Université de Versailles Saint-Quentin-en-Yvelines

DEA Réalité Virtuelle et Maîtrise des Systèmes Complexes

Planification et génération de trajectoires d'un robot bipède en environnement non structuré

INRIA Rhône-Alpes - Projet BIP

Responsables de Stage :
Nathalie Cislo & Bernard Espiau

Jean-Matthieu Bourgeot

Février - Juin 2001

Table des matières

Table des figures

Remerciements

Je tiens à remercier tout particulièrement Nathalie Cislo ainsi que Bernard Espiau pour leur accueil, leur aide et le temps qu'ils ont bien voulu m'accorder tout au long de ce stage au sein du projet BIP de l'INRIA Rhône Alpes.

Je voudrais aussi remercier Fanny Benattar, James Lainé et Fabien Lydoire, avec qui j'ai partagé mon bureau durant ce stage, pour leur aide, leur convivialité et leur patience.

Je remercie également Christine Azevedo pour sa bonne humeur et ses remarques toujours pertinentes.

Enfin je remercie l'ensemble des membres du projet BIP sans qui ce stage n'aurait pas pu avoir lieu.

Chapitre 1

Introduction

Le projet BIP

L'INRIA Rhône-Alpes et le Laboratoire de Mécanique des Solides (LMS) de l'Université de Poitiers ont conçu un robot bipède anthropomorphe à quinze degrés de liberté : BIP2000 [ES00], [BIP]. Les deux laboratoires ont participé ensemble à la réalisation du premier prototype tout en ayant des rôles complémentaires. Le LMS s'est chargé de la partie mécanique du prototype, le service Moyens Robotiques de l'INRIA a réalisé la partie électronique du robot, l'équipe BIP de l'INRIA a eu la responsabilité de la partie contrôle commande du système. Ce travail commun a permis d'aboutir à une plate-forme expérimentale robuste pour l'étude de la marche bipède robotisée.

Le robot BIP2000 est un robot anthropomorphe à 15 degrés de liberté (ddl). Ce modèle ne prétend pas modéliser tous les degrés de liberté d'un vrai corps humain car il ne possède ni tête, ni bras. Néanmoins ces 15 ddl sont suffisants pour étudier la partie locomotion du corps humain. Sur la figure 1.3, les articulations actives sont représentées. A l'heure actuelle toutes les études ont été réalisées dans le plan Sagittal du bipède. Dans ce plan le robot se compose de 7 solides et de 6 articulations d'axes parallèles (2 pour les chevilles, 2 pour les genoux et 2 pour les hanches, voir la figure 1.1).

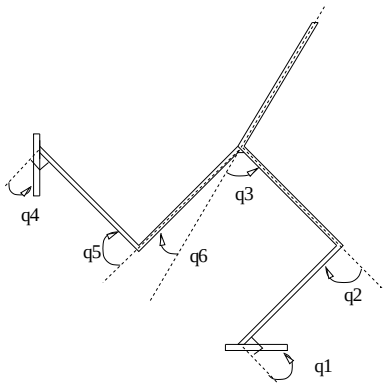


FIG. 1.1 – Les 6 articulations du plan Sagittal [AA00]

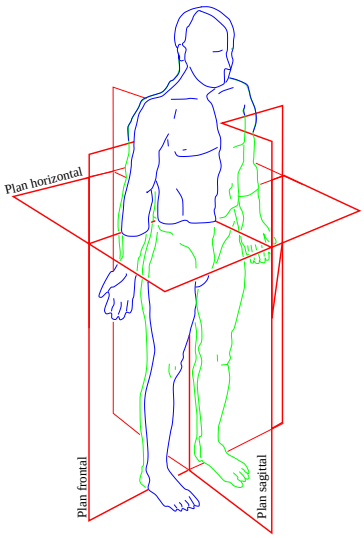


FIG. 1.2 – Les plans du corps humain [Ali99]

A ce jour le robot est capable de marcher, en utilisant ces 6 ddl, sur un sol plat, de maintenir un contrôle postural pendant des mouvements de forte amplitude, en équilibre statique sur un pied. [ES00] et [AA00] donnent une description complète de BIP et de ses premières démarches.

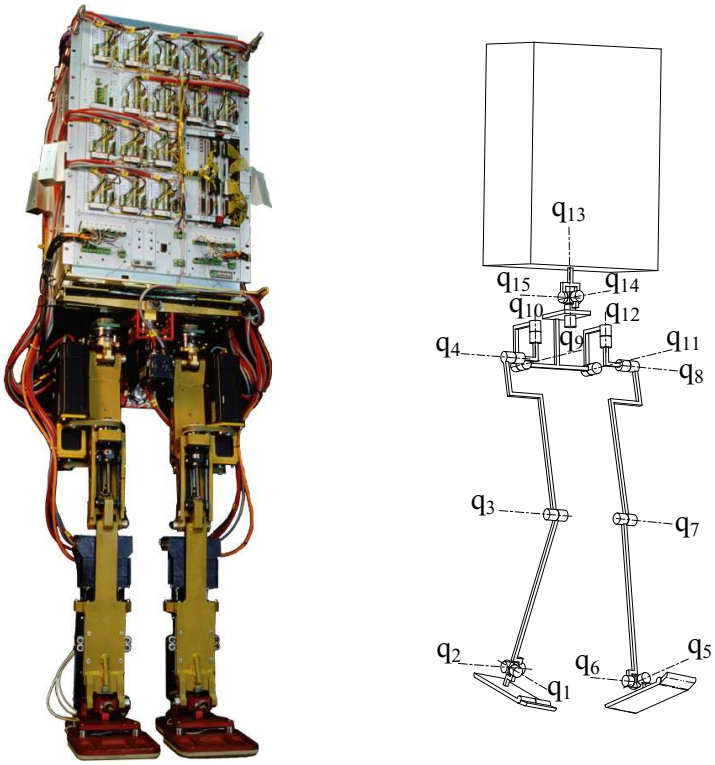


FIG. 1.3 – Le robot BIP et ses 15 articulations actives

La stabilité statique d'un robot bipède

Le bipède est statiquement stable si la projection de son centre de masse sur le sol se trouve à l'intérieur du polygone de sustentation.

Polygone de sustentation :
Lorsque le bipède est sur un seul pied (phase de simple support), le polygone de sustentation correspond à la surface de contact du pied sur le sol. Lorsque le robot est sur ses deux pieds (phase de double support) le polygone de sustentation correspond à l'empreinte des deux pieds et à la surface interpodale.

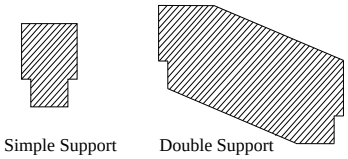


FIG. 1.4 – Polygone de sustentation

Démarche statiquement stable

La démarche du bipède est statiquement stable si la stabilité statique du robot est vérifiée à chaque instant de son déplacement. Pour l'instant, nous n'étudions que ce type de démarche, cela implique que le mouvement du bipède ne doit pas être trop rapide, pour ne pas introduire des forces d'inertie trop importantes qui risquent de compromettre la stabilité statique du robot.

La démarche du bipède peut être vue comme une suite d'empreintes sur le sol (voir la figure 1.5). La problèmatique est donc de générer des trajectoires (dans l'espace articulaire) qui permettent

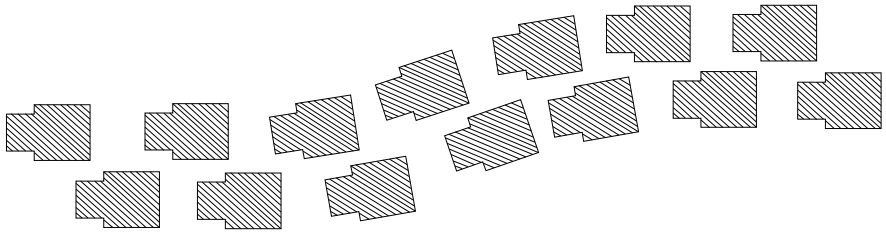


FIG. 1.5 – Suite d'empreintes

de déplacer le pied d'une position à une autre tout en garantissant la stabilité statique du bipède.

Génération des trajectoires

Actuellement, entre deux empreintes, les trajectoires sont générées par interpolation entre trois positions statiquements stables :

- la posture de départ, q_{init},
- la posture d'arrivée, q_{final},
- une position intermédiaire qui place le pied libre à une certaine distance du sol.

En dehors de ces trois postures qui sont statiquement stables, nous n'avons pas de garantie sur la stabilité durant le déplacement entre ces trois états.

Cette méthode convient pour des trajectoires simples : en ligne droite sur terrain plat par exemple [AA00]. Mais nous voulons développer une méthode permettant de générer ces trajectoires de façon automatique, et cela dans un contexte plus général : en 3D et en environement non structuré par exemple.

Des algorithmes existant permettent de calculer des postures stables pour une position et une orientation donnée du pied libre (le pied de support étant bien entendu fixe sur le sol). La méthode la plus simple, mais aussi la plus exigeante en temps de calcul, serait de décomposer le mouvement en une multitude de postures "clefs". Si ces postures sont assez proches les unes des autres, une simple interpolation suffirait à maintenir la stabilité.

Le problème qui nous intéresse dans la première partie est donc de développer une méthode de génération de trajectoires (dans l'espace articulaire) qui garantisse la stabilité du bipède entre deux postures clefs assez espacées l'une de l'autre. Ainsi on pourra , au mieux, n'utiliser que deux postures clefs pour décrire un pas (le début et la fin). Le chapitre 2 présente l'état de l'art des techniques de génération de trajectoires, essentiellement utilisées dans le domaine des robots manipulateurs. Puis le chapitre 3 présente l'adaptation de la méthode du Gradient Projeté (GPM) aux robots bipèdes. Enfin le chapitre 4 présente des résultats de simulation. La deuxième partie du rapport traite du problème de la planification de chemin en environnement non structuré. Le chapitre 5 présente l'état de l'art, le chapitre 6 pose une classification des différents types de terrain que peut rencontrer le bipède. Le chapitre 7 propose une méthode de planification dont les résultats se trouvent au chapitre 8.

Première partie

Génération de trajectoires

Chapitre 2

État de l'Art

Le problème auquel nous nous intéressons est de pouvoir déplacer le robot entre deux positions statiquement stables (voir figure 2.1).

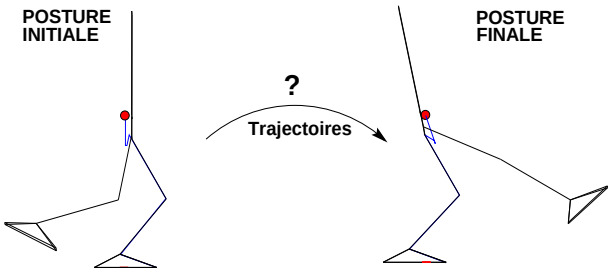


FIG. 2.1 – Deux positions stables

Pour préserver la stabilité statique du bipède au cours de ce déplacement, la projection du centre de masse du robot (CdM) doit toujours être située à l'intérieur de la surface de sustentation du robot.

Pour trouver les trajectoires (dans l'espace articulaire) réalisant cette condition, l'idée est d'utiliser les travaux de Espiau et Boulic [EB98] qui montrent que :

- la position du centre de masse d'une chaîne articulaire arborescente (le robot bipède) peut être représentée par le point terminal d'une chaîne cinématique série (un bras manipulateur par exemple) ;
- le robot équivalent a le même nombre d'articulations que le robot réel et les variables articulaires sont égales à celles de la chaîne arborescente, par contre les dimensions et les masses de chaque corps sont fonction des caractéristiques de la chaîne arborescente.

Dans ces conditions, l'étude de la position du CdM du robot bipède est équivalente à l'étude des déplacements du point terminal d'une chaîne articulaire série (voir figure 2.2). Pour l'étude de ce type de système, on peut s'inspirer des travaux déjà effectués dans le domaine des robots manipulateurs. Sciavicco et Siciliano [SS00] donnent les bases de la modélisation et du contrôle des robots manipulateurs.

Pour contrôler la convergence de q vers q_{final} sans déplacer le CdM, il est nécessaire d'utiliser la redondance du système. Le chapitre 4 de [SBE91] donne les définitions et les problèmes relatifs aux robots redondants, ainsi que quelques éléments de réponse par l'approche des fonctions de tâche. Il faut garder à l'esprit que la redondance d'un robot est fortement liée à la tâche que l'on veut lui faire accomplir. Par exemple, dans le plan, un bras 3ddl est redondant si on veut placer son extrémité en un point précis, alors qu'il n'est pas redondant si on veut fixer aussi l'orientation de l'outil.

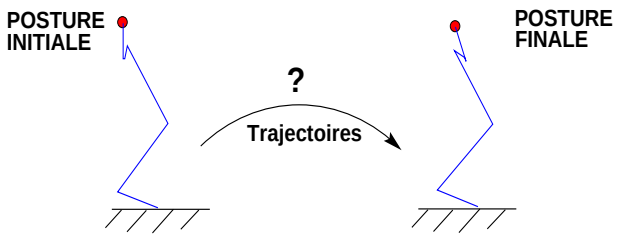


FIG. 2.2 – Le problème aprés transformation

Par exemple, sur la figure 2.3, on voit que l'on peut réorganiser la configuration interne d'un bras 3ddl sans déplacer la position de son extrémité.

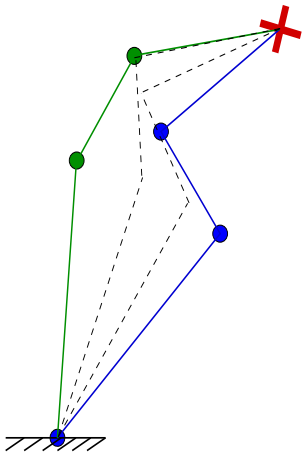


FIG. 2.3 – Redondance du bras 3ddl

2.1 Les fonctions de tâches

Pour simplifier la notion de fonction de tâche [SBE91] on peut dire qu'une fonction de tâche est une fonction qui s'annule lorsque le robot réalise cette tâche.

Par exemple $e(q,t) = f(q) - e_d(t)$ est une fonction de tâche où :
– $f(q)$ est la fonction cinématique directe du robot,
– $e_d(t)$ la trajectoire désirée.

De plus une tâche est redondante si et seulement si $\frac{\partial e}{\partial q}$ est de rang plein.
Dans le cas qui nous intéresse, on pourrait prendre la fonction de tâche suivante :

$$e_1(q) = f(q) - Xg$$

où $f(q)$ donne la position du point terminal du robot série (donc la position du CdM du robot bipède). Xg est la position désirée du CdM.

Si $e_1(q) = 0$ tout au long du mouvement, la stabilité statique du bipède est préservée.

Pour réaliser la condition $e_1 = 0$, il faut résoudre le problème de cinématique inverse du manipulateur.

La cinématique directe d'un manipulateur est décrite par la relation :

$$\begin{aligned} x &= f(q) \\ \dot{x} &= J\dot{q} \end{aligned} \tag{2.1}$$

$$\text{avec } \dim x = m, \dim x = n \text{ et } J = \frac{\partial f}{\partial q}.$$

Si le robot n'est pas redondant, la solution de la cinématique inverse est unique (si elle existe) :

$$\dot{q} = J^{-1}\dot{x}$$

Or dans notre cas le système est redondant donc J n'est pas carré, on a $n > m$. Ce problème peut donc admettre plusieurs solutions, voir même une infinité si la redondance du robot est élevée. Pour choisir la meilleur solution il faut introduire un critère à minimiser.

On cherche la solution qui minimise la vitesse des articulations :

$$\min \tfrac{1}{2}\dot{q}^T\dot{q}$$

Solution qui est obtenue grâce à la pseudo-inverse : $J^\dagger = J^T(JJ^T)^{-1}$

$$\dot{q} = J^\dagger\dot{x}$$

Méthode utilisant la pseudo-inverse :
D'une façon plus générale on peut vouloir minimiser ce critère :

$$\min \frac{1}{2}\dot{q}^T W\dot{q}$$

Où W est une matrice de pondération.
La pseudo-inverse généralisée devient : $J_W^\dagger = W^{-1}J^T(JW^{-1}J^T)^{-1}$
La solution de cinématique inverse est toujours donnée par :

$$\dot{q} = J_W^\dagger\dot{x} \tag{2.2}$$

De plus si on choisit $W = H$ (H étant la matrice d'inertie du robot), la solution choisie minimise l'énergie cinétique.

En utilisant cette formule, on peut donc résoudre le problème de cinématique inverse d'un système redondant. Ici la redondance sert seulement à réduire la consommation du système, il serait intéressant de pouvoir utiliser les mouvements supplémentaires pour exécuter une autre tâche (une tâche secondaire e_2). Dans cette optique la résolution du problème de cinématique inverse est la solution du problème suivant :

Minimiser la tâche secondaire e_2,
Sous la contrainte $e_1 = 0$.

Dans notre cas, on prend comme tâche principale : le maintien de la stabilité statique, et comme tâche secondaire la convergence des variables articulaires q vers les valeurs de consignes q_{final}, correspondant à la posture désirée. [CM00] et [CD95] donnent deux approches possibles pour résoudre ce problème d'optimisation sous contraintes : les méthodes GPM et WLN.

2.2 Méthodes d'optimisation locale

2.2.1 La méthode GPM

La méthode GPM (Gradient Projection Method) calcule la direction du déplacement $\frac{dq}{dt}$ qui tend à réduire la tâche secondaire e_2. Ce vecteur est ensuite projeté sur le noyau de la tâche

principale, l'erreur de la tâche secondaire est donc réduite sans modifier le bon déroulement de la tâche principale.

$$\dot{q} = J^{\dagger}\dot{x}_d + (I - J^{\dagger}J).\dot{q}_0 \tag{2.3}$$

où :
$\dot{x}d$ permet de réaliser e_1, par exemple $\dot{x}_d = Kp.(f(q) - Xg)$,
$\dot{q}_0$ est le gradient de e_2 : $\dot{q}_0 = k_0(\frac{\partial e_2}{\partial q})^T$,
$(I - J^{\dagger}J)$ est l'opérateur de projection sur le noyau de J.

2.2.2 La méthode WLN

La méthode WLN (Weighted Least Norm) est une méthode des moindres carrés pondérés où les coefficients de pondération varient en fonction de e_1 et e_2. Dans l'équation 2.2, W n'est plus une matrice constante mais vaut :

$$W = \begin{bmatrix} w_1 & 0 & 0 & ... & 0 \\ 0 & w_2 & 0 & ... & 0 \\ ... & ... & ... & ... & 0 \\ 0 & 0 & 0 & ... & w_n \end{bmatrix}$$

Où les éléments w_i de la diagonale de W sont définis comme suit :

$$w_i = 1 + \left|\frac{\partial e_2}{\partial q_i}\right|$$

Il est à noter qu'en plus de réaliser les tâches e_1 et e_2, ces deux méthodes tendent aussi à minimiser les vitesses articulaires. Dans la référence [CD95], Chan et Dubey comparent les deux méthodes sur un exemple à 7 degrés de liberté. Le principal problème de la méthode GPM est qu'elle détermine la direction du déplacement $\frac{\partial e_2}{\partial q}$, mais pas sa norme. La norme de $\frac{\partial e_2}{\partial q}$ doit être ajustée par un coefficient k_0 choisi par essais-erreurs. Un coefficient constant ne peut pas être utilisé si le noyau de J_1 est trop petit.

Lorsque l'on étudie plus en détail les contraintes liées au bipède, on s'aperçoit qu'il n'y a pas qu'une seule contrainte secondaire mais plusieurs. Par exemple on peut prendre en compte :
- les limites géométriques des articulations (lorsque la liaison arrive en butée),
- les limites énergétiques des moteurs (il faut que les tensions calculées soient admissibles par les moteurs),
- les conditions imposées par les obstacles environnant le bipède.

C'est pourquoi il faut introduire la notion de fonctions de tâches multiples. [CS97] et [NHY87] donnent deux solutions pour la prise en compte de tâches multiples. La première référence utilise un schéma en parallèle où les tâches ont la même priorité alors que la deuxième référence place les tâches en cascade ce qui permet de leur affecter des priorités les unes par rapport aux autres. Nenchev et Sotirov [NS94] proposent une méthode permetttant de modifier de façon dynamique les priorités de chacune des tâches.

Toutes les méthodes présentées ci-dessus ont un caractère local, c'est à dire qu'à chaque instant la méthode calcule la direction privilégiée qui réduit l'erreur des tâches secondaires sans modifier la tâche principale. Ces méthodes ont l'avantage de pouvoir être calculées en ligne si un critère garantit qu'il existe bien une solution, sinon le mouvement risque de s'arrêter en cours de route (par exemple, si on se trouve dans la situation où la tâche $e_1$est réalisée et où la direction, dans laquelle on veut se déplacer pour réduire e_2, est orthogonale au noyau de e_1).

Nous avons toujours la possibilité de faire tout le calcul hors ligne, pour être sûr qu'il existe bien un chemin réalisable, mais dans ce cas là il est préférable d'utiliser une approche globale. [KW88] et [GTJ95] donnent quelques exemples d'utilisation des méthodes globales pour le contrôle de systèmes redondants.

2.3 Méthodes d'optimisation globale

Les méthodes GPM et WLN ont l'avantage de pouvoir être calculées en temps réel. En effet le calcul de $\dot{q}$ ne nécessite que les informations capteurs présentes à l'instant présent. Cependant, l'inconvenient de ce type de méthode est la possibilité d'obtenir une solution non optimale dans le cas de long parcours, où lorsque la contrainte est de passer le plus loin possible des points singuliers.

Pour tenir compte de la globalité du chemin lors de l'optimisation il faut considèrer le critère à minimiser sous la forme d'une intégrale. Le problème est alors de la forme :

Minimiser la tache secondaire $I = \int_{ti}^{tf} e_2(t).dt$,
Sous la contrainte $e_1 = 0$.

Un critère typique de minimisation est la norme du vecteur de vitesses angulaires : $\min I = \int_{ti}^{tf} (\dot{q}^T \dot{q}).dt$

2.3.1 Méthode basée sur le calcul de la variation de la fonction augmentée

Kazerouian et Wang [KW88] utilisent le calcul de la variation de I^*(la fonction augmentée de $I = \int_{ti}^{tf} (\dot{q}^T \dot{q}).dt$,) pour minimiser globalement la vitesse des articulations.

Il en résulte que la solution globale vérifie l'équation :

$$\ddot{q} = J^T (JJ^T)^{-1} (\ddot{X} - \dot{J}\dot{q})$$

Le problème revient à résoudre un système de n équations différentielles du second ordre. Dans notre cas, les bornes sont imposées à droite et à gauche.

Nakamura et Hanafusa [NH87] montrent qu'il est possible de mettre ce même problème sous la forme d'un système de $2n$ équations du premier ordre en utilisant le principe du maximum de Pontryagin.

2.3.2 Principe du maximum de Pontryagin

Le problème de contrôle optimal peut se mettre sous la forme :

$$\begin{aligned} \dot{q} &= J_1^{\dagger} \dot{e}_1 + (I - J_1^{\dagger} J_1).y \\ e_2 &= \int_{ti}^{tf} p(q,t).dt \end{aligned}$$

L'Hamiltonien s'écrit :

$$H(\psi, q, t, y) = -p + \psi^T g \text{ où } \psi \text{ est un vecteur de } \mathbb{R}^n.$$

La trajectoire optimale $q(t)$ est solution du système :

$$\dot{q} = (\frac{\partial H}{\partial \psi})^T, \dot{\psi} = -(\frac{\partial H}{\partial q})^T$$

Cela donne un système de $2n$ équations du premier ordre.

2.4 Optimisation au niveau cinétique, plutôt qu'au niveau cinématique

Le contrôle de robot implique la génération de couples moteurs tel que le robot suive les trajectoires désirées. Une trajectoire est qualifiée de bonne lorsqu'elle nécessite un minimum d'accéleration. Il est donc intéressant de résoudre le problème de la redondance au niveau des accélérations articulaires plutôt qu'au niveau des vitesses. [KW88] et [GL91] montrent des exemples d'utilisations.

Pour déterminer le contrôle au niveau des accélérations il suffit de dériver l'équation 2.1, ce qui donne :

$$\ddot{x} = J\ddot{q} + \dot{J}\dot{q}$$

et on obtient la généralisation de l'équation 2.3 :

$$\ddot{q} = J^{\dagger}(\ddot{X} - \dot{J}\dot{q}) + (I - J^{\dagger}J)(J^{\dagger}\dot{X} - J^{\dagger}JZ + \dot{Z})$$

De cette équation Kazerounian et Wang [KW88] concluent qu'une optimisation locale au niveau des accélérations angulaires est équivalente à une optimisation globale au niveau des vitesses.

2.5 Éléments de choix entre les méthodes locales ou globales

Les méthodes globales permettent de trouver la solution optimale au niveau énergétique et aussi de passer au plus loin des points singuliers. Par contre elles nécessitent beaucoup plus de calculs et elles doivent donc être calculées hors ligne. Ainsi les trajectoires sont générées avant le début du mouvement, et ne peuvent pas prendre en compte un incident (une perturbation par exemple) durant le suivi de trajectoire. A l'inverse, les méthodes locales calculent q_{ref} à chaque instant, et tiennent ainsi compte des changement de condition (voir Figure 2.4).

Si une perturbation écarte les variables articulaires de leur valeur de référence, le controleur $C(p)$ fait en sorte que le système rattrape les valeurs de consigne. Mais le chemin pris pour annuler cet écart est pris de façon arbitraire, l'exemple suivant en montre les conséquences.

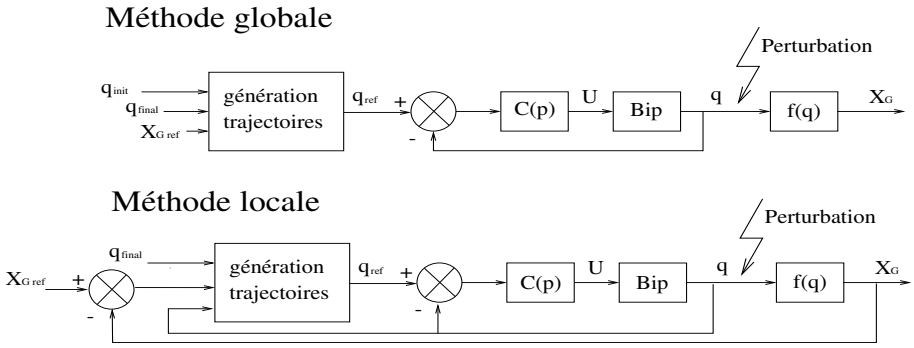


FIG. 2.4 – Schéma des deux approches

Un petit exemple sur un bras 2ddl où la méthode globale ne fonctionne pas (figure 2.5) :

Phase 1 : une position d'équilibre.

On se place dans le cas simple où le bras n'est composé que de deux corps,et en régime permanent (les valeurs de références q_{ref} données par la génération de trajectoire globale sont constantes $q_{1ref} = -5°$ et $q_{2ref} = 5°$).

Phase 2 : une perturbation arrive...
Admettons qu'une perturbation intervienne, et place le système dans la phase 2 de la figure 2.5. Le contrôleur $C(p)$ est censé annuler les erreurs sur q_1 et q_2.

Phase 3 : Le contrôleur annule les écarts.
Le contrôleur $C(p)$ annule les erreurs sur q_1 et q_2. Mais si par exemple, la constante de temps de la boucle de régulation de la deuxième articulation est beaucoup plus faible que celle de la première, on peut se trouver dans le cas où :
– l'erreur sur q_2 est annulée et,
– l'erreur sur q_1 est toujours présente.

Phase 4 : La perturbation est complètement absorbée.
Au bout d'un temps suffisamment long le contrôleur $C(p)$ annule les erreurs sur q_1 et q_2, et on se retrouve dans la configuration de départ (phase 4).

En observant la figure 2.5, on remarque que l'extrémité du bras est sortie de la zone d'équilibre (phase 3). L'utilisation d'une méthode d'optimisation locale aurait permis de calculer en temps réel une trajectoire qui rattrappe la position de consigne sans compromettre la stabilité.

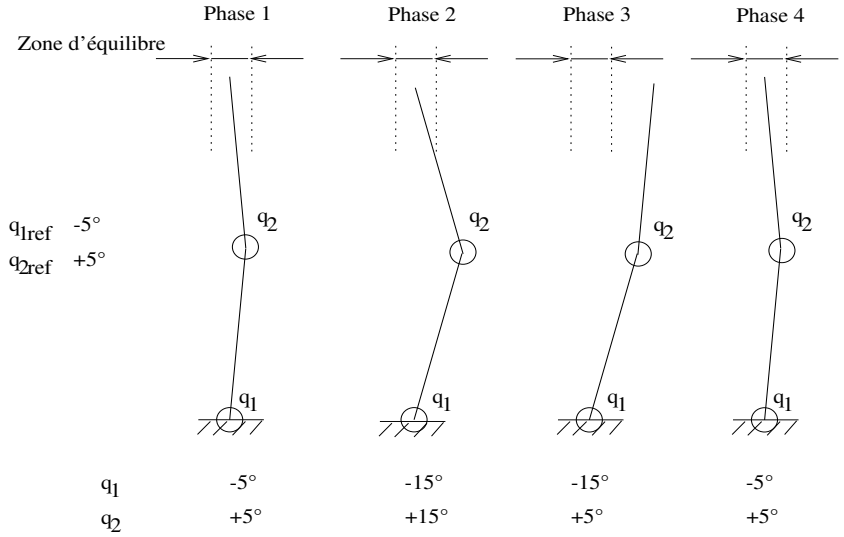


FIG. 2.5 – Exemple d'échec pour la méthode globale

2.6 Conclusion

En conclusion, nous allons donc utiliser une méthode locale pour résoudre ce problème de cinématique inverse. Nous allons utiliser la méthode du Gradient Projeté (GPM), puisque c'est la méthode qui permet de prendre en compte le plus facilement les différentes contraintes imposées au robot.

Ces contraintes sont mises sous la forme de fonctions de tâche :
- maintenir la position du centre de pression (CP) ;
- converger vers la posture souhaitée ;
- éviter les butées articulaires ;
- empécher les contacts avec le sol.

Le chapître suivant développe la méthode GPM et son adaptation au bipède.

Chapitre 3

La méthode du gradient projeté (GPM)

Pour résoudre le problème de génération de trajectoires, nous utilisons une méthode locale pour des raisons de rapidité de calcul. Dans les méthodes locales, nous choisissons la méthode du gradient projeté parce que c'est la méthode qui permet d'intégrer le plus simplement les tâches secondaires respectant les contraintes du bipède.

3.1 Formulation générale

Enoncé :

A chaque instant, il s'agit de calculer les vitesses $\dot{q}^\star$ des articulations afin de réaliser la tâche principale et de minimiser la tâche secondaire.

Soit $e_1^\star$ la position de consigne pour la tâche principale, alors $\dot{e}_1^\star = -k(e_1 - e_1^\star)$,

Soit $\dot{q}_0$ le gradient de la tâche secondaire $\dot{q}_0 = \frac{\partial e_2}{\partial q}$.

Les vitesses articulaires sont données par :

$$\dot{q}^\star = J^\dagger \dot{e}_1^\star + (I - J^\dagger J)\dot{q}_0$$

Démonstration (tirée de [SS00]) :

Rappel sur la pseudo inverse :

Pour inverser l'équation de cinématique directe (2.1) d'un robot manipulateur redondant, sachant que l'on cherche $\dot{q}$ permettant de déplacer le manipulateur de $\dot{x}_d$, il est possible d'utiliser la pseudo inverse. La pseudo inverse est obtenue en minimisant le critère suivant :

$$g(\dot{q}) = \frac{1}{2}\dot{q}^T W \dot{q}$$

Où W est une matrice de pondération symétrique définie positive. Pour résoudre ce problème en tenant compte de la contrainte (2.1), le critère est modifié par les multiplicateurs de Lagrange λ et devient :

$$g(\dot{q}, \lambda) = \frac{1}{2}\dot{q}^T W \dot{q} + \lambda^T(\dot{e}_1^\star - J\dot{q})$$

La solution $\dot{q}$ qui minimise $g(\dot{q}, \lambda)$ doit satisfaire les conditions suivantes :

$$\left(\frac{\partial g}{\partial \dot{q}}\right)^T = 0 \qquad \left(\frac{\partial g}{\partial \lambda}\right)^T = 0$$

Deux ces deux conditions, on en déduit que :

$$\begin{array}{c} \dot{q} = W^{-1}J^T\lambda \\ \dot{x}_d = J\dot{q} \end{array} \tag{3.1}$$

En les combinant, on obtient :

$$\dot{x}_d = JW^{-1}J^T\lambda$$

Si J est de rang plein, $JW^{-1}J^T$ est carrée et de rang plein, on peut l'inverser :

$$\lambda = (JW^{-1}J^T)^{-1}\dot{e}_1^\star$$

Par subtitution dans (3.1), on obtient la solution de notre problème de cinématique inverse.

$$\dot{q}^\star = W^{-1}J^T(JW^{-1}J^T)^{-1}\dot{e}_1^\star$$

En prenant $J = I$, on reconnait la formule de la pseudo-inverse.

$$J^\dagger = J^T(JJ^T)^{-1}$$

$$\dot{q}^\star = J^\dagger\dot{e}_1^\star$$

La solution obtenue minimise localement la norme des vitesse angulaire, par contre en utilisant la pseudo inverse pondérée avec $W = H$ (H étant la matrice d'inertie), la solution minimise localement l'énergie cinétique.

Forme générale :

Si $\dot{q}^\star$ est une solution du problème de cinématique inverse, et si P est une matrice de projection de $\mathbb{R}^n$ sur le noyau de J, alors $\dot{q}^\star + P.\dot{q}_0$ est aussi une solution.

L'introduction du terme $P.\dot{q}_0$ permet d'utiliser les degrés de mobilités redondants afin d'accomplir une ou plusieurs tâches secondaires. Il reste à déterminer P.

Ceci est réalisé en minimisant le critère suivant :

$$g(\dot{q}, \lambda) = \frac{1}{2}(\dot{q} - \dot{q}_0)^T(\dot{q} - \dot{q}_0) + \lambda^T(\dot{e}_1^\star - J\dot{q})$$

La première condition nécessaire donne :

$$\dot{q} = J^T\lambda + \dot{q}_0$$

d'où

$$\lambda = (JJ^T)^{-1}(\dot{e}_1^\star - J\dot{q}_0)$$

On obtient finalement

$$\dot{q}^\star = J^\dagger\dot{e}_1^\star + (I - J^\dagger J)\dot{q}_0$$

$(I - J^\dagger J)$ est donc une des matrices de projection P, qui permettent de projeter $\dot{q}_0$ sur le noyau de e_1. Ainsi on détermine le vecteur qui minimise au mieux la tâche secondaire sans modifier la valeur de la tâche principale (e_1). Si $\dot{e}_1^\star = 0$, la chaine cinématique est reconfigurée sans modification de la position du point terminal.

3.2 Les tâches secondaires

Dans l'expression précédente, $\dot{q}_0$ est le gradient ($\frac{\partial e_2}{\partial q}$) de la tâche secondaire. Or dans notre cas nous voudrions pouvoir réaliser plusieurs tâches secondaires (noté e_{21} , e_{22} ,e_{23} ...), comme par exemple :

- Faire converger le bipède vers la posture souhaitée.
- Respecter les contraintes liées aux articulations du robot.
- Eviter que le robot rentre en contact avec d'éventuels obstacles (le sol par exemple).

Pour réaliser toutes ces tâches en parallèle, il faut prendre $\dot{q}_0 = k_1\frac{\partial e_{21}}{\partial q} + k_2\frac{\partial e_{22}}{\partial q} + k_3\frac{\partial e_{23}}{\partial q} + ...$ Le problème est de trouver un bon compromis pour les gains k_1, k_2 et k_3 qui permettent de converger quelque soit la situation.

3.2.1 Tâche e_{21} : converger vers la posture souhaitée

Nous avons vu qu'il est possible d'accomplir une tâche secondaire sans modifier le bon déroulement de la tâche principale. Dans notre cas, le positionnement du centre de pression est contrôlé par la tâche principale. Parallèlement, la tâche secondaire fait converger la position du robot vers sa position désirée q_{final}.

Soit la tâche secondaire $e_{21} = \frac{1}{2}(q^\star - q)^2$, qui tend à minimiser l'écart entre la posture actuelle du robot q et la posture souhaitée $q^\star$. Alors :

$$\dot{q}_0 = \frac{\partial e_{21}}{\partial q} = (q - q^\star)$$

La façon la plus simple de choisir $q^\star$, est de prendre $q^\star = q_{final}$. Mais dans ce cas l'accélération produite au démarrage est forte puisque $(q - q^\star)$ est maximal, et plus on se rapproche de q_{final}, plus $\dot{q}_0$ diminue. On obtient alors une réponse de la forme d'une exponentielle $(1 - e^{-t/\tau})$. Il est préférable de choisir un autre $q^\star$qui limite les variations brusques de $\dot{q}_0$. En choisissant une trajectoire de référence d'une forme comparable à la figure 3.1. On obtient à

- $t = t_{initial}$, $\dot{q}_0 = \frac{1}{2}(q_{initial} - q^\star) = 0$ donc la vitesse au démarrage est nulle.
- $t = t_{final}$, $\dot{q}_0$ tend vers zéro, donc la vitesse tend vers zero en fin de mouvement.

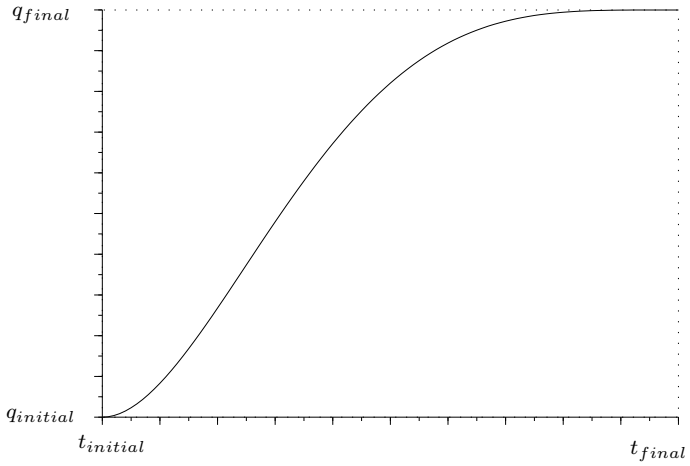


FIG. 3.1 – Trajectoire de référence pour les variables articulaires

En d'autres termes, nous essayons de faire suivre aux variables articulaires une trajectoire de référence, sans modifier la position du centre de masse (ou de pression). Si ces deux objectifs ne sont pas réalisables en même temps, cette méthode garantit en priorité le bon positionnement du centre de masse. Dans ce cas, les variables articulaires s'éloignent de leur trajectoire de référence, puis elles y reviennent ensuite.

La convergence vers la posture souhaitée n'est pas la seule tâche secondaire qu'il faille imposer au robot. Il faut aussi veiller à ne pas rentrer en butée au niveau des articulations, et éviter les contacts avec le sol.

3.2.2 Tâche e_{22} : éviter les butées articulaires

Pour éviter que les articulations n'arrivent en butées, il faut utiliser un critère de la forme de la figure 3.2. Ce critère est quasiment nul lorsque les articulations sont à l'intérieure de leur plage de fonctionnement, puis il tend vers l'infini lorsqu'une articulation se rapproche trop de sa butée.

Ce critère est modélisé par la fonction de tâche suivante :

$$e_{22} = \sum_{i}^{Ndof} \frac{(q_{i,max} - q_{i,min})^2}{(q_{i,max} - q_i)(q_i - q_{i,min})}$$

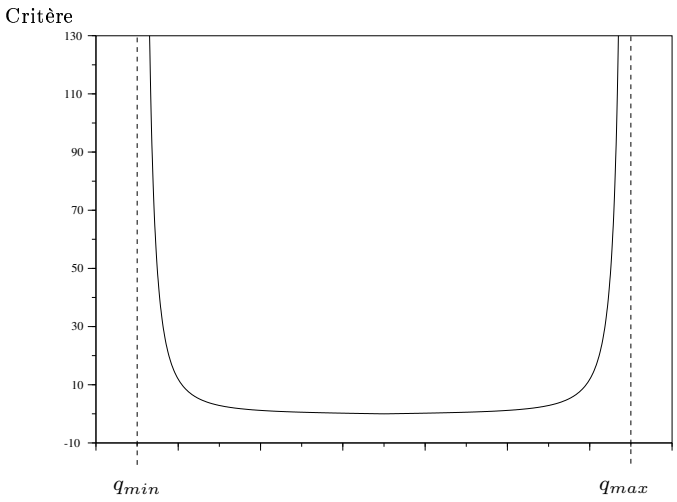


FIG. 3.2 – Critère sur les butées articulaires

avec le gradient de cette tâche :

$$\frac{\partial e_{22}}{\partial q} = \frac{(q_{max} - q_{min})^2(q_{max} - 2q + q_{min})}{(q_{max} - q)^2(q_{min} - q)^2}$$

$q_{i,max}$ et $q_{i,min}$ sont respectivement les valeurs maxima et minima que peut attreindre l'articulation q_i.

3.2.3 Tâche e_{23} : empêcher les contacts avec le sol

De même que pour les butées articulaires, si le robot suit exactement la trajectoire de référence, le pied libre peut entrer en collision avec le sol. Pour éviter cela, il faut pénaliser la fonction de coût avec un critère qui tend vers l'infini lorsque le pied se rapproche trop du sol.

On veut maximiser la distance avec le sol, soit :

$$e_{23} = max \, \|p_i(q) - o\|$$

ou bien

$$e_{23} = min \left\| \frac{1}{p_i(q) - o} \right\|$$

avec

$$\frac{\partial e_{23}}{\partial q} = \frac{\partial p_i(q)}{\partial q} . \frac{1}{(p_i(q) - o)^2}$$

Où
- les p_i correspondent aux fonctions de cinématique directe, qui donnent l'altitude des quatre coins du pied libre.
- o représente l'altitude du sol.

3.3 Tâches incompatibles

Dans notre cas les tâches secondaires sont exécutées en parallèle, c'est à dire que l'exécution des tâches e_{2i} n'influe pas sur la tâche principale, mais les tâches secondaires influent les unes sur les autres. Si deux tâches secondaires imposent des effets incompatibles entre eux, cela provoque une croissance rapide du terme $\dot{q}_0$ et la solution diverge. Ceci se produit par exemple si la tâche e_{21} tend à faire passer le pied très profondement dans le sol, et que la tâche e_{23} s'y oppose.

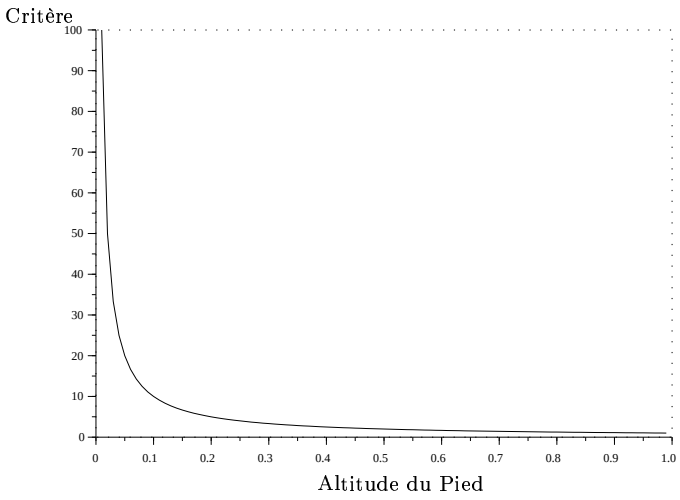


FIG. 3.3 – Critère sur l'altitude du pied

3.4 Priorité entre tâches

Pour remèdier à ce problème d'incompatibilité entre tâches, il est possible de hiérarchiser les tâches en fonction de leur importance. Si bien qu'une tâche de priorité plus faible ne peut pas rentrer en conflit avec une tâche plus élevée. Pour réaliser cela, il suffit de projeter le gradient de la tâche de plus faible priorité sur le noyau de la tâche précédente et ainsi de suite jusqu'à la projection sur le noyau de la tâche principale.

D'où la formulation suivante :

$$\dot{q} = J_1^{\dagger}\dot{e}_1 + (I - J_1^{\dagger}J_1)(J_2^{\dagger}\dot{e}_2 + (I - J_2^{\dagger}J_2)\dot{q}_0)$$

Cette formulation permet de réaliser,
- la tâche principale e_1,
- la tâche e_2 sans modifier la tâche e_1,
- les tâches secondaires (à travers le gradient $\dot{q}_0$) sans modifier les tâches e_1 et e_2.

Par contre il est à noter que la réalisation de la tâche principale e_1 perturbe la tâche e_2 (ainsi que les tâches secondaires). Dans notre étude, ceci n'a pas trop d'importance puisque les variations de la tâche principale (e_1) doivent être trés faibles (e_1 contrôle le centre de masse ou de pression).

Par contre pour que e_1 ne perturbe pas e_2, il est toujours possible d'utiliser la formulation qui permet de compenser les effets de e_1 sur e_2 [NHY87].

$$\dot{q} = J_1^{\dagger}\dot{e}_1 + (I - J_1^{\dagger}J_1)\tilde{J}_2^{\dagger}(\dot{e}_2 - J_2J_1^{\dagger}\dot{e}_1) + (I - J_1^{\dagger}J_1)(I - \tilde{J}_2^{\dagger}\tilde{J}_2)\dot{q}_0$$

$$\text{où } \tilde{J}_2 = J_2(I - J_1^{\dagger}J_1)$$

Chapitre 4

Résultats de simulation

Ce chapitre présente les résultats obtenus en simulation. Deux types de minimisation ont été testés : la minimisation des vitesses articulaires et la minimisation de l'énergie cinétique. Le deuxième type permet d'obtenir des trajectoires plus proche de la réalité.

4.1 Minimisation des vitesses articulaires

Dans un premier temps, nous appliquons au bipède la méthode du gradient projeté en utilisant la forme simple de la pseudo inverse.

$$J^{\dagger} = J^{T}(JJ^{T})^{-1}$$

La solution obtenue minimise donc la norme des vitesses articulaires et cette méthode donne des résultats satisfaisants pour le contrôle postural. Par contre l'algorithme trouve une solution qui n'est pas du tout anthropomorphe. Là où l'être humain plierait son genou, le robot garde la jambe tendue et préfère utiliser sa hanche en déplaçant son pied vers l'extérieur (Voir la figure 4.1)

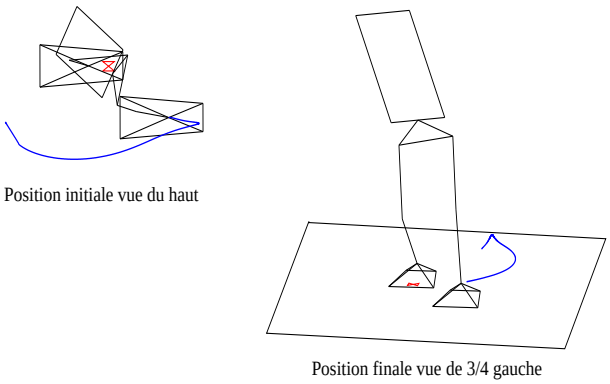


FIG. 4.1 – Trajectoire (de la cheville) générée par la minimisation des vitesses articulaires

4.2 Minimisation de l'énergie cinétique

Pour générer des trajectoires plus proches de la réalité, nous minimisons l'énergie cinétique dépensée par le bipède au lieu de minimiser la norme des vitesses articulaires. Pour cela il faut utiliser la forme générale de la pseudo inverse.

$$J_H^\dagger = H^{-1} J^T (J H^{-1} J^T)^{-1}$$

Où H est la matrice d'inertie du bipède.

Sur l'exemple de la figure 4.2, le robot suit une trajectoire plus classique. Pour passer son pied de l'arrière vers l'avant, il utilise la mobilité de son genou pour déplacer son pied libre en ligne droite. Pour cela, il plie la jambe libre au niveau du genou, alors que dans le cas précédent la jambe restait tendue.

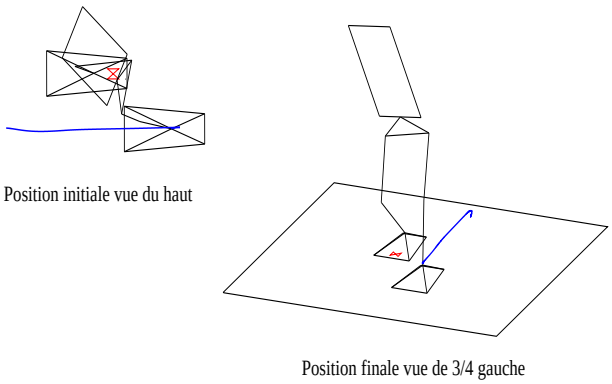


FIG. 4.2 – Trajectoire générée par minimisation de l'énergie cinétique

4.3 Analyse d'un exemple

Dans cette section, on trouve les résultats d'une simulation où le bipède fait passer sa jambe de l'arrière vers l'avant. Sur les figures 4.3 et 4.4, on peut voir les positions de départ et d'arrivée du bipède, ainsi que la trajectoire de la cheville du pied libre.

On constate qu'à mi-chemin, la trajectoire passe par un palier horizontal, ceci est dû à la tâche "éviter le sol" qui empêche le pied de descendre trop bas.

De même, la discontinuité, que l'on constate entre les instants 8 et 9 secondes sur toutes les courbes, est aussi dûe à la tâche "éviter le sol". C'est à ce moment là que le pied libre repasse au dessus de la limite au-delà de laquelle la tâche "éviter le sol" ne produit plus d'effet.

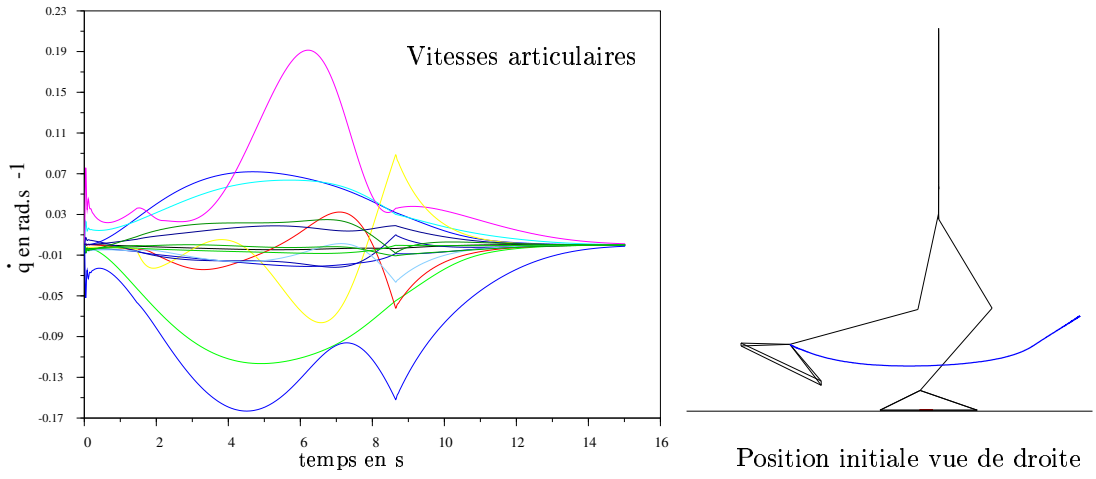
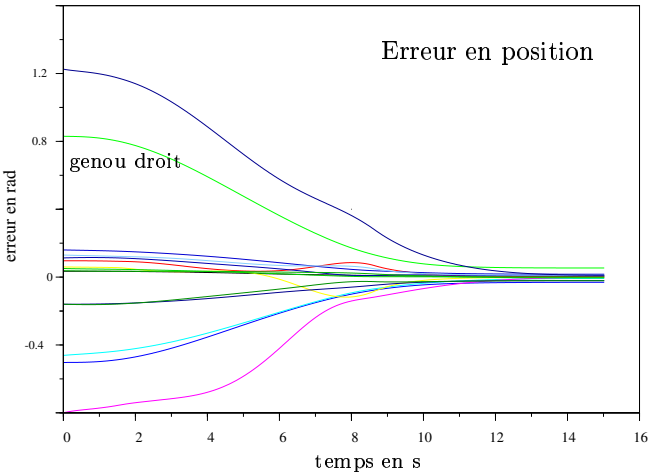
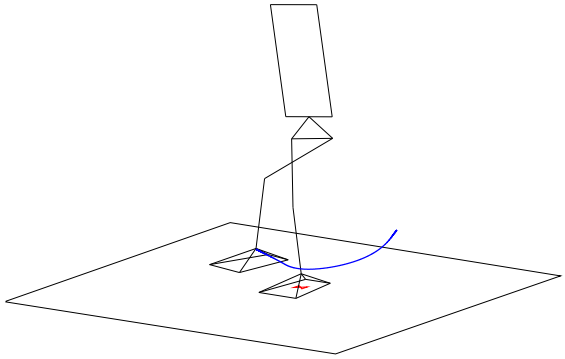


FIG. 4.3 – Position de départ et vitesses articulaires

Les courbes de la figure 4.3 montrent que la vitesse du robot tend vers zéro en fin de parcours. Par contre on remarque de petites oscillations au démarrage, celles ci sont dûes à la tâche "éviter les butées" qui tend à éloigner les articulations de leurs butées lors du démarrage.

Sur les courbes de la figure 4.4, on constate que les erreurs en position tendent vers zéro, ce qui veut dire que le bipède converge bien vers la posture souhaitée. Mais on observe un léger écart statique sur pratiquement toutes les articulations. Cet écart statique est dû à la tâche "éviter les butées", qui empêche d'aller trop près des limites articulaires, la posture souhaitée n'est donc pas accessible si elle est trop proche des limites. En effet l'écart statique le plus important se situe sur l'articulation correspondnat au genou droit, or c'est justement sur cette articulation que l'écart, entre la position souhaitée et les limites articulaires, est le plus faible.





Position finale vue de 3/4 gauche

FIG. 4.4 – Position finale et erreurs

Deuxième partie

Planification

Chapitre 5

État de l'Art

Le problème de la planification est de déterminer un chemin dans l'espace des postures admissibles permettant de relier la posture de départ à la configuration d'arrivée, sans rentrer en collision avec les éventuels obstacles et surtout en respectant les contraintes cinématiques du robot. Vu la complexité des contraintes cinématiques et dynamiques liées au bipède, nous allons nous placer dans un domaine d'étude simplifié.

Dans [CE01], Nathalie Cislo et Bernard Espiau étudient la planification de BIP2000 dans un univers peu structuré, composé de plans horizontaux et hexagonaux placés à diffèrentes altitudes.

Dans ce rapport, l'univers dans lequel se déplace le bipède est composé de plans inclinés (voir figure 5.1). Le problème est donc de trouver un chemin dans cet espace suivant lequel le robot puisse trouver une suite de postures admissibles. Dans un premier temps nous allons passer en revue les différentes techniques de planification de chemins pour un robot mobile, qu'il soit à roues ou à pattes.

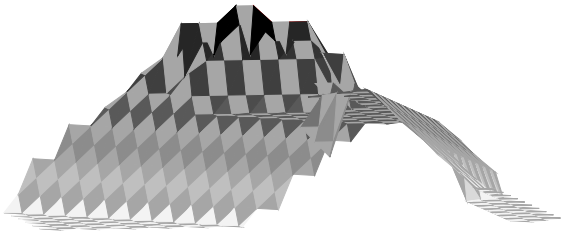


FIG. 5.1 – Exemple d'un univers non structuré

5.1 Robots mobiles

Typiquement la planification de chemins impose la minimisation de la longueur du trajet, mais le fait de prendre en compte des contraintes propres au robot entraîne la minimisation d'autres facteurs plus importants, comme par exemple :

- la courbure de la trajectoire pour les robots mobiles non holonomes,
- le nombre de pas (ou la longueur des pas) pour des robots à pattes,
- l'inclinaison du terrain.

Les techniques de planification peuvent se regrouper en deux groupes.

- la planification avec connaissances a priori de l'environnement,
- la planification locale.

5.1.1 Planification avec connaissances à priori de l'environnement

La connaissance a priori de l'environnement par le planificateur permet de connaître à l'avance les zones accessibles ou non par le robot, on peut définir l'espace accessible par le robot.

Espace accessible

L'espace accessible (ou espace de configuration), est l'espace que le robot peut atteindre compte tenu des obstacles et de la forme géométrique du robot. Si par exemple le robot est cylindrique (de rayon R), l'espace non accessible est l'espace correspondant aux obstacles augmenté d'une distance R (voir figure 5.2)

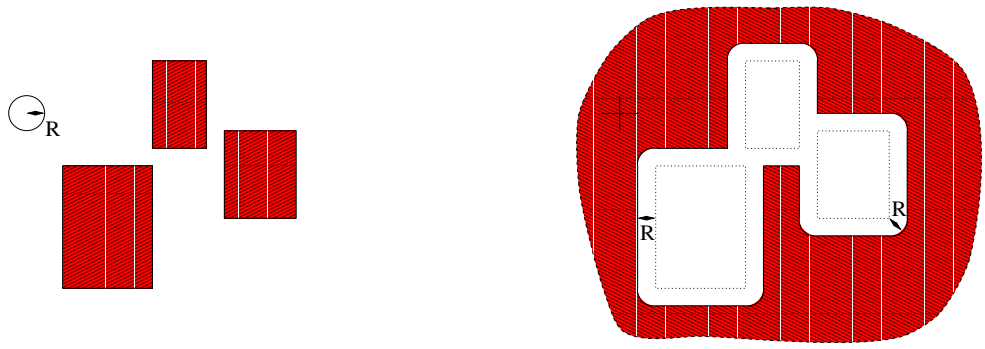


FIG. 5.2 – Espace accessible

Le problème est qu'il y a une infinité de chemins possibles. Pour chercher le meilleur chemin dans l'espace accessible, il faut discrétiser cet espace en un nombre fini de chemin.

Pour discrétiser l'espace on peut soit :

- découper l'espace en cases régulières de même dimension. Chaque case correspond au noeud d'un graphe. Puis un algorithme de recherche dans un graphe détermine le meilleur chemin. (voir figure 5.3)
- ou bien, on peut extraire de l'espace seulement les points caractéristiques de l'environnement. Par exemple le graphe de visibilité ne retient que les points générateurs de l'espace non accessible. Ceci a pour avantage de réduire la taille de graphe de recherche.

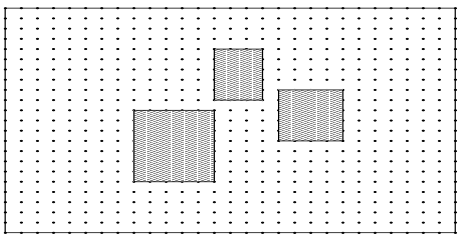


FIG. 5.3 – Discrétisation régulière de l'espace

Graphe de Visibilité

Une méthode possible pour discrétiser l'espace est de construire le graphe de visibilité, pour cela on associe un noeud à chaque sommet de l'espace accessible, plus deux noeuds pour les points de départ et d'arrivée. Les branches du graphe correspondent à toutes les liaisons entre deux noeuds qui ne traversent pas l'espace non accessible. Le choix d'un chemin reliant le départ et l'arrivée dans ce graphe permet d'éviter les obstacles (voir figure 5.4).

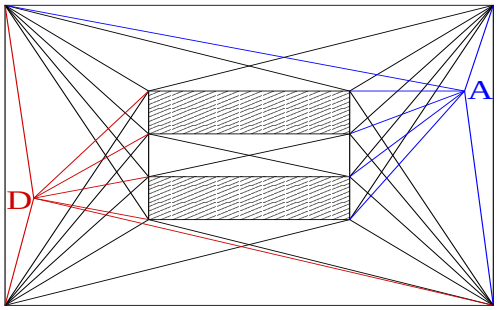


FIG. 5.4 – Graphe de visibilité

Il reste à déterminer le meilleur choix, pour cela on peut utiliser la méthode de Dijkstra, ou des méthodes heuristiques ;

Les algorithmes de recherche dans un graphe créent au fur et à mesure du déroulement de l'algorithme une structure de graphe orienté appelée arbre de recherche. A l'origine, cette structure ne comporte que le noeud correspondant à l'état de départ. Puis tant que l'état d'arrivée n'est pas découvert, ces algorithmes construisent l'arbre de recherche en développant chacun des noeuds.
La différence essentielle entre les deux méthodes suivantes réside dans la façon dont sont développés les noeuds lors de la recherche.

L'algorithme exhaustif de Dijkstra

Cet algorithme est dit exhaustif parce qu'il développe l'ensemble des noeuds d'une même profondeur (recherche dite en largeur d'abord). Mais seuls les chemins à coût minimaux sont retenus. Ceci permet de réduire le nombre de chemins étudiés, par contre il est nécessaire d'avoir traité tout les noeuds pour affirmer que le chemin trouvé est le plus court.
Pour résoudre ce problème, et limiter le nombre de noeuds à développer, l'utilisation d'algorithmes heuristiques s'impose.

Les algorithmes heuristiques (A*)

L'algorithme précédent développe tout les noeuds exhaustivement, le cheminement dans le graphe s'effectue de manière aveugle. Les méthodes heuristiques se basent sur une connaissance du coût de chaque noeud en fonction de la destination souhaitée. Ceci permet de déterminer les noeuds à développer prioritairement. Ainsi l'heuristique permet de limiter le nombre de noeuds à développer.

On attribue à chaque noeud (n), le coût du chemin allant du départ au noeud (n), soit $g_{d\to n}$ ce coût. De même $h_{n\to f}$ correspond au coût entre le noeud (n) et la position finale. Le coût de chaque noeud est donc $f(n) = g_{d\to n} + h_{n\to f}$. Il suffit donc de ne développer que les noeuds ayant un coût minimal pour trouver le chemin de coût minimum.

Par contre, il n'est pas toujours possible de connaître la vraie valeur de $h_{n\to f}$. Il faut donc utiliser une valeur estimée. Le plus souvent, on choisit la valeur de la distance de vol entre le noeud (n) et le noeud d'arrivée.

L'utilisation de l'une ou l'autre de ces deux méthodes permet de déterminer plus ou moins vite, le chemin le plus court entre le départ et l'arrivée. Mais par son type de construction le graphe de visibilité génère des chemins qui passent au plus près des obstacles. Malheureusement, ceci est un handicap pour une utilisation avec des robots mobiles qui possédent une faible précision en position.

Ce problème est résolu en utilisant le diagramme de Voronoi

Diagramme de Voronoi

Le graphe de Voronoi, permet de choisir des chemins qui passent au plus loin des différents obstacles. En contre partie le chemin calculé n'est pas nécessairement le plus court en terme de distance.

Le diagramme de Voronoi s'obtient en prenant tout les points équidistants de deux (ou plus) points des obstacles (voir figure 5.5).

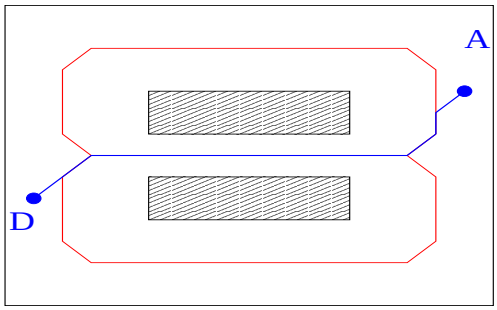


FIG. 5.5 – Diagramme de Voronoi

5.1.2 Planification locale

Dans les méthodes précédentes, le planificateur possède une carte détaillée de l'environnement tout entier. Il n'est pas toujours possible de le connaitre; par exemple si la portée des capteurs est limitée, ou si une partie de l'environnement n'est pas visible au début du déplacement. Dans ce cas il faut utiliser une méthode de planification locale. On peut classer ces méthodes en deux groupes :

- les méthodes du champ de potentiel,
- les méthodes par tests aléatoires.

Méthode du champ de potentiel

Les méthodes utilisant un champ de potentiel ne voient plus le robot à l'intérieur de son espace accessible tout entier, mais il est vu comme une particule sous l'influence d'un champ de potentiel artificiel (noté U). Les variations locales de ce champ de potentiel sont sensées refléter la structure de l'espace accessible.

Les fonctions de potentiel sont généralement vues comme étant la somme de fonctions attractives qui tendent à pousser le robot vers son but, et de fonctions répulsives qui repoussent le robot loin des obstacles.

La planification se fait de façon itérative, à chaque itération la force artificielle $\vec{F}(q) = -\vec{\nabla}.U(q)$ détermine la meilleure direction possible du mouvement à l'instant présent.

Le problème principal de ce type de méthode est que le robot peut être attiré par un minimum local du champ de potentiel, dans ce cas le chemin reste bloqué dans un puit de potentiel. [Cha96] propose une technique permettant d'éviter les minimuns locaux.

Méthode par tests aléatoires

Les méthodes de recherche aléatoire ne sont pas utilisées de manière indépendante, mais elles sont couplées avec une autre méthode locale ou classique. Elles permettent de construire le graphe correspondant à l'espace accessible, en mémorisant les différents essais. [BKL+97] présente une application de cette technique pour un robot manipulateur à 7 degrés de liberté. Une comparaison est faite entre l'utilisation conjointe avec un planificateur local ou un planificateur classique (basé sur une connaissance a priori de l'environnement).

5.2 Robot à roues

Toutes ces méthodes ne peuvent pas s'appliquer telles quelles pour les robots à roues, parce qu'il faut prendre en compte les contraintes non holonomes comme par exemple le roulement sans glissement des roues sur le sol. Ceci peut être réglé par l'utilisation de trajectoires prédéfinies [JC89] qui respectent les contraintes de non holonomie.

Le bipède n'étant pas soumis à cette contrainte nous n'allons pas développer ce point.

5.3 Robot à pattes

Les robots à pattes ne sont pas soumis aux contraintes non holonomes des robots à roues, ils peuvent se déplacer latéralement et ils peuvent tourner sur place. Par contre ils sont soumis à d'autre contraintes :

- ils doivent choisir le lieu où déposer le pied ;
  - le sol, non ponctuel, doit être uniforme ; le pied ne peut pas être posé sur une arête,
  - l'inclinaison du sol doit être en accord avec le coefficient de frottement pied/sol,
- la posture doit être admissible au niveau énergétique. A l'inverse du robot à roues, le robot à pattes doit fournir de l'énergie même lorsqu'il est à l'arrêt pour pouvoir maintenir sa position debout et compenser l'effet de la gravité.

Les robots à pattes sont répartis en deux familles :

- les robots multipattes, qui possèdent au moins 3 pattes. Leur problème est de synchroniser les différentes pattes entre elles.
- les robots bipède, qui possèdent une marge de stabilité bien moindre que les robots multipattes.

5.3.1 Les robots multipattes

Les premières études de démarches s'adaptant à un terrain variable, se basent sur une discrétisation du terrain en cellules. McGhee et Iswandhi [MI79] classent ces cellules en deux groupes GO (les cellules accessibles) et NO-GO (les cellules interdites). Le séquencement des pattes se fait à l'intérieur des cellules de type GO, de façon à maximiser la stabilité du robot. Les cellules sont non-accessibles (de type NO-GO), si elles imposent à la patte du robot de se trouver proche ou hors de sa zone de fonctionnement. Le problème de cette méthode est la quantité de calcul fait pour trouver un chemin à l'intérieur des cellules GO.

En effet, la quantité de calculs augmente de façon exponentielle avec le nombre de degrés de liberté du système contrôlé. Pour réduire ceci, [CK96] propose une méthode d'optimisation ordinale, qui détermine non pas le meilleur chemin, mais un chemin suffisament bon par rapport au critère choisi. Si les ressources le permettent, le calcul peut être prolongé pour obtenir un meilleur résultat. Cette technique est appliquée sur un robot marcheur à 6 pattes.

5.3.2 Les robots bipèdes

Le problème de la génération de chemin pour un robot bipède est de trouver une suite de positions pour les pieds du robot le long desquelles la stabilité du robot est conservée. De plus ces positions doivent satisfaire aux contraintes mécaniques et énergétiques du bipède.

- Au niveau énérgétique :

La dynamique du robot est régie par :

$$\Gamma = M.\ddot{q} + W(q, \dot{q}) + G(q)$$

N'étudiant que la démarche statiquement stable, les vitesses et accélérations du bipède sont négligeables.

$$\text{d'où } \Gamma = G(q)$$

La contrainte énergétique est donc $\Gamma \in [\Gamma_{min}; \Gamma_{max}]$ pour chaque posture.

- Au niveau mécanique :

Il faut que les articulations se trouvent à l'intérieur de leur plage de fonctionnement :

$$q \in [q_{min}; q_{max}]$$

- Au niveau de la stabilité :

En plus de ces deux contraintes, le robot doit toujours être en équilibre statique ; le centre de pression doit se trouver à l'intérieur de la surface de sustentation du bipède.

Si ces trois contraintes sont respectées, le pas est "faisable".

Dans [CE01], Nathalie Cislo et Bernard Espiau recherchent des pas faisables à l'intérieur d'un univers décomposé en cellules hexagonales. Pour réduire la complexité de la recherche, le pied peut prendre une seule position et trois orientations $(-\frac{\pi}{3}; 0; \frac{\pi}{3})$ dans chaque cellule. La recherche dans cet univers se fait à l'aide d'un algorithme $A^\star$ minimisant la distance parcourue et l'énergie dépensée.

Dans [LDS+00], Lorch et Denk planifient le chemin d'un bipède équipé d'un système de vision, en mettant bout à bout en temps réel des primitives de trajectoires calculées hors ligne. Pour juger de la performance des trajectoires générées par aggrégation de primitives précalculées, ils utilisent le critère suivant :

$$\min = \sum_{j=1}^{N} (x_c(q^{(j)}) - x_c^{\star(j)})^2 + (y_c(q^{(j)}) - y_c^{\star(j)})^2$$

qui minimise l'écart entre la position réelle et la position souhaitée du centre de pression.

Le choix des séquences de pas se fait par une méthode locale, en effet tant que le bipède est loin d'un obstacle il avance avec une longueur de pas constante. Si un obstacle survient il modifie la longueur de ses pas. Le calcul se fait sur l'horizon des trois pas futurs.

Avant de développer une nouvelle méthode pour le robot bipède, nous allons d'abord étudier les capacités de déplacement du robot en fonction du type de terrain sur lequel il évolue.

Chapitre 6

Classification des différentes classes de terrain

Avant de développer une méthode de planification propre au robot bipède, il faut d'abord connaître les capacités de franchissement d'un robot bipède en fonction du terrain sur lequel il évolue. Pour cela, le terrain est subdivisé en 3 classes.

- *La classe des sols plats* : les sols plats ne présentent pas a priori de difficultés majeures de franchissement pour un bipède.

- *La classe des plans inclinés* : il faut tenir compte du non glissement des pieds sur le sol et il faut gérer la discontinuité du terrain au raccord entre deux pentes d'inclinaison différente.

- *La classe des marches d'escaliers* : les marches d'escalier peuvent être vues comme une sous classe des sols plat, puisque ce n'est qu'une suite de plans horizontaux à différentes altitudes. Mais le brusque changement d'altitude qu'il y a entre deux marches, nous oblige à les traiter à part.

L'étude de ces différentes classes de terrain sera faite en considérant le robot BIP2000. De plus le robot est toujours considéré en équilibre statique.

6.1 Classe des sols plats

Sur sol plat un bipède n'a pas trop de problème pour se déplacer. La figure 6.1 montre l'espace accessible par le pied gauche du robot BIP2000 pour une position donnée du pied droit. L'espace est accessible dans toutes les directions jusqu'à une distance de 40 cm.

Au maximum, le bipède peut faire des pas de 60 cm, mais en pratique cela n'est pas conforme avec une démarche anthropomorphe. En effet, dans l'hypothèse d'un maintien de l'équilibre statique, lorsque le bipède fait de très grand pas, il doit incliner fortement son tronc pour pouvoir maintenir son centre de pression à l'intérieur de son pied de support.

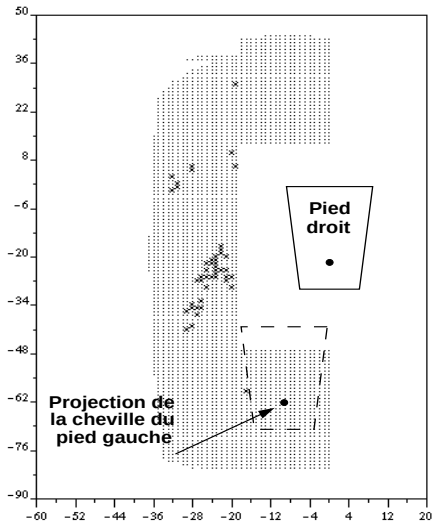


FIG. 6.1 – Espace accessible par le pied libre de BIP2000 [CE01]

6.2 Classe des plans inclinés

On étudie ici l'effet de l'inclinaison du sol sur la longueur maximum des pas que peut réaliser le bipède. Pour chaque test, on étudie le robot avec les deux pieds parallèles et espacés d'une distance comparable à la largeur des hanches. Le sol est supposé suffisamment adhérent pour que le pied ne glisse pas lorsque l'inclinaison augmente.

6.2.1 Étude de l'effet du tangage

L'angle de tangage est défini comme étant l'angle que fait le sol avec l'horizontale dans le plan sagittal[1] (voir figure 6.2)

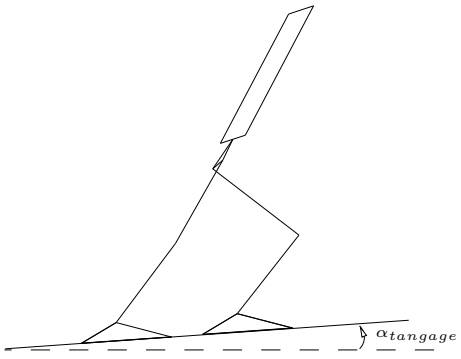


FIG. 6.2 – Angle de tangage

Le robot BIP2000 se comporte bien en tangage, la longueur maximum des pas décroît régulièrement avec l'inclinaison (voir figure 6.3). La limite du robot se trouve au niveau des articulations des chevilles du plan sagittal. Plus l'inclinaison augmente, plus l'orientation des chevilles augmente, jusqu'à arriver en butées.

[1] plan sagittal : voir figure 1.2

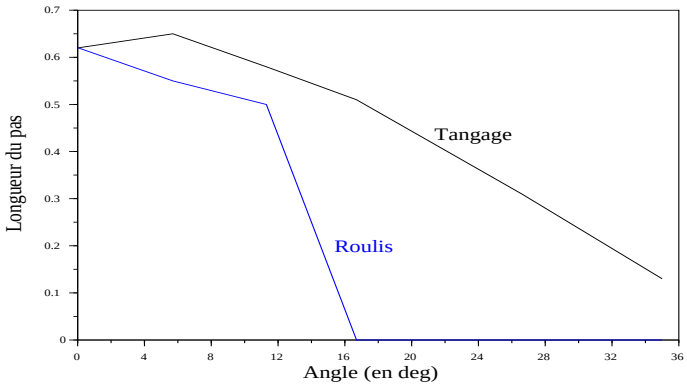


FIG. 6.3 – Evolution de la longueur des pas en fonction de la pente ou du roulis

6.2.2 Étude de l'effet du roulis

L'angle de roulis est définie comme étant l'angle que fait le sol avec l'horizontale dans le plan frontal[2] (voir figure 6.2).

Les mêmes tests sont effectués avec un angle de roulis. Le robot est beaucoup moins performant que dans le cas du tangage. Le robot doit se pencher sur le coté pour pouvoir ramener son centre de pression à l'intérieur de son pied de support. Or l'amplitude des articulations des chevilles du bipède est beaucoup moins importante dans le plan frontal que dans le plan sagittal.

La figure 6.3 montre que les capacités du bipède chutent trés vite lorsque l'angle de roulis augmente. De plus lorsque l'angle de roulis est supérieur à une dizaine de degrés, le robot ne peut même plus maintenir une position debout (pour cet écartement des pieds).

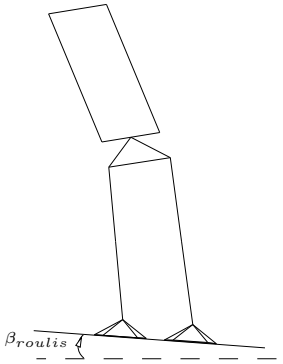


FIG. 6.4 – Angle de roulis

6.3 Classes des marches d'escalier

Pour simplifier la modélisation d'un environemment peu structuré, l'environnement choisi ne comporte pas de marches d'escaliers. Par contre le comportement du bipède lors de la montée et de la descente des marches d'escaliers est étudié dans [Ali99]. Il en résulte que le robot possède de bonnes capacités à monter les escaliers, à l'inverse la descente pose plus de problèmes. En effet lors de la descente, l'être humain passe par une phase où son centre de gravité sort du polygone de sustentation (phase de stabilité dynamique). Or le robot est étudié pour l'instant en équilibre statique. Ceci implique donc une extension excessive du torse, d'où une limitation de sa capacité à descendre des escaliers (voir figure 6.5).

[2]plan frontal : voir figure 1.2

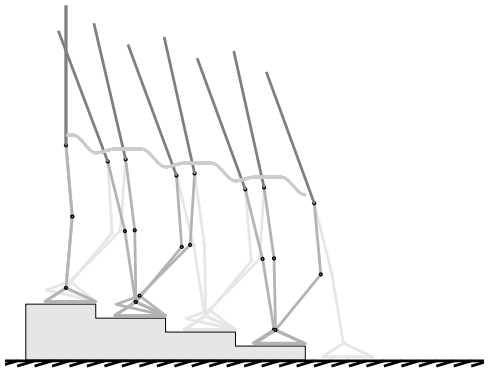


FIG. 6.5 – Descente d'escaliers : tronc en extension arrière [Ali99]

De ces tests il résulte que le bipède a des difficultés pour gravir des pentes de biais. Il faut donc développer un planificateur qui évite de monter (ou descendre) les pentes en travers. Il est préférable d'attaquer les pentes de face.

Chapitre 7

Méthode de planification proposée

Comme nous l'avons vu plus haut, la faible mobilité du robot pose des problèmes lorsqu'on veut discrétiser l'espace et forcer le bipède à poser son pied à un endroit précis. L'idée est donc de ne plus discrétiser le terrain, mais d'utiliser une méthode de planification locale autour d'une position de référence. Cependant, une discrétisation du terrain sera nécessaire pour déterminer la trajectoire de référence.

7.1 Trajectoire de référence

La trajectoire de référence est calculée en utilisant les techniques classiques de robotique mobile. Nous avons vu au chapitre précédent que le bipéde perdait de la mobilié lorsque le sol s'inclinait. Pour maximiser les chances de réussite du planificateur local, la trajectoire de référence est calculée de façon à minimiser les inclinaisons le long du parcours.

7.1.1 Déterminer le chemin le plus court

Le problème est de déterminer un chemin qui va de la posture de départ (indice d) à la posture finale (indice f). Le robot n'étant pas soumis à des contraintes non holonomes, il n'est pas nécessaire de tenir compte des orientations de départ et de fin. Il est toujours possible de faire tourner le bipède sur place.

Un algorithme $A^\star$ permet de calculer un tel chemin, en prenant les heuristiques suivantes :

$$h_{n \to f} = dist(n, f)$$
$$g_{n-1 \to n} = dist(n-1, n)$$

La figure 7.1 montre le chemin ainsi calculé. Le chemin est bien le plus court, mais le problème est qu'il passe par le point le plus haut de notre univers. Il serait préférable, pour satisfaire les contraintes énergiques, de déterminer une trajectoire qui contourne le sommet. Ceci est réalisable en minimisant la valeur de la pente le long du parcours.

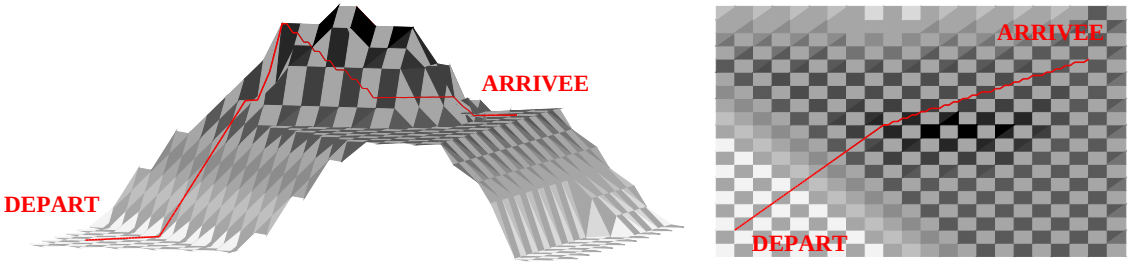


FIG. 7.1 – Chemin de référence, le plus court

7.1.2 Déterminer le chemin le plus court et de plus faible pente

Pour déterminer le chemin le plus court et de plus faible pente, on modifie la valeur des heuristiques utilisées dans l'algorithme $A^\star$.

$$h_{n\to f} = dist(n, f)$$
$$g_{n-1\to n} = dist(n-1, n) + k_i. \mid \alpha_{inclinaison} \mid$$

Ainsi en plus de minimiser la distance parcourue, on pénalise le passage par des cellules de forte inclinaison. Le gain k_i permet de faire le réglage entre des chemins qui ne contournent que les gros obstacles, et des chemins qui font beaucoup de virages pour éviter les obstacles.

Sur la figure 7.2, le chemin obtenu contourne bien le sommet, par contre la montée de la première pente se fait en biais, ceci oblige le robot à se pencher vers la droite, or nous avons vu au chapitre 6 que les capacités du bipède sont faibles lorsqu'il doit se pencher sur le coté.

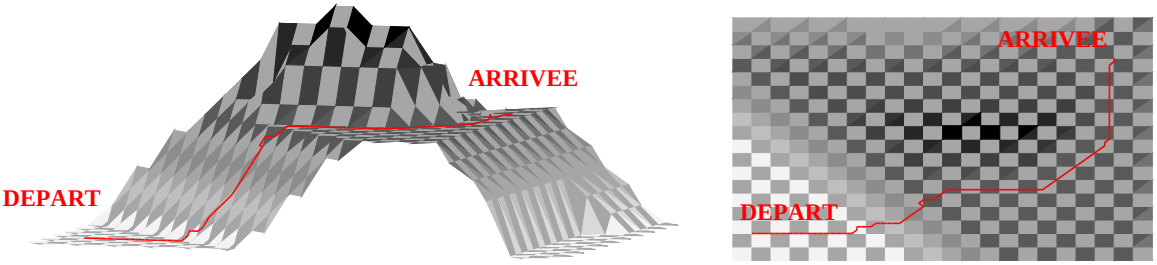


FIG. 7.2 – Chemin de référence, de plus faible pente

Pour éviter les ascensions en biais, il faut décomposer l'inclinaison du terrain en roulis et en tangage.

7.1.3 Déterminer le chemin le plus court, de plus faible pente et sans roulis

Les nouvelles heuristiques utilisées par l'algorithme $A^\star$ deviennent :

$$h_{n\to f} = dist(n, f)$$
$$g_{n-1\to n} = dist(n-1, n) + k_t. \mid \alpha_{tangage} \mid + k_r \mid \beta_{roulis} \mid$$

L'angle d'inclinaison est décomposé en tangage et roulis ainsi on peut choisir de pénaliser plus fortement le passage par des secteurs où le robot se penche sur le coté.

En prenant $k_r > k_i$ on pénalise les montées de pente en biais (le bipède a plus de facilité pour monter les pentes de front).

Un exemple de trajectoire de référence est présent sur la figure 7.3. On constate que lorsque c'est possible la trajectoire contourne les secteurs pentus, et lorsque qu'il n'y a pas d'autre possibilité, la montée se fait perpendiculairement aux lignes de pente.

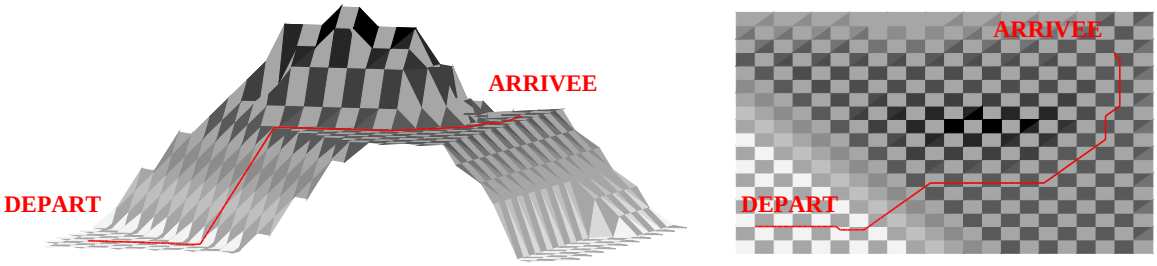


FIG. 7.3 – Chemin de référence, en tenant compte du roulis

7.2 Suivi du chemin de référence

Pour déterminer les pas que doit réaliser le robot, il faut savoir où ce dernier se trouve par rapport au chemin de référence. On définit deux grandeurs :

- l_{os} : écart entre le bipède et le chemin
- θ_{os} : écart entre l'orientation du bipède et l'orientation de la tangente au chemin.

De plus, une position de secours est définie, elle sera utilisée si le planificateur propose plus de n_{max} postures non valides.

La posture de secours correspond à une posture où les deux pieds sont côte à côte espacés de la largeur des hanches.

La stratégie, trés simple dans un premier temps correspond au schéma de la figure 7.4.

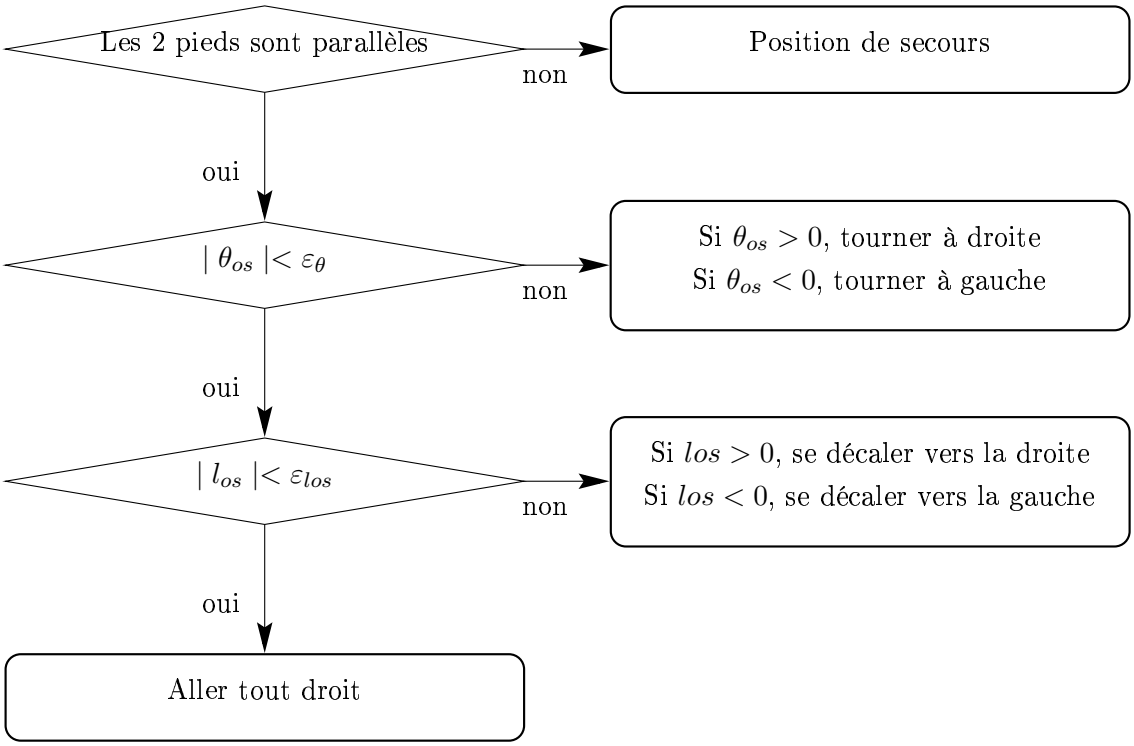


FIG. 7.4 – Stratégie utilisée dans le suivi de chemin

Pour chacune de ces actions, si la posture souhaitée n'est pas réalisable :
– soit parce que la posture n'est pas admissible au niveau énergétique,
– soit parce que le pied ne peut pas être posé à cet endroit.

Alors une nouvelle posture est calculée avec un angle de rotation, ou une longueur de pas plus faible. Si aprés n_{max} essais aucune posture faisable n'est trouvée, on retourne à la posture de secours.

Chapitre 8

Résultats de simulation

Ce chapitre présente les résultats obtenus avec la stratégie développée dans les chapitres précédents. C'est à dire générer un chemin de référence par l'utilisation d'un algorithme $A^{\star}$. Puis suivre ce chemin en contrôlant l'écart latéral et l'écart d'orientation du bipède par rapport au chemin de référence.

Le chemin de référence suivi est celui de la figure 7.3, c'est à dire le chemin qui minimise l'inclinaison et le roulis des pentes traversées.

8.1 Résultats sur une portion de sol horizontal

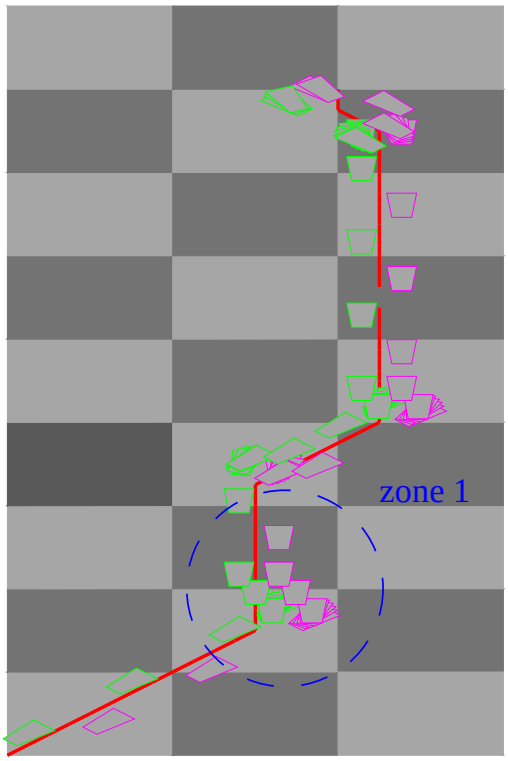


FIG. 8.1 – Suivi de chemin sur une portion de sol plat

La figure 8.1 montre qu'il est possible de déterminer des postures pour suivre un chemin. Le bipède tourne sur place par pas de $\pm 10°$ pour pouvoir suivre l'orientation du chemin. La longueur

du pas de base a été fixée à 40cm.
A l'intérieur de la zone 1 de la figure 8.1, le robot s'est écarté de sa trajectoire de référence en sortie de virage. Le planificateur récupère bien le tracé en faisant trois petits pas de coté.

8.2 Résultats lors du franchissement d'une pente

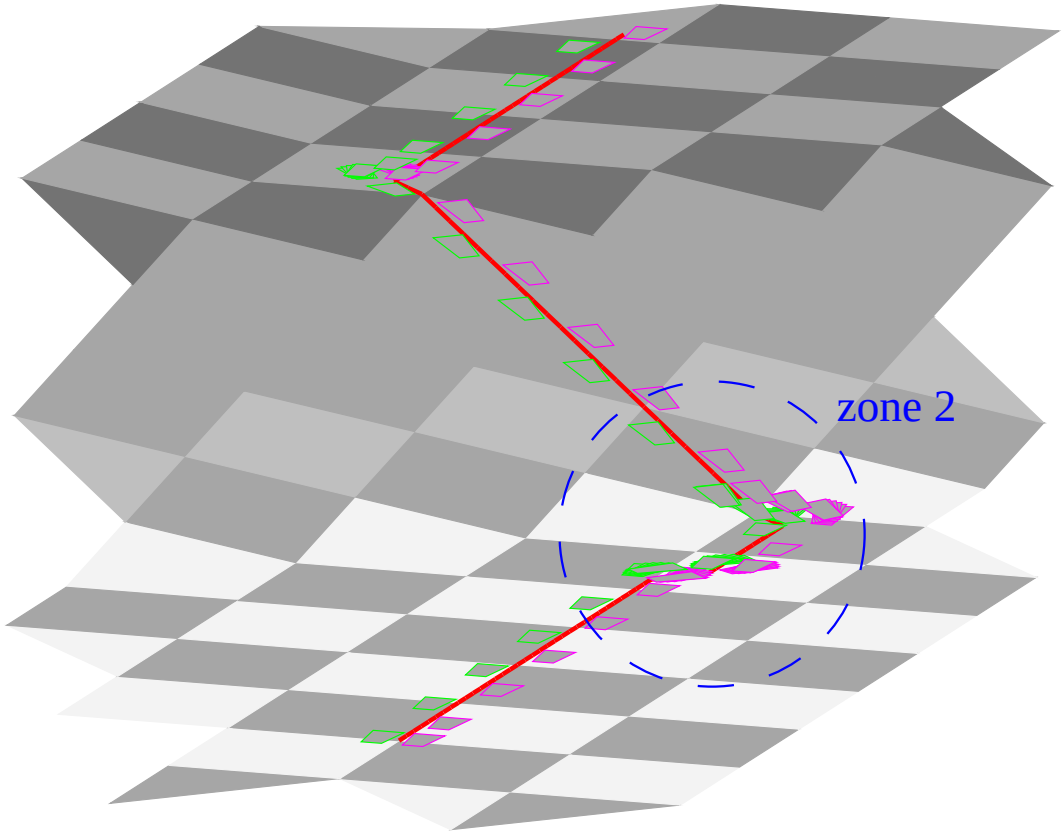


FIG. 8.2 – Suivi de chemin lors de la monté

Sur la figure 8.1, le bipède franchit une pente de l'ordre de 20°. On remarquera à l'intérieur de la zone 2 que le chemin de référence, généré par l'algorithme $A^{\star}$, fait un petit crochet à droite pour pouvoir orienter le bipède dans l'axe de la montée (ceci afin d'éviter une montée en travers, comme cela était le cas en utilisant le chemin de la figure 7.2).

Chapitre 9

Conclusion

Les objectifs de ce stage étaient de déterminer une méthode de génération automatique de trajectoires pour un robot bipède, et de mettre en oeuvre une technique de planification de chemin pour le robot dans un environnement non structuré.

9.1 Contributions

La première partie traite de la génération automatique de trajectoires entre deux postures statiquement stables pour le robot bipède. Un état de l'art a tout d'abord rappelé les différentes techniques de génération de trajectoires propres aux robots manipulateurs, et les différences entre une étude globale ou locale du problème.

Ensuite nous avons adapté la technique du gradient projeté au robot bipède, en mettant en évidence les problèmes liés à ce système : la gestion de la position du centre de masse, la gestion des contacts avec l'environnement et la gestion de la priorité entre les fonctions de tâches. Quelques essais en simulation montrent la faisabilité de cette approche.

La deuxième partie du stage se concentre sur le développement d'un planificateur de chemin en environnement non structuré pour le bipède. En raison du caractère trés récent des travaux sur ce sujet, l'étude bibliographique ne porte que sur le thème général de la planification pour les robots mobiles.

Nous avons mis en oeuvre une technique de planification basée sur le suivi d'une trajectoire de référence. Cette trajectoire de référence est calculée en fonction des données topographiques du terrain de façon à maximiser les chances de réussite du suivi de chemin. Cette approche fonctionne pour le robot BIP2000 dans l'univers peu strucuré mis en place dans cette étude (assemblage de plans horizontaux ou inclinés)

9.2 Perspectives

Pour la partie génération de trajectoire, il serait intéressant d'intégrer dans la méthode, des fonctions de tâches qui permettent de mieux prendre en compte l'impact entre le pied et le sol lorsque l'on veut utiliser cette méthode pour générer des trajectoires de marche. Il serait bon aussi de déterminer une valeur approchée de la matrice d'inertie H du bipède afin de réduire la quantité de calculs faits lors de la génération des trajectoires.

En ce qui concerne la partie relative au planificateur, une première extension serait d'introduire des marches d'escaliers dans l'univers proposé au bipède pour voir le comportement du planificateur devant de tels obstacles. Il est toujours possible de mettre au point des stratégies plus complexes

qui permettraient d'éviter certains blocages du suivi du chemin lorsque le terrain devient trop chaotique.

Bibliographie

[AA00] C. Azevedo et N. Andreff. – *Etude expérimentale des premières démarches du robot.* – Rapport de Recherche n4017, BIP2000, INRIA, 2000.

[Ali99] B. El Ali. – *Contribution à la commande du centre de masse d'un robot bipède.* – Thèse de doctorat, Institut National Polytechnique de Grenoble, December 1999.

[BIP] `http ://www.inrialpes.fr/bip`. – Page web du projet BIP.

[BKL+97] J. Barraquand, L. Kavraki, J. Latombe, T. Li et P. Raghavan. – A random sampling scheme for path planning. *The International Journal of Robotics Research*, vol. 16, n6, December 1997, pp. 759–774.

[CB97] E.S. Conkur et R. Buckingham. – Clarifying the definition of redundancy as used in robotics. *Robotica*, vol. 15, n5, 1997, pp. 583–586.

[CD95] T. Chang et R. Dubey. – A weighted least-norm solution based scheme for avoiding joints limits for redundant manipulators. *IEEE Trans. on Robotics and Automation*, vol. 11, n2, April 1995, pp. 286–292.

[CE01] N. Cislo et B. Espiau. – Path-planning for biped locomotion in a 3d partially structured environment. *In : Proc. of the 32nd International Symposium on Robotics*, pp. 1539–1544. – Seoul, April 2001.

[Cha96] Hsuan Chang. – A new technique to handle local minimum for imperfect potential field based motion planning. *In : Proc of the Int. Conf. on Robotics and Automation*, pp. 108–112. – Minneapolis, April 1996.

[CK96] C.H. Chen et V. Kumar. – Motion planning of walking robots in environment with uncertainty. *In : Proc of the Int. Conf. on Robotics and Automation*, pp. 3277–3282. – Minneapolis, April 1996.

[CM00] F. Chaumette et E. Marchand. – A new redundancy-based iterative scheme for avoiding joint limits, application to visual servoing. *In : Proc. of the 2000 IEEE Int. Conf. on Robotics and Automation*, pp. 1720–1725. – San Francisco, CA, April 2000.

[CP94] S. Cameron et P. Probert. – *Advanced guided vehicles, aspects of the Oxford AGV Project.* – World Scientific, 1994.

[CS97] F.T. Cheng et M.S. Shih. – Multiple-goal priority considerations of redundant manipulators. *Robotica*, vol. 15, n6, 1997, pp. 675–691.

[EB98] B. Espiau et R. Boulic. – *On the computation and control of the mass center of articulated chains.* – Rapport de Recherche n3479, BIP2000, INRIA, 1998.

[ES00] B. Espiau et P. Sardain. – The anthropomorphic biped robot bip. *In : Proceedings of the 2000 IEEE Conference on Robotics and Automation*, pp. 3996–4001.

[GL91] S. Gupta et J.Y.S. Luh. – Closed loop control of manipulators with redundant joints using the hamilton-jacobi-bellman equation. *In : Proc of the 1991 IEEE Int. Conf. on Robotics and Automation*, pp. 472–477. – Sacramento, CA, April 1991.

[GTJ95] K. Gotlih, I. Troch et K. Jezernik. – Global optimal control of redundant robot. *Robotica*, vol. 14, n2, 1995, pp. 131–140.

[JC89] P. Jacobs et J. Canny. – Planning smooth paths for mobile robots. *In : Proceedings : 1989 IEEE International Conference on Robotics and Automation.* pp. 2–7. – Scottsdale, Arizona, 1989.

[KW88] K. Kazerounian et Z. Wang. – Global versus local optimization in redundancy resolution of robotic manipulators. *The Int. Journal of Rob. Research*, vol. 7, n5, 1988, pp. 3–12.

[Lat91] J-C. Latombe. – *Robot Motion Planning.* – Kluwer Academic Publishers, 1991.

[LDS+00] O. Lorch, J. Denk, J. Fernandez Seara, M. Buss et G. Schmidt. – Co-ordination of perception and locomotion planning for goal-oriented walking. *In : CLAWAR*, pp. 183–192. – Spain, October 2000.

[MI79] R. McGhee et G. Iswandhi. – Adaptive locomotion of a multilegged robot over rough terrain. *In : 1979 IEEE Trans. Systems Man and Cybernetics*, pp. 176–182.

[NH87] Y. Nakamura et H. Hanafusa. – Optimal redundancy control of robot manipulators. *Journal of Robotics Research*, vol. 6, n1, Spring 1987, pp. 32–42.

[NHY87] Y. Nakamura, H. Hanafusa et T. Yoshikawa. – Task-priority based redundancy control of robot manipulators. *The Int. Journal of Robotics Reseach*, vol. 6, n2, 1987, pp. 3–15.

[NS94] D.N. Nenchev et Z.M. Sotirov. – Dynamic task-priority allocation for kinematically redundant robotic mechanisms. *In : Proc of the 1994 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 518–524.

[SBE91] C. Samson, M.Le Borgne et B. Espiau. – *Robot Control : the Task Function Approach.* – Clarendon Press, 1991.

[SS00] L. Sciavicco et B. Siciliano. – *Modelling and control of robot manipulators.* – Springer, 2000.

[Wie00] Pierre-Brice Wieber. – *Modĺisation et commande d'un robot marcheur anthropomorphe.* – Thèse de doctorat, Ecole des Mines de Paris, Décembre 2000.

Résumé

Les robots bipèdes présentent de par leur structure complexe, une forte instabilité, et des contraintes dynamiques et cinématiques qui restreignent leurs mouvements.

Cette étude, en deux parties, porte sur la génération automatique de trajectoires (au niveau articulaire) entre deux postures stables pour un robot bipède. Ceci est fait en adaptant la méthode du gradient projeté au robot bipède.

La deuxième partie de l'étude porte sur la planification de chemin en environnement non structuré. La planification est réalisée en faisant le suivi d'un chemin déterminé à l'avance en fonction des données topographiques du terrain.

Ces deux études sont appliquées au robot BIP, un robot anthropomorphe comportant 15 articulations actionnées.

MOTS-CLÉS : Robots Marcheurs, Méthode locale, Gradient Projeté, Fonction de Tâche, Planification, Suivi de Chemin, Univers Non Structuré