenant à la grille temporelle $\{t_0, t_1..., t_{N-1}, t_N\}$. Cependant, aucune information n'est donnée pour les instants se trouvant entre deux instants.

Dans les publications [Lengagne 07, Lengagne 08], nous mettons en évidence à quel point une discrétisation classique des contraintes continues est risquée pour le respect des contraintes et donc pour l'équilibre et l'intégrité du robot humanoïde HOAP-3. Nous considérons les contraintes présentées dans la section 3.8.2 et la durée du mouvement comme fonction de coût pour obtenir les résultats suivants :

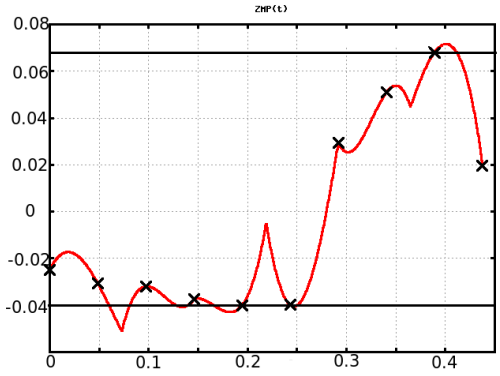


FIG. 4.6 – Représentation d'une grandeur contrainte (le ZMP), obtenue par une optimisation utilisant une grille de 10 instants.

Sur la Figure 4.6, les points pris en compte par l'optimisation (marqués par des croix) respectent bien les contraintes. Cependant, entre ces points, l'évolution continue de la grandeur montre bien une violation des contraintes. Intuitivement, la première idée pour résoudre ce problème serait d'augmenter le nombre de points de la grille. Nous montrons dans la section 4.6.1, que cette solution permet de diminuer la valeur de la violation jusqu'à la supprimer. Néanmoins, lors de nos expérimentations (voir section 4.6.1), nous avons constaté une augmentation du critère avec le nombre de points de la grille d'échantillonnage.

Les méthodes de planification de mouvements, qui se basent sur une discrétisation par grille temporelle, ne permettent donc pas de garantir l'optimalité du mouvement et la validité des contraintes. Malheureusement, dans les deux cas le logiciel d'optimisation considère le mouvement comme optimal et sans violation de contraintes. Nous considérons que l'obtention d'un mouvement sous-optimal est moins important que la génération d'un mouvement dangereux pour le robot. Pour éviter toute violation de contraintes, nous proposons une méthode de discrétisation des contraintes basée sur l'Analyse par Intervalles.

4.4 L'Analyse par Intervalles

4.4.1 Définition

Initialement, l'Analyse par Intervalles a été développée pour quantifier les erreurs introduites par la représentation en nombres flottants de nombre réels et a été étendue à la validation numérique [Moore 66, Neumaier 90, Sunaga 58]. Certains auteurs, comme Hansen [Hansen 04], assimilent le début de l'Analyse par Intervalles à la parution de l'ouvrage de Moore [Moore 66], en 1966.

L'idée principale est très simple : remplacer les nombres réels par des intervalles auxquels ils appartiennent. Il est alors possible d'obtenir des encadrements numériques garantis par la transposition aux intervalles des fonctions classiques opérant sur des réels [Jaulin 01b].

Pour cela, on définit un intervalle réel $[a] = [\underline{a},\ \bar{a}]$ comme un sous-ensemble continu de $\mathbb{R}$. L'ensemble de tous les intervalles réels de $\mathbb{R}$ est noté par $\mathbb{IR}$. Les opérations réelles (addition, soustraction, multiplication, division) sont étendues aux intervalles. Considérons un opérateur $\circ \in \{+, -, *, \div\}$ ainsi que $[a]$ et $[b]$ deux intervalles, alors on définit :

$$[a] \circ [b] = [inf_{u\in[a],v\in[b]}\, u \circ v, \quad sup_{u\in[a],v\in[b]}\, u \circ v] \tag{4.6}$$

De la même façon, les fonctions réelles peuvent être étendues aux intervalles. Considérons la fonction $\mathbf{g} : \mathbb{R}^n \longmapsto \mathbb{R}^m$; l'ensemble des valeurs de cette fonction pour un intervalle $[a]$ est donné par :

$$\mathbf{g}([\mathbf{a}]) = \{\mathbf{g}(\mathbf{u}) \mid \mathbf{u} \in [\mathbf{a}]\} \tag{4.7}$$

La fonction intervalle $[\mathbf{g}] : \mathbb{IR}^n \longmapsto \mathbb{IR}^m$ est une fonction d'inclusion de la fonction réelle $\mathbf{g}$ si

$$\forall [\mathbf{a}] \in \mathbb{IR}^n,\ \mathbf{g}([\mathbf{a}]) \subseteq [\mathbf{g}]([\mathbf{a}]) \tag{4.8}$$

Une fonction d'inclusion $\mathbf{g}$ peut être obtenue en remplaçant chaque occurrence de la variable réelle a par la variable intervalle correspondante $[a]$ et chaque fonction réelle par la fonction intervalle adéquate. Dans ce cas, la fonction obtenue est appelée fonction d'inclusion naturelle dont les performances dépendent de l'expression formelle de $\mathbf{g}$.

4.4.2 Surapproximation

Définition : L'Analyse par Intervalles est donc un outil intéressant qui permet d'obtenir un résultat numérique garanti sur des opérations mettant en scène des ensembles de valeurs. Cependant son utilisation provoque intrinsèquement de la surapproximation [Chabert 07] que nous allons présenter à travers un exemple simple.

Soit deux fonctions :

$$f_1(t) = 3 \times t + 1 \tag{4.9}$$
$$f_2(t) = 2 \times t - 2 \tag{4.10}$$

La valeur de t appartient à l'intervalle $[0, 1]$. Nous souhaitons évaluer dans quel intervalle vont évoluer les fonctions suivantes :

$$f_3(t) = f_1(t) + f_2(t) \tag{4.11}$$
$$f_4(t) = f_1(t) - f_2(t) \tag{4.12}$$

Chacun voit que $f_3(t) = 5 \times t - 1$ et que $f_4(t) = t + 3$ et par conséquent que les résultats seront $f_3([t]) = [-1, 4]$ et $f_4([t]) = [3, 4]$. Mais quels sont les résultats donnés par l'Analyse par Intervalles ? Pour le savoir, nous allons calculer les fonctions d'inclusion de f_1 et f_2 :

$$\begin{aligned} f_1([t]) = &\quad 3 \times t + 1 \quad = 3 \times [0,1] + 1 \quad = \quad [1,4] \quad = [\underline{f_1}, \overline{f_1}] \\ f_2([t]) = &\quad 2 \times t - 2 \quad = 2 \times [0,1] - 2 \quad = \quad [-2,0] \quad = [\underline{f_2}, \overline{f_2}] \end{aligned} \tag{4.13}$$

pour reporter leurs valeurs dans les équations de f_3 et f_4 :

$$\begin{aligned} f_3([t]) = &\quad f_1([t]) + f_2([t]) \quad = [\underline{f_1} + \underline{f_2}, \overline{f_1} + \overline{f_2}] \quad = [1 + (-2), 4 + 0] \quad = [-1, 4] \\ f_4([t]) = &\quad f_1([t]) - f_2([t]) \quad = [\underline{f_1} - \overline{f_2}, \overline{f_1} - \underline{f_2}] \quad = [1 - 0, 4 - (-2)] \quad = [1, 6] \end{aligned} \tag{4.14}$$

La valeur de $f_3([t])$ est bien celle prévue, mais le calcul de $f_4([t])$ montre bien des différences. Le calcul de la fonction d'inclusion donne un intervalle qui contient la solution ($[3, 4] \subseteq [1, 6]$). Nous définissons la surapproximation comme la différence entre l'intervalle calculé et l'intervalle réel (c'est à dire l'intervalle minimum qui contient toutes les solutions).

Pour réduire la surapproximation, il existe un grand nombre de méthodes qui se basent, notamment, sur la bissection de l'intervalle et/ou sur un test de monotonie de la fonction sur l'intervalle.

Bissection : La bissection consiste à couper l'intervalle en deux sous-intervalles et à effectuer le calcul sur chaque sous-intervalle : le résultat final sera l'union des résultats :

$$\begin{aligned} \text{à partir de l'intervalle} \quad & [t] = [\underline{t}, \overline{t}] \\ \text{on définit} \quad & [t_1] = [\underline{t}, \tau] \\ \text{et} \quad & [t_2] = [\tau, \overline{t}] \\ \text{avec} \quad & \tau \in [t] \\ \text{alors} \quad & f([t]) = f([t_1]) \cup f([t_2]) \end{aligned} \tag{4.15}$$

Reprenons l'exemple de $f_4(t)$ et bissectons l'intervalle $[t] = [0, 1]$ en $[t_1] = [0, 0.5]$ et $[t_2] = [0.5, 1]$ on obtient :

$$\begin{aligned} f_4([t]) &= f_4([t_1]) \cup f_4([t_2]) \\ &= ([1, 2.5] - [-2, -1]) \cup ([2.5, 4] - [-1, 0]) \\ &= [2, 4.5] \cup [2.5, 5] \\ f_4([t]) &= [2, 5] \end{aligned} \tag{4.16}$$

On obtient donc l'intervalle $f_4([t]) = [2, 5]$, on voit que la bissection a réduit la surapproximation (de $[1, 6]$ à $[2, 5]$). Cependant le résultat n'est toujours pas l'intervalle réel $[3, 4]$, pour encore diminuer la surapproximation il faudra bissecter d'avantage.

Monotonie : Le test de la monotonie consiste à étudier la dérivée de la fonction par rapport à la valeur intervalle. Si cette dérivée est strictement positive (ou négative), on sait alors que la fonction est strictement croissante (ou décroissante) et que donc les optima sont obtenus pour les valeurs des bornes de l'intervalle.

Reprenons notre exemple pour illustrer la monotonie :

$$\begin{array}{rcl} f_1(t) = 3 \times t + 1 & \rightarrow & f_1'(t) = 3 \\ f_2(t) = 2 \times t - 2 & \rightarrow & f_2'(t) = 2 \end{array} \tag{4.17}$$

Donc :

$$f_4'([t]) = f_1'([t]) - f_2'([t]) \quad = 3 - 2 \quad = 1 \tag{4.18}$$

La dérivée de la fonction $f_4(t)$ est strictement positive, par conséquent :

$$f_4([t]) = [f_4(\underline{t}), f_4(\overline{t})] \quad = [f_1(\underline{t}) - f_2(\underline{t}), f_1(\overline{t}) - f_2(\overline{t})] \quad = [1 - (-2), 4 - 0] \quad = [3, 4] \tag{4.19}$$

Sur cet exemple simple, l'étude de la dérivée nous donne l'intervalle exact sans aucune surapproximation. Cependant pour des fonctions plus complexes, l'étude du gradient doit être associée à la bissection pour donner de bons résultats.

Conclusion : Nous avons donc vu que l'Analyse par Intervalles permet d'obtenir des résultats numériques garantis, ce qui peut être utile dans beaucoup d'applications comme l'optimisation, la commande ou la planification. Toutefois cette garantie nécessite une implémentation numérique vérifiée qui effectue un contrôle de l'arrondi.

Nous avons présenté les deux principales méthodes permettant de diminuer la surapproximation, cependant il existe d'autres méthodes basées, par exemple, sur le développement en série de Taylor ou d'autres techniques d'optimisation globale [Hansen 04].

4.4.3 Exemples d'applications de l'Analyse par Intervalles

Loin de vouloir faire une liste exhaustive des applications de l'Analyse par Intervalles, nous présentons quelques travaux se rapportant à la robotique, afin d'exposer un aperçu de la diversité des problèmes résolvables grâce à cet outil.

Optimisation :

L'optimisation d'une fonction revient à chercher le minimum (ou le maximum) de cette fonction sur un intervalle donné. Contrairement aux méthodes d'optimisation classiques par descente, l'utilisation de l'Analyse par Intervalles permet de ne pas tenir compte des minima locaux et de trouver le minimum global [Hammer 93],[Hansen 04],[Kolev 00].

La Figure 4.7 montre la recherche du minimum global pour une fonction comportant plusieurs minima locaux et le principe d'un algorithme d'optimisation basé sur l'Analyse par Intervalles : l'algorithme *Midpoint Test*. Cet algorithme calcule la valeur de la fonction $\tilde{f}$ au milieu d'un intervalle . Tout intervalle $[t]$ dont la borne inférieure Inf($f([t])$)est supérieure à $\tilde{f}$ est rejeté, les autres intervalles seront bissectés et l'opération relancée.

Les minima locaux peuvent également être gérés par des algorithmes de type génétique qui sont des algorithmes d'optimisation globale mais non garantis. En effet, le fait qu'il ne se basent que sur des échantillons de l'espace de recherche ne leur permet pas de garantir

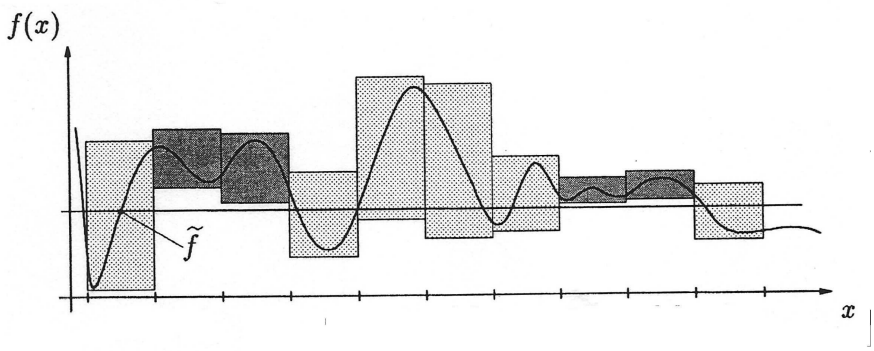


FIG. 4.7 – Optimisation utilisant l'Analyse par Intervalles [Hammer 93].

l'absence (ou la présence) de solution à un problème, contrairement aux méthodes par intervalles.

L'optimisation basée sur l'Analyse par Intervalles permet d'obtenir un résultat global et garanti. Cette propriété importante a été exploitée, par exemple, par Jaulin pour la résolution des problèmes d'optimisation pour la planification de chemin [Jaulin 01a] pour montrer la faisabilité du problème par l'existence d'un chemin possible et pour déterminer le meilleur chemin.

Commande :

Pour la commande d'un système, il est important de prouver la stabilité d'un système avant de pouvoir l'implémenter. Walter propose une méthode d'inversion basée sur l'analyse par intervalles afin de déterminer l'ensemble des paramètres qui permettent d'avoir un système stable (pôles à partie réelle négative) [Walter 94]. Alors que Vehi, Fans ou encore Merlet l'utilisent dans le cas d'une commande prédictive pour s'assurer du bon respect des contraintes [Vehi 01],[Fang 05],[Merlet 01].

L'Analyse par Intervalles peut également s'appliquer aux observateurs, qui sont des éléments importants de la commande de systèmes. Kieffer utilise un encadrement du bruit de mesure et de l'état initial pour estimer un ensemble d'état en garantissant que toutes les solutions contenues sont possibles pour les valeurs mesurées [Kieffer 98].

4.4.4 Implémentation

L'Analyse par Intervalles est donc un outil intéressant puisqu'il permet de s'assurer du fonctionnement d'un robot à travers ses équations. Afin de faciliter son utilisation il peut être implémenté sous différents langage de programmation :

- MATLAB, par la *Toolbox* INTLAB [Rump 99]
- SCILAB, par la *Toolbox* INT4SCI `http://www-sop.inria.fr/coprin/logiciels/Int4Sci/index.php`
- C/C++ grâce à la bibliothèque PROFIL-BIAS [Knoppel 99] ou ALIAS [ali]

4.5 Planification de mouvements sûrs

4.5.1 Calcul des contraintes

Dans la section 4.3.3, nous avons mis en évidence qu'une discrétisation classique ne permettait pas de vérifier la validité des contraintes entre les instants de la grille. Et la section précédente montre bien que l'Analyse par Intervalles permet de calculer de manière garantie un encadrement d'une fonction sur un intervalle (temporel dans notre cas).

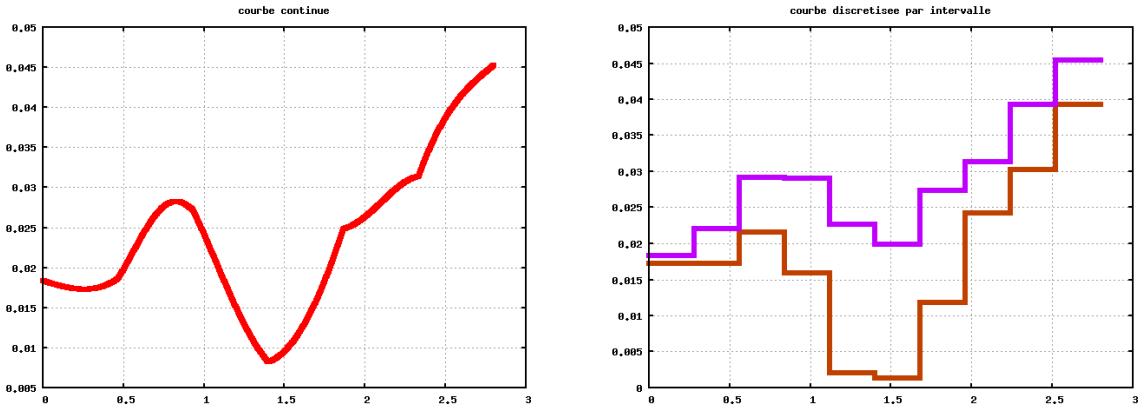
Nous définissons, donc, la planification garantie comme une planification qui utilise une discrétisation temporelle par intervalle que nous appelons *discrétisation garantie*. Elle nous permet d'utiliser les mêmes logiciels que pour une planification classique.

L'idée principale de la discrétisation garantie est de fournir au logiciel d'optimisation, les extrema des contraintes 'inégalité' continues $g_{i_c}(\mathbf{X}, t)$ calculés sur des intervalles de temps $[t] = [\underline{t}, \overline{t}]$ au lieu de ne retourner que certains points de ces fonctions (voir Figure 4.8). Par conséquent l'équation 4.5 est remplacée par :

$$\forall i_c, \forall [t] \in \mathbb{IT} \quad \max_{\forall \tau \in [t]} \; g_{i_c}(\mathbf{X}, \tau) \leq 0 \tag{4.20}$$

Avec $\mathbb{IT} = \{[t]_1, [t]_2, ..., [t]_{k-1}, [t]_k\}$ où les intervalles de temps sont définis par $[t]_n = [t_{n-1}, t_n]$.

Comme nous l'avons montré précédemment, le calcul des contraintes sur ces intervalles produirait une surapproximation qui donnerait des résultats trop peu précis. Chacun de ces intervalles sera donc bissecté en N_b sous-intervalles $[t]_k = \cup_{l=1,...,N_b}[t]_{k,l}$ pour obtenir une précision suffisante. Nous définissons le sous-intervalle $[t]_{i_c,k,l_{max}}$ comme celui qui contient le maximum de la contrainte $g_{i_c}(\mathbf{X}, t)$ sur l'intervalle $[t_k]$.



(a) Courbe continue (b) Valeurs discrétisées par intervalles

FIG. 4.8 – Discrétisation garantie : la courbe continue est assimilée son encadrement pour plusieurs intervalles de temps.

Cette méthode permet de s'assurer que le logiciel d'optimisation dispose bien des valeurs extrêmes de la fonction, et qu'il sera en mesure de modifier le mouvement (les paramètres $\mathbf{X}$) en cas de violation d'une contrainte (cf. Figure 4.9).

L'utilisation de la discrétisation garantie permet donc de s'assurer du respect des contraintes 'inégalité' continues sur toute la durée du mouvement, contrairement à la discrétisation classique qui ne vérifie les contraintes qu'à des instants précis sans se soucier de l'évolution des contraintes entre ces instants.

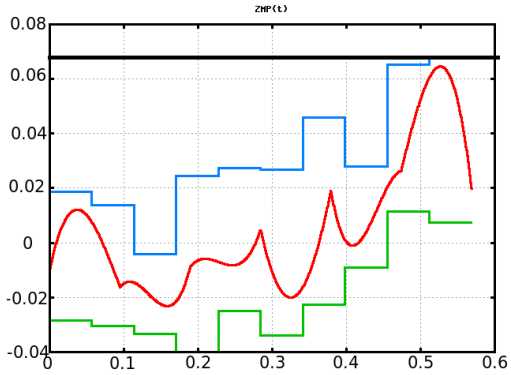


FIG. 4.9 – Planification garantie : Représentation d'une grandeur contrainte (le ZMP) : les encadrements de la fonction respectent les limites, donc la fonction continue les respectent.

Afin de privilégier la simplicité des calculs, nous avons remplacé la courbe à discrétiser par un ensemble de fonctions constantes (cf. Figure 4.8). Il serait également possible de la remplacer par un ensemble de fonctions qui ne soient pas constantes (droites, polynômes, fonctions convexes ou concaves).

4.5.2 Calcul du gradient des contraintes

Afin d'améliorer la convergence, la plupart des logiciels d'optimisation ont besoin de l'estimation de la dérivée des contraintes (et du critère) par rapport aux paramètres : $\frac{\partial g(\mathbf{X},t)}{\partial \mathbf{X}}$. Avec une discrétisation par grille (cf Section 4.3.1), le gradient est calculé pour les instants de la grille, soit par une expression formelle [Suleiman 07, Lee 05] soit par différentiation automatique [Bendtsen]. La discrétisation garantie présentée dans la section 4.5.1, obtient la valeur des contraintes en divisant l'intervalle $[t_k]$ en N_b sous-intervalles $[t]_k = \cup_{l=1,\ldots,N_b}[t]_{k,l}$. Il existe donc un sous-intervalle $[t]_{i_c,k,l_{max}}$ qui contient le maximum de l'équation : $g_{i_c}(\mathbf{X},t)$ for $t \in [t]_k$ comme le montre la Figure 4.10.

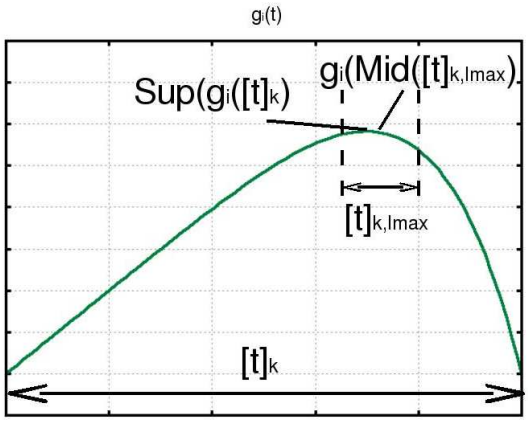


FIG. 4.10 – Représentation de l'approximation de $\text{Sup}[g]_i([t]_k)$ par $g_i(Mid[t]_{k,l_{max}}$ utilisé pour le calcul du gradient.

Nous ne pouvons pas connaître l'instant exact, appartenant à $[t]_{i_c,k,l_{max}}$, où la fonction est maximale. Par conséquent nous choisissons d'approcher le gradient de ce maximum

$\frac{\partial}{\partial \mathbf{X}} g_i(\mathbf{X}, t)$ par la valeur du gradient de la fonction au milieu de l'intervalle $[t]_{k,l_{max}}$ comme formulé dans l'équation 4.21 :

$$\frac{\partial}{\partial \mathbf{X}} \text{Sup}[g]_i(\mathbf{X}, [t]_k) \approx \frac{\partial}{\partial \mathbf{X}} g_i(\mathbf{X}, Mid[t]_{k,l_{max}}) \tag{4.21}$$

Cette approximation nécessaire, fait que la largeur de l'intervalle $[t]_{k,l_{max}}$ aura un effet sur la validité du calcul du gradient, et donc influera sur l'efficacité du logiciel d'optimisation. Un sous-intervalle $[t]_{i_c,k,l_{max}}$ de largeur très petite, permettra d'avoir une évaluation précise du gradient ce qui permettra à l'algorithme de converger rapidement avec peu d'itérations, mais augmentera le temps de calcul des contraintes à chaque itération d'une itération.

Le choix du degré de bissection devra tenir compte du paradoxe : moins d'itérations mais des itérations plus longues.

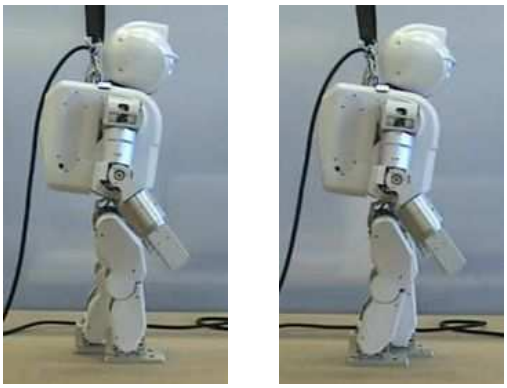
4.5.3 Librairie Développée

Toutes les notions présentées dans ce chapitre (calcul d'une fonction, de son gradient pour des instants ou pour des intervalles) ont été intégrées dans une librairie : La *Guaranteed Discretization Librar*y (GDL) disponible à cette adresse `http://www.lirmm.fr/~lengagne/GDL`.

Cette librairie C++ est basée sur une classe de type template que l'utilisateur pourra compléter à sa guise et nous a permis d'implémenter le modèle du robot HOAP-3 pour faire la planification de ses mouvements.

4.6 Résolution du problème d'optimisation

Dans cette section, nous allons appliquer les méthodes d'optimisation classiques et garanties en considérant le modèle 3D du robot HOAP-3 affectés des limites présentés dans le chapitre 3, pour générer les trajectoires articulaires des membres inférieurs pour un mouvement de pas en avant. Le robot commence le mouvement avec le pied gauche situé à 5 cm derrière le pied droit pour le finir avec le pied gauche 5 cm devant le pied droit qui sera le pied de support (cf. Figure 4.6). Les calculs ont été effectués sur un PC : 3.2GHz, 1Go de RAM ayant le système d'exploitation Kubuntu.



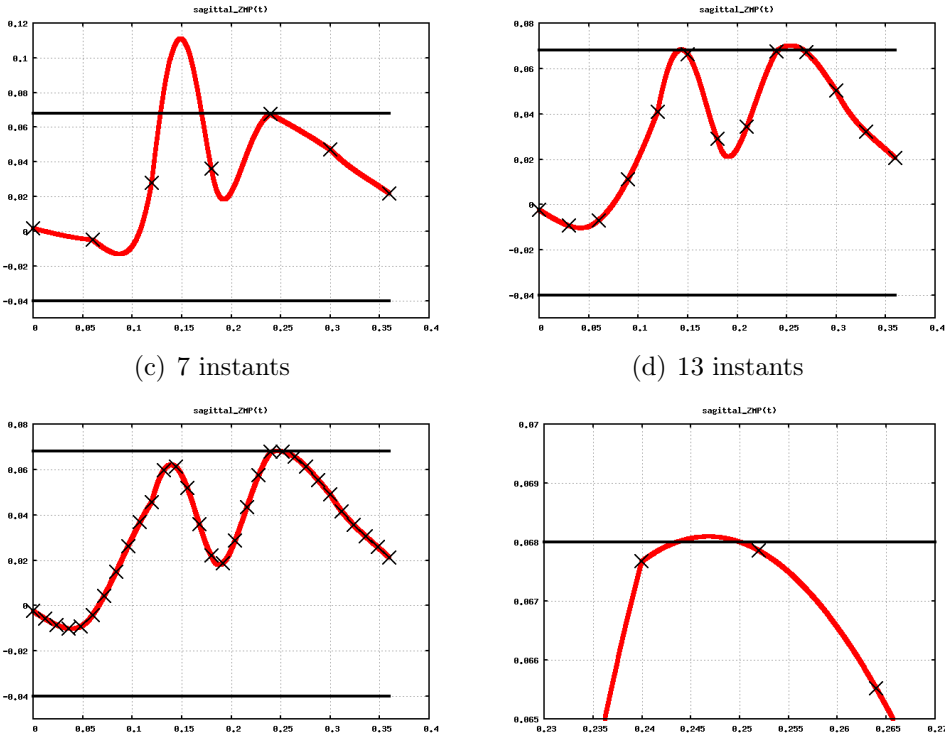
(a) Début du mouvement (b) Fin du mouvement

Nombre de points de la grille	Nombre de contraintes	Nombre d'itérations	Valeur du critère (s)	Temps de calcul (IPOPT+ NLP)	Nombre de limites non respectées	Grandeurs au dessus des limites
7	558	48	3.691e-01	49.9 s (2.2 + 47.7)	17	(13)$\dot{q}$: 49.8% (2) Γ : 6.7% (2) ZMP : 71.4%
13	1026	73	3.896e-01	84.9 s 1 mn 24.9s (3.5 + 81.4)	14	(11)$\dot{q}$: 4.5% (1) Γ : 0.05% (2) ZMP : 4.7%
31	2430	34	3.914e-01	65.0 s 1 mn 05s (4.8 + 60.2)	14	(10)$\dot{q}$: 0.5% (2) Γ : 0.02% (2) ZMP : 0.65%
61	4770	45	3.925e-01	154.8 s 2 mn 347.8s (10.2 + 144.6)	6	(5)$\dot{q}$: 0.16% (1) Γ : 0.2%
121	9450	47	3.943e-01	287.6 s 4min 47.6s (21.0 + 266.6)	2	(1)$\dot{q}$: 0.0007% (1) Γ : 0.007%
301	23490	60	4.010e-01	801.1 s 13 mn 21.1 s (63.9 + 737.2)	0	pas de violation
601	46890	52	4.169e-01	1449.8 s 24mn 9.8s (122.6 + 1327.2)	0	pas de violation

TAB. 4.1 – Temps de calcul et résultats pour des optimisations utilisant des grilles temporelles pour la discrétisation des contraintes.

4.6.1 Grille temporelle

La Figure 4.11 représente l'évolution du ZMP dans le plan sagittal pour des optimisations utilisant une discrétisation à base de grilles temporelles de tailles différentes (7, 13, 31). Ces figures confirment le fait que cette méthode ne garantit pas le respect des contraintes sur toute la durée du mouvement : les points pris en compte sont dans les limites mais la fonction continue dépasse les limites à certains instants. Même si cette violation est minime (cf. Figure 4.11(f)) elle ne peut pas être acceptée car elle pourrait entraîner la chute du robot ou la destruction d'un de ses composants.



(c) 7 instants (d) 13 instants

(e) 31 instants (f) 31 instants (zoom)

FIG. 4.11 – Représentation du ZMP dans le plan sagittal pour des optimisations utilisant une grille de 7,13 et 31 instants.

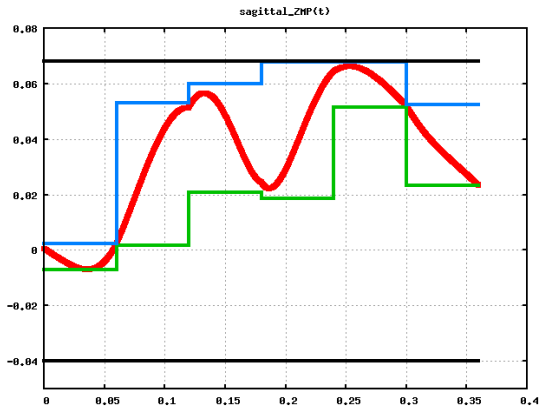
Le tableau 4.1 montre les temps de calcul de plusieurs optimisations utilisant différentes grilles temporelles. Le temps de calcul est réparti en deux :

- IPOPT donne le temps de calcul passé dans l'algorithme d'optimisation,
- NLP le temps de calcul passé dans les fonctions d'évaluations (contrainte, critère et gradient).

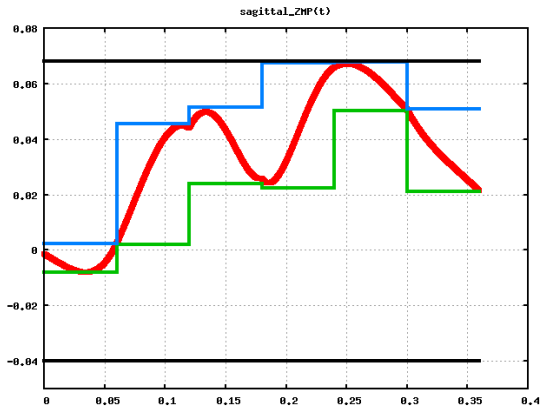
Ce tableau montre bien la violation de contraintes pour des grilles temporelles comportant peu d'instants. Il apparaît que les contraintes sont moins violées quand le nombre d'instants de la grille augmente mais ceci augmente également le temps de calcul. Malheureusement, on remarque que la valeur du critère continue à augmenter avec le nombre de points (301 à 601) même si il n'y a plus de violations de contraintes. D'après ces résultats on peut déduire que la solution intuitive qui consisterait à augmenter le nombre de points pour s'assurer du respect des limites produirait une solution sous-optimale. Cette méthode de discrétisation est donc difficile à configurer. Un calcul rapide peut amener un résultat violant les limites du robot, et un calcul trop lent produira une solution sous-optimale.

4.6.2 Intervalles

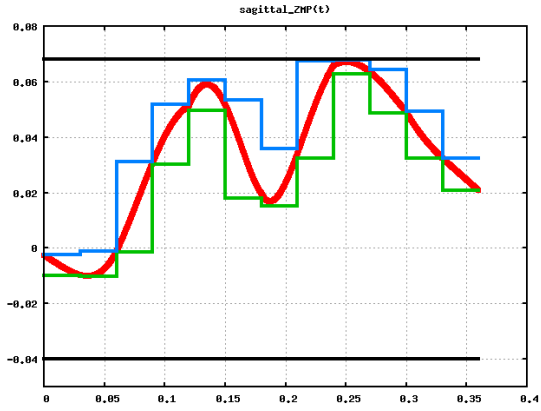
La Figure 4.12 représente l'évolution du ZMP dans le plan sagittal pour des optimisations utilisant la discrétisation garantie. On remarque que les encadrements de cette fonction (valeurs retournées à l'algorithme) respectent bien les limites et que par conséquent l'évolution continue de cette fonction ne les dépasse à aucun moment.



(a) 6 intervalles (2^{15} sous-intervalles)



(b) 6 intervalles (2^{16} sous-intervalles)



(c) 12 intervalles (2^{15} sous-intervalles)

FIG. 4.12 – Représentation du ZMP dans le plan sagittal pour des optimisations utilisant la discrétisation garantie.

Ces courbes mettent également en évidence l'effet de la bissection sur la surapproximation. Sur la Figure 4.12(a) on remarque un écart entre l'évaluation intervalle de la fonction et les extrema réels. Cet écart est réduit sur la Figure 4.12(b) pour laquelle le processus de bissection a été plus fin.

Le Tableau 4.2 montre les résultats d'optimisation utilisant la discrétisation garantie des contraintes. On retrouve bien l'effet de la bissection sur le temps de calcul qui a été exposé dans la section 4.5.2, à savoir qu'une bissection plus fine permet de diminuer le nombre d'itérations car le calcul du gradient est plus précis, mais aussi de diminuer la surapproximation sur les contraintes ce qui permet d'avoir une solution avec un critère plus petit. Malheureusement, on peut constater que le temps de calcul est de l'ordre de la journée alors que les méthodes de discrétisation classique produisent un résultat en quelques dizaines de minutes. Cette augmentation du temps de calcul est liée à la complexité du calcul qui nécessite une bissection importante.

Nombre d'intervalles	nombre de sous-intervalles	nombre de contraintes	nombre d'itérations	valeur du critère (s)	temps de calcul (IPOPT : NLP)
6	2^{15}	480	269	3.974e-01	158627.4 s 44h 03mn 47.4s (274.9 + 158352.5)
12	2^{15}	948	86	3.933e-01	102969.2 s 28h 36mn 09.2s (810.0 + 102159.2)
6	2^{16}	480	145	3.950e-01	116343.2 s 32h 19mn 03.2s (429.2 + 115914.0)

TAB. 4.2 – Temps de calcul et résultats pour des optimisations utilisant la discrétisation garantie des contraintes.

Dans [Lengagne 09b], nous considérons un modèle 2D, plus simple, du robot nécessitant une bissection en 2^8 sous-intervalles et nous montrons une évolution du temps de calcul de 8s, pour les méthodes classiques, à 90 s pour notre méthode garantie, ce qui reste acceptable. Cependant avec un modèle 3D, il faut trouver une méthode pour réduire le temps de calcul, tout en garantissant la validité des contraintes. C'est pourquoi nous proposons d'utiliser une méthode hybride.

4.7 Optimisation hybride

4.7.1 Principe

Étant donné les temps de calcul de la méthode garantie et le risque inhérent à l'utilisation des méthodes classiques nous proposons une méthode d'optimisation hybride. Cette méthode itérative consiste à faire des optimisations qui utilisent la discrétisation classique. Puis nous vérifions la validité des contraintes 'inégalité' continues par une discrétisation garantie afin de relancer l'optimisation en prenant en compte de nouvelles limites en fonction des contraintes violées.

Pour l'itération q on modifie les contraintes inégalités continue du problème 4.1 par :

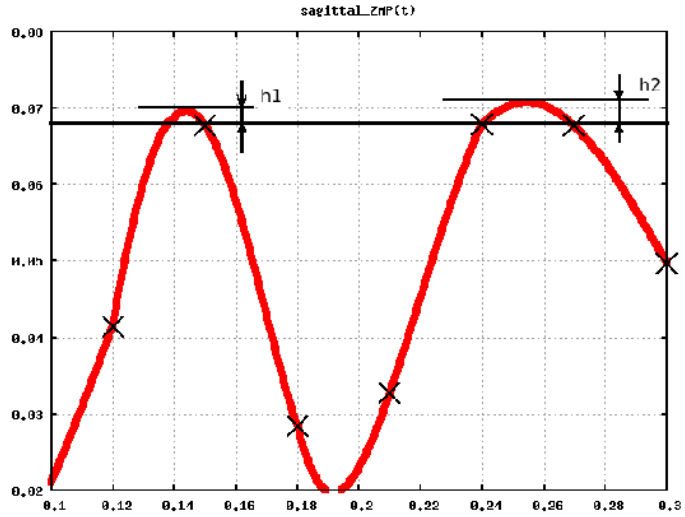
$$\forall i, \forall t_k \in \mathbb{T} \quad g_{i_c}(\mathbf{X}, t_k) \leq -v_{i,k,q} \tag{4.22}$$

Pour la première itération, $v_{i,k,q}$ est initialisé à zéro, puis il est calculé grâce à l'Analyse par Intervalles de la manière suivante.

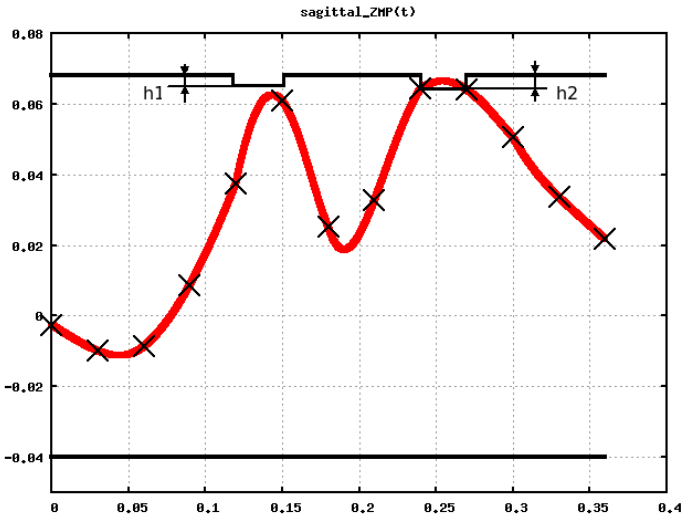
$$\begin{gathered} \mu_{i,k,q} = max\left(\text{Sup}[g]_{i_c}(\mathbf{X}, [t_{k-1}, t_k]), \text{Sup}[g]_{i_c}(\mathbf{X}, [t_k, t_{k+1}]), 0\right) \\ v_{i,k} = v_{i,k,q-1} + \mu_{i,k,q} \end{gathered} \tag{4.23}$$

Où $\mu_{i,k,q}$ est la valeur maximale de la violation à l'itération courante q (si il y a violation sinon il vaut zéro) de la contrainte i entre les instants t_{k-1} et t_{k+1} pour le mouvement défini par les paramètres $\mathbf{X}$.

La Figure 4.14 montre le principe de l'optimisation hybride. La Figure 4.13(a) représente le résultat de la première itération. La discrétisation garantie permet de s'apercevoir de la présence d'une ou de plusieurs violations de contraintes et de calculer leurs valeurs



(a) première itération



(b) deuxième itération

FIG. 4.13 – Représentation du ZMP dans le plan sagittal aux deux premières itérations de l'optimisation hybride

(h_1, h_2). Ensuite, le processus d'optimisation est relancé en ayant pris en compte ces violations. La Figure 4.13(b) montre la modification des limites ainsi que leur effet sur la suppression des violations.

4.7.2 Résultats

Le Tableau 4.3 montre les temps de calcul de chaque itération. La génération d'un mouvement optimal garanti a été réalisée en $(41.4 + 33.8 + 7.6 + 8.4 + 4 \times 380 = 1611.2s)$ qui correspond à 26mn 51.2s. On peut s'apercevoir que cette solution possède un critère inférieur à celui fourni par des méthodes classiques. On peut donc allier la sécurité et l'optimalité apportées par la discrétisation garantie et le temps de calcul réduit de la discrétisation classique pour obtenir une génération de mouvements optimaux sûrs en un temps de calcul raisonnable.

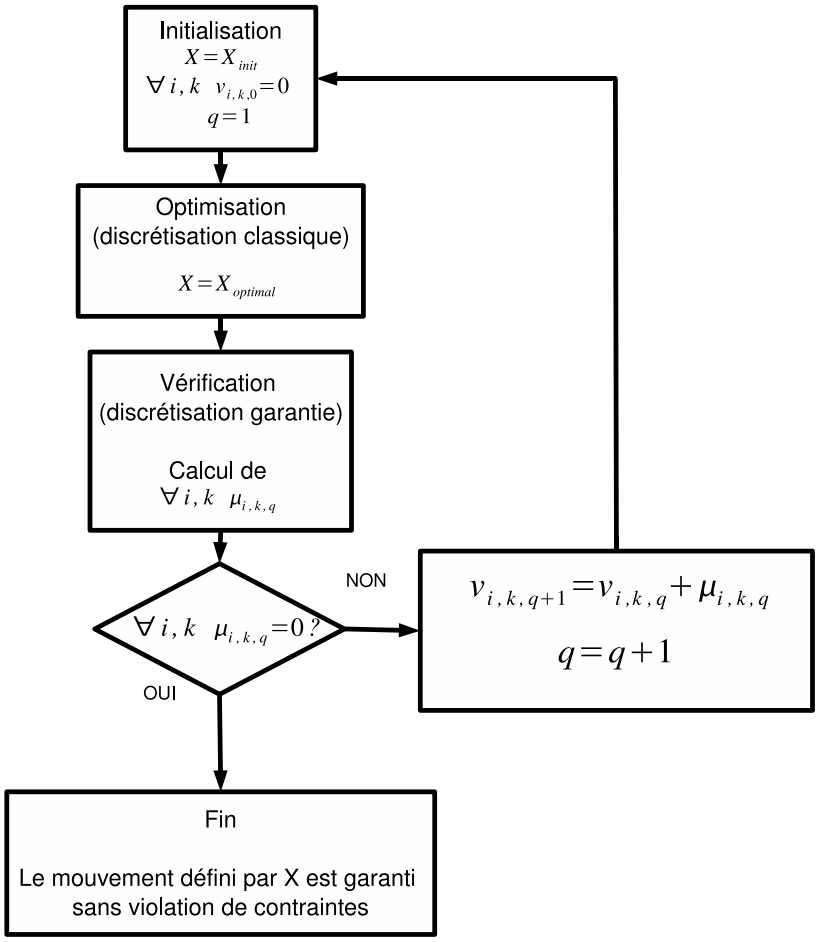


FIG. 4.14 – Algorithme hybride

4.8 Comparaison des méthodes de planification

Le Tableau 4.4 présente les temps de calcul et les résultats des méthodes de planification pour des mouvements ne violant pas les contraintes. La méthode de planification classique fournit un résultat pour une grille de 301 instants en 13 minutes. Cependant, si on considère que pour arriver à ce résultat, l'utilisateur a dû lancer le processus d'optimisation pour des grilles de (7, 13, 31, 61 ,121) on peut considérer que le temps de calcul total est d'approximativement 22 minutes.

Ce tableau reflète bien le fait que la planification classique est rapide mais sous-optimale (car le critère est plus élevé) par rapport à une méthode de planification garantie qui est beaucoup plus longue.

Nous avons développé une méthode de planification hybride qui permet de regrouper les qualités des deux méthodes présentées précédemment. En effet, la méthode de planification permet d'obtenir une mouvement optimal qui ne viole pas les contraintes et possède un temps de calcul semblable à celui des méthodes classiques.

itération	nombre d'itérations	valeur du critère (s)	temps de calcul	violation maximale des contraintes	temps de calcul total (s)
1	34	3.885e-01	41.4s	5.4 %	421.4
2	30	3.992e-01	33.8s	0.25 %	835.2
3	5	3.993e-01	7.6s	$1e^{-5}$ %	1222.8
4	5	3.993e-01	8.4s	pas de violation	1611.2

TAB. 4.3 – Temps de calcul des optimisations de la méthode hybride. Chaque itération est suivie d'un calcul de validité des contraintes qui dure 380 s.

planification	critère (s)	temps de calcul
classique	4.010 e-01	(13 mn) 22mn
garantie	3.933e-01	28h36mn
hybride	3.933e-01	27mn

TAB. 4.4 – Comparaison des méthodes de planification (classique, garantie, hybride).

4.9 Conclusion

Dans ce chapitre, nous avons présenté la planification de mouvement comme la résolution d'un problème de programmation semi-infinie. Les méthodes classiques de planification utilisent un processus de discrétisation des contraintes 'inégalité' continues basées sur une grille temporelle afin de pouvoir résoudre le problème de planification par des logiciels d'optimisation. Nous avons montré que cette méthode est dangereuse car elle ne donne aucune information sur la validité des contraintes entre deux instants de la grille.

La planification de mouvements sûrs utilise la discrétisation garantie des contraintes 'inégalité' continue afin de certifier la validité des contraintes sur toute la durée du mouvement. Cependant, cette méthode est très coûteuse en temps de calcul car elle nécessite un grand nombre de bissection.

Afin de pouvoir générer un mouvement respectant les limites du robot dans un laps de temps comparable aux méthodes classiques, nous avons présenté une méthode hybride qui utilise la discrétisation garantie pour vérifier ces contraintes pour des mouvements générés grâce à une discrétisation classique.

Nous avons donc proposé une méthode de planification de mouvements qui assure la sûreté de fonctionnement du robot pour des temps de calcul qui avoisinent les méthodes classiques. Nous avons utilisé la méthode hybride pour générer un ensemble de mouvements de pas utilisés pour la navigation globale du robot dans le chapitre 6.

Chapitre 5

Re-planification rapide de mouvements

Sommaire

5.1 Introduction

Nous définissons la re-planification de mouvements comme la génération d'un nouveau mouvement en se basant sur un mouvement déjà existant. Les mouvements optimaux générés dans le chapitre 4, sont parfaitement conçus et adaptés pour une situation donnée. Pour d'autres situations ils peuvent s'avérer sous-optimaux, inefficaces ou même dangereux. La re-planification de mouvements permet, donc, de pouvoir générer rapidement des nouveaux mouvements qui seront mieux adaptés à la situation actuelle que les mouvements optimaux pré-calculés.

Dans ce chapitre, nous allons tout d'abord présenter le mouvement que nous voulons adapter, puis nous montrerons comment calculer un sous-ensemble faisable autour des paramètres de ce mouvement. Finalement, la re-planification reviendra à chercher une solution dans ce sous-ensemble la mieux adaptée à la nouvelle tâche.

5.2 Le problème étudié

5.2.1 Enjeux

Afin de mieux percevoir ces enjeux, nous allons nous intéresser au mouvement de coup de pied dans une balle. Lors des précédentes éditions de compétitions comme la Robocup, le mouvement de coup de pied était généralement un mouvement calculé hors ligne [Carpin 06]. Il était toujours le même quelle que soit la position de la balle ou du but. Dans nos travaux de thèse, nous proposons une méthode de re-planification de mouvement et l'appliquons dans le cas d'un changement de position de la balle, mais elle peut également s'appliquer pour un changement de position du but, ou à d'autres mouvements.

5.2.2 Paramètres du mouvement

Dans un premier temps, nous allons générer un mouvement optimal de coup de pied qui sera caractérisé par position du point d'impact par rapport à l'extrémité du pied de support du robot (la hauteur h et la longueur x) qui aura lieu à la moitié du mouvement (cf. Figure 5.1). La deuxième étape concernera la re-planification de ce mouvement avec de nouveaux paramètres x et h.

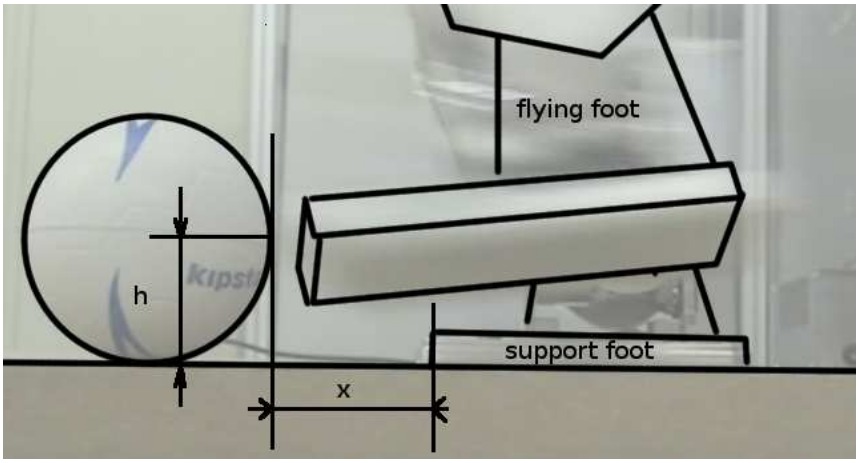


FIG. 5.1 – Schéma des paramètres (x et h) du coup de pied.

Le critère que nous voulons minimiser est l'énergie électrique consommée exprimée par l'équation 3.7. De plus, nous voulons imposer des postures initiale et finale identiques durant lesquelles le robot a les pieds côte à côte. Le modèle considéré est celui en simple support avec les contraintes présentées dans la section 3.8.2.

5.2.3 Mouvement optimal

Nous avons utilisé la méthode hybride présentée dans la section 4.7 pour planifier le mouvement de coup de pied dans une balle de $6cm$ de diamètre. Nous souhaitons un impact situé à $3cm$ de hauteur et éloigné du pied de support de $1cm$ de longueur ($x = 1cm$, $h = 3cm$). Nous choisissons d'utiliser 5 paramètres de B-splines par articulations, ce qui donne 61 paramètres à optimiser (5×12 paramètres et la durée du mouvement T).

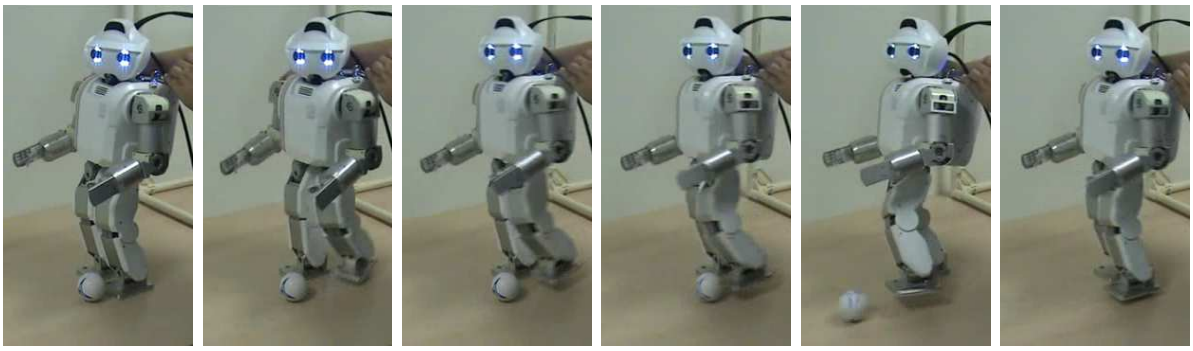
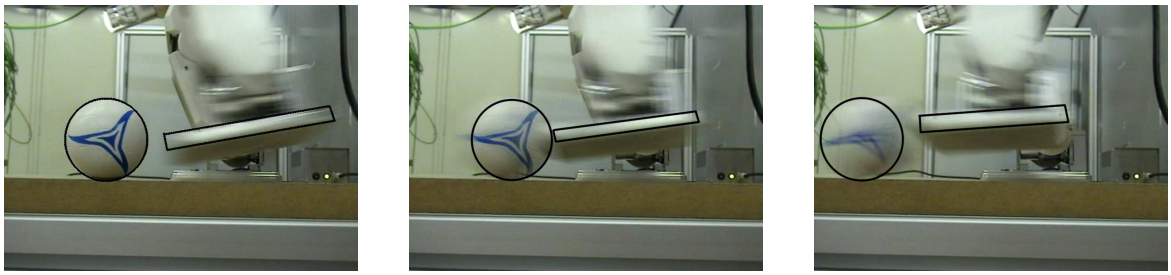


FIG. 5.2 – Mouvement optimal avec la balle placée à $1cm$



(a) avant la collision (b) collision (c) après la collision

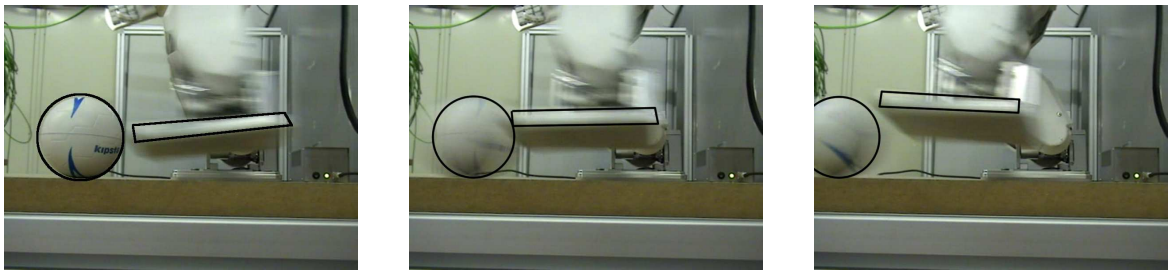
FIG. 5.3 – Représentation de l'impact pour un mouvement optimal avec la balle à $x = 1cm$

Le tableau 5.1 montre le résultat de l'optimisation du mouvement de coup de pied avec la méthode hybride. Le temps de calcul total est de 5787 secondes (soit 1 heure et 37 minutes) pour le mouvement optimal présenté sur la Figure 5.2 avec un impact représenté sur la Figure 5.3. On s'aperçoit que le pied entre en contact avec la balle à la hauteur voulue, c'est à dire $3cm$. Le mouvement optimal calculé hors ligne est donc parfaitement adapté quand la balle est à la position désirée.

itération	nombre d'itérations	valeur du critère (J)	temps de calcul	violation maximale des contraintes	temps de calcul total (s)
1	134	41.39	763 s	12.4%	1063 s
2	117	45.61	650 s	7.5 %	2013 s
3	134	46.47	747 s	1.5 %	3060 s
4	139	46.72	774 s	0.12 %	4134 s
5	104	46.76	579 s	0.045 %	5013 s
6	85	46.80	474 s	pas de violation	5787 s

TAB. 5.1 – Temps de calcul de l'optimisation d'un mouvement de coup de pied ayant les paramètres ($x = 1cm$, $h = 3cm$) avec la méthode hybride. Chaque itération est suivie d'un calcul de validité des contraintes qui dure 300 s.

Mais que se passe-t-il si la balle n'est pas à l'endroit attendu ? La figure 5.4 montre l'impact de la balle pour le mouvement optimal obtenu précédemment alors que la balle est positionnée à $x = 3cm$. Il apparaît clairement que le pied percute la balle plus haut



(a) avant la collision (b) collision (c) après la collision

FIG. 5.4 – Représentation de l'impact pour un mouvement optimal avec la balle à $x = 3cm$ au lieu de $x = 1cm$.

que précédemment, ce qui affectera la transmission d'énergie et entraînera une trajectoire de la balle différente de celle voulue (plus courte dans ce cas).

5.2.4 Calcul d'un mouvement adapté

Afin de générer un mouvement adapté, il est possible de relancer le processus d'optimisation en prenant comme condition initiale le mouvement optimal obtenu dans la section 5.2.3 en considérant les mêmes contraintes 'inégalité' continues et de nouvelles contraintes 'égalité' liées à la position de la balle.

itération	nombre d'itérations	valeur du critère (J)	temps de calcul de calcul	violation maximale des contraintes	temps de calcul total (s)
1	114	44.23	732 s	3.23 %	1032 s
2	99	45.20	641 s	2.38 %	1973 s
3	63	45.36	410 s	pas de violation %	2683 s

TAB. 5.2 – Temps de calcul pour l'optimisation du mouvement adapté ayant les paramètres ($x = 3cm$, $h = 3cm$) avec la méthode hybride. Chaque itération est suivie d'un calcul de validité des contraintes qui dure 300 s.

Le tableau 5.2 montre les temps de calcul de l'optimisation du mouvement adapté. La re-planification est plus rapide que l'optimisation initiale (45 minutes contre 1 heure 37 minutes précédemment). De plus le mouvement obtenu parait bien adapté à la position de la balle. Cependant le temps de calcul (2683 secondes soit 45 minutes) ne permet pas une application de cette méthode pendant une application réelle, il faut donc utiliser une autre méthode pour faire la re-planification de ce mouvement de coup de pied.

5.3 La Méthode de Re-Planifcation Rapide

5.3.1 Principe

Nous proposons donc une méthode, publiée dans [Lengagne 09a], qui va modifier le mouvement de coup de pied précédemment calculé, avec un temps de calcul assez court pour être effectué en ligne. Dans notre cas, nous considérons que le temps de génération d'un mouvement adapté doit être de quelques secondes pour être acceptable.

Nous définissons X_{free}, l'ensemble des paramètres $\mathbf{X}$ qui vérifie l'ensemble des contraintes 'inégalité' continue :

$$X_{free} = \{\mathbf{X}, \forall i, \forall t \in [0,T], g_{i_c}(\mathbf{X}, t) \leq 0\} \tag{5.1}$$

Le problème du temps de calcul est principalement dû aux contraintes 'inégalité' continues $\forall t \in [0,T],\ g_c(X,t) < 0$ qui sont très longues à calculer. L'idée principale de notre méthode de re-planification est de remplacer la recherche d'une solution dans X_{free} par la recherche d'une solution dans un sous-ensemble faisable de X_{free} : $[X]$, sous-ensemble qui serait de forme simple et dont les limites sont définies par des équations linéaires par rapport aux paramètres. La recherche dans ce sous-ensemble permet de ne pas calculer l'ensemble des équations $g_{i_c}(X,t)$ qui sont non-linéaires et coûteuses en temps de calcul.

Les contraintes 'égalité', quant à elles, sont généralement des contraintes géométriques qui permettent de définir le mouvement, il faut donc prendre en compte un nouvel ensemble de contraintes 'égalité' h'_k qui caractérise la nouvelle position de la balle.

Le critère à minimiser F' peut être identique à celui de l'optimisation hors ligne. Cependant pour avoir un temps de calcul faible il est intéressant de considérer un critère qui ne dépend que des paramètres d'optimisation ou alors de ne pas tenir compte du critère ce qui revient à résoudre un problème de satisfaction de contraintes.

Le re-planification rapide d'un mouvement revient donc à calculer, hors-ligne, le sous-ensemble faisable $[X]$ qui est une approximation intérieure de X_{free}, puis à résoudre le problème présenté dans l'équation 5.2, afin de générer un mouvement adapté.

$$\begin{array}{cc} \text{minimiser} & F'(\hat{X}, t) \\ \text{avec} & \hat{X} \in [X] \\ \text{et} & \forall j \quad h'_j(\hat{X}) = 0 \end{array} \tag{5.2}$$

5.3.2 Sous-Ensemble Faisable

Des études récentes, se sont intéressées au calcul d'ensembles faisables, en utilisant l'Analyse par Intervalles, afin de pouvoir dimensionner les actionneurs de robots parallèles ou série [Ramdani 08, Oetomo 09]. Évidemment, pour que le robot puisse s'adapter à un grand nombre de situations, il nous faut calculer le plus grand sous-ensemble $[X]$. Ce sous-ensemble doit contenir le vecteur optimal $\tilde{X}$ (obtenu précédemment par optimisation) et uniquement des solutions appartenant à l'espace libre X_{free} qui satisfont les contraintes 'inégalité' continues qui sont liées aux limites physiques.

Dans notre cas, nous ne disposons d'aucune information sur la forme ou les propriétés de l'ensemble faisable. Nous cherchons, donc, une approximation intérieure de l'espace libre X_{free} : le sous-ensemble $[X]$, qui sera une boîte à n dimensions obtenue en résolvant le problème suivant :

$$\begin{array}{cc} \text{maximiser} & \delta \in \mathbb{R}^+ \\ \text{avec } \forall b & [X_b] = \tilde{X}_b + \delta \times [\mathbb{W}_b] \\ \text{et } \forall b & 0 \in [\mathbb{W}_b] \\ \text{et} \forall i_c,\ \forall X \in [X],\ \forall t \in [0,T] & g_{i_c}(X,t) < 0 \end{array} \tag{5.3}$$

Nous définissons δ comme la largeur normalisée de la boîte et $[\mathbb{W}]$ un vecteur intervalle, choisi par l'utilisateur, qui va influencer la forme de la boîte afin d'ignorer ou d'accentuer la recherche suivant certaines directions de la boîte $[X]$. Il est évident que le résultat final dépendra de $[\mathbb{W}]$ et que son choix doit être fait judicieusement.

5.3.3 Algorithme

L'algorithme de calcul du sous-ensemble faisable $[X]$ comporte deux étapes :
- Un calcul initial qui permet d'obtenir un sous-ensemble de l'espace libre,
- Une procédure d'expansion qui va augmenter ce sous-ensemble suivant certaines directions.

Calcul initial

La figure 5.5 montre le principe de cet algorithme qui calcule le sous-ensemble $[X]$. Le principe général de l'algorithme, présenté Figure 5.5 est de commencer à partir d'un sous-ensemble très grand (δ grand), et de le réduire en rejetant toute solution $[z]$ qui ne respecte pas l'ensemble des contraintes. Nous choisissons la matrice $[\mathbb{W}]$ de telle manière que chacune de ces composantes soit la distance entre le vecteur optimal $\tilde{X}$ et la frontière de l'espace libre X_{free}.

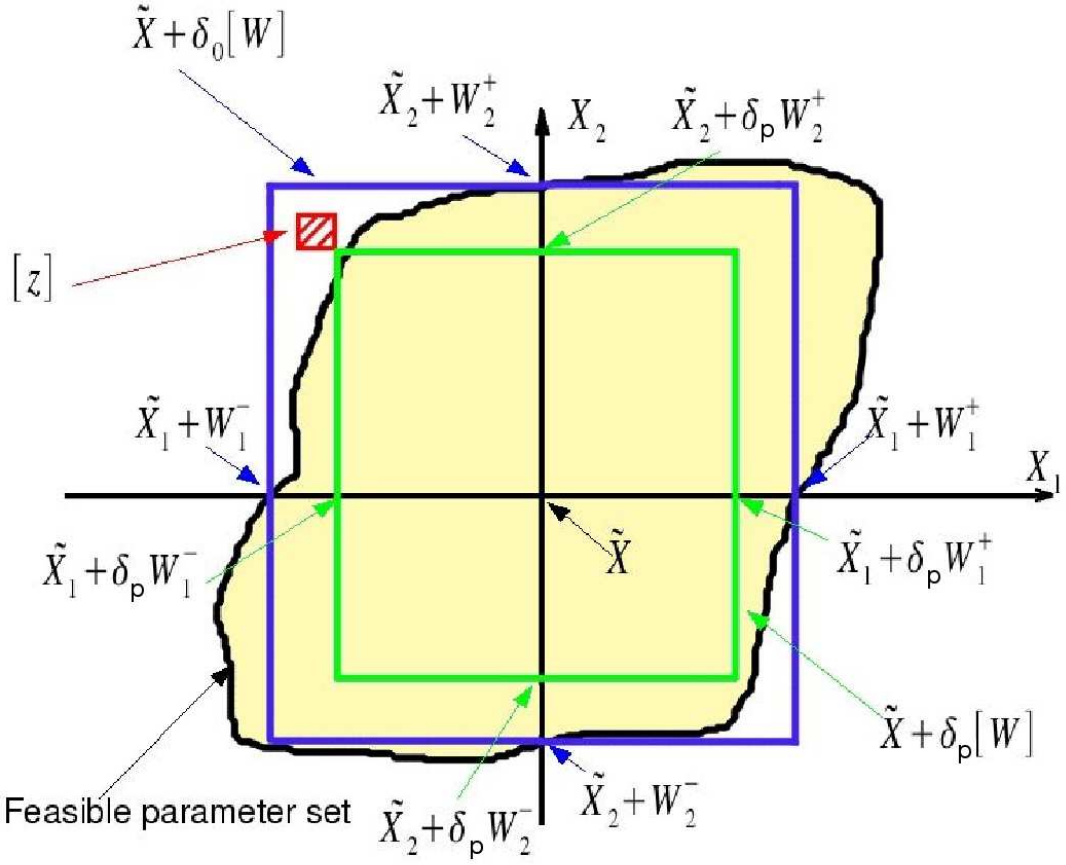


FIG. 5.5 – Exemple d'une ensemble faisable et de son approximation intérieure : $[X]$

L'utilisation de ALIAS [ali], un algorithme de résolution de problème d'optimisation utilisant l'Analyse par Intervalles, nous permet de calculer la boîte $[z]$ qui viole au moins une contrainte 'inégalité'. La boîte $[z]$ est calculée en résolvant :

$$\begin{array}{cc} \text{dans} & [X] = \tilde{X} + \delta_p[\mathbb{W}] \\ \text{trouver} & [z] \subset [X] \\ \text{tel que } \exists i_c, \exists t \in [t] & Sup[g]_{i_c}([z], t) > 0 \end{array} \tag{5.4}$$

Une fois l'existence de $[z]$, prouvée et sa valeur calculée, l'algorithme réduit la boîte en calculant la nouvelle valeur δ_{p+1} tel que :

$$[z] \ \cap \ \tilde{X} + \delta_{p+1}[\mathbb{W}] = \emptyset \tag{5.5}$$

L'algorithme s'arrêtera quand il n'y aura plus de solution $[z]$ au problème 5.4. Finalement, le sous-ensemble $[X] = \tilde{X} + \tilde{\delta}[\mathbb{W}]$ constitue un sous-ensemble faisable des contraintes 'inégalité' continues.

Expansion

En observant la Figure 5.6, on peut remarquer que le sous-ensemble $[X]$ pourrait être agrandi suivant les directions X_1^+ ou X_2^-. Nous nous intéressons, donc, à l'ajout d'une étape d'expansion du sous-ensemble faisable afin de l'augmenter et d'obtenir le sous-ensemble faisable étendu $[\mathbb{X}]$.

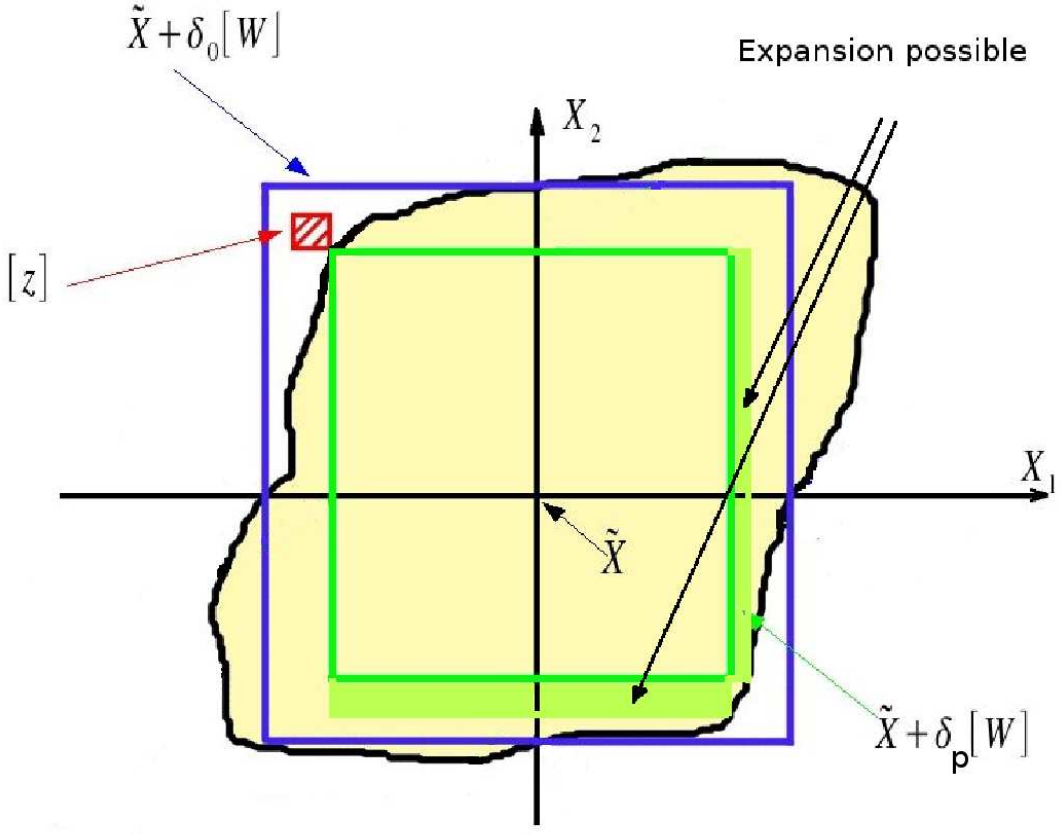


FIG. 5.6 – Représentation des expansions possibles du sous-ensemble $[X]$

L'expansion du sous-ensemble faisable suivant une direction définie par r (le numéro du paramètre à étendre) et $\circ \in \{-;+\}$ (le sens négatif ou positif de l'expansion) se fait en résolvant le système :

$$\begin{array}{cc} \text{maximiser} & \rho_r \in \mathbb{R}^+ \\ \text{avec} & [\mathbb{X}] = [X] + \rho_r \times \mathbb{E}_r^\circ \\ \text{et } \forall i_c, \ \forall X \in [\mathbb{X}], \ \forall t \in [0,T] & g_{i_c}(X,t) < 0 \end{array} \tag{5.6}$$

Avec $\mathbb{E}_r^\circ$, un vecteur de même dimension que $[X]$, ne contenant que des intervalles nuls sauf pour la composante r qui vaut $[-1;0]$ pour $\circ \equiv -$ et $[0;1]$ pour $\circ \equiv +$.

L'expansion totale du sous-ensemble implique d'effectuer une expansion dans toutes les directions. Cependant l'ordre dans lequel sera étendu le sous-ensemble aura un impact sur le bénéfice obtenu. En effet, la Figure 5.6 montre bien qu'une expansion suivant X_1^+ diminuerait l'effet d'une deuxième expansion suivant X_2^- (et inversement).

5.4 Re-planification du mouvement de coup de pied

5.4.1 Choix des paramètres à adapter

Calcul initial

Le mouvement optimal calculé dans la section 5.2.3 est défini à partir de 61 paramètres (60 coefficients de splines ainsi que la durée du mouvement). Nous décidons de ne pas adapter le mouvement suivant tous les paramètres parce que :

- le sous-ensemble faisable autour du mouvement optimal serait trop étroit pour pouvoir contenir des paramètres correspondant à des mouvements très différents,
- certains paramètres n'ont pas d'effet sur la position et l'orientation du pied au moment de l'impact.

Nous allons donc limiter le nombre de paramètres à adapter. Dans notre cas, nous nous intéressons à adapter le mouvement en fonction de la distance entre la balle et le pied d'appui au moment de l'impact x. Nous proposons, donc, d'adapter les trajectoires articulaires qui ont un effet important sur la position du pied dans le plan sagittal, à savoir les genoux et les articulations de tangage des hanches et des chevilles nommés XLEG_JOINT[3,4,5] sur la Figure 3.20, $X \in \{R, L\}$.

De plus la collision entre le pied du robot et la balle a lieu à la moitié de la durée du mouvement, on voit sur la Figure 3.1 que le troisième coefficient aura un effet prépondérant sur les trajectoires articulaires au moment de l'impact. Nous choisissons donc d'adapter uniquement les 6 paramètres correspondant au troisième coefficient de B-splines des articulations responsables du mouvement dans le plan sagittal.

Ordre pour l'expansion

L'ordre des différentes directions pour la phase d'expansion du sous-ensemble faisable est très important pour l'efficacité de cette opération. Il nous faut classer les paramètres suivant deux critères :

- La sensibilité sur la position du pied : le but est d'obtenir un sous-ensemble qui permette la re-planification la plus large possible. Il faut donc privilégier les paramètres qui auront le plus d'influence sur la position du pied.
- La sensibilité par rapport aux contraintes : afin d'obtenir un sous-ensemble le plus grand possible, il faut accentuer la recherche pour les paramètres qui n'amènent pas rapidement à une violation de contraintes.

En résumé, il faut déterminer quels sont les paramètres qui vont maximiser les variations de la position du pied sans provoquer de violation de contraintes. Nous choisissons d'établir l'ordre suivant :

1. Cheville de la jambe en phase de vol,
2. Genou de la jambe en phase de vol,

3. Hanche de la jambe en phase de vol,
4. Hanche de la jambe de support,
5. Genou de la jambe de support,
6. Cheville de la jambe de support.

5.4.2 Calcul du sous-ensemble faisable

Articulation	$\tilde{X}_b$	$\underline{X}_b$	$\overline{X}_b$
Hanche de la jambe de support	2.65	1.10	3.90
Genou de la jambe de support	15.01	14.98	17.14
Cheville de la jambe de support	-12.45	-13.54	-10.81
Hanche de la jambe en phase de vol	-14.18	-17.18	-12.77
Genou de la jambe en phase de vol	44.95	36.26	44.98
Cheville de la jambe en phase de vol	-28.14	-28.30	-24.73

TAB. 5.3 – Tableau des valeurs du sous-ensemble obtenu après le calcul initial.

Les Tableaux 5.3 et 5.4 présentent le sous-ensemble obtenu avant et après le processus d'expansion. La valeur initiale du paramètre à adapter est notée $\tilde{X}_b$, alors que $\underline{X}_b$ et $\overline{X}_b$ définissent les bornes inférieures et supérieures du sous-ensemble faisable $[X]$.

Sur ces deux tableaux, on peut remarquer que la largeur du sous-ensemble faisable, suivant un paramètre n'est pas constante. Cependant, il est intéressant de voir que certains paramètres peuvent évoluer dans une gamme importante de valeurs (presque 7 degrés avec l'expansion, jusque 13 degré après) sans provoquer de violation de contraintes.

Articulation	$\tilde{X}_b$	$\underline{\mathbb{X}}_b$	$\overline{\mathbb{X}}_b$
Hanche de la jambe de support	2.65	-2.65	7.10
Genou de la jambe de support	15.01	14.91	22.14
Cheville de la jambe de support	-12.45	-16.35	-7.33
Hanche de la jambe en phase de vol	-14.18	-22.18	-9.17
Genou de la jambe en phase de vol	44.95	31.26	45.07
Cheville de la jambe en phase de vol	-28.14	-28.70	-19.73

TAB. 5.4 – Tableau des valeurs du sous-ensemble obtenu après l'expansion.

Le Tableau 5.4 montre l'utilité du processus d'expansion, par rapport au tableau 5.3. En effet, l'ensemble faisable a augmenté suivant toutes les directions. La Figure 5.7 représente l'ensemble (x, h) réalisable pour tous les mouvements contenus dans le sous-ensemble faisable, elle montre également que l'utilisation du processus d'expansion a augmenté l'ensemble (x, h) réalisable.

Si on ne considère que les mouvements qui généreront un coup de pied situé à $h = 3cm$, nous pouvons affirmer que le sous-ensemble faisable $[\mathbb{X}]$ contient des mouvements permettant une position de la balle $x \in [-1cm; 5cm]$ soit une augmentation de 500% de l'éloignement de la balle. De la même manière en ne considérant que les mouvements qui correspondent à une balle situé à $x = 1cm$, il sera possible de trouver, dans $[\mathbb{X}]$, des mouvements dont la hauteur d'impact se situera entre $h \in [2cm; 3.7cm]$.

La Figure 5.7 représente l'ensemble des valeurs (x, h) possibles à partir du mouvement optimal calculé dans la section précédentes. Même si le sous-ensemble $[X]$ des paramètres

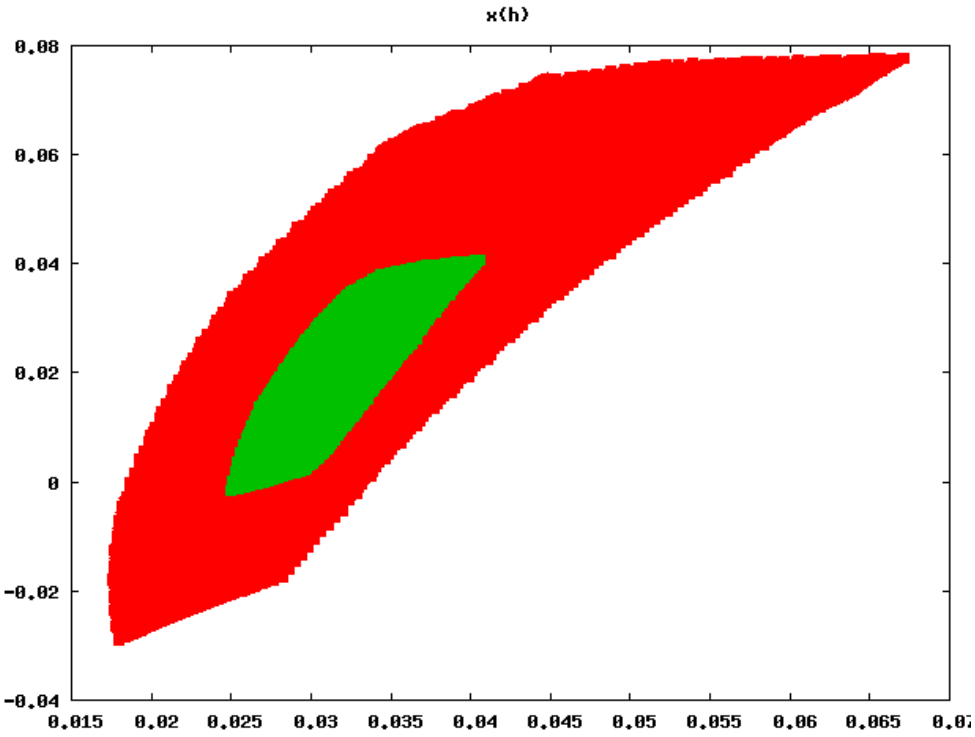


FIG. 5.7 – Représentation de l'ensemble des caractéristiques (x, h) pour les mouvements du sous-ensemble faisable avant l'expansion (en vert) et après l'expansion (en rouge) autour du mouvement optimal $x = 1cm$, $h = 3cm$. x en ordonnées et h en abscisse sont exprimé en mètre.

faisables du mouvement a une forme de boîte, on s'aperçoit que l'ensemble des valeurs (x, h) possibles n'est pas une boîte, mais peut être délimité par des arcs de cercles. Ce phénomène peut être rapproché du fait que les articulations du robot sont rotoïdes.

5.4.3 Recherche du Mouvement Adapté

Nous souhaitons adapter le mouvement de coup de pied pour qu'il ait les caractéristiques $x = 3cm$ et $h = 3cm$. La Figure 5.7 montre que ces caractéristiques sont réalisables par un (ou plusieurs) mouvement appartenant à l'espace faisable $[\mathbb{X}]$. Pour cela il nous faut résoudre le problème d'optimisation suivant :

$$\begin{array}{ll} \text{trouver} & \hat{X} \in [\mathbb{X}] \\ \text{tel que} & h(\frac{T}{2}) = 3cm \\ & x(\frac{T}{2}) = 3cm \end{array} \tag{5.7}$$

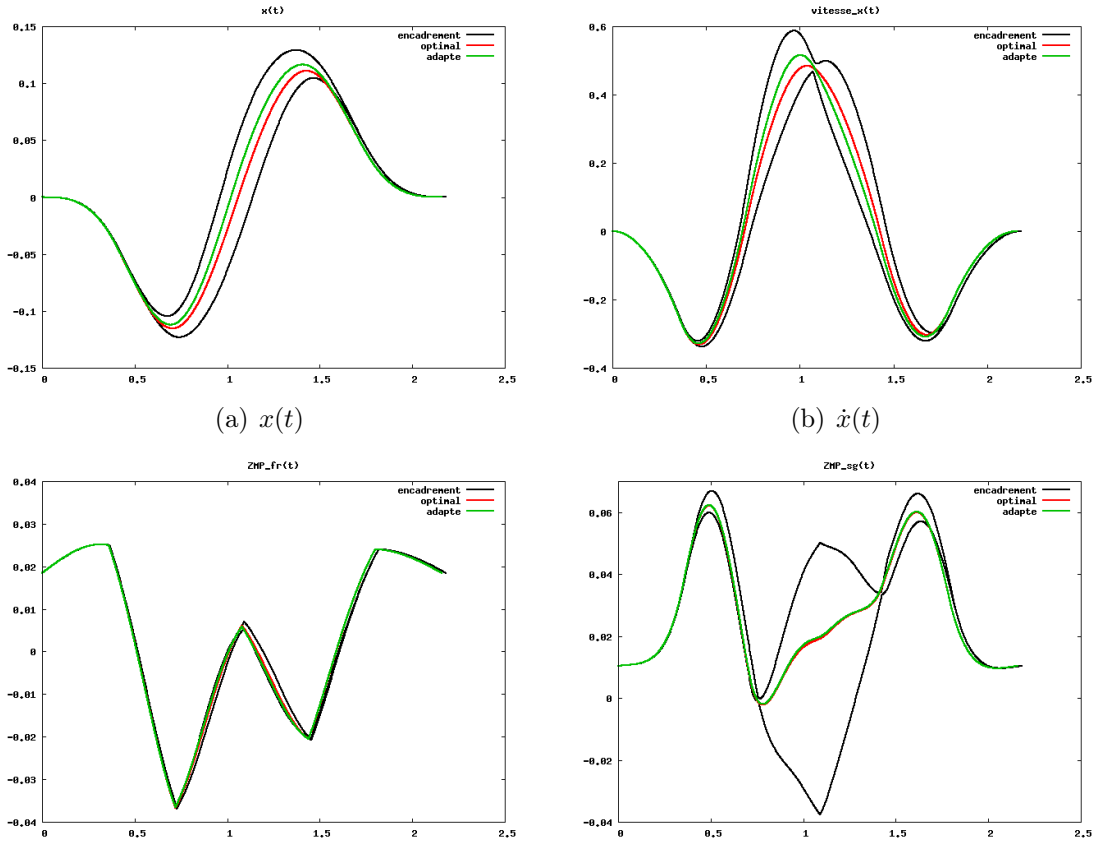
La résolution de ce problème d'optimisation s'est faite en 22 itérations pour un temps de calcul de 1.5 seconde et un critère de 47.45 Joules avec une modification des paramètres présenté sur la tableau 5.5.

Dans la section 5.2.4, nous avons généré un mouvement optimal par un processus d'optimisation de tous les paramètres, le critère obtenu était de $45.36J$. La re-planification que nous venons de présenter ne prend en compte que six paramètres sans considérer de critère, elle a généré un mouvement consommant $47.45J$, soit 4.6% de plus que le mouvement optimal.

Articulation	$\tilde{X}_b$	$\hat{X}_b$
Hanche de la jambe de support	2.65	7.03
Genou de la jambe de support	15.01	22.11
Cheville de la jambe de support	-12.45	-7.33
Hanche de la jambe en phase de vol	-14.18	-22.14
Genou de la jambe en phase de vol	44.95	45.05
Cheville de la jambe en phase de vol	-28.14	-28.69

TAB. 5.5 – Tableau des valeurs optimales et adaptées.

La méthode présentée nous permet donc d'obtenir un mouvement avec une consommation d'énergie très proche de l'optimale en seulement 1.5 seconde de temps de calcul, nous pouvons donc la définir comme une méthode de re-planification rapide.



(a) $x(t)$ (b) $\dot{x}(t)$

(c) ZMP dans le plan frontal (d) ZMP dans le plan sagittal

FIG. 5.8 – Représentation temporelle de plusieurs grandeurs pour le mouvement optimal, le mouvement adapté ainsi que l'ensemble des mouvements faisables appartenant à $[\mathbb{X}]$.

La Figure 5.8 montre l'évolution temporelle de la position et de la vitesse de l'extrémité du pied $x(t)$, $\dot{x}(t)$ et du ZMP dans le plan frontal et sagittal pour les mouvements optimal et re-planifié. L'encadrement correspond aux valeurs maximales des mouvements appartenant à $[\mathbb{X}]$.

La Figure 5.8(a) montre la gamme des positions x possibles, le mouvement re-planifié produit bien une valeur x égale à 3cm à la moitié du mouvement. On peut également remarquer la vitesse du pied, représentée Figure 5.8(b), au moment de l'impact varie très

peu par rapport à la valeur du mouvement optimal (la vitesse de l'impact était fixé à $0.5m/s$), ceci est dû au choix des paramètres de re-planification, en effet on peut voir sur la Figure 3.1 que le troisième coefficient de splines ne modifie pas la vitesse angulaire à $t = \frac{T}{2}$, le modèle cinématique ne diffère, donc, qu'au niveau des positions angulaires.

Nous avons choisi d'adapter les paramètres qui ont une influence dans le plan sagittal, le ZMP dans le plan frontal ne peut donc que très légèrement varier, comme le prouve la Figure 5.8(c). Inversement la modification de la position de la balle se fait dans le plan sagittal ce qui produit un ZMP dans le plan sagittal pouvant varier dans des gammes de valeurs importantes. Cependant le mouvement re-planifié produit une évolution de cette grandeur semblable au mouvement optimal, ce qui peut expliquer que le mouvement re-planifié sera proche de l'optimal, comme le montre la consommation d'énergie ($46.8J$ pour l'optimal, contre $47.45J$ pour le mouvement re-planifié) ou encore la Figure 5.9 qui représente le mouvement de coup de pied re-planifié.

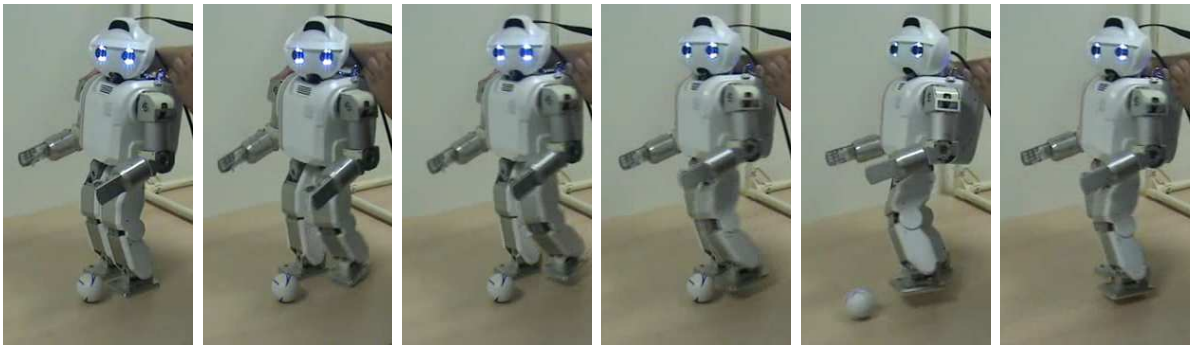
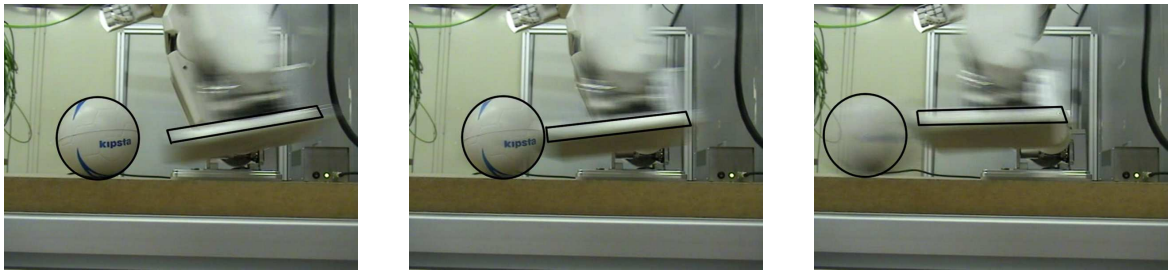


FIG. 5.9 – Mouvement re-planifié avec la balle placée à $3cm$

Le mouvement re-planifié produit l'impact désiré (cf. Figure 5.10), le pied entre en collision avec la balle placée à $x = 3cm$ à la hauture voulue de $h = 3cm$.



(a) avant la collision (b) collision (c) après la collision

FIG. 5.10 – Représentation de l'impact pour le mouvement re-planifié avec la balle placée à $3cm$

5.5 Conclusion

La re-planification de mouvements est le processus qui génère un nouveau mouvement en se basant sur un mouvement déjà existant, elle est utilisée pour adapter des mouvements optimaux à un environnement légèrement différent de celui pour lequel ils ont été calculés.

Dans ce chapitre, nous avons mis en évidence que la re-planification de mouvement ne peut pas se faire par un processus d'optimisation classique, même si les paramètres initiaux sont ceux d'un mouvement proche, à cause du temps de calcul trop important. Nous avons établi que le temps de l'optimisation était fortement augmenté par le calcul des contraintes 'inégalité' qui transcrivent les limites du robot. Toutes ces limites permettent de définir l'ensemble de paramètres X_{free} qui contient les paramètres de tous les mouvements respectant les limites physiques.

La première étape de notre méthode de re-planification est de calculer un sous-ensemble $[X]$ qui soit une approximation intérieure de l'ensemble X_{free} et dont les frontières s'expriment linéairement par rapport aux paramètres du mouvement. Dans notre cas, nous avons utilisé un sous-ensemble sous forme de boîte, ce qui correspond à limiter la valeur des paramètres entre deux valeurs, mais il est envisageable de considérer d'autres formes à base de polygone ou d'ellipse tout en conservant la propriété de linéarité.

La seconde étape étant de rechercher dans ce sous-ensemble $[X]$ une solution qui satisfait les nouvelles contraintes 'égalité' liées à la modification de l'environnement. A partir d'un mouvement de coup de pied optimal correspondant à une position de la balle de $1cm$, nous avons pu générer un mouvement quasiment optimal pour une position de la balle de $3cm$ en un temps de 1.5 secondes.

Chapitre 6

Expérimentations

Sommaire

6.1 Introduction

Précédemment, nous avons montré comment générer et adapter des mouvements optimaux pour les robots humanoïdes. Le but de ce chapitre est de valider expérimentalement les mouvements optimaux sûrs issus des méthodes de planification présentées dans le chapitre 4. Nous présentons donc une méthode de navigation simple qui va permettre d'enchaîner plusieurs mouvements de base afin de permettre au robot de rejoindre une cible mobile comme présenté sur la Figure 6.1.

Pour commencer, nous avons dû choisir des mouvements simples afin de constituer une base de données. Puis, nous avons testé une méthode classique de séquencement de pas, pour enfin développer une méthode directionnelle qui permet de gérer rapidement la navigation du robot HOAP-3 vers une cible mobile durant une expérimentation.

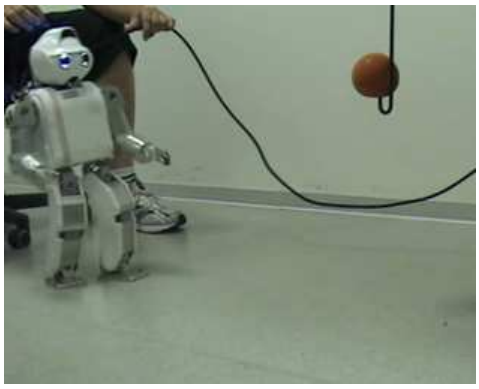


FIG. 6.1 – Expérimentation : le robot doit rejoindre une cible mobile.

6.2 Constitution d'une base de données

6.2.1 Postures

La posture du robot est définie par la position statique de tous les corps du robot au début ou à la fin d'un pas, elle dépend de la valeur des positions articulaires et se caractérise par la position du repère du pied en phase de vol, dans le repère du pied de support, à travers 3 paramètres L, l, α représentés sur la Figure 6.2 :

- L : la position, suivant l'axe z, du repère du pied droit exprimé dans le repère du pied gauche,
- l : la position, suivant l'axe y, du repère du pied droit exprimé dans le repère du pied gauche,
- α : l'orientation du repère du pied droit exprimé dans le repère du pied gauche.

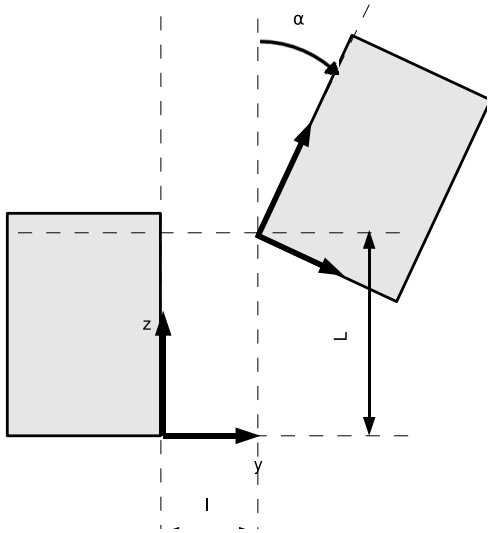


FIG. 6.2 – Caractérisation de posture

Nous souhaitons gérer la navigation du robot HOAP-3 sur un sol plan. La première étape est de définir le nombre de postures de la base de données ainsi que leurs propriétés. Dans notre cas, nous souhaitons obtenir un temps de calcul suffisamment faible pour

permettre une navigation réactive, fluide et proche de celle d'un humain. Trop peu de postures ne permettront pas d'obtenir la navigation voulue, alors qu'un nombre trop grand de postures ralentirait le processus. Nous avons donc choisi quatre postures différentes et leurs symétriques qui peuvent s'assimiler à quatre actions élémentaires.

N°	posture	L	l	α
1	pied en avant	5 cm	2 cm	0 °
2	pied en avant tourné vers l'extérieur	5 cm	2 cm	30 °
3	pied au milieu tourné vers l'extérieur	0 cm	2 cm	30 °
4	pied au milieu	0 cm	2 cm	0 °

TAB. 6.1 – Actions élémentaires

Nous disposons donc de quatre postures et de leurs postures symétriques dont les propriétés sont données par le tableau 6.1.

6.2.2 Pas

Nous définissons un pas comme le mouvement qui permet de passer d'une posture à une autre. Nous disposons de 8 postures (4 quand le robot est en appui sur le pied gauche et 4 quand il est sur le pied droit fixe), ce qui nous donne 64 (8×8) pas possibles. Afin de réduire ce nombre, nous :

- imposons le changement de pied d'appui entre deux pas,
- interdisons de tourner deux fois de suite (à droite puis à gauche et inversement) afin d'éviter la 'marche en canard',
- interdisons de faire deux changements d'appui consécutifs.

Finalement, il nous reste donc 24 pas possibles (12 pas et leurs symétriques) qui sont représentés par la Figure 6.3.

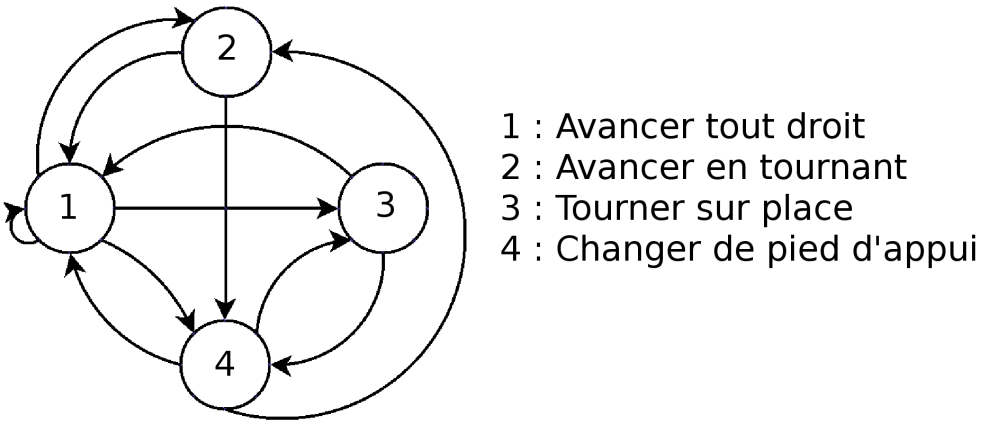
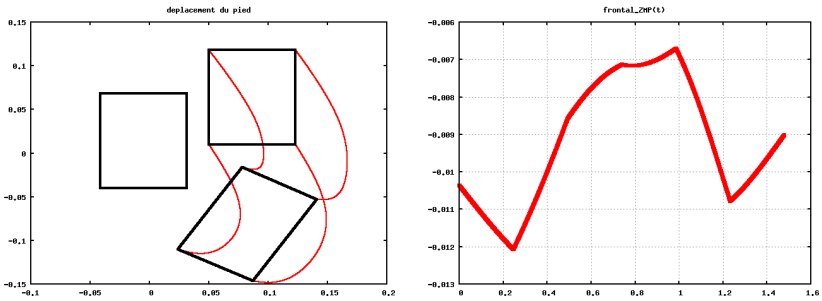


FIG. 6.3 – Graphe d'enchaînement des postures. Les postures sont représentées par des cercles et les pas par des flèches.

Un pas est composé d'un mouvement en simple support suivi d'un mouvement en double support permettant de changer le pied d'appui, afin de pouvoir effectuer le pas suivant.

La Figure 6.4 montre l'évolution du mouvement en simple support du pas qui permet de relier la posture 2 à la posture 1, ce pas est utilisé pour aller tout droit après avoir

tourné. Comme dans le chapitre 4, les mouvements de la base de données commencent et se terminent avec des vitesses et accélérations angulaires nulles.



(a) trajectoire du pied droit (b) évolution du ZMP dans le plan frontal

FIG. 6.4 – Exemple de pas reliant les postures 2 et 1.

6.3 Génération de la séquence de pas

Dans la section 2.1.3, nous avons donné un bref aperçu des méthodes existantes pour la planification de séquences. Dans cette section nous allons présenter une méthode intuitive pour la génération d'une séquence de pas.

6.3.1 Fonctionnement

La génération de séquences de pas se base sur la construction d'un arbre composé d'état et de branches. Un état correspond à la posture du robot se situant à une position donnée. Une branche est une succession d'états (donc une séquence de pas) qui ont permis d'amener le robot depuis l'état initial jusqu'à l'état courant. La position initiale du robot constitue la base de l'arbre. Pour déployer l'arborescence on ajoute, à chaque branche, les états valides qui peuvent succéder à l'état précédent jusqu'à rejoindre l'état final pour lequel le robot a un pied sur le point final.

Nous définissons un état comme valide quand :

- il existe un pas permettant de rallier la posture précédente à la posture courante (voir section 6.2.2),
- l'état (position et posture du robot) ne produit pas de collision avec l'environnement.

6.3.2 Détection de collision

L'algorithme doit, donc, disposer d'un test de collision pour savoir si un état est valide ou non. Nous avons développé deux tests de collisions :

- un test complet qui détermine exactement si il y a collision ou non,
- un test rapide qui a un temps de calcul très inférieur mais qui peut générer des faux positifs (le test indique une collision qui n'a pas lieu).

Test Complet

Le test de collision complet se fait en recherchant un point commun entre l'empreinte du pied et l'obstacle. Dans un soucis de simplicité, nous définissons les empreintes et

les obstacles comme des rectangles orientés qui sont donc définis par quatre droites (cf. Figure 6.5(a)). Tout point de l'obstacle ou de l'empreinte répond donc à l'ensemble des équations :

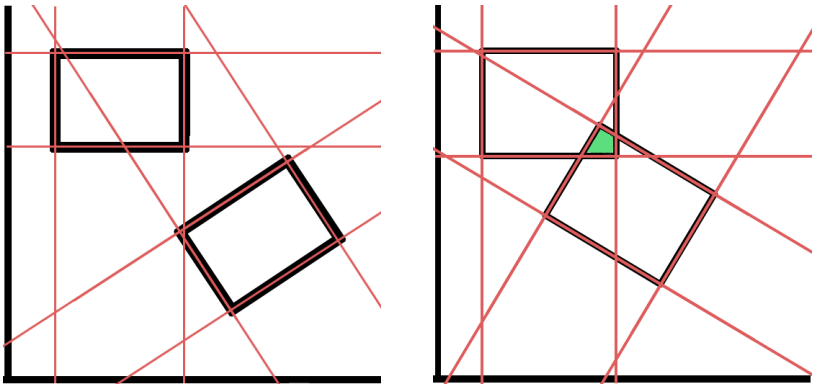
$$\begin{array}{cc} \text{empreinte du pied} & \text{obstacle} \end{array}$$

$$g_p(y,z) = \left\{ \begin{array}{l} a_{p,1}y + b_{p,1}z + c_{p,1} \leq 0 \\ a_{p,2}y + b_{p,2}z + c_{p,2} \leq 0 \\ a_{p,3}y + b_{p,3}z + c_{p,3} \leq 0 \\ a_{p,4}y + b_{p,4}z + c_{p,4} \leq 0 \end{array} \right\} \quad g_o(y,z) = \left\{ \begin{array}{l} a_{o,1}y + b_{o,1}z + c_{o,1} \leq 0 \\ a_{o,2}y + b_{o,2}z + c_{o,2} \leq 0 \\ a_{o,3}y + b_{o,3}z + c_{o,3} \leq 0 \\ a_{o,4}y + b_{o,4}z + c_{o,4} \leq 0 \end{array} \right\} \tag{6.1}$$

Pour déterminer si l'empreinte est en collision avec l'obstacle il suffit de déterminer s'il existe une solution au problème :

$$\begin{array}{rl} \text{trouver} & y, z \\ \text{tel que} & g_p(y,z) \leq 0 \\ \text{et} & g_o(y,z) \leq 0 \end{array} \tag{6.2}$$

S'il n'y a pas de solution on peut en conclure qu'il n'y a pas de collision (cf. Figure 6.5(a)), en revanche s'il existe une solution il est évident que le pied sera en collision avec l'obstacle (cf. Figure 6.5(b)). Contrairement à la méthode des axes séparateurs, souvent utilisées dans les jeux vidéos, la recherche d'un point commun peut être facilement étendue à des obstacles avec des formes complexes ou concaves. La résolution du problème 6.2 est confiée au logiciel C-FSQP [Lawrence] qui permet d'obtenir un résultat en quarante microsecondes. Lors d'une planification de séquence ce test peut être effectué un grand nombre de fois, nous décidons donc d'utiliser un test préalable qui soit plus rapide.

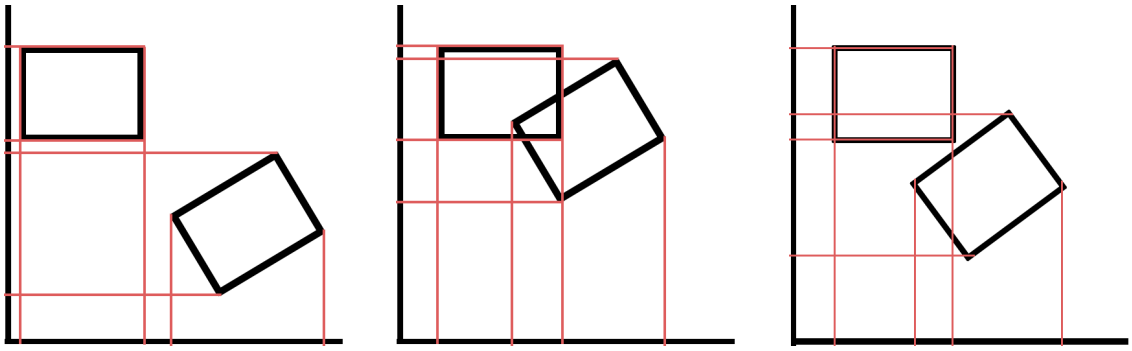


(a) Pas de collision (b) Collision détectée

FIG. 6.5 – Test complet

Test Simple

Afin de diminuer le temps de calcul nous avons implémenté un test plus simple qui va projeter l'empreinte et l'obstacle sur les axes du repère global. Si les intersections ne se chevauchent pas suivant un axe, il est certain que l'empreinte n'est pas en collision avec l'obstacle (cf . Figure 6.6(a)). Malheureusement, cette méthode peut produire des faux



(a) Pas de collision (b) Collision (c) Faux positif

FIG. 6.6 – Test Simple : l'empreinte et l'obstacle sont projetés sur les axes

positifs, sur la Figure 6.6(c) on s'aperçoit que les projections s'intersectent sur les deux axes alors que le pied et l'obstacle ne sont pas en collision.

Ce test simple ne nécessite pas la résolution d'un problème de satisfaction de contraintes, il est donc rapide puisque son temps de calcul est de l'ordre de 0.5 microsecondes. Cependant il n'est fiable que lorsque qu'il informe d'une non collision, dans le cas contraire il faudra faire appel au test complet pour déterminer la présence ou non de collision.

6.3.3 Application

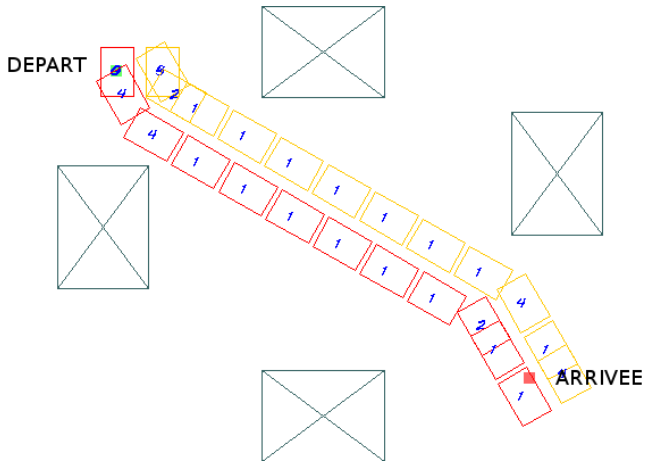


FIG. 6.7 – Séquence de pas générés

Nous avons testé cette méthode de planification de séquence de pas pour calculer les déplacements du robot HOAP-3 dans un environnement comportant 4 obstacles (voir Figure 6.7).

La méthode implémentée est efficace car elle permet de trouver une solution sous la condition qu'elle existe. Cependant, le temps de calcul dépend de manière exponentielle du nombre de pas effectués. Pour un chemin comportant moins de dix pas, le temps de calcul sera inférieur à une dizaine de secondes, mais augmentera de façon exponentielle avec le nombre de pas. Cette lenteur est due à la taille de l'arbre qui augmente à chaque itération. Nous proposons donc de corriger ce défaut avec un algorithme que nous qualifions de directionnel.

6.4 Méthode directionnelle

6.4.1 Idée principale

Afin de réduire le temps de calcul, la plupart des méthodes existantes tentent de diminuer la taille de l'arbre en ne déployant que certaines branches. Si nous souhaitons diminuer la taille de l'arbre à l'extrême, il faut développer une méthode qui va générer uniquement le "tronc" de l'arbre (qui correspond au chemin entre la position initiale et finale).

De la même manière, que l'on guiderait une personne ayant les yeux bandés en lui donnant des indications (tourne à droite, va tout droit, etc), notre méthode directionnelle décidera du pas à exécuter en fonction de la direction θ de la cible par rapport au robot de manière à orienter la marche du robot vers l'objectif (voir Figure 6.8).

Cette méthode reconsidère la position de la cible après chaque pas réalisé, elle permet, donc, de pouvoir se diriger vers une cible mobile. Nous disposons d'un système de vision qui effectue le suivi visuel de l'objectif et qui retourne l'information de position de la balle. Cette information sera prise en compte pour lancer un mouvement calculé hors-ligne.

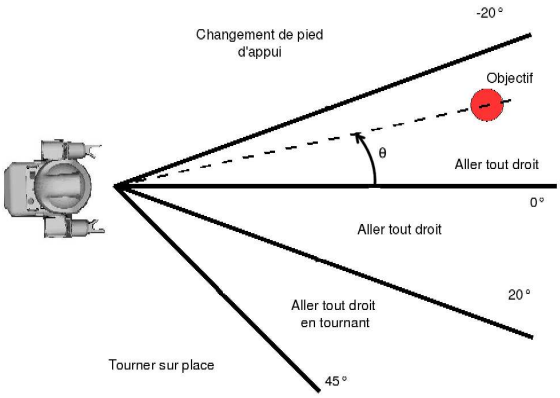


FIG. 6.8 – Algorithme directionnel pour le robot en appui sur le pied gauche.

Nous avons utilisé cet algorithme pour générer la séquence de pas, en l'absence d'obstacles, qui est présentée sur la Figure 6.9. On peut remarquer que le chemin produit est cohérent, il ne comporte pas de virages inutiles. Néanmoins, le faible nombre de pas de la bibliothèque produit un chemin assez morcelé. En augmentant le nombre de pas et en considérant un algorithme adapté à ce nouveau nombre, il doit être possible d'obtenir une courbe plus lisse.

6.4.2 Éviter les obstacles

Implémentation

Nous gérons l'évitement d'obstacle de la même manière que l'avancée du robot vers son objectif. En cas de présence d'un obstacle nous allons choisir une direction qui permettra de l'éviter, cette direction sera ensuite envoyée à l'algorithme de décision de pas, comme le montre la Figure 6.10.

La Figure 6.11 représente un cas où le robot doit éviter un obstacle pour rejoindre sa cible. En cas d'obstacle nous calculons l'angle ψ qui représente la direction permettant de

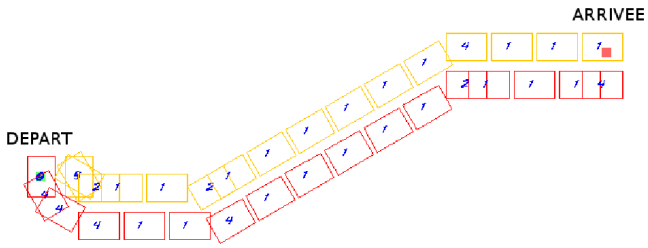


FIG. 6.9 – Résultat de l'algorithme directionnel sans obstacle.

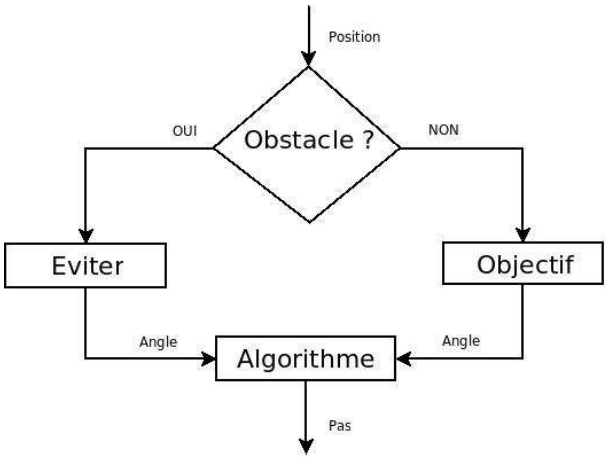


FIG. 6.10 – Algorithme incluant la détection d'obstacle.

l'éviter suivant la procédure suivante :

1. ajout d'un espace de sécurité autour de l'obstacle,
2. calcul des trois sommets de l'encadrement, les plus proches du robot,
3. calcul des angles $\alpha_1 \geq \alpha_2 \geq \alpha_3$ qui représentent la direction des sommets dans le repère du robot,
4. calcul de l'angle θ qui représente la direction de l'objectif dans le repère du robot grâce à un test comparatif.

Pour suivre cette procédure, nous devons définir les angles :
- α_i : angle représentant la direction du sommet i d'un obstacle dans le repère du robot,
- θ : angle représentant la direction de l'objectif dans le repère du robot,
- ψ : angle qui sera retourné à l'algorithme de choix du pas à exécuter

Le test comparatif qui permet de calculer l'angle ψ répond à la règle suivante :

$$\begin{array}{ccc} \text{si} & \text{alors} & \\ \theta \geq \alpha_1 & \psi = \theta & \text{l'obstacle ne gêne pas} \\ \alpha_1 \geq \theta \geq \alpha_2 & \psi = \alpha_1 & \text{évitement de l'obstacle par la gauche} \\ \alpha_2 \geq \theta \geq \alpha_3 & \psi = \alpha_3 & \text{évitement de l'obstacle par la droite} \\ \alpha_3 \geq \theta & \psi = \theta & \text{l'obstacle ne gêne pas} \end{array} \tag{6.3}$$

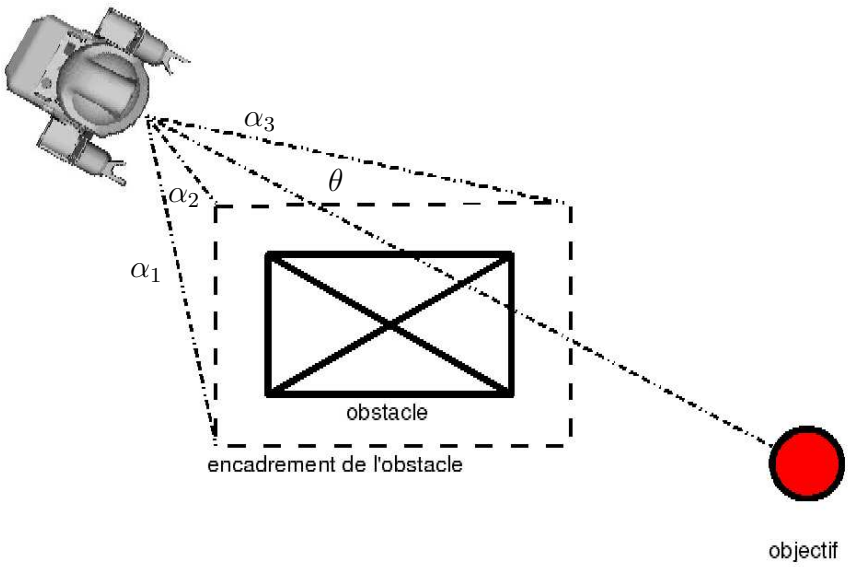


FIG. 6.11 – Évitement d'obstacle

Applications

Nous avons appliqué cette méthode pour générer la séquence de pas présentée sur la Figure 6.12. Le robot est en mesure de rejoindre l'objectif en longeant les obstacles qui se trouvent sur sa route, avec un temps de calcul très faible puisque chaque pas est obtenu à partir de simples tests comparatifs.

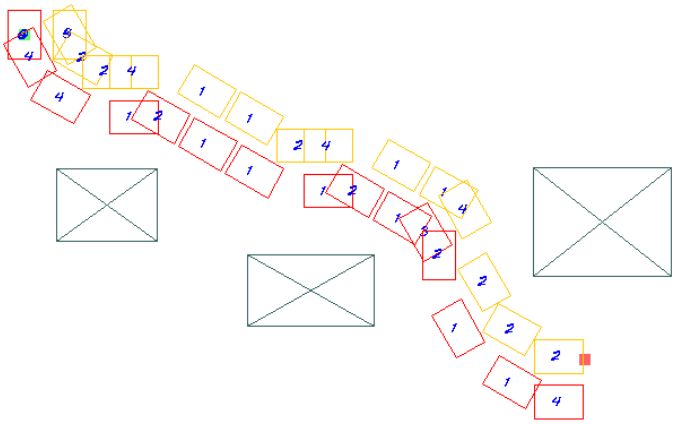


FIG. 6.12 – Séquence de pas en présence d'obstacle

Limite de la stratégie d'évitement

Nous avons choisi de développer une stratégie de navigation très simple, afin de pouvoir facilement utiliser une base de données de pas obtenue grâce à notre méthode de planification de mouvements sûrs (voir chapitre 4). Cette stratégie nous a permis de réaliser rapidement une expérimentation, mais, malheureusement, elle n'est pas adaptée pour des environnements encombrés en présence d'un grand nombre d'obstacles. La Figure 6.13, montre le résultat d'une planification avec plusieurs obstacles et des zones de passage

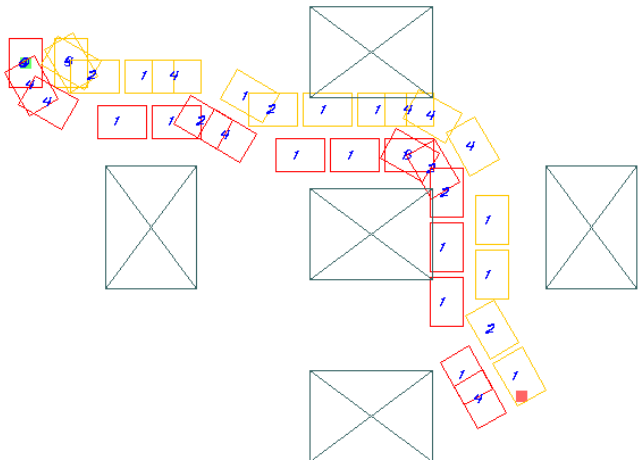


FIG. 6.13 – Séquence de pas en présence d'un grand nombre d'obstacles

étroites. Elle montre bien qu'il peut y avoir des collisions entre les pieds du robot et des obstacles ne se trouvant pas en direction de l'objectif (car ils ne sont pas pris en compte).

Pour remédier à ce problème, il faudrait ajuster correctement la largeur de l'espace de sécurité autour des obstacles ou développer une méthode de fusion des obstacles proches les uns des autres.

6.4.3 Résultats Expérimentaux

Nous avons donc décidé de valider notre méthode de navigation ainsi que la méthode de planification de mouvements sûrs sur le robot HOAP-3. Le programme global de navigation pour le robot est présenté sur la Figure 6.14.

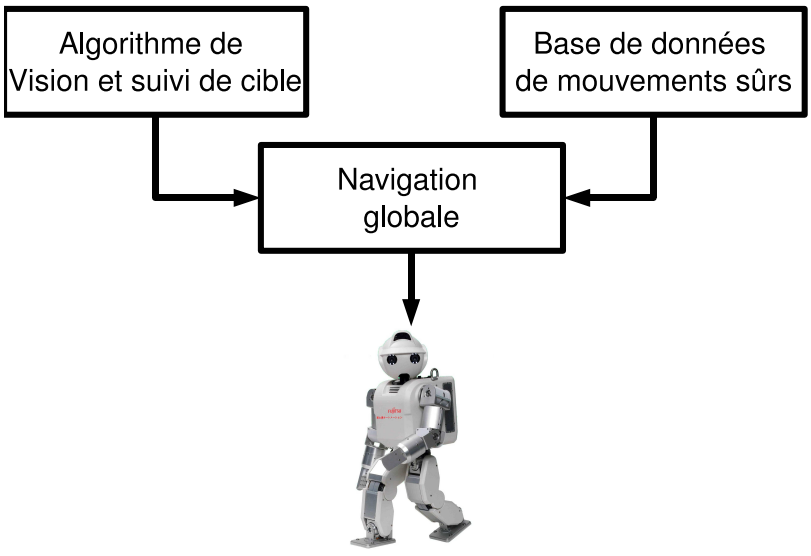


FIG. 6.14 – Programmation du robot HOAP-3

Nous avons généré une base de données de 24 (12×2) mouvements sûrs présentés sur la Figure 6.3 par la méthode d'optimisation hybride (cf. section 4.7). La mesure de la

direction de l'objectif se fait en ajoutant l'orientation de la tête par rapport au torse du robot et la direction de la balle mesurée par une méthode classique de vision et de suivi de cible (cf. Figure 6.15).

FIG. 6.15 – Système de vision de la balle.

La Figure 6.16 montre les résultats de l'expérimentation sur le robot HOAP-3. Cette expérimentation nous permet de valider les mouvements générés par la méthode de planification de mouvements sûrs, en effet aucun mouvement ne déséquilibre le robot. La méthode directionnelle de navigation est également validée, car le robot enchaîne toujours des pas lui permettant de se diriger dans la direction de la balle.

6.5 Conclusion

Dans ce chapitre nous avons présenté deux méthodes de génération de séquence de pas qui permettent à un robot de se déplacer dans son environnement à partir de quelques mouvements stockés dans une base de données. La première méthode se base sur la construction d'un arbre, elle permet de générer la totalité de la séquence de la position initiale du robot à sa position finale. Cette méthode permet un évitement des obstacles, cependant elle s'avère très longue en temps de calcul lorsque la séquence comporte plus d'une dizaine de pas et n'est pas adaptée lorsque la position finale se déplace.

Nous avons, donc, présenté une méthode directionnelle qui permet de choisir quel est le mouvement à effectuer en fonction de l'état actuel du robot et de la position de la cible. Nous avons constaté quelques limites en cas de nombreux obstacles à éviter. Cependant, elle nous a permis de réaliser une expérimentation de suivi d'une cible mobile où nous avons validé les mouvements générés par notre méthode de planification de mouvements sûrs. De plus, la base de données ne comporte que des mouvements qui commencent et qui se terminent avec une vitesse et une accélération angulaire nulle, ce qui oblige le robot à s'arrêter entre deux pas. Pour produire un déplacement plus fluide, il serait intéressant d'ajouter des mouvements qui commencent et qui se terminent avec des vitesses angulaires non nulle, tout en faisant attention à assurer la continuité entre deux mouvements.

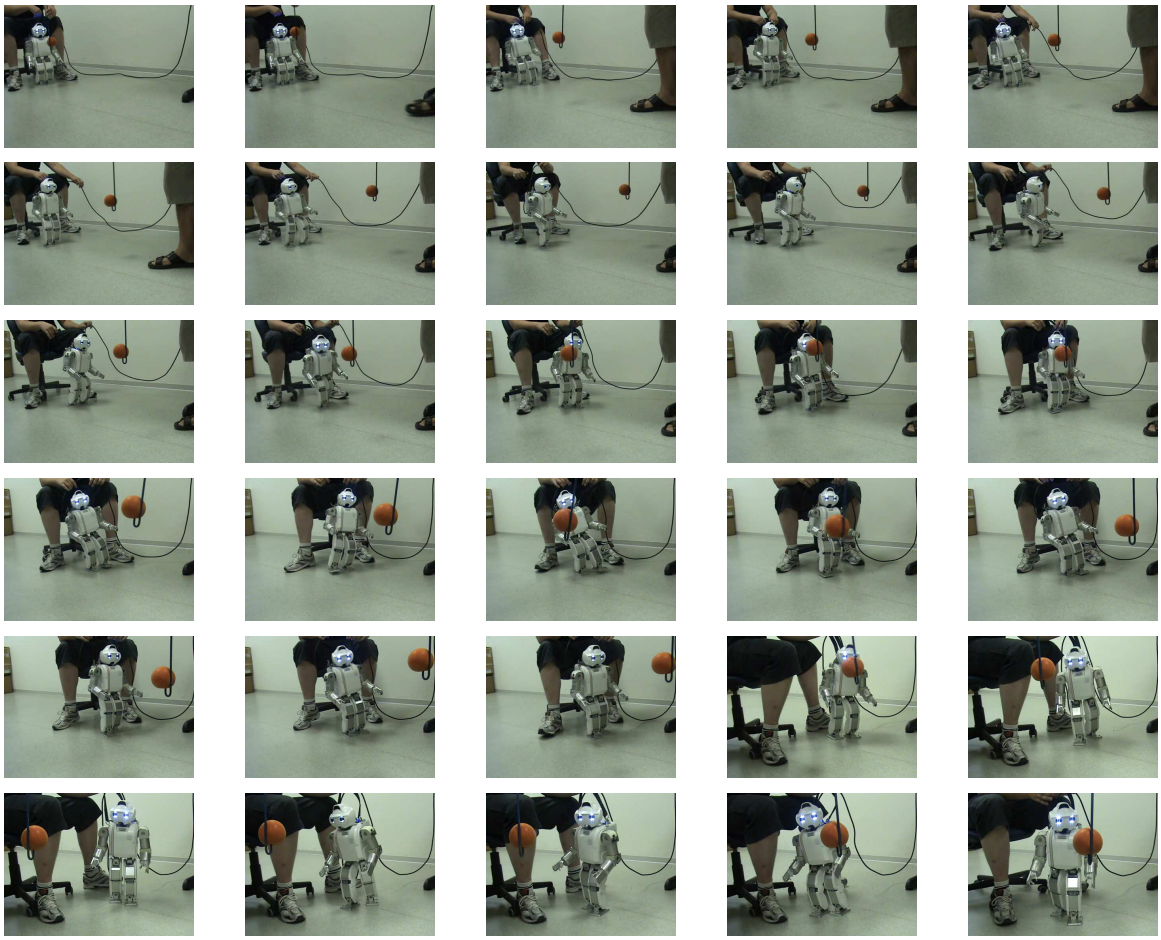


FIG. 6.16 – Expérimentation avec le robot HOAP-3.

Chapitre 7

Conclusion générale

7.1 Conclusion

Après avoir donné un aperçu des méthodes de planification de chemin, de mouvements et de séquences, nous avons détaillé le fonctionnement et les limites des robots humanoïdes. Nous avons montré que les méthodes de planification actuelles utilisent une discrétisation temporelle par grille des contraintes représentant les limites physiques du robot. Avec cette méthode de discrétisation, les contraintes ne sont calculées et retournées à l'algorithme d'optimisation que pour quelques instants, l'algorithme ne dispose, donc, pas d'information concernant l'évolution des contraintes entre ces points. Nous avons montré que cette méthode de discrétisation peut conduire à un dépassement de certaines limites du robot pouvant mettre en péril son intégrité ou son équilibre.

La première contribution de la thèse est d'avoir proposé une méthode de discrétisation garantie, qui va calculer les contraintes sur des intervalles de temps couvrant toute la durée du mouvement. Ce calcul est fait grâce à un outil mathématique : l'Analyse par Intervalles. L'algorithme recevra donc les valeurs extrêmes des grandeurs à contraindre et évitera le dépassement de leurs limites. Cette méthode permet d'avoir la certitude que le mouvement généré respecte les limites physiques du robot. Cependant, cette méthode est très longue en temps de calcul, ce qui nous a conduit à développer une méthode hybride. Pour obtenir la garantie sur les contraintes, dans un laps de temps raisonable, la méthode de planification hybride consiste à effectuer, alternativement, une optimisation utilisant une discrétisation par grille, puis une vérification des contraintes basée sur une discrétisation par intervalle afin de modifier la valeur des contraintes jusqu'à ne constater violation. Finalement, nous avons obtenu une méthode de planification de mouvements sûrs avec des temps de calcul comparables à celui des méthodes classiques. Nous sommes, donc en mesure de générer des mouvements optimaux parfaitement adaptés à une situation. Mais comment réagir si la situation change ?

La seconde contribution de la thèse est d'avoir proposer une méthode de re-planification de ces mouvements dans le cas où la situation avait légèrement changé. Le problème étant qu'une re-planification consistant à relancer le même processus d'optimisation serait trop longue en temps de calcul. Notre méthode consiste à remplacer le calcul des limites, très long, par le calcul de limite sur les paramètres du mouvement. Pour cela nous avons calculé un sous-ensemble de mouvements faisables autour du mouvement optimal qui est une approximation intérieure de l'ensemble faisable des paramètres. Le processus de re-planification consistait, alors, à trouver un mouvement adapté à la situation dans ce sous-ensemble. Le calcul du sous-ensemble faisable utilise également l'Analyse par Inter-

valles afin de vérifier qu'il ne contient aucun mouvement mettant en danger l'intégrité ou l'équilibre du robot. Nous avons appliqué cette méthode sur un mouvement de coup de pied en déplaçant le point d'impact entre la situation du mouvement optimal et la situation courante. La re-planification réussit à générer rapidement un nouveau mouvement en moins de deux secondes.

Finalement, nous avons développé une heuristique pour la navigation globale du robot à partir d'une base de données de mouvements de pas. Le robot devant rejoindre une cible fixe ou mobile, calcule l'angle instantané de la cible par rapport à sa position, puis choisit le mouvement approprié pour avoir la cible devant lui. Cette expérimentation nous a servi de protocole de validation pour la planification de mouvements sûrs.

7.2 Perspectives

Cette thèse propose d'incorporer un outil mathématique (l'Analyse par Intervalles) à des méthodes de planification déjà existantes afin d'obtenir une méthode de planification de mouvements sûrs. Cet outil a également permis de réaliser une re-planification rapide de ces mouvements sans compromettre la sûreté de fonctionnement du robot.

Cette thèse peut être considérée comme un des premiers essais d'utilisation de l'Analyse par Intervalles pour les robots humanoïdes. La discrétisation garantie permet d'obtenir un encadrement d'une grandeur sur un intervalle de temps, cependant nous avons utilisé les méthodes les plus simples pour faire ce calcul. Afin d'améliorer le temps de calcul, il faudrait tester ou développer d'autres méthodes qui la rendrait plus facilement utilisable et plus attrayante. De plus, nous sommes concentrés sur la discrétisation des contraintes suivant une seule dimension. Une généralisation de la discrétisation garantie à n dimensions peut parfaitement être envisagée et serait même utile lors de la présence d'incertitude dans le modèle.

Le calcul d'un sous-ensemble faisable autour d'un mouvement optimal est aussi un apport important de l'Analyse par Intervalles aux robots humanoïdes. Il pourrait être intéressant de valider cette méthode en recherchant la meilleure forme possible pour ce sous-ensemble qui permettrait d'augmenter la gamme des mouvements possibles. Une autre piste serait de pouvoir décrire entièrement l'ensemble faisable des paramètres grâce à cette technique ce qui permettrait de grandement accélerer les processus de planification de mouvements en ligne.

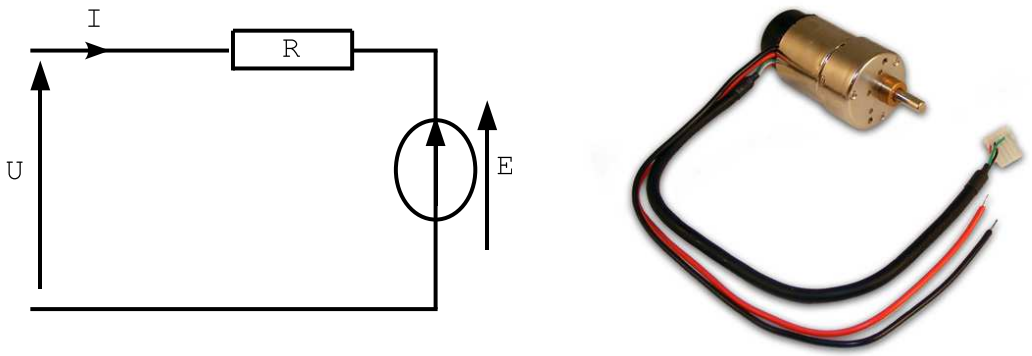
Annexe A

Modélisation des moteurs électriques

Sommaire

A.1 Modèle électrique équivalent

Les moteurs à courant continu sont essentiels à la production de mouvements pour les robots humanoïdes et sont composés de deux parties, le rotor qui est la partie tournante et le stator qui est la partie fixe. Le stator produit un champ magnétique constant soit grâce à un aimant permanent, soit grâce à un bobinage parcouru par un courant continu alors appelé électro-aimant. Le rotor est composé de conducteurs parcourus par un courant continu, qui sont soumis à des forces (forces dites de "Laplace" dues au champ magnétique du stator) et entraînent la rotation du rotor.



(a) modèle électrique (b) Photographie

FIG. A.1 – Le moteur à courant continu

Le schéma électrique équivalent d'un moteur à courant continu est représenté en Figure A.1(a). Les grandeurs sont :

- U : la tension électrique appliquée aux bornes du moteur,
- I : le courant électrique consommé par le moteur,
- R : la résistance du bobinage du rotor,

– E : la force contre électromotrice équivalente à une source de tension électrique.

Ce schéma nous permet d'établir l'équation suivante :

$$U = E + R \times I \tag{A.1}$$

Il existe également une inductance équivalente, mais qui peut être négligée dans le cas d'une alimentation à courant continu. Les grandeurs mécaniques du moteur (vitesse de rotation Ω_{mot} et couples moteurs Γ_{mot}) sont reliées aux grandeurs électriques par le coefficient électromagnétique K_{em} qui dépend de la valeur du champ magnétique généré par le stator, dans le cas d'un moteur à aimant permanent ce coefficient est constant.

$$E = K_{em} \times \Omega_{mot} \tag{A.2}$$

$$\Gamma_{mot} = K_{em} \times I \tag{A.3}$$

En connaissant les paramètres R et K_{em}, il est donc possible de contrôler le comportement du moteur (Γ_{mot}, Ω_{mot}) à partir de la tension d'alimentation U.

A.2 Bilan de puissance

Nous avons vu dans la section précédente la partie électrique du moteur. Cependant, dans le cas de robots, les moteurs sont souvent associés à des systèmes de réduction afin de produire un moteur équivalent qui aura sa vitesse divisée et son couple multiplié par le coefficient de réduction K_r (Cf. Équation A.4).

$$\begin{aligned} \Omega &= \frac{\Omega_{mot}}{K_r} \\ \Gamma &= K_r \times \Gamma_{mot} \end{aligned} \tag{A.4}$$

Il est intéressant d'effectuer le bilan de puissance de l'ensemble moteur + réducteur, pour cela on définit :

– La puissance utile : $P_u = \Gamma \times \Omega$ est la puissance mécanique fournie à l'articulation,
– La puissance électrique consommée par le moteur $P_{elec} = U \times I$,
– La puissance électromécanique $P_{em} = E \times I = \Gamma_{mot} \times \Omega_{mot}$ est la puissance mécanique transmise au rotor.

Si on considère un système parfait ces trois puissances sont égales, dans le cas contraire il faut prendre en compte des phénomènes de perte que l'on peut répartir en deux parties :

– Les pertes par effet Joule $P_{Joule} = R.I^2$ qui sont dues au passage du courant électrique dans le bobinage du rotor.
– Les pertes mécaniques qui sont dûes aux frottements du moteur mais aussi du réducteur (dans ce cas elles sont ramenées au rotor à travers le rapport de réduction K_r) et aux courants de Foucault au rotor. En effet le rotor étant en mouvement, il voit donc un champ magnétique variable ce qui crée des courants en surface. Les pertes mécaniques peuvent s'exprimer comme une perte en couple en fonction de la vitesse de rotation $\Gamma = K_r \times \Gamma_{mot} - (K_v \times \Omega + K_s)$ où les coefficients K_v et K_s dépendent de la vitesse de rotation Ω et de son signe.

La puissance électrique consommée par un moteur peut se calculer par :

$$P_{elec} = P_{Joule} + P_{meca} + P_u \tag{A.5}$$

La puissance électrique peut donc s'exprimer en fonction des grandeurs mécaniques fournies en sortie du réducteur :

$$P_{elec} = \frac{R}{(K_{em}K_r)^2}(\Gamma + K_v \times \Omega + K_s)^2 + \Gamma \times \Omega \tag{A.6}$$

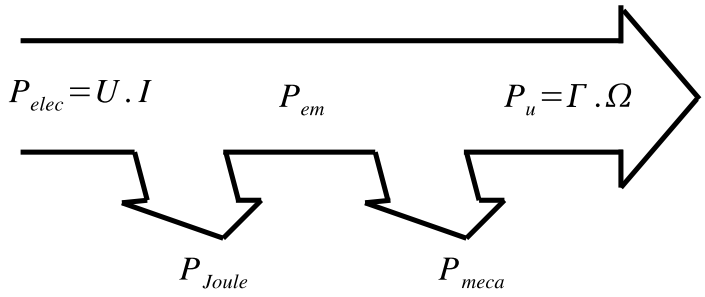


FIG. A.2 – Bilan de puissance d'un moteur à courant continu

Les pertes dûes au réducteur sont négligeables devant le couple, on suppose donc $K_v = 0$ et $K_s = 0$. Les ordres de grandeurs des autres coefficients sont les suivants :

- Résistance du rotor : quelques ohms : $R = 1 \sim 10(\Omega)$
- Coefficient électromagnétique : $K_{em} = 1e^{-3} \sim 1e^{-2}(V/rad/s)$
- Coefficient de réduction : $K_r = 100 \sim 300$

N'ayant pas les caractéristiques des moteurs du robot HOAP-3 nous considérons le rapport : $\frac{R}{(K_{em}K_r)^2}$ égal à 0.1. Dans le chapitre sur la planification (Chap. 4), quand nous choisirons le critère $F(t)$ comme étant l'énergie, nous essaierons de minimiser la grandeur suivante :

$$F(t) = \int_0^T P_{elec}.dt = \int_0^T 0.1\Gamma^2 + \Gamma \times \Omega.dt \tag{A.7}$$

Annexe B

Calcul des fonctions B-splines

Sommaire

Le calcul des B-splines est expliqué dans [de Boor 78].

B.1 Définition

Étant donné $m+1$ nœuds T_k dans $T = \{T_0, T_1, \ldots, T_k, T_{k+1}, \ldots, T_m\}$ comme étant une séquence non décroissante.

Une courbe spline de degré n est une courbe paramétrique composée de fonctions B-splines de degré n

Le calcul des B-splines ce fait de manière récursive i en partant de la séquence T.

Pour $i = 0$ on impose

$$\begin{array}{cc} si\ T_j < t < T_{j+1} & b_{j,0}(t) = 1 \\ sinon & b_{j,0}(t) = 0 \end{array} \tag{B.1}$$

Ensuite on applique la formule récursive suivante :

$$b_{j,i}(t) = \frac{t - T_j}{T_{j+i} - T_j} b_{j,i-1}(t) + \frac{T_{j+i+1} - t}{T_{j+i+1} - T_{j+1}} b_{j+1,i-1}(t) \tag{B.2}$$

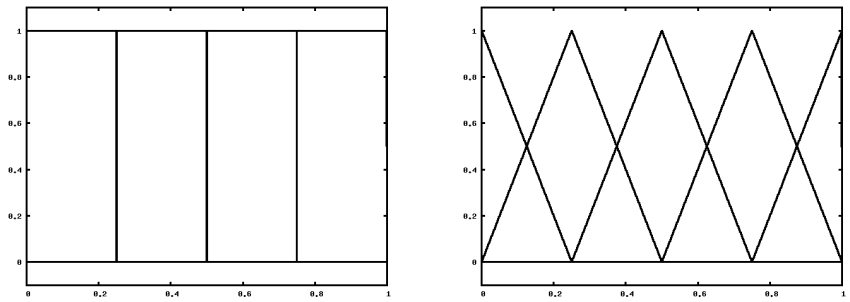
Lorsque plusieurs noeuds t_J sont confondus on pose $\frac{0}{0} = 0$. Quand les nœuds sont équidistants, les B-splines sont dites uniformes.

B.2 Représentation

Dans nos travaux, nous utilisons des B-splines d'ordre 3 et considérons N_s fonctions de splines (5 ou 7). Nous présentons les calculs pour N_s fonctions B-splines.

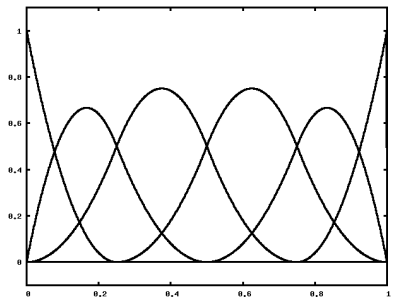
Nous fixons le vecteur $T = 0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1$ (normalisé par rapport a T_{MAX})

Nous avons utilisé la récurrence présentée en Équation(B.2) et obtenu les résultats présentés en Figure B.1

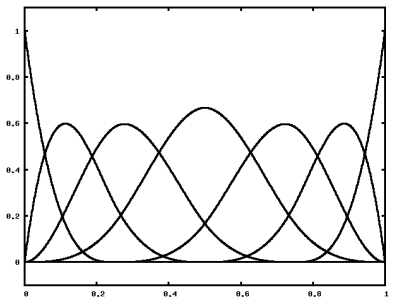


(a) i = 0

(b) i = 1



(c) i = 2



(d) i = 3

FIG. B.1 – Représentation des Fonctions B-splines

B.3 Calcul analytique

Les calculs présentés sont ceux de la figure B.2

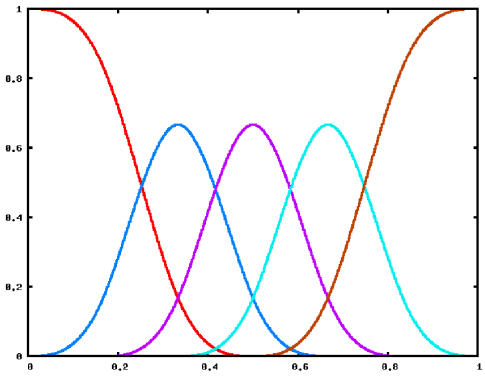


FIG. B.2 – Représentation des fonctions de base

on pose $\delta = \frac{T_{max}}{N_s - 1}$

T	$[0;\delta]$	$[\delta;2\delta]$	$[2\delta;3\delta]$	$[i\delta;(i+1)\delta]$	$[(N_s-4)\delta;(N_s-3)\delta]$	$[(N_s-3)\delta;(N_s-2)\delta]$	$[(N_s-2)\delta;T_{MAX}]$
$b_0(t)$	$f_{00}(t-\frac{\delta}{2})$	$f_{01}(t-\frac{\delta}{2})$	$f_{33}(t-\frac{\delta}{2})$	0	0	0	0
$b_1(t)$	$f_{30}(t-\frac{\delta}{2})$	$f_{31}(t-\frac{\delta}{2})$	$f_{32}(t-\frac{\delta}{2})$	0	0	0	0
$b_2(t)$	0	$f_{30}(t-\frac{3\delta}{2})$	$f_{31}(t-\frac{3\delta}{2})$	0	0	0	0
$b_3(t)$	0	0	$f_{30}(t-\frac{5\delta}{2})$	0	0	0	0
...	...	...	...	...	...	...	...
$b_i(t)$	0	0	0	$f_{33}(t-\frac{2(i-2)\delta}{2})$	0	0	0
$b_{i+1}(t)$	0	0	0	$f_{32}(t-\frac{2(i-1)\delta}{2})$	0	0	0
$b_{i+2}(t)$	0	0	0	$f_{31}(t-\frac{2(i)\delta}{2})$	0	0	0
$b_{i+3}(t)$	0	0	0	$f_{30}(t-\frac{2(i+1)\delta}{2})$	0	0	0
...	...	...	...	...	...	...	...
$b_{N_s-4}(t)$	0	0	0	0	$f_{33}(T_{MAX}-t-\frac{\delta}{2})$	0	0
$b_{N_s-3}(t)$	0	0	0	0	$f_{32}(T_{MAX}-t-\frac{\delta}{2})$	$f_{30}(T_{MAX}-t-\frac{3\delta}{2})$	0
$b_{N_s-2}(t)$	0	0	0	0	$f_{31}(T_{MAX}-t-\frac{3\delta}{2})$	$f_{31}(T_{MAX}-t-\frac{\delta}{2})$	$f_{00}(T_{MAX}-t-\frac{\delta}{2})$
$b_{N_s-1}(t)$	0	0	0	0	$f_{30}(T_{MAX}-t-\frac{5\delta}{2})$	$f_{01}(T_{MAX}-t-\frac{\delta}{2})$	$f_{30}(T_{MAX}-t-\frac{\delta}{2})$

TAB. B.1 – tableau des fonctions B-splines qui imposent une vitesse et accélération initiales et finales nulles

avec les fonction suivantes :

$$f_{00}(t) = \frac{47}{48} - \frac{t}{8\delta} - \frac{t^2}{4\delta^2} - \frac{t^3}{6\delta^3} \tag{B.3}$$

$$f_{01}(t) = \frac{11}{12} + \frac{t}{4\delta} - \frac{t^2}{\delta^2} + \frac{t^3}{3\delta^3} \tag{B.4}$$

$$f_{30}(t) = \frac{1}{48} + \frac{t}{8\delta} + \frac{t^2}{4\delta^2} + \frac{t^3}{6\delta^3} \tag{B.5}$$

$$f_{31}(t) = \frac{5}{48} - \frac{3t}{8\delta} + \frac{5t^2}{4\delta^2} - \frac{t^3}{2\delta^3} \tag{B.6}$$

$$f_{32}(t) = \frac{-157}{48} + \frac{51t}{8\delta} - \frac{13t^2}{4\delta^2} + \frac{t^3}{2\delta^3} \tag{B.7}$$

$$f_{33}(t) = \frac{125}{96} - \frac{75t}{24\delta} + \frac{5t^2}{4\delta^2} - \frac{t^3}{6\delta^3} \tag{B.8}$$

$$\tag{B.9}$$

Annexe C

Calcul des forces de contact pour un mouvement en double support

Sommaire

C.1 Modèle dynamique

Simple support

En simple support, le modèle dynamique d'un robot à n degrés de liberté se met sous la forme :

$$\begin{bmatrix} \Gamma \\ F_{ref} \\ M_{ref} \end{bmatrix} = M(x)\ddot{x} + H(x, \dot{x}) \tag{C.1}$$

Avec :

$$x = [q_1, q_2, \ldots, q_n, x_{ref}, y_{ref}, z_{ref}, \theta_{ref}, \alpha_{ref}, \gamma_{ref}]^T \tag{C.2}$$

et

- q_i la valeur de l'angle pour l'articulation i,
- $x_{ref}, y_{ref}, z_{ref}, \theta_{ref}, \alpha_{ref}, \gamma_{ref}$ la position et orientation du corps de référence exprimé dans un repère absolu (dans notre cas le corps de référence est le pied droit),
- F_{ref} de taille 3 est la force exercée par le robot sur le corps de référence,
- M_{ref} de taille 3 est le moment exercé par le robot sur le corps de référence,
- $M(x)$ de taille $(n+6)$ $times(n+6)$ la matrice d'inertie,
- $H(x, \dot{x})$ de taille $(n+6)$ le vecteur représentant l'effet de Coriolis, des forces centrifuges et de la gravité.

Dans notre cas, nous considérons 12 articulations. Nous souhaitons connaître la valeur des couples articulaires Γ (12 composantes), ainsi que des efforts F_{ref} et M_{ref} (3 composantes chacun), à partir de la connaissance du vecteur x et de ses dérivées.

Le système d'équation C.1 se décompose en 18 équations qui permettent de déterminer les 18 inconnues et peut donc être résolu.

double support

Pour un mouvement en double support, le modèle dynamique s'exprime sous la forme suivante :

$$\begin{bmatrix} \Gamma \\ F_{ref} \\ M_{ref} \end{bmatrix} = M(x)\ddot{x} + H(x,\dot{x}) + J^T \begin{bmatrix} F_c \\ M_c \end{bmatrix} \tag{C.3}$$

Où F_c et M_c représentent les efforts de contact (au niveau du pied gauche dans notre cas001) et J est la matrice jacobienne des contacts.

Comme dans le cas du simple support nous cherchons à évaluer la valeur des 12 couples articulaires et des efforts sur le corps de référence, mais aussi les composantes de l'effort s'exerçant sur le pied gauche. Le système d'équation C.3 est toujours composé de 18 équations, mais avec plus de 18 inconnues. Ce système ne peut pas être résolu car il est sous-déterminé.

Pour lever l'indétermination, il faut poser des hypothèses sur les efforts de contacts. Dans le simulateur HuManS [Wieber 06], les efforts de contacts sont souhaités les plus proches de la normale à la surface de contact, alors que dans [Miossec 08], Miossec préfère avoir des efforts de contact qui minimisent les couples articulaires. Dans ces deux cas, les forces de contact sont obtenues par la résolution d'un problème d'optimisation sous contraintes. Cependant cette solution est lourde en temps de calcul car elle nécessite la résolution d'un problème d'optimisation à chaque pas d'échantillonnage [Lengagne 06].

Nous choisissons de faire des hypothèses qui vont ajouter des équations au système C.3 pour le rendre résolvable. De plus, la formulation de la Jacobienne donnée par le logiciel HuManS définit les efforts de contacts du pied gauche comme quatre forces de contacts s'exerçant aux quatre coins du pied.

Nous devons donc ajouter 12 équations (4×3) pour pouvoir résoudre ce système.

C.2 Hypothèses

Pour résoudre ce système, nous allons imposer des relations sur les forces de contact afin de lever l'indétermination. Nous utilisons quatre hypothèses :

- les forces de contact s'exercent uniquement selon l'axe normal au sol (de haut en bas suivant l'axe x),
- les forces de contact en double support sont proportionnelles aux forces de contact en simple support,
- Le ZMP calculé en simple support est identique à celui calculé en double support.
- les forces de contact sur le pied de support sont pondérées par la distance les séparant du ZMP.

Notations

Pour permettre le calcul des efforts de contacts nous allons utiliser les modèles en simples et double support. Nous définissons donc F_0 et M_0 les efforts sur le corps de référence pour le modèle en simple support et F_c et M_c en double support.

$$\begin{bmatrix} \Gamma_0 \\ F_0 \\ M_0 \end{bmatrix} = M(x)\ddot{x} + H(x, \dot{x}) \tag{C.4}$$

$$\begin{bmatrix} \Gamma_c \\ F_c \\ M_c \end{bmatrix} = M(x)\ddot{x} + H(x, \dot{x}) + J^T \begin{bmatrix} F_{c_1} \\ F_{c_2} \\ F_{c_3} \\ F_{c_4} \end{bmatrix} \tag{C.5}$$

Des équations C.4 et C.5, on peut déduire que :

$$\begin{bmatrix} \Gamma_c \\ F_c \\ M_c \end{bmatrix} = \begin{bmatrix} \Gamma_0 \\ F_0 \\ M_0 \end{bmatrix} + J^T \begin{bmatrix} F_{c_1} \\ F_{c_2} \\ F_{c_3} \\ F_{c_4} \end{bmatrix} \tag{C.6}$$

Et donc que :

$$F_c = F_0 + A \times (F_{c_1} + F_{c_2} + F_{c_3} + F_{c_4}) \tag{C.7}$$

$$M_c = M_0 + A \times (F_{c_1} \wedge P_1 + F_{c_2} \wedge P_2 + F_{c_3} \wedge P_3 + F_{c_4} \wedge P_4) \tag{C.8}$$

Avec A la matrice d'orientation définie par l'orientation du pied gauche par rapport au pied droit α de la manière suivante :

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \tag{C.9}$$

et les vecteurs P_i sont les vecteurs entre le centre du repère du pied droit et le coin i du pied gauche.

Efforts de contact verticaux

Les efforts tangentiels (suivant l'axe Y et Z), n'ont pas d'effet sur l'équilibre du robot. Afin de faciliter les calculs nous choisissons de les considérer comme nuls, d'où ;

$$\begin{aligned} F_{C1}(y) = F_{C2}(y) = F_{C3}(y) = F_{C4}(y) = 0 \\ F_{C1}(z) = F_{C2}(z) = F_{C3}(z) = F_{C4}(z) = 0 \end{aligned} \tag{C.10}$$

Efforts de contact proportionnels à l'effort en simple support

Le but du mouvement en double support est de changer de pied de support. Nous imposons donc des efforts nuls pour un pied au début du mouvement, puis nuls pour l'autre pied à la fin du mouvement.

$$F_{C1}(x) + F_{C2}(x) + F_{C3}(x) + F_{C4}(x) = -K \times F_0(x) \tag{C.11}$$

Le coefficient K est défini comme :

$$K = \frac{t}{T} \tag{C.12}$$

Où T est la durée du mouvement est t l'instant courant.

ZMP

Afin de faciliter la prise en compte de l'équilibre pendant la phase de double support, nous souhaitons avoir le ZMP au même endroit que ce soit en simple ou en double support. Nous ajoutons donc les égalités :

$$\begin{array}{l} F_{ZMP} = M_0(y)/F_0(x) = M_C(y)/F_C(x) \\ S_{ZMP} = M_0(z)/F_0(x) = M_C(z)/F_C(x) \end{array} \tag{C.13}$$

Efforts proportionnels à la distance au ZMP

Afin d'obtenir le nombre suffisant d'équation, nous décidons d'imposer une relation entre deux efforts de contact qui se base sur la distance les séparant du ZMP.

$$\begin{aligned} F_{C4}(x) &= \Delta \times F_{C1}(x) \\ \Delta &= D_1/D_4 \\ D_1 &= \sqrt{(P_1(y) - F_{ZMP})^2 + (P_1(z) - S_{ZMP})^2} \\ D_4 &= \sqrt{(P_4(y) - F_{ZMP})^2 + (P_4(z) - S_{ZMP})^2} \end{aligned} \tag{C.14}$$

Récapitulatif

Ayant défini les équations nécessaire pour résoudre notre système, il est possible d'en déduire la valeur des forces de contact aux quatre coins du pied.

$$\begin{aligned} M_0(y)/F_0(x) = M_C(y)/F_C(x) \\ M_0(z)/F_0(x) = M_C(z)/F_C(x) \\ F_C = A \times (F_0 + F_{C1} + F_{C2} + F_{C3} + F_{C4}) \\ M_C = M_0 + A \times (F_0 \wedge P + F_{C1} \wedge P_1 + F_{C2} \wedge P_2 + F_{C3} \wedge P_3 + F_{C4} \wedge P_4) \\ F_{C1}(y) = F_{C2}(y) = F_{C3}(y) = F_{C4}(y) = 0 \\ F_{C1}(z) = F_{C2}(z) = F_{C3}(z) = F_{C4}(z) = 0 \\ F_{C1}(x) + F_{C2}(x) + F_{C3}(x) + F_{C4}(x) = -K \times F_0(x) \\ F_{C4}(x) = \Delta \times F_{C1}(x) \end{aligned} \tag{C.15}$$

C.3 Formulation

Ainsi, en utilisant la méthode de substitution sur notre système, on obtient les solutions du système dépendant les une des autres.

$$F_{C4}(x) = \Delta \times F_{C1}(x) \tag{C.16}$$
$$F_{C3}(x) = C_3 \times F_{C1} + H_3 \tag{C.17}$$
$$F_{C2}(x) = C_2 \times F_{C1} + H_2 \tag{C.18}$$
$$F_{C1}(x) = (-K \times F_0(x) - H_3 - H_2)/(1 + C_2 + C_3 + \Delta) \tag{C.19}$$

Lors de la résolution du problème, pour éviter de traîner la somme ou le produit de terme constant, nous définissons des variables intermédiaires. Ces variables sont des

constantes pour la plupart qui ne nécessitent pas d'être recalculées au cour du temps.

$$a_1 = -P_1(z) \times cos(\alpha) - P_1(y) \times sin(\alpha) \quad \text{(C.20)}$$
$$a_2 = -P_2(z) \times cos(\alpha) - P_2(y) \times sin(\alpha) \quad \text{(C.21)}$$
$$a_3 = -P_3(z) \times cos(\alpha) - P_3(y) \times sin(\alpha) \quad \text{(C.22)}$$
$$a_4 = -P_4(z) \times cos(\alpha) - P_4(y) \times sin(\alpha) \quad \text{(C.23)}$$
$$b_1 = -P_1(z) \times sin(\alpha) + P_1(y) \times cos(\alpha) \quad \text{(C.24)}$$
$$b_2 = -P_2(z) \times sin(\alpha) + P_2(y) \times cos(\alpha) \quad \text{(C.25)}$$
$$b_3 = -P_3(z) \times sin(\alpha) + P_3(y) \times cos(\alpha) \quad \text{(C.26)}$$
$$b_4 = -P_4(z) \times sin(\alpha) + P_4(y) \times cos(\alpha) \quad \text{(C.27)}$$
$$C_2 = (a_1 \times b_3 - a_3 \times b_1 + \Delta \times (a_4 \times b_3 - a_3 \times b_4))/(a_2 \times b_3 - a_3 \times b_2) \quad \text{(C.28)}$$
$$C_3 = (a_1 \times b_2 - a_2 \times b_1 + \Delta \times (a_1 \times b_2 - a_2 \times b_1))/(a_3 \times b_2 - a_2 \times b_3) \quad \text{(C.29)}$$
$$H_2 = K \times b_3 \times M_0(y) + K \times a_2 \times M_0(z) \quad \text{(C.30)}$$
$$H_3 = K \times b_2 \times M_0(y) + K \times a_2 \times M_0(z) \quad \text{(C.31)}$$

Bibliographie

[ali] *ALIAS-C++ A C++ Algorithms Library of Interval Analysis for equation Systems.*

[Arechavaleta-Servin 07] Gustavo Arechavaleta-Servin. *An optimality principle governing human walking.* PhD thesis, Institut National des Sciences Appliquées, Toulouse, 112p., 2007. Doctorat.

[Arechavaleta 08] G. Arechavaleta, J-P. Laumond, H. Hicheur & A. Berthoz. *An Optimality Principle Governing Human Walking.* IEEE Transactions on Robotics, vol. 24, no. 1, pages 5–14, February 2008.

[Arisumi 07] H. Arisumi, J. R. Chardonnet, A. Kheddar & K. Yokoi. *Dynamic Lifting Motion of Humanoid Robots.* In Robotics and Automation, 2007 IEEE International Conference on, pages 2661–2667, Roma, April 2007.

[Arisumi 08] Hitoshi Arisumi, Sylvain Miossec, Jean-Rémy Chardonnet & Kazuhito Yokoi. *Dynamic Lifting by whole body motion of human robots.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 668–675, september 2008.

[Ayaz 06] Y. Ayaz, K. Munawar, M. Bilal Malik, A. Konno & M. Uchiyama. *Human-Like Approach to Footstep Planning Among Obstacles for Humanoid Robots.* In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 5490–5495, Beijing,, October 2006.

[Bekris 07] K. E. Bekris & L. E. Kavraki. *Greedy but Safe Replanning under Kinodynamic Constraints.* In Robotics and Automation, 2007 IEEE International Conference on, pages 704–710, Roma,, April 2007.

[Bendtsen] Claus Bendtsen & Ole Stauning. *FADBAD, a flexible c++ package for automatic differentation.*

[Bohlin 00] R. Bohlin & L. E. Kavraki. *Path planning using lazy PRM.* In Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on, volume 1, pages 521–528, San Francisco, CA, USA, 2000.

[Bouyarmane 09] Karim Bouyarmane, Adrien Escande, Florent Lamiraux & Abderrahmane Kheddar. *Potential Field Guide for Humanoid Multicontacts Acyclic Motion Planning.* In IEEE International Conference on Robotics and Automation, may 2009.

[Bretl 08] T. Bretl & S. Lall. *Testing Static Equilibrium for Legged Robots*. IEEE Transactions on Robotics, vol. 24, no. 4, pages 794–807, August 2008.

[Brock 00] O. Brock & O. Khatib. *Real-time re-planning in high-dimensional configuration spacesusing sets of homotopic paths*. In Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on, volume 1, pages 550–555, San Francisco, CA, USA, 2000.

[Carpin 06] Stefano Carpin & Enrico Pagello. *The challenge of motion planning for human robots playing soccer*. In Workshop on Humanoid Soccer Robots of the 2006 IEEE-RAS International Conference on Humanoid Robots, pages 71–77, December 2006.

[Chabert 07] Gilles Chabert & Luc Jaulin. *computing the pessimism of inclusion functions*. Reliable Computing, vol. 13, pages 489–505, 2007.

[Chestnutt 03] J. Chestnutt, J. Kuffner, K. Nishiwaki & S. Kagami. *Planning biped navigation strategies in complex environments*. In IEEE/RAS Int. Conf humanoid Robots, 2003.

[Chestnutt 05] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins & T. Kanade. *Footstep Planning for the Honda ASIMO Humanoid*. In Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, pages 629–634, April 2005.

[Choi 07] Jaesik Choi & E. Amir. *Factor-guided motion planning for a robot arm*. In Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on, pages 27–32, San Diego, CA,, October/November 2007.

[Cotton 08] S. Cotton, A. Murray & P. Fraisse. *Statically Equivalent Serial Chains for Modeling the Center of Mass of Humanoid Robots*. In IEEE-RAS Internationnal conference on Humanoid robots, 2008.

[de Boor 78] Carl de Boor. A pratical guide to splines, volume 27. Springer-Verlag, New York, 1978.

[Dutra 03] Max S. Dutra, Armando C. de Pina Filho1and & Vitor F. Romano. *Modeling of a bipedal locomotor using coupled nonlinear oscillators of Van der Pol*. Biological Cybernetics, vol. 88, no. 4, pages 286–292, april 2003.

[Duysens 98] Jacques Duysens & Henry W.A.A. Van de Crommert. *Neural control of locomotion ; Part 1 : The central pattern generator from cats to human*. Gait & Posture, vol. 7, no. 7, pages 131–141, march 1998.

[Escande 06] A. Escande, A. Kheddar & S. Miossec. *Planning support contact-points for humanoid robots and experiments on HRP-2*. In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 2974–2979, Beijing,, October 2006.

[Escande 08] Adrien Escande. *Planification de points d'appui pour la génération de mouvements acycliques application aux humanoïdes.* PhD thesis, Universite d'evry-val d'essonne, décembre 2008.

[Fang 05] H. Fang & J. P. Merlet. *Dynamic interference avoidance of 2-DOF robot arms using interval analysis.* In Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on, pages 3809–3814, August 2005.

[Fraichard 99] T. Fraichard. *Trajectory planning in a dynamic workspace : a state-time approach.* Advanced Robotics and Automation, IEEE Transactions on, vol. 13, pages 75–94, 1999.

[Gouaillier 08] David Gouaillier, Vincent Hugel, Pierre Blazevic, Chris Kilner, Jerome Monceaux, Pascal Lafourcade, Brice Marnier, Julien Serre & Bruno Maisonnier. *The NAO humanoid : a combination of performance and affordability.* CoRR, vol. abs/0807.3223, 2008. informal publication.

[Haken 85] H. Haken, J. A. S. Kelso & H. Bunz. *A theoretical model of phase transitions in human hand movements.* Biological Cybernetics, vol. 51, no. 5, pages 347–356, fevrier 1985.

[Hammer 93] R. Hammer, M. Hocks, U. Kulisch & D.Ratz. Numerical toolbox for verified computing 1. Springer-Verlag, 1993.

[Hansen 04] E. Hansen & G.W. Walster. Global optimization using interval analysis. Marcel Dekker, 2nd edition, 2004.

[Harada 03] K. Harada, S. Kajita, K. Kaneko & H. Hirukawa. *ZMP analysis for arm/leg coordination.* In Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, volume 1, pages 75–81, October 2003.

[Hauser 05] K. Hauser, T. Bretl & J. C. Latombe. *Non-gaited humanoid locomotion planning.* In Humanoid Robots, 2005 5th IEEE-RAS International Conference on, pages 7–12, Tsukuba,, December 2005.

[Hettich 93] R. Hettich & K. O. Kortanek. *Semi-infinite programming : theory, methods, and applications.* SIAM Rev., vol. 35, no. 3, pages 380–429, 1993.

[Hirukawa 06] Hirohisa Hirukawa, Shizuko Hattori, Kensuke Harada, Shuuji Kajita, Kenji Kaneko, Fumio Kanehiro, Kiyoshi Fujiwara & Mitsuharu Morisawa. *A Universal Stability Criterion of the Foot Contact of Legged Robots - Adios ZMP.* In IEEE International Conference on Robotics and Automation (ICRA)., pages 1976– 1983, may 2006.

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

[Ijspeert 08] Auke Jan Ijspeert. *Central pattern generators for locomotion control in animals and robots : A review.* Neural Networks, vol. 21, no. 4, pages 642–653, may 2008.

[IPO 06] *Introduction to IPOPT : a tutorial for downloading, installing and using IPOPT*, april 7th 2006.

[Jaulin 01a] Luc Jaulin. *path planning using intervals and graphs.* Reliable Computing, vol. 7, no. 1, pages 1–15, fevrier 2001.

[Jaulin 01b] Luc Jaulin, Michel Kieffer, Olivier Didrit & Eric Walter. Applied interval analysis. Springer, 2001.

[Jean-Claude Latombe 91] J.C. Jean-Claude Latombe. robot motion planning. kluwer academic publishers, 1991.

[Kajita 02] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kazuhito Yokoi & Hirohisa Hirukawa. *A Realtime Pattern Generator for Biped Walking.* In IEEE International Conference on Robotics and Automation (ICRA)., may 2002.

[Kajita 09] Shuuji Kajita, Hirohisa Hirukawa, Kensuke Harada & Kazuhito Yokoi. Introduction à la commande des robots humanoïdes. springer, 2009.

[Kanehiro 08] Fumio Kanehiro, Wael Suleiman, Florent Lamiraux, Eiichi Yoshida & Jean-Paul Laumond. *Integrating dynamics into motion planning for humanoid robots.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 660–665, september 2008.

[Kaneko 04] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi & T. Isozumi. *Humanoid robot HRP-2.* In Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, volume 2, pages 1083–1090, April/May 2004.

[Kaneko 08] Kenji Kaneko, Kensuke Harada, Fulio Kanehiro, Go Miyamori & Kazuhito Yokoi. *Humanoid Robot HRP-3.* In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008.

[Kavraki 96] L. E. Kavraki, P. Svestka, J.-C. Latombe & M. Overmars. *Probabilistic Roadmaps for Path Planning in High Dimensional Configuration Spaces.* IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pages 566–580, 1996.

[Khalil 02] Wisama Khalil & Etienne Dombre. Modeling, identification & control of robots. Hermes Sciences Europe, 3 edition, march 2002.

[Khatib 95] Oussama Khatib. *Inertial Properties in Robotic Manipulation : An Object-Level Framework.* International Journal of Robotics Research, vol. 14, page pp., 1995.

[Kieffer 98] M. Kieffer, L. Jaulin & E. Walter. *Guaranteed recursive nonlinear state estimation using intervalanalysis.* In Decision and Control, 1998. Proceedings of the 37th IEEE Conference on, volume 4, pages 3966–3971, Tampa, FL, USA, December 1998.

[Knoppel 99] Olaf Knoppel. *PROFIL/BIAS V2.0.* Technische Universitï£¡t Hamburg-Harburg Technische Iformatik II, D-21071 Hamburg Germany, february 1999.

[Kolev 00] L. Kolev & D. Penev. *An interval method for global inequality constraint optimization problems.* In Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on, volume 4, pages 617–620, MAY 2000.

[Konno 08] A. Konno, T. Myojin, T. Tsujita & M. Uchiyama. *Optimization of impact motions for humanoid robots.* In Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pages 647–652, Nice, September 2008.

[Kuffner 00] J. J. Jr. Kuffner & S. M. LaValle. *RRT-connect : An efficient approach to single-query path planning.* In Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on, volume 2, pages 995–1001, San Francisco, CA, USA, 2000.

[Kuffner 01] J. J. Jr Kuffner, K. Nishiwaki., S. Kagami., M. Inaba & H. Inoue. *Footstep planning among obstacles for biped robots.* In Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on, volume 1, pages 500–505, Maui, HI, USA, 2001.

[Lavalle 98] Steven M. Lavalle. *Rapidly-exploring random trees : A new tool for path planning.* Rapport technique, Computer Science Dept., IOwa State univ., 1998.

[LaValle 00] Steven M. LaValle & James J. Kuffner. *Rapidly-Exploring Random Trees : Progress and Prospects.* In 4th Int'l Workshop on the Algorithmic Foundations of Robotics, 2000.

[LaValle 06] Steven M. LaValle. Planning algorithms. Cambridge University Press, 2006.

[Lawrence] Craig Lawrence, Jian L. Zhou & Andre L. Tits. *User's Guide for CFSQP Version 2.5 : A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints.* Electrical Engineering Department, Institute for Systems Research University of Maryland, College Park, MD 20742.

[Lee 05] Sung-Hee Lee, Junggon Kim, F.C. Park, Mumsang Kim & James E. Bobrow. *Newtom-Type Algorithms for Dynamics-Based Robot movement Optimization.* In IEEE Transactions on robotics, volume 21, pages 657– 667, 2005.

[Lengagne 06] Sebastien Lengagne. Optimisation de mouvement multi-contacts pour le robot hrp-2. Master's thesis, Ecole centrale de Nantes, 2006.

[Lengagne 07] Sébastien Lengagne, Nacim Ramdani & Philippe Fraisse. *Guaranteed computation of constraints for safe path planning.* In

IEEE-RAS 7th International Conference on Humanoid Robots, 2007.

[Lengagne 08] Sébastien Lengagne, Nacim Ramdani & Philippe Fraisse. *A new method for generating safe motions for humanoid robots.* In IEEE-RAS Internationnal conference on Humanoid robots, 2008.

[Lengagne 09a] Sébastien Lengagne, Nacim Ramdani & Philippe Fraisse. *Planning and Fast Re-Planning of Safe Motions for Humanoid Robots : Application to a Kicking Motion.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2009.

[Lengagne 09b] Sébastien Lengagne, Nacim Ramdani & Philippe Fraisse. *Safe motion planning computation for databasing balanced movement of Humanoid Robots.* In EEE International Conference on Robotics and Automation, ICRA, 2009.

[Lozano-Perez 83] T.A. Lozano-Perez. *Spatial planning : a configuration space approach.* IEEE Trans on Computers, vol. 32, pages –, 1983.

[Merlet 01] Jean-Pierre Merlet. *A Generic Trajectory Verifier for the Motion Planning of Parallel Robots.* Journal of mechanical design (J. mech. des.), vol. 123, no. 4, pages 510–515, 2001.

[Mettler 08] B. Mettler. *An extremal fields approach for the analysis of human planning and control performance.* In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pages 2152–2158, Pasadena, CA,, May 2008.

[Miossec 06] Sylvain Miossec, Kazuhito Yokoi & Abderrahmane Kheddar. *Development of a Software for Motion Optimization of Robots - Application to the Kick Motion of the HRP-2 Robot.* In IEEE International Conference on Robotics and Biomimetics, pages 299–304, 2006.

[Miossec 08] S. Miossec, S.Lengagne & K. Yokoi A. Kheddar and. *Motion optimization of robotic systems and validation of HRP-2 robot.* In RSJ National Conference, 2008.

[Moore 66] R.E. Moore. Interval analysis. Prentice-Hall, Englewood Cliffs, 1966.

[Nakaoka 03] S. Nakaoka, A. Nakazawa, K. Yokoi, H. Hirukawa & K. Ikeuchi. *Generating whole body motions for a biped humanoid robot from captured human dances.* In Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on, volume 3, pages 3905–3910, September 2003.

[Nakaoka 04] S. Nakaoka, A. Nakazawa, K. Yokoi & K. Ikeuchi. *Leg motion primitives for a dancing humanoid robot.* In Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, volume 1, pages 610–615, April/May 2004.

[Nakaoka 07] Shin'Ichiro Nakaoka, Atsushi Nakazawa, Fumio Kanehiro, Kenji Kaneko, Mitsuharu Morisawa, Hirohisa Hirukawa &

Katsushi Ikeuchi. *Learning from Observation Paradigm : Leg Task Models for Enabling a Biped Humanoid Robot to Imitate Human Dances.* Int. J. Rob. Res., vol. 26, no. 8, pages 829–844, 2007.

[Neumaier 90] A. Neumaier. Interval methods for systems of equations. Cambridge university press, Cambridge, 1990.

[Oetomo 09] D. Oetomo, D. Daney & J. P. Merlet. *Design Strategy of Serial Manipulators With Certified Constraint Satisfaction.* IEEE Transactions on Robotics, vol. 25, no. 1, pages 1–11, February 2009.

[Olaru 09] Ionut Olaru, Sebastien Krut & Francois Pierrot. *Novel Mechanical Design of Biped Robot SHERPA Using 2 DOF Cable Differential Modular Joints.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (soumis à), 2009.

[Pepy 09] Romain Pepy. *vers une planification robuste et sûre pour les systèmes autonomes.* PhD thesis, L2S, Supelec, 2009.

[Pettré 03] Julien Pettré, Jean-Paul Laumond & Thierry Siméon. *A 2-stages locomotion planner for digital actors.* In SCA '03 : Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 258–264, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Piazzi 98] A. Piazzi & A. Visioli. *Global minimum-time trajectory planning of mechanical manipulators using interval analysis.* International Journal of Control, vol. 71, pages 631–652(22), November 1998.

[Piazzi 00] Aurelio Piazzi & Antonio Visioli. *Global Minimum-Jerk Trajectory Planning of Robot Manipulators.* In IEEE Transactions on Industrial Electronics, volume 47, pages 140–149, febru 2000.

[Ramdani 08] N. Ramdani, M. Gouttefarde, F. Pierrot & J. P. Merlet. *First results on the design of high speed parallel robots in presence of uncertainty.* In Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pages 2410–2415, September 2008.

[Renner 06] Reimund Renner & Sven Behnke. *Instability Detection and Fall Avoidance for a Humanoid using Attitude Sensors and Reflexes.* In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 2967–2973, october 2006.

[Rump 99] S.M. Rump. *INTLAB - INTerval LABoratory.* In Tibor Csendes, editeur, Developments in Reliable Computing, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999. `http://www.ti3.tu-harburg.de/rump/`.

[Sanchez-Lopez 03] Abraham Sanchez-Lopez. *Contribution à la plannification de mouvements en robotiques : Approches probabilistes et approches déterministes.* PhD thesis, Université Montpellier II, juillet 2003.

[Shawn 01] Rusaw Shawn. *Sensor-based motion planning in SE(2) and SE(3) via nonsmooth analysis*. Rapport technique, Oxford University Computing Laboratory, Oxford, UK, UK, 2001.

[Shin 08] Dongjun Shin, I. Sardellitti & O. Khatib. *A hybrid actuation approach for human-friendly robot design*. In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pages 1747–1752, Pasadena, CA,, May 2008.

[Suleiman 07] Wael Suleiman, Eiichi Yoshida, Jean-Paul Laumond & André Monink. *On Humanoid Motion Optimization*. In IEEE-RAS 7th International Conference on Humanoid Robots, 2007.

[Suleiman 08] W. Suleiman, E. Yoshida., J-P. Laumond. & A. A. Monin. *Optimizing Humanoid Motions Using Recursive Dynamics and Lie Groups*. In Information and Communication Technologies : From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on, pages 1–6, Damascus,, April 2008.

[Sunaga 58] T. Sunaga. *Theory of interval algebra and its application to numerical analysis*. RAAG Memoirs, Ggujutsu Bunken Fukuykai, vol. 2, pages 547–564, 1958.

[Tsujita 08a] T. Tsujita, A. Konno, S. Komizunai, Y. Nomura, T. Owa, T. Myojin, Y. Ayaz & M. Uchiyama. *Humanoid robot motion generation for nailing task*. In Advanced Intelligent Mechatronics, 2008. AIM 2008. IEEE/ASME International Conference on, pages 1024–1029, Xian, July 2008.

[Tsujita 08b] Teppei Tsujita, Atsushi Konno, Shunsuke Komizunai, Yuki Nomura, Takuya Owa, Tomoya Myojin, Yasar Ayaz & Masaru Uchiyama. *Analysis of nailing task motion for a humanoid robot*. In Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pages 1570–1575, Nice, September 2008.

[Uno 89] Y. Uno, M. Kawato & R. Suzuki. *Formation and control of optimal trajectory in human multijoint arm movement*. Biological Cybernetics, vol. 6, no. 2, pages 89–101, juin 1989.

[Vehi 01] J. Vehi, N.Luo, J. Rodellar & J. Armengol. *Digital Control via Interval Analysis*. Nonlinear Analysis, vol. 47, pages 203–212, 2001.

[von Stryk 93] Oskar von Stryk. *Numerical solution of optimal control problems by direct collocation*, 1993.

[Vukobratović 72] Miomir Vukobratović. *On the stability of anthropomorphic systems*. Mathematical Biosciences, vol. 15, pages 1–37, 1972.

[Walter 94] Eric Walter & Luc Jaulin. *Guaranteed characterization of stability domains via set inversion*. IEEE Transactions on Automatic Control, vol. 39, no. 4, pages 886–889, April 1994.

[Wieber 06] Pierre-Brice Wieber, Florence Billet, Laurence Boissieux & Roger Pissard-Gibollet. *The HuMAnS toolbox, a homogeneous framework for motion capture, analysis and simulation*. In the

ninth ISB Symposium on 3D analysis of human movement, 2006.

[Yagi 99] M. Yagi & V. Lumelsky. *Biped robot locomotion in scenes with unknown obstacles*. In Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, volume 1, pages 375–380, Detroit, MI, USA, 1999.

[Yanase 06] Toshihiko Yanase & Hitoshi Iba. *Evolutionary motion design for humanoid robots*. In GECCO '06 : Proceedings of the 8th annual conference on Genetic and evolutionary computation, pages 1825–1832, New York, NY, USA, 2006. ACM.

[Yoshida 05] Eiichi Yoshida, Igor Belousov, Claudia Esteves & Jean-Paul Laumond. *Humanoid motion planning for dynamic tasks*. In Humanoid Robots, 2005 5th IEEE-RAS International Conference on, pages 1–6, December 2005.

[Zaier 06] Riadh Zaier & Fumio Nagashima. *Motion Pattern Generator and Reflex System for Humanoid Robots*. In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 840–845, october 2006.

[Zaier 08] Riadh Zaier & Shinji Kanda. *Adaptive Locomotion Controller and Reflex System for Humanoid Robots*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2492–2497, september 2008.

[Zinn 04] M. Zinn, O. Khatib, B. Roth & J. K. Salisbury. *Playing it safe [human-friendly robots]*. IEEE Robotics & Automation Magazine, vol. 11, no. 2, pages 12–21, June 2004.

Titre : Planification et re-planification de mouvements sûrs pour les robots humanoïdes.

Résumé :
Ces travaux de thèse traitent de la génération de mouvements optimaux pour les robots humanoïdes. La plupart des méthodes de génération de mouvements sont inspirées de celles utilisées pour les robots manipulateurs. Elles se basent sur l'utilisation d'un algorithme d'optimisation qui nécessite une paramétrisation du mouvement ainsi qu'une discrétisation temporelle des contraintes définissant les limites physiques du robot. Nous montrons qu'une discrétisation faite à partir d'une grille temporelle peut compromettre la sécurité et l'intégrité des robots. De ce fait, nous proposons une nouvelle méthode de discrétisation garantie qui calcule les extrema des contraintes sur des intervalles de temps couvrant toute la durée du mouvement. Cette méthode de discrétisation pour le calcul des contraintes, nécessite un temps de calcul important. Nous avons, donc, développé une méthode hybride qui assure la validité des contraintes pour des temps de calcul comparables à celui des méthodes classiques. Cette méthode nous permet ainsi de générer une base de données de mouvements que nous avons utilisée lors d'une expérimentation de suivi de cible mobile. Nous sommes, donc, en mesure de générer un mouvement optimal parfaitement adapté à une configuration de l'environnement. Cependant, aucune méthode ne dispose d'un temps de calcul qui permette de réagir rapidement à une modification de l'environnement. Par conséquent, nous présentons une méthode de re-planification qui permet de générer un nouveau mouvement à partir d'un mouvement optimal calculé précédemment. Pour cela, nous calculons, hors-ligne, un sous-ensemble faisable autour des paramètres du mouvement qui vérifient les limites du robot. La re-planification consiste, alors, à chercher, en ligne, dans ce sous-ensemble les paramètres qui satisfont la nouvelle configuration de l'environnement. Nous avons testé la méthode de re-planification avec un mouvement de coup de pied où la position de la balle varie et nous obtenons un mouvement adapté en 1.5 s de temps de calcul.

Mots clés :
Robots humanoïdes, Planification de mouvements, Re-planification de mouvements, Validité des Contraintes, Analyse par Intervalles, Optimisation, Sous-Ensembles Faisables.

Title : Planning and re-planning of safe motions for humanoid robots.

Abstract :
These works deal with the computation of optimal motions for the humanoid robots. Most of the motion planning methods come from the motion planning of the manipulator robots. They rely on optimization algorithms which need a motion parametrization and a time-discretization of the constraints that define the physical limits of the robot. We show that a time-grid discretization is hazardous for the safety and the integrity of the robot. That is why, we propose a new method for the guaranteed discretization that computes the extrema of the constraints over time-interval that covers the whole motion duration. This method of discretization is time consuming. Thus, we developped a hybrid method that ensures the constraint validity within the same range of time of the state-of-the-art methods. With this method, we created a database of motions to follow a moving target. Consequently, we can generate an optimal motion that fits to the environment. However, there is no method which is fast enough to compute a new motion adapted to a new environment. Thus, we present a re-planning method that produces a new motion from a previous one. To do it, we compute, offline, a feasable sub-set around the motion that respects the constraint validity. The re-planning process consists in finding, in this sub-set, a new motion that is adapted to the new environment. We tested this re-planning method with a kicking motion where the position of the ball changes and we are able to find and adapted motion within 1.5s of CPU-time.

Keywords :
Humanoid Robots, Motion Planning, Motion Re-planning, Constraints Validity, Interval Analysis, Optimization, Feasable Sub-Sets.

Discipline : Génie Informatique, Automatique et Traitement du signal

Intitulé et adresse du laboratoire :
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM)
UMR CNRS / Université Montpellier II, No. 5506
161 rue Ada, 34092 Montpellier, France.