MASTER EEAS

Spécialité : Systèmes Automatiques, Informatiques et Décisionnels

Parcours : Systèmes Réseaux et Informatiques Critiques

Année Universitaire 2005-2006

PLANIFICATION DE LA LOCOMOTION HUMANOÏDE

POIRIER Mathieu

Groupe :	GEPETTO
Responsable de stage :	Jean-Paul Laumond
Adresse de l'organisme :	LAAS - CNRS 7 avenue du Colonel Roche 31077 TOULOUSE Cedex 4 FRANCE

Informations Pratiques

Lieu du stage

LAAS-CNRS de Toulouse
Grande salle de robotique Gérard Bauzil
7, av du Colonel Roche
31077 Toulouse

Maitre de stage

Jean-Paul Laumond

Coordonnées

LAAS-CNRS
7, av du Colonel Roche
31077 Toulouse
France

Jean-Paul Laumond (Tuteur)
Directeur de recherche 1ère classe
Bureau B90
Tel : 05 61 33 69 46
jpl@laas.fr

Mathieu Poirier (Stagiaire)
poirier_mathieu@yahoo.fr
Tel :06 15 52 00 62
mpoirier@laas.fr

Résumé

Planification de la locomotion pour un humanoïde. En robotique, la planification de robot mobile a connu un bel essor depuis les dix dernières années. La planification humanoïde n'en est qu'à ces premiers pas. L'adaptation de certaines techniques issues de la robotique mobile à roues, et notamment les outils de planification pour les systèmes non-holonome, nous permettent de réaliser ces premières étapes.

De plus, de nombreuses recherches et résultats ont aussi été présentés sur les avancés en matière de générateur de marche pour humanoïde. De nombreux critères existent pour réaliser la locomotion humanoïde, et l'intégration de celui-ci à la planification, ainsi qu'à la validation dynamique des chemins produits est indispensable.

Ce rapport présente donc les adaptations que j'ai apporté à la méthode locale utilisant la platitude différentielle, issue de la planification de robot mobile à roues, ainsi que son intégration au sein d'une nouvelle plate-forme de planification humanoïde : HPP, développée au LAAS. Ensuite je me suis intéresser à l'intégration d'un générateur de marche développé par le JRL-Japon, pour la plate-forme humanoïde HRP2.14. Et enfin, je présente quelques tests réalisés en simulation mais aussi sur la plate-forme physique.

⋆ ⋆
⋆

Abstract

locomotion planning for humanoïd robot. In Robotics, the planification for mobile robot with wheels have made a lot of progress in the past ten years. The humanoïd planification is just making its first steps. The conversion of some techniques outcome from mobile robot planning, as the tools used for non-holonomic system, allow us to realize those first steps.

Plus, many researchs and results have been relating to the progress of pattern generator for humanoïd robot. Many measures are known to generate humanoïd locomotion, so its integration to the planification and the dynamic validation from the trajectory created, is essential.

So, this report presents the modifications that I provide to the steering method using flat interpolation, outcome from the mobile robot with wheels planification. Then, the work about its integration into the new humanoïd planification software : HPP, develop in LAAS. Next, my work about the integration of JRL-Japan's pattern generator for HRP2.14 hardware. and finaly, I present some of the tests I made in simulation but also on the hardware system.

Remerciements

Je tiens à remercier mon responsable de stage M. Jean-Paul Laumond ainsi que tous les membres de l'équipe GEPETTO pour leur disponibilité, leur conseil et leur sympathie durant toute la durée de ce stage.

Merci à l'équipe Robots_Admin pour leur travail permanent sur les robots et leurs systèmes.

Je remercie également tous les doctorants et stagiaires avec qui j'ai travaillé.

Merci enfin au laboratoire du LAAS-CNRS pour m'avoir accueilli.

Table des matières

Chapitre 1

Introduction

1.1 Problématique Générale

1.1.1 La robotique

L'objectif d'une partie de la recherche en robotique est de réaliser des machines autonomes, capables de travailler dans des environnements difficiles et souvent inconnus. De telles machines doivent posséder des capacités de perception, de raisonnement, de déplacement et d'action sur le monde qui les entoure.

Notre travail s'inscrit dans le domaine de la planification de mouvement dont le but est de calculer des déplacements pour des robots. Ce travail appliqué depuis plusieurs décennies sur les robots mobiles à roues tentes aujourd'hui d'être réalisé sur des robots humanoïdes.

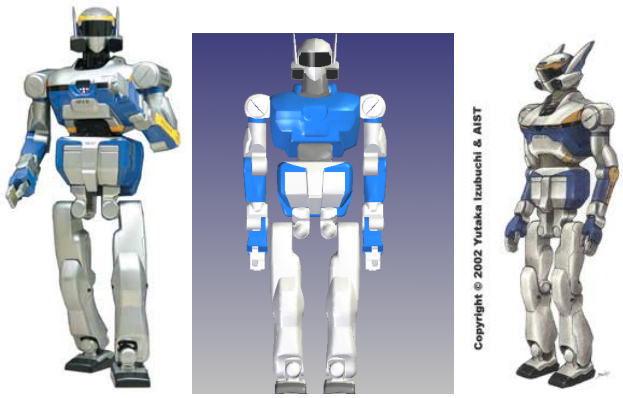


FIG. 1.1 – Plate-forme humanoïde HRP-2.14 Kawada Industrie

1.1.2 La planification de mouvement

La problème de la planification de mouvement consiste à calculer automatiquement un chemin entre deux configurations d'un robot dans un environnement statique connu.

La position d'un robot dans son espace de travail, appelée configuration, peut-être représentée par un vecteur de paramètres $(q_i, ..., q_j)$. Par exemple, la configuration d'une voiture (voir fig. 1.1) peut être représentée par (x, y, θ, Φ), x et y étant les coordonnées du milieu de l'essieu arrière, θ l'orientation de l'essieu arrière et Φ l'angle des roues avant. Pour un humanoïde nous pouvons faire une première simplification en ramenant ce vecteur à (x, y, θ), x et y étant les coordonnées du bassin et θ l'orientation du bassin.

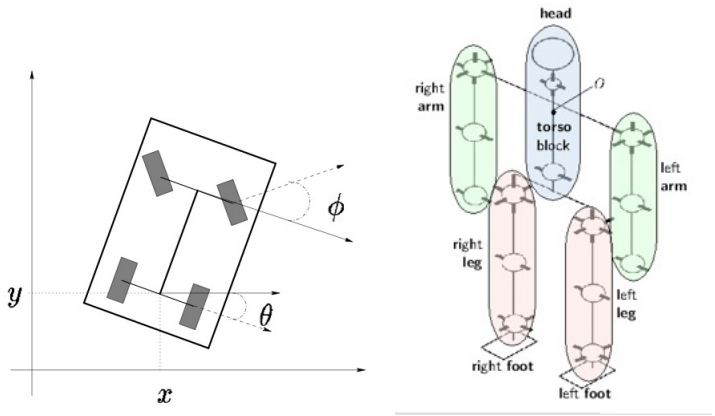


FIG. 1.2 – Configuration d'une voiture et d'un humanoïde

L'ensemble de ces positions est appelé espace des configurations *CS*. Certaines parties de l'espace de travail du robot sont occupées par des obstacles. On appelle espace des configurations libres CS_{libre} l'ensemble des configurations pour lesquelles l'intersection entre le robot et les obstacles est vide.

Contraintes non holonomes. En plus des contraintes physiques imposant au système de rester dans CS_{libre}, des contraintes non-holonomes s'appliquent aux vitesses du système. L'espace des vitesses accessibles à une dimension inférieure à celle de l'espace des configurations. Par exemple un robot mobile à roues est soumis à des contraintes de roulement sans glissement. La configuration d'un robot a trois paramètres (x, y, θ) alors que sa vitesse n'en a que deux : une vitesse de rotation autour de son centre, et une vitesse linéaire dans la direction de ses roues (Fig. 1.3). La planification de mouvement doit donc trouver un chemin dans CS_{libre} en respectant les contraintes non-holonomes.

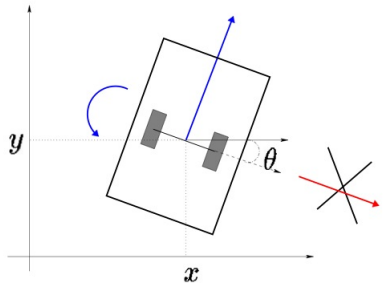


FIG. 1.3 – Contrainte holonomes sur une voiture

1.1.3 Les algorithmes de planification probabiliste

Plusieurs démarches ont été proposées pour contourner la complexité inhérente aux problèmes de planification de mouvement. Parmi celles-ci, depuis une dizaines d'année, des approches probabilistes ont permis d'envisager la résolution de problèmes mettants en jeu un grand nombre de variables articulaires.

Méthodes de type *Probabilistic RoadMap (PRM) [1]* L'algorithme PRM, consiste à tirer aléatoirement une configuration uniformément dans l'espace des configurations et à la stocker sous forme de graphe si elle est dans l'espace libre.

Une fois que l'on dispose du graphe, il faut relier la configuration initiale à la configuration finale, et déterminer si elles appartiennent à une même composante connexe. Si c'est le cas, une trajectoire répondant au problème posé pourra être construite par concaténation **de chemins locaux**.

Vient ensuite une phase d'optimisation/lissage, qui, selon un critère de coût établi, permet de tendre vers un meilleur chemin répondant au problème posé (Fig. 1.4)

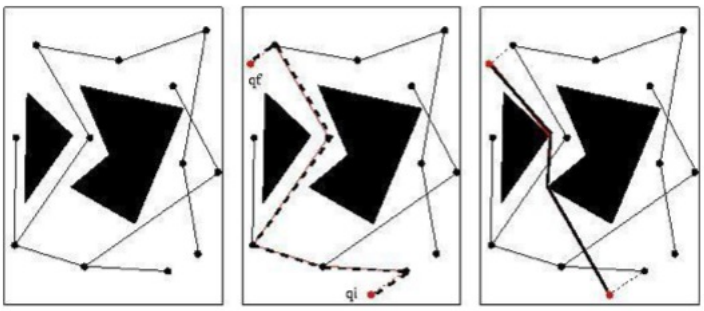


FIG. 1.4 – Le graphe - La trajectoire - L'optimisation

1.2 Spécification de la planification humanoïde

La plupart des méthodes décrites ci-dessus ont été largement appliquées dans le domaine de la robotique mobile à roues, mais il n'existe encore aucune application pour les humanoïdes.

La planification humanoïde est un peu plus complexe que la planification de mouvement pour des robots mobiles à roues, dans le sens où pour être sûr de la trajectoire réalisée par le robot nous devons prendre en compte la dynamique du robot. La planification des robots mobiles à roues n'utilise que la cinématique.

Pour une trajectoire de base donnée, un humanoïde exécute celle-ci de manière différente en fonction de sa vitesse ou du poids qu'il transporte. Afin de garder son équilibre, un humanoïde doit plus ou moins compenser avec son corps. Ainsi il déforme sa trajectoire, au risque de rentrer en collision avec un obstacle. Nous devons donc valider chaque trajectoire en tenant compte de la dynamique du robot pour éviter les mauvaises surprises. (Fig. 1.5)

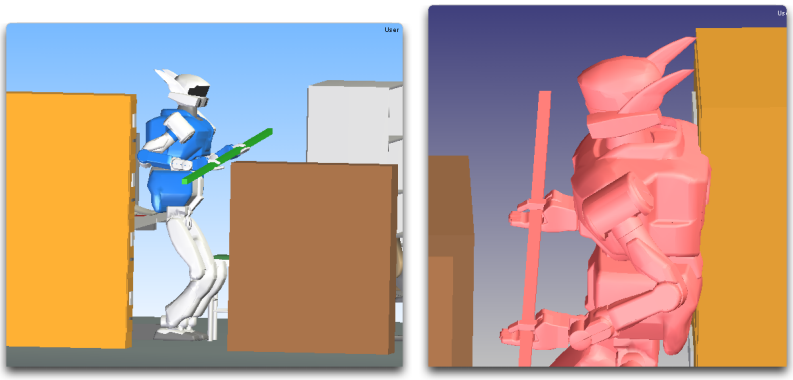


FIG. 1.5 – Mouvement planifié pour un humanoïde, avec cinématique et puis avec dynamique

Pour simuler la dynamique du robot sur des trajectoires, nous utilisons un **PatternGenerator** [3]. Le PatternGenerator (lire chapitre 4) est un programme qui prend, en entrée, une pile d'empreintes devant être exécutée par l'humanoïde et qui renvoie, en sortie, les valeurs de tous les angles de toutes les articulations du robot échantillonnées toutes les 5 ms, soit, pour HRP-2.14, 40 degrés de liberté ou articulations[1] .

Nous divisons la planification humanoïde en plusieurs étapes :

[1]Remarque : les périodes d'échantillonnages sont fixée par le fonctionnement de la plate-forme HRP-2.14 .

– **étape 1** : Dans un premier temps, nous nous occupons que de la partie basse du robot. Pour cela, nous identifions le modèle du robot à une boite rectangulaire représentant une enveloppe corporelle du robot et ayant pour configuration un simple vecteur (x, y, θ). Ceci nous permet de facilement calculer une trajectoire avec les méthodes connues, à l'aide de KineoWorks2.(Fig. 1.6)

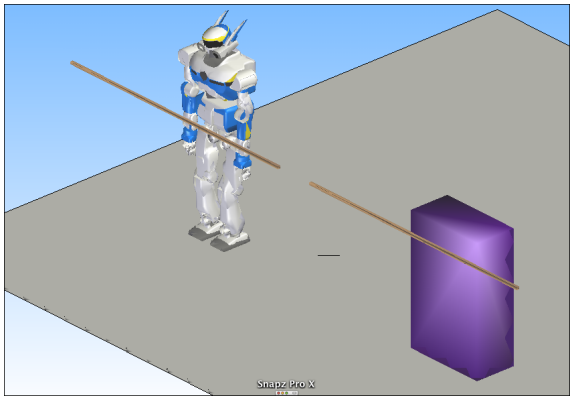


FIG. 1.6 – L'humanoïde et son modèle simplifié (boite englobante) pour la planification

– **étape 2** : Une fois la trajectoire trouvée, nous positionnons des empreintes de pas autour de la trajectoire. Ces empreintes servent de repère pour le PatternGenerator qui calcule, en sortie, toutes les valeurs des angles à appliquer au robot toutes les 5 ms.

> **Remarque :** Pendant la deuxième étape, le PatternGenerator calcule les angles de la partie haute du corps selon une heuristique qui compense les effets de couples qui se produisent lorsque l'on ne tient pas compte de la partie haute du corps.

– **étape 3** : Une fois que nous disposons de toutes ces configurations, à 40 degrés de liberté, nous les appliquons, une à une au modèle complet du robot dans le logiciel KineoWorks2, afin de valider la trajectoire. si celle-ci est validée, sans collisions, on passe à l'étape 4 sinon il faut revenir a l'étape 1 pour modifier celle-ci, ou trouver une autre trajectoire réalisable par le robot.
– **étape 4** : Cette étape réalisée par d'autres collègues, permet de prendre en compte la partie supérieur du corps dans la planification. Une fois la partie supérieure ajoutée à la première planification, on rejoue l'ensemble une dernière fois pour valider la trajectoire du robot avec l'ensemble de ces mouvements. Si la trajectoire n'est pas validée à cette étape nous pouvons revenir à l'étape 1 ou 4 (Fig. 1.7).

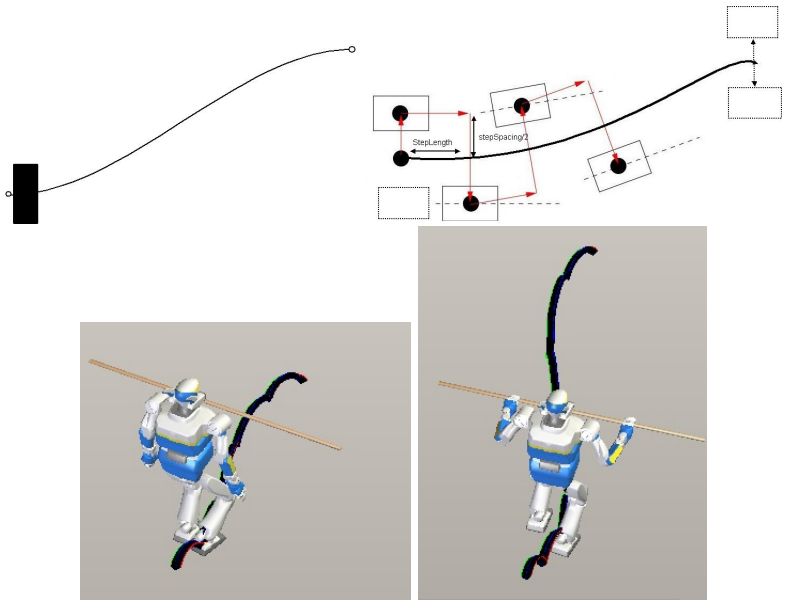


FIG. 1.7 – Processus de planification humanoïde : étape 1 - étape 2 - étape 3 - étape 4

1.3 But du stage

Ma contribution durant ce stage se situe dans les étapes 1 à 3 de la planification humanoïde. Dans un premier temps, j'ai récupéré une méthode locale déjà implémentée sur des robots mobiles à roues de l'équipe RIA. Puis j'ai travaillé sur l'adaptation de cette méthode à la planification humanoïde, afin de l'intégrer au schéma. Puis je me suis intéressé à l'intégration d'un PatternGenerator et enfin à la validation de la trajectoire humanoïde, sans sa partie supérieure. *L'ensemble de mon travail prend place dans la construction d'une plate-forme de développement pour la planification humanoïde basée autour du logiciel KineoWorks2 qui fonctionnera en parallèle du logiciel propriétaire fournit par General Robotix : OpenHRP, pour l'utilisation de la plate-forme physique HRP-2.14.*

Je vais donc dans un premier temps présenter la méthode locale utilisée pour la planification de la première étape. Ensuite je rentrerai un peu plus dans les détails afin de comprendre la difficulté d'utilisation d'une telle méthode. Puis j'expliquerai comment nous avons commencé à intégrer tous ces outils (méthode locale et PatternGenerator) afin de développer une plate-forme de planification humanoïde : **HPP Humanoïde Path Planner** . Et enfin je parlerai des quelques expérimentations que j'ai pu réalisées, tout d'abord en simulation grâce au logiciel KineoWorks2 et OpenHRP et ensuite sur la plate-forme physique de HRP-2.14 .

Chapitre 2

Une méthode locale

2.1 Introduction

Qu'est ce qu'une méthode locale ? Une méthode locale, aussi appelée **steering method** est une méthode ou fonction utilisée en planification robotique pour relier deux configurations données. Il n'existe pas de méthode générale permettant de calculer la commande à donner à un système non linéaire pour l'amener d'une configuration à une autre. Cela n'a rien d'étonnant dans la mesure où le problème inverse consistant à calculer la position finale d'un système soumis à une entrée connue et partant d'un point donné n'a pas non plus de solution générale. Pour certains types de systèmes (en l'occurrence pour les robots de type voiture) des solutions ont cependant été proposées.

Dans la première étape de notre planificateur le robot humanoïde est approximé par une boite, une boite englobante de la partie basse du robot qui peut s'apparenter à un système proche d'une voiture. Son vecteur de configuration est donc ramené à un vecteur (x, y, θ), x et y étant les coordonnées du bassin et θ l'orientation du bassin. Nous réduisons ainsi le système à 3 degrés de liberté et nous pouvons appliquer des méthodes locales déjà utilisées pour la planification de robot à roues de type voiture.

Pourquoi une méthode locale basée sur la platitude différentielle ? Pour que la locomotion humanoïde paraisse le plus réaliste possible et pour des questions de physique : accélération trop importante, les vitesses appliquées au robot doivent rester continues. Nous aurions pu utiliser une méthode locale linéaire mais un problème se pose lorsqu'il faut joindre 3 configurations (Fig. 2.1) : une discontinuité apparaît sur la vitesse. De même pour la locomotion, le robot tournerait sur place avant de joindre la 3ième configuration. La méthode, basée sur la platitude différentielle, que nous allons décrire par la suite permet d'éviter se genre de désagrément, peu importe le nombre de configurations à joindre.

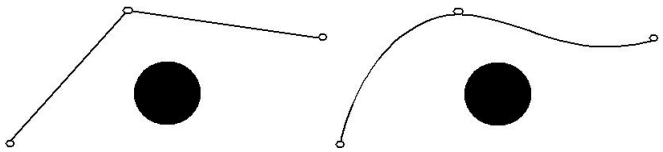


FIG. 2.1 – Discontinuité de la méthodes linéaire, continuité de platitude différentielle : chemin reliant 3 configurations avec un obstacle

2.2 La Platitude

La méthode que nous avons utilisée repose sur la propriété de platitude différentielle du système robot-remorque ou de type voiture [5].

Définition : *Un système $q \in \mathbf{R}^n$ est dit plat s'il existe une fonction $y = h(q)$, appelée sortie plate, à valeurs dans $\mathbf{R}^m$ et une fonction A telle que*

$$q = A(y_1, ..., y_1^{(\alpha_1)}, ..., y_m, ..., y_m^{(\alpha_m)})$$

avec { $y_1, .., y_m$ } différentiellement indépendants.

La propriété de platitude est telle que tout chemin de la sortie plate dans son espace, correspond à un chemin faisable du système dans son espace des configurations. La donnée de la valeur de la sortie plate et de ses dérivées permet de calculer l'état du système sans intégrer aucune équation différentielle.

Illustrons cette propriété dans le cas du robot-remorque à attache centrée (Fig. 2.2). Soit $y = (y_1, y_2)$ le milieu de l'axe des roues de la remorque. Si $y(s)$ désigne la courbe décrite par y, l'orientation du vecteur tangent à $y(s)$ donne l'orientation de la remorque :

$$\theta = larctan\frac{\dot{y_2}}{\dot{y_1}}$$

En dérivant une fois de plus cette expression, on peut montrer :

$$\Phi = \arctan(l.\kappa)$$

où κ est la courbure de $y(s)$ et l la longueur de l'attache de la remorque.

On peut donc reconstruire la trajectoire du système à partir de celle du centre de la remorque, y est donc la sortie plate.

Dans le cas de l'attache décentrée, nous avons un résultat analogue mais la sortie plate n'est située en aucun point particulier de la remorque ou du robot, et sa position par rapport au système varie suivant l'angle que la remorque fait avec le robot *(donc tout le long du chemin en général)*.

Pour un système plat, trouver un chemin entre deux configurations en respectant les contraintes non holonomes, revient donc à trouver une courbe avec des contraintes sur les dérivées aux extrémités.

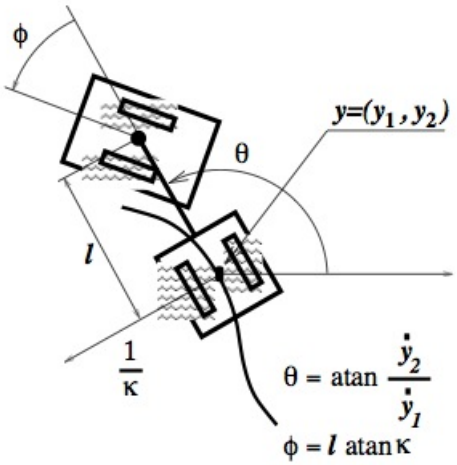


FIG. 2.2 – sortie plate

2.3 La méthode locale

Dans notre cas nous voulons construire une courbe reliant deux points ayant comme contraintes aux extrémités la tangente à la courbe et la courbure.

2.3.1 Courbes canoniques

Notre méthode repose sur la notion de courbe canonique associée à une configuration. La courbe canonique est l'unique courbe passant par cette configuration et gardant la courbure constante. Cette courbe correspond à celle que l'on obtiendrait si on soudait la remorque au robot et si l'on faisait avancer le robot. Il s'agit d'un cercle, ou d'une droite si la courbure est nulle (Fig.2.3).

2.3.2 Combinaison convexe des courbes canoniques

On utilise alors une fonction de pondération α pour effectuer une combinaison convexe de ces deux courbes canoniques (Fig. 2.4). α définie et croissante de [0,1] dans [0,1] vérifiant :

(2.1)

$$\begin{array}{lll} \alpha(0) = 0 & \alpha'(0) = 0 & \alpha''(0) = 0 \\ \alpha(1) = 1 & \alpha'(1) = 0 & \alpha''(1) = 0 \end{array}$$

Appelons $\gamma_1(u)$ (resp. $\gamma_2(u)$) la courbe canonique passant par la configuration initiale lorsque $u = 0$ (resp. par la configuration finale quand $u = 1$) (Fig.2.3).

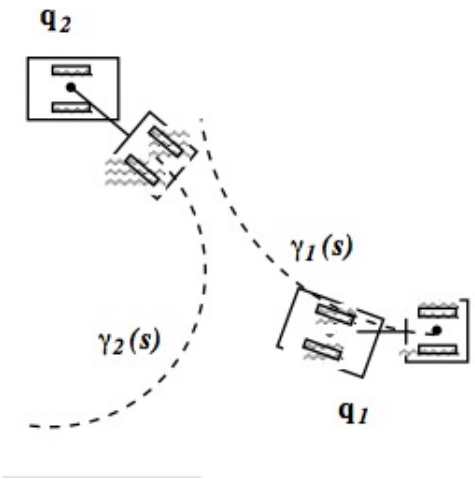


FIG. 2.3 – Courbes canoniques

On définit la courbe suivante (paramètre de 0 à 1) :

$$\gamma(u) = (1 - \alpha(u)).\gamma_1(u) + \alpha(u).\gamma_2(u)$$

On vérifie facilement que :

$$\begin{array}{ll} \gamma(0) = \gamma_1(0) & \gamma(1) = \gamma_2'(1) \\ \gamma'(0) = \gamma_1'(0) & \gamma'(1) = \gamma_2'(1) \\ \gamma''(0) = \gamma_1''(0) & \gamma''(1) = \gamma_2''(1) \end{array}$$

Donc γ a la même tangente et courbure que γ_1 en 0 et que γ_2 en 1, ainsi lorsque la sortie plate parcourt γ entre [0,1], le système exécute un chemin entre q_1 et q_2.

2.3.3 Paramètre des courbes canoniques

Le choix du paramètre, bien qu'arbitraire, est guidé par la volonté que le système respecte la propriété de *commandabilité en espace petit* [1].

[1] Un système est dit localement commandable en espace petit si pour tout voisinage U d'une configuration q, l'ensemble des configurations accessibles par un chemin inclus dans un U est un voisinage de q. [5]

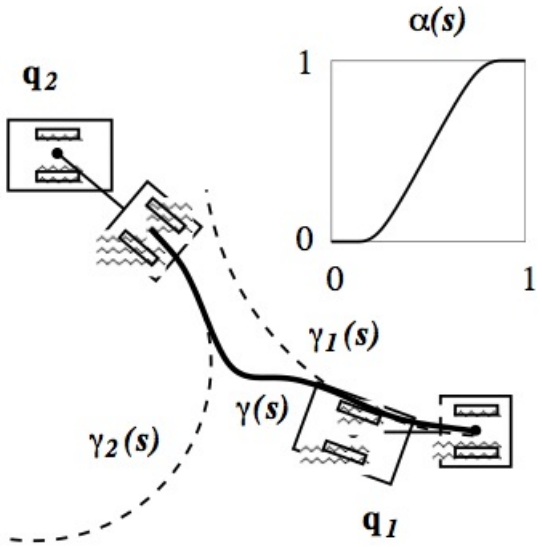


FIG. 2.4 – Interpolation des courbes canoniques

L'idée consiste à rester autant que possible près des courbes canoniques. Par exemple si le point d'arrivée se trouve sur la courbe canonique du point de départ, la trajectoire engendrée sera exactement cette courbe canonique. Pour cela, on appelle v la distance sur la courbe canonique du point de départ au projeté sur cette courbe du point d'arrivée (Fig. 2.5).

Si l'on appelle $\Gamma(q, s)$ la configuration sur la courbe canonique de q d'abscisse curviligne s, nous pouvons alors exprimer nos courbes canoniques par :

$$\gamma_1(u) = \Gamma(q_1, u.v)$$
$$\gamma_2(u) = \Gamma(q_2, (u-1).v)$$

On remarque que si $q_2 \in \Gamma(q_1, s)$ alors $\gamma_1(u) = \gamma_2(u) = \gamma(u)$.

Reste maintenant à trouver une fonction α aussi simple que possible mais vérifiant les conditions citées plus haut.

2.3.4 Choix de α

Naturellement le plus simple est de prendre un polynôme. Nous avons six contraintes (voir équation 2.1), nous devons donc prendre un polynôme de degré cinq : $\alpha(u) =$

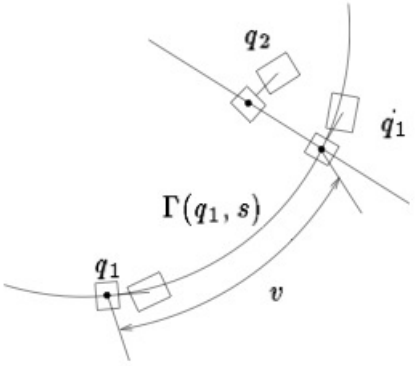


FIG. 2.5 – Calcul de v

$a_0 + a_1.u + a_2.u^2 + a_3.u^3 + a_4.u^4 + a_5.u^5$, puis en résolvant le système (2.1) nous obtenons :

$$\alpha(u) = 10.u^3 - 15.u^4 + 6.u^5$$

Nous avons donc une trajectoire associée à un couple configuration de départ-configuration d'arrivée.

Chapitre 3

Implémentation de la méthode locale

Une partie de mon stage a donc consisté à implémenter les trajectoires locales décrites dans la partie précédente. Lorsqu'on implémente un chemin local, on doit fournir plusieurs fonctions dont deux très importantes

- la fonction **d'interpolation**, elle-même, qui doit être capable de retourner la configuration associée au robot à un endroit donnée de la courbe.
- une second fonction appelée **MaxAbsoluteDerivative** qui retourne les majorants de la valeur absolue des dérivés du vecteur de configuration d'un chemin local entre deux configurations. Les résultats de celle-ci sont ensuite utilisés pour calculer la distance minimale assurant la non-collision du robot avec un obstacle de l'environnement.

Je présente d'abord l'intérêt de ces fonctions puis le choix d'implémentation pour accélérer les calculs de la deuxième fonction.

3.1 La fonction d'interpolation

Je ne vais pas m'étendre sur cette fonction car elle ne fait qu'implémenter les calculs vu dans le chapitre précédent. Celle-ci renvoie à l'utilisateur, pour une position u donnée, comprise entre [0, 1], la courbe étant paramétrée dans le même intervalle, le vecteur de configuration du robot, soit (x, y, θ, κ) (Fig. 3.1), κ la courbure, étant un degré de liberté fictif ajouté au vecteur de configuration afin d'assurer la continuité entre plusieurs chemins locaux.

3.2 Test anti-collision

Ayant un chemin local, nous devons tester s'il est en collision. Cette opération ne peut pas être effectuée analytiquement sur l'ensemble du chemin. La stratégie employée par les logiciels de planification tel KineoWorks2 ou Move3D [6] est de tester la non

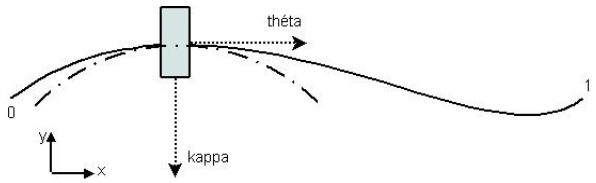


FIG. 3.1 – Exemples de configuration de la boite englobante sur la trajectoire

collision d'un nombre fini de configurations sur le chemin, en s'assurant que l'intervalle entre chaque point est également sans collision.

3.2.1 Description d'un test de collision

Etant donné un chemin du système robotique, on veut savoir s'il est valide, c'est-à-dire si le système suivant ce chemin entrera ou non en collision avec son environnement.

Le test d'anti-collision est basé sur des appels répétés à un détecteur de collision statique (par ex : KCD [8]), selon un pas de discrétisation du chemin non uniforme. La Figure 3.2 décrit ce mécanisme de validation de chemin. Etant donné un chemin local $\gamma(u)$ entre deux configurations q_i et q_f, on place le système dans sa première configuration courante $q_c = \gamma(0)$ et on calcule à quelle distances d_rob du robot se trouvent les obstacles les plus proches. Le planificateur local comporte une fonction appelée **MaxAbsoluteDerivative** fournissant un incrément de paramètre du du chemin assurant que d_rob soient le majorant des distances parcourues par tous les points du robot. Cette fonction nous assure que tout l'intervalle $[u, u+du]$ est donc sans collision. On met le système dans la configuration $\gamma(u + du)$ et on répète la même procédure. De cette manière, on parcourra γ avec un minimum d'appels au détecteur de collision statique, qui engendre toujours beaucoup de calcul.

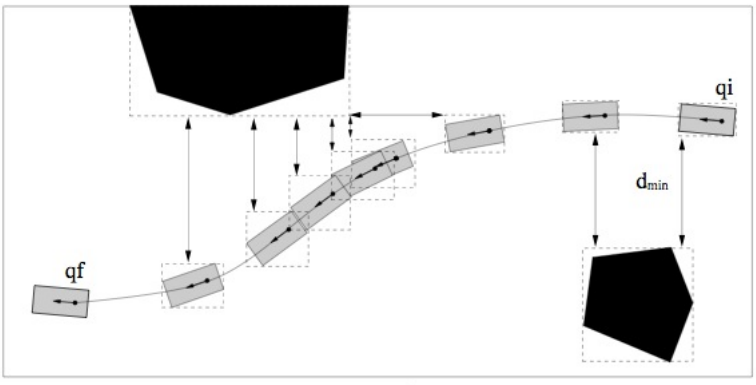


FIG. 3.2 – Validation d'un chemin dans un planificateur tel KineoWorks2

Il est clair que plus le système se rapprochera d'un obstacle lors du parcours de son chemin, plus le pas du sera petit. Si le chemin local comporte une collision, la procédure précédente converge vers la première configuration en collision. Pour éviter cette situation, on remplace d_rob par **max(d_min, d_rob)** ou d_min est une distance choisie par l'utilisateur. Lorsque le système se rapproche de l'obstacle, les incréments sur le chemin local deviennent réguliers et une éventuelle collision peut être détectée (Fig. 3.3).

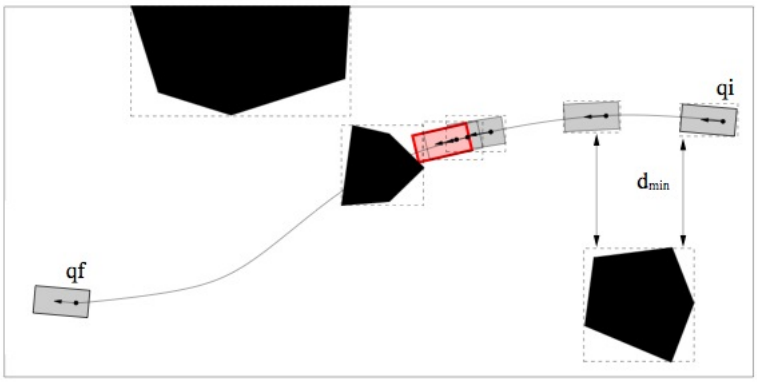


FIG. 3.3 – Validation d'un chemin dans un planificateur tel KinéoWorks2

3.2.2 La fonction MaxAbsoluteDerivative

Comme décrite précédemment, la fonction MaxAbsoluteDerivative calcule, pour une configuration donnée sur un chemin local, un incrément du paramètre du assurant qu'aucun point du robot ne se déplace de plus d'une distance donnée en entrée.

Dans notre cas, il s'agit de calculer les majorations sur le vecteur (x, y, θ, κ) (Fig. 3.1), sachant que κ n'est pas traité car non influant, il nous reste à majorer les paramètres x, y et θ, soit les fonctions :

$$\gamma(u) = (1 - \alpha(u)).\gamma_1(u) + \alpha(u).\gamma_2(u) \quad \text{avec} \quad \gamma(u) = \begin{bmatrix} \gamma_x(u) \\ \gamma_y(u) \end{bmatrix}$$

sachant

$$\gamma_1(u) = \Gamma(q_1, v.u)$$
$$\gamma_2(u) = \Gamma(q_2, v.(u-1))$$

et

$$\theta(u) = \arctan \frac{\dot{y}}{\dot{x}}$$

Calculs de $||\gamma'(u)||$:

$||\Gamma'(u)|| = 1$, pour tout $q \in ||\frac{1}{du}\Gamma(q,u)|| = 1$

Mais pour q fixe, $\gamma'_1(u) = v\frac{d}{du}(\Gamma(q_1, v.u))$ (resp. $\gamma'_2(u)$).

D'où $||\gamma'_1(u)|| = |v|$ et $||\gamma'_2(u)|| = |v|$.

sachant $\gamma'(u) = (1-\alpha(u)).\gamma'_1(u) + \alpha(u)\gamma'_2(u) + \alpha'(u)(\gamma_2(u) - \gamma_1(u))$

Il faut donc résoudre :

$$\boxed{||\gamma'(u)|| \leq |1-\alpha(u)|.|v| + |\alpha(u)||v| + |\alpha'(u)|.|\gamma_2(u) - \gamma_1(u)|}$$

pour trouver les majorants $|x'|$ et $|y'|$.

Calculs de $|\theta'(u)|$:

$\theta' = \frac{y".x' - y'.x"}{x'^2+y'^2}$ avec $\gamma(u) = \begin{bmatrix} \gamma_x(u) \\ \gamma_y(u) \end{bmatrix}$

$\theta' = \frac{\gamma"_y(u).\gamma'_x(u) - \gamma'_y(u).\gamma"_x(u)}{||\gamma'(u)||^2}$ soit $\theta' = \frac{det(\gamma'(u),\gamma''(u))}{||\gamma'(u)||^2}$

Alors $|\theta'| = \frac{|det(\gamma'(u),\gamma''(u))|}{||\gamma'(u)||^2}$

D'où $|\theta'| = \frac{||\gamma'(u)||.||\gamma''(u))||}{||\gamma'(u)||^2}$

Il faut donc résoudre :

$\boxed{|\theta'| = \frac{||\gamma''(u))||}{||\gamma'(u)||}}$ pour trouver le majorant de $|\theta'|$

En résumé :

Pour résoudre ces deux équations, il nous faut donc trouver un majorant de $||\gamma''(u))||$, un majorant et un minorant de $||\gamma'(u)||$

Une majoration globale.

Une première méthode consisterait à assurer la sécurité à 100%. Ceci revient à calculer analytiquement les majorations et minorations des vitesses linéaires et angulaires du robot pour tout le chemin en majorant d'abord les normes des dérivées de la trajectoire de façon uniforme sur [0, 1].

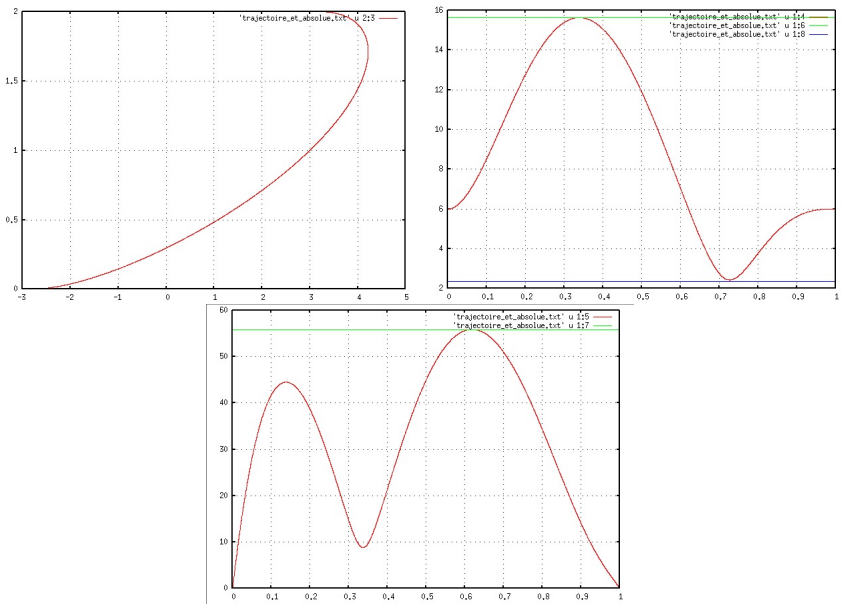


FIG. 3.4 – une trajectoire, la majoration, la minoration de $\gamma'(u)$ et la majoration de $\gamma"(u)$

En fait, cette première méthode est tellement sûre que l'on perd beaucoup de précision lors de l'évitement d'obstacle, ce qui pose problème lorsque l'on veut que le robot passe dans des endroits plus confinés.

Une majoration par intervalle

Nous avons donc opté pour une majoration sur intervalle, car la fonction MaxAbsoluteDerivative n'est pas forcement appelée par le planificateur sur l'intervalle entier de [0, 1]. Il est souvent intéressant de trouver le d_rob sur un bout de la trajectoire.

Pour garder tout de même une certaine performance, en terme de temps de calcul, nous avons mis au point un algorithme récursif, qui permet de trouver très rapidement les majorants et le minorant de $\gamma'(u)$ et $\gamma"(u)$ de manière "graphique" en n'ayant qu'un nombre réduit de calcul à faire. Ceci est ensuite stocké sous forme d'arbre dans le chemin locale, ce qui permet de retrouver très rapidement le minorant ou le majorant d'une partie de la trajectoire sans effectuer de calcul supplémentaire.

La seconde méthode implémentée est donc la même que la précédente pour ce qui est du calcul de du et des vitesses maximales en fonction des normes des dérivées de la trajectoire mais utilise une méthode de maximisation plus fine pour les normes des dérivées de la trajectoire. Pour la dérivée troisième de la trajectoire on utilise la même

méthode, puis pour les autres on utilise une fonction de maximisation à laquelle on peut donner la précision que l'on veut ici $\frac{v}{10}$ ou $\epsilon = 0.01$.

Le principe de cette maximisation est d'utiliser une procédure de dichotomie permettant de calculer un maximum. La formule utilisée pour maximiser une fonction f est (en connaissant M tel que $|f'(u)| < M$) :

$$\frac{f(u_{i+1}) + f(u_i)}{2} - M.\frac{u_{i+1} - u_i}{2} \leq f(u) \leq \frac{f(u_{i+1}) + f(u_i)}{2} + M.\frac{u_{i+1} - u_i}{2}$$

La marge d'erreur étant $min(f_{max} - f(u_0), f_{max} - f(u_{end}))$ (Fig. 3.5).

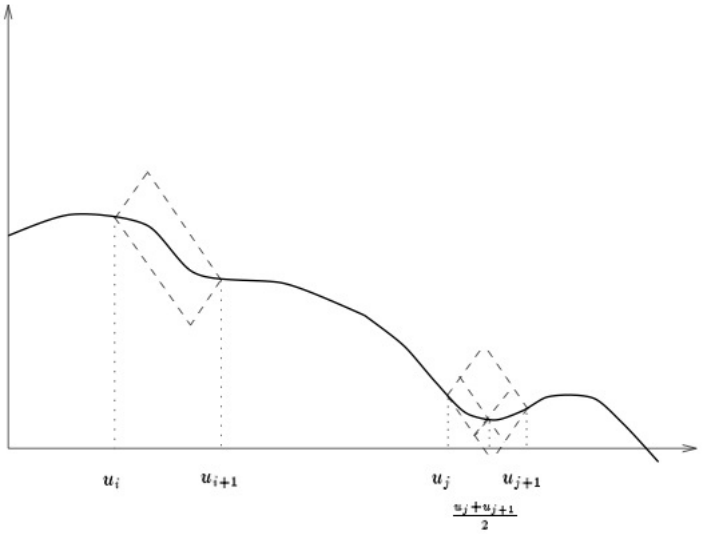


FIG. 3.5 – Détermination du maximum ou minimum par la méthode

La dichotomie consiste à couper l'intervalle [0,1] en plus petits intervalles tant que l'erreur est supérieure à celle désirée ou que l'intervalle minimum ϵ n'est pas atteint.

Si l'on ramène ceci à notre problème, il nous faut donc majorer $\gamma'''(u)$, en déduire le majorant de $\gamma''(u)$ et réappliquer cette même méthode pour en déduire le majorant et le minorant de $\gamma'(u)$ sur l'intervalle donnée (Fig. 3.6).

Soit on trouve $||\gamma'''(u)|| \leq M_3$ pour tout $u \in [0, 1]$

Puis pour tout u_1 et $u_2 \in [0, 1]$ et $u_2 > u_1$

$$\underbrace{||\gamma''(u_1)|| - M_3|u_2 - u_1|}_{droite} \leq ||\gamma''(u_2)|| \leq \underbrace{||\gamma''(u_1)|| + M_3|u_2 - u_1|}_{droite}$$

de même

$$\underbrace{||\gamma'(u_1)|| - M_2|u_2 - u_1|}_{droite} \leq ||\gamma'(u_2)|| \leq \underbrace{||\gamma'(u_1)|| + M_2|u_2 - u_1|}_{droite}$$

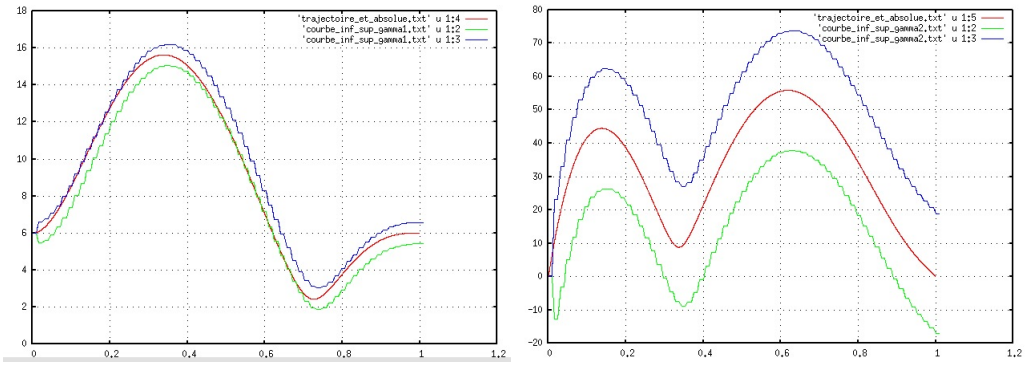


FIG. 3.6 – majorant, minorant par intervalle de $||\gamma'(u)||$ et $||\gamma''(u)||$

Résumé de la majoration de $\gamma^{(3)}(u)$

Pour la démonstration complète voir *annexe A*.

$$\gamma''(u) = (1-\alpha(u)).\gamma_1''(u)+\alpha(u)\gamma_2''(u)+2\alpha'(u)(\gamma_2'(u)-\gamma_1'(u))+\alpha''(u)(\gamma_2(u)-\gamma_1(u))$$

d'où

$$\begin{aligned}\gamma'''(u) = (1-\alpha(u)).\gamma_1'''(u)\\ +\alpha(u).\gamma_2'''(u)\\ +3\alpha'(u).(\gamma_2''(u)-\gamma_1''(u))\\ +3\alpha''(u).(\gamma_2'(u)-\gamma_1'(u))\\ +\alpha'''(u).(\gamma_2(u)-\gamma_1(u))\end{aligned}$$

soit

$$||\gamma'''(u)|| \leq |1-\alpha(u)|.||\gamma_1'''(u)||$$

$$
\begin{aligned}
&+|\alpha(u)|.||\gamma_2'''(u)|| \\
+|3\alpha'(u)|.||\gamma_2''(u) - \gamma_1''(u)|| \\
+|3\alpha''(u)|.||\gamma_2'(u) - \gamma_1'(u)|| \\
+|\alpha'''(u)|.||\gamma_2(u) - \gamma_1(u)||
\end{aligned}
$$

Donc sur [0, 1]

$$
\begin{aligned}
||\gamma'''(u)|| &\leq |\kappa_1^2 v^3| \\
&+|\kappa_1^2 v^3| \\
+3.\frac{45}{8}.v^2.&\Big[|\kappa_2|.(|\tau_1 - \overline{\tau_2}| + |\kappa_1 - \kappa_2|.|v|) + |\kappa_2 - \kappa_1|\Big] \\
&+51,957.|v|.(|\overline{\tau_2} - \tau_1| + |\kappa_2 - \kappa_1|.|v|) \\
+60.&\left(||\gamma_2(0) - \gamma_1(0)|| + |v|.|\overline{\tau_2} - \tau_1| + |\kappa_2 - \kappa_1|.\frac{v^2}{2}\right) \\
&\text{avec} \quad \overline{\tau_2} = \tau_2 - \kappa_2.|v|
\end{aligned}
$$

Construction et recherche dans l'arbre

Une fois la majoration de $\gamma'''(u)$ calculées, nous appliquons l'algorithme récursif afin de trouver les majorants de $\gamma''(u)$ sur chaque intervalle élémentaire. Ces informations sont stockées de la manière suivante :

> **Remarque :** Les majorants et minorants d'un intervalle élémentaire sont calculés et stockés dans la borne supérieure de l'intervalle.

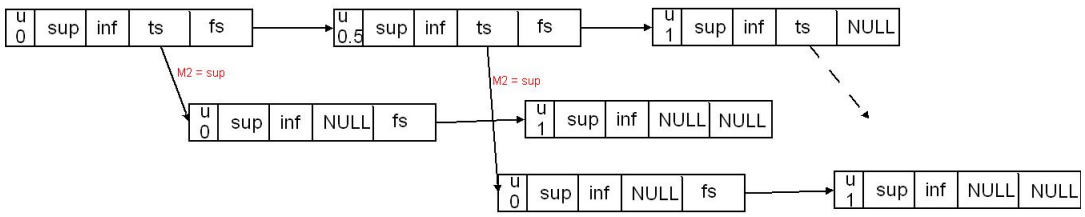


FIG. 3.7 – Exemple d'arbre utilisé pour le stockage des informations

Ou :

- u : la valeur du paramètre
- sup : le majorant de l'intervalle élémentaire
- inf : le minorant de l'intervalle élémentaire

– fs : le fils suivant, la borne d'intervalle élémentaire suivante sinon NULL
– ts : la racine de l'arbre de la dérivée inférieure si celle-ci existe sinon NULL.

L'arbre est donc construit de la manière suivante (Fig. 3.9). Nous traçons les droites de coefficient directeur M_3 et $-M_3$ à partir de la dérivé du point initial et du point final. Nous sauvegardons les points de croisement des droites comme min et max de l'intervalle. Puis nous réitérons l'opération en coupant l'intervalle en 2 et en mettant à jour le min et max. Nous faisons ceci jusqu'à atteindre la taille des intervalles élémentaires choisis (Fig. 3.8).

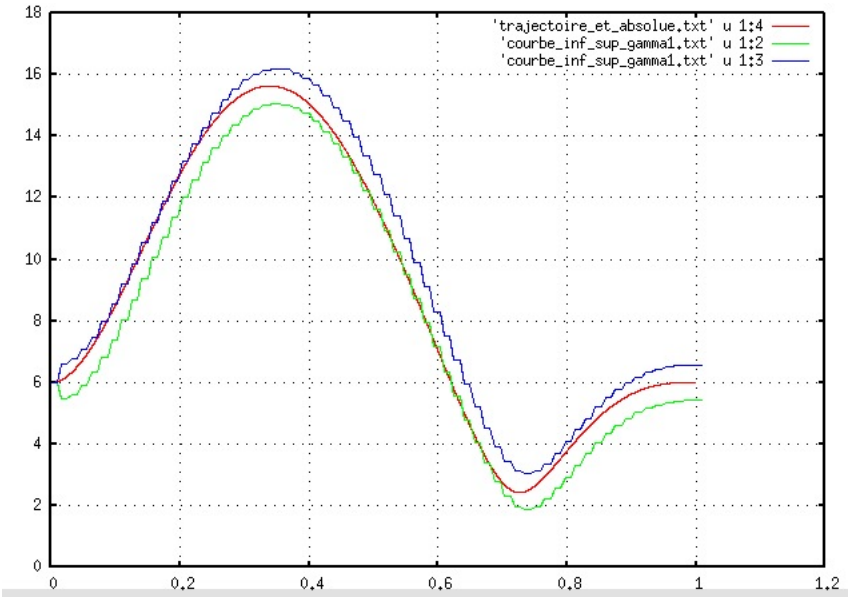


FIG. 3.8 – majorant, minorant par intervalle de $||\gamma'(u)||$

Ensuite nous réitérons l'algorithme pour chaque majorant de $\gamma''(u)$. Nous donnons à la fonction M_2 et le pointeur sur l'arbre suivant *(nt)* qui devient la nouvelle racine. Ceci permet donc de trouver et stocker les majorants et minorants de $\gamma'(u)$.

Finalement pour retrouver les bornes sur un intervalle donné il suffit de parcourir toutes les bornes supérieures de tous les intervalles élémentaires compris dans l'intervalle recherché et d'en sortir le majorant le plus grand et le minorant le plus petit.

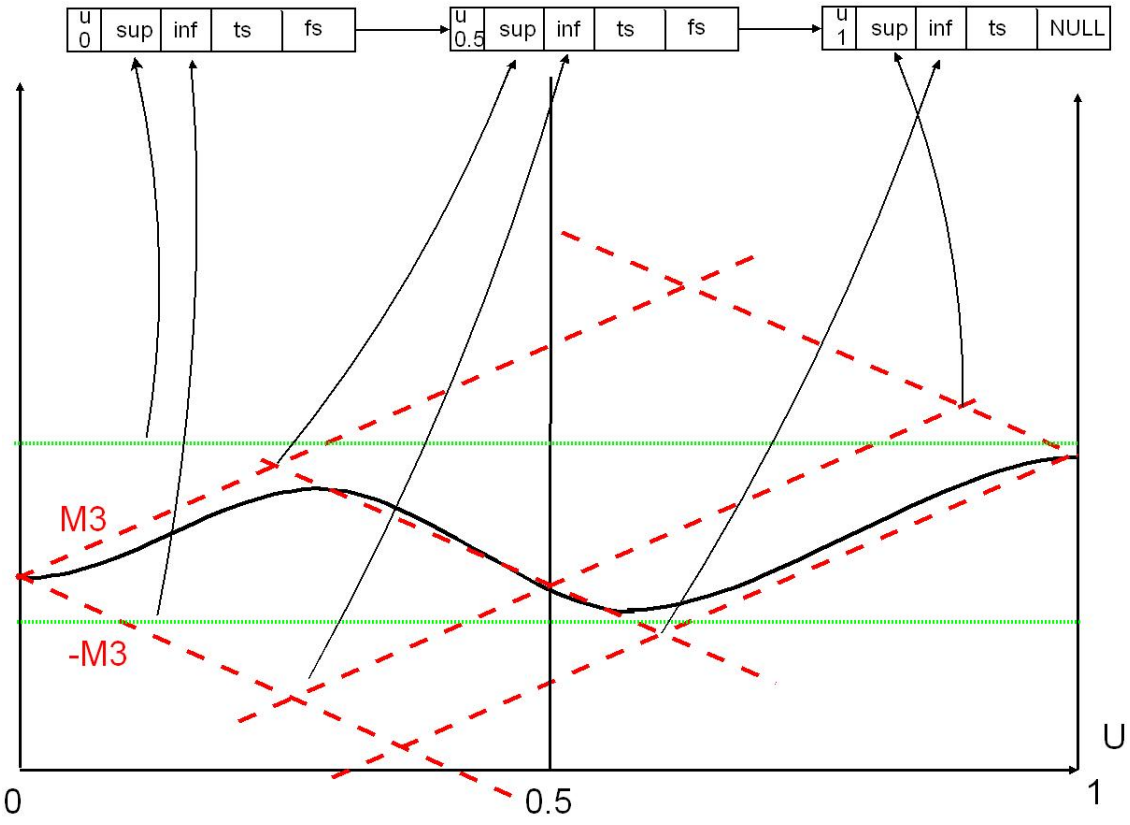


FIG. 3.9 – Construction de l'arbre sur deux itérations

Chapitre 4

Intégration de la plateforme HPP : HumPathPlanner

Le logiciel de planification HumPathPlanner : HPP est en cours de développement dans le but de faire de la planification de mouvement avec la nouvelle plate-forme robotique HRP-2.14. Basé sur le moteur de planification de KineoWorks2, nous y intégrons de nouveaux outils grâce à la librairie HPP. De plus, HPP comprend un module d'interface Corba, afin de pouvoir communiquer avec le robot, ainsi qu'un module d'interface graphique KineoPathPlanner : KPP, afin de visualiser, pendant les phases de développement, les différents outils implémentés. (Fig. 4.1)

Nous avons donc dans un premier temps intégré la méthode locale, décrite dans les chapitres précédent, au moteur de planification KineoWorks2. Il nous reste maintenant à mettre en place les outils des étapes 2 et 3 (Fig. 1.7) : soit transformer la trajectoire initiale de la boite en une pile d'empreintes, et d'intégrer le PatternGenerator afin de récupérer la trajectoire complète de l'humanoïde.

4.1 De la trajectoire simplifiée à la pile d'empreintes

Apres avoir trouvé une trajectoire pour la boite englobante du robot, il nous faut convertir celle-ci en une entrée acceptable par le PatternGenerator. C'est la fonction *computeAStepSequenceFromAPath(pathBox, stepLength, stepSpacing)* qui va s'en occuper. Pour cela il faut lui fournir un chemin dont le vecteur de configuration est (x, y, θ, κ), la taille des pas et l'écartement des pieds souhaité. κ n'est plus utilisé dans cette partie mais est un reste de la planification par la méthode locale basée sur la sortie plate. La fonction renvoie ensuite, en sortie, une pile d'empreinte de pas. Ces empreintes sont décrites de manières relatives les unes par rapport aux autres (Fig. 4.2 et Fig. 4.3) .

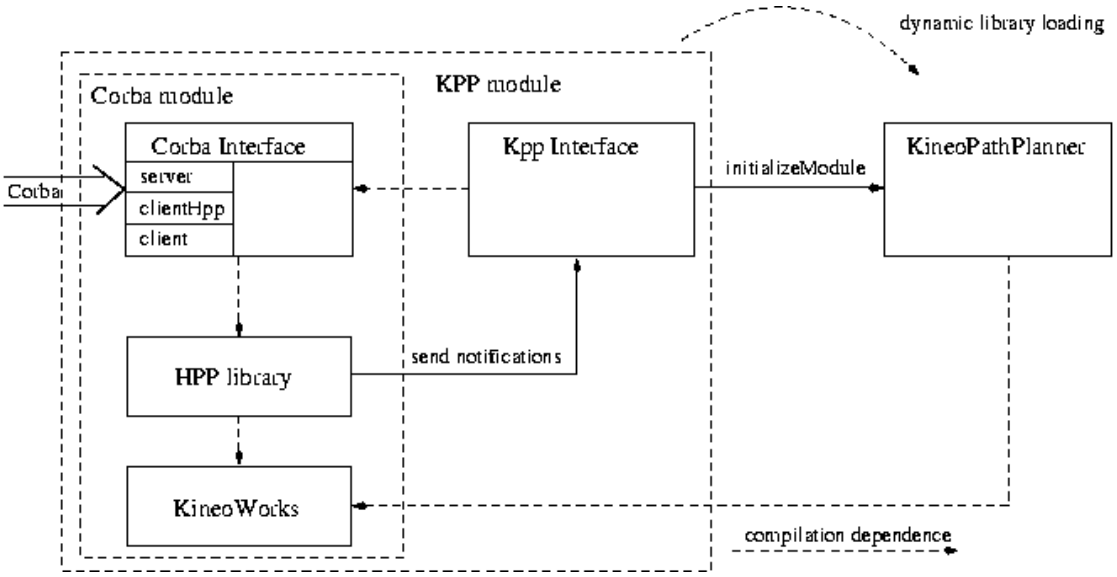


FIG. 4.1 – Intégration de la librairie HPP dans la plateforme de planification humanoïde

Remarque : Nous faisons en sorte que le robot parte et arrive toujours les pieds parallèles. Ceci afin de garantir un maximum de stabilité au départ et à l'arrivée de la trajectoire.

4.2 De la pile d'empreintes à la trajectoire complète du robot : PatternGenerator

Une fois la pile d'empreintes obtenue, nous utilisons la fonction *computeAllQFromAStepSequence()* pour générer à l'aide du PatternGenerator la valeur de tous les degrés de liberté du robot, permettant ainsi d'exécuter la trajectoire que nous venons de lui décrire. En sortie, cette fonction nous renvoie donc une nouvelle trajectoire, celle-ci pour le robot complet et échantillonnée toutes les 5 ms. Elle contient, toutes les valeurs des angles mais aussi la position du ZMP (voir paragraphe suivant) calculé pour le robot, toutes les 5 ms.

Qu'est ce que le ZMP ? Le ZMP[4] ou Zero Moment Point est un critère d'équilibre couramment utilisé et essentiel pour la marche humanoïde. Il représente le point au sol où le moment tangentiel dû à l'inertie et à la gravité est nul. En équilibre, il est la projection du centre de masse (COM) au sol. Le ZMP doit toujours rester dans le polygone convexe formé par les pieds : polygone de sustentation. Si un seul pied est au

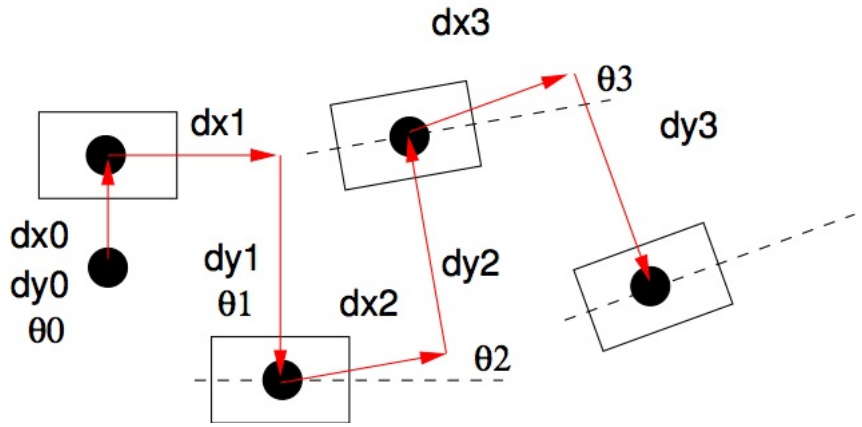


FIG. 4.2 – Positionnement relatif des empreintes les unes par rapport aux autres

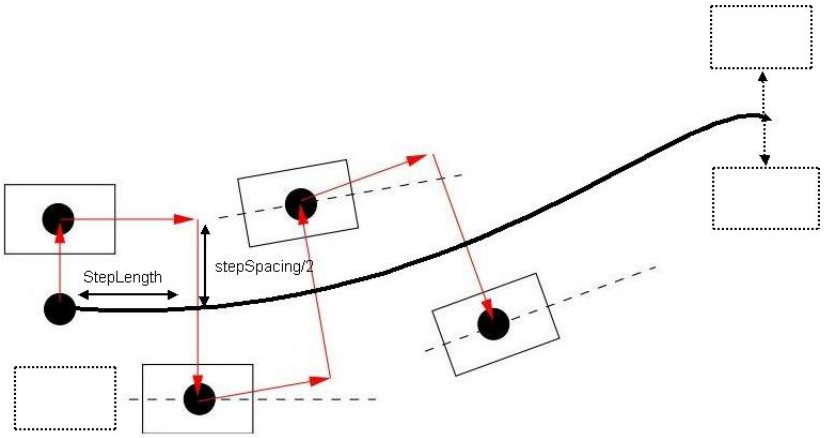


FIG. 4.3 – Placement des empreintes de pas autour d'une trajectoire de boite englobante

sol, le ZMP se situe sous le pied en appui. Si le ZMP sort du polygone, l'équilibre n'est plus assuré et l'humanoïde a de grande chance de tomber.(Fig. 4.4) :

$$p_y = y - \frac{z_c}{g}\ddot{y} \qquad p_x = x - \frac{z_c}{g}\ddot{x} \tag{4.1}$$

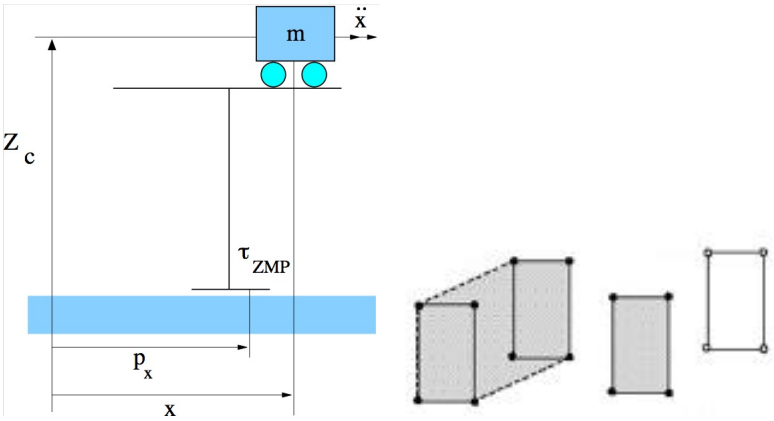


FIG. 4.4 – Représentation du ZMP par un modèle simplifié (Cart Table Model ou pendule inverse) et exemples de polygones de sustentation

Principe du générateur de marche : PatternGenerator. Le PatternGenerator développé au sein du JRL-Japon par Olivier Stasse sur les travaux et papiers de Kajita-san [3] est utilisé ici comme une boîte noire. Mon travail a donc été d'intégrer le PatternGenerator et ses fonctions à la plate-forme HPP.

Le PatternGenerator transforme la pile d'empreintes relatives que nous venons de créer et génère une pile d'empreintes absolues. Ensuite, il en déduit la discrétisation de la trajectoire d'un ZMP de référence.

Ce PatternGenerator utilise une méthode de ZMP Preview Controler [3] (Annexe. B). Cette méthode consiste à calculer la valeur du ZMP en avance, ou dans le futur afin de calculer les erreurs que l'on devrait avoir et les corriger en amont. Le PatternGenerator permet donc de corriger au temps t les futurs erreurs pour que celles-ci ne viennent pas perturber le système. Ceci est aussi utilisé pour anticiper les changements d'appuis du robot et rendre plus lisse le passage du ZMP lors d'une phase de double appuis (Fig. 4.7).

En même temps que la trajectoire réelle du ZMP, le PatternGenerator calcule la trajectoire du centre de masse (COM) qui permet par *cinématique inverse* de calculer les angles des jambes pour une position du COM et une empreinte données. Le reste du corps, ici le haut, n'est pour l'instant pas explicitement pris en compte. Les angles de la partie supérieure du robot suivent une heuristique permettant de réduire les effets du couple qui se produiraient en z si on ne bougeait pas les membres supérieurs.

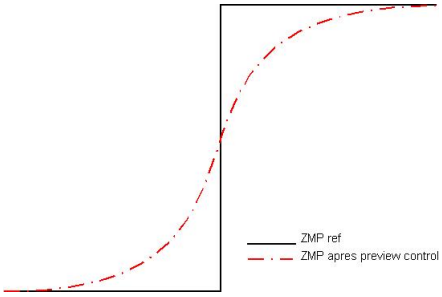


FIG. 4.5 – Modification du ZMP par le Preview Control

4.3 Amélioration

Cependant quelques améliorations ont été ou vont être apportées au PatternGenerator en collaboration avec le JRL-Japon afin que l'on puisse agir sur la partie supérieur du robot et réaliser l'étape 4 de la planification humanoïde (Yoshida-san [7]).

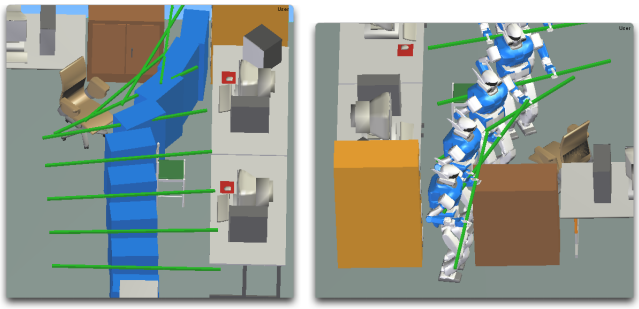


FIG. 4.6 – Modification du ZMP par le Preview Control

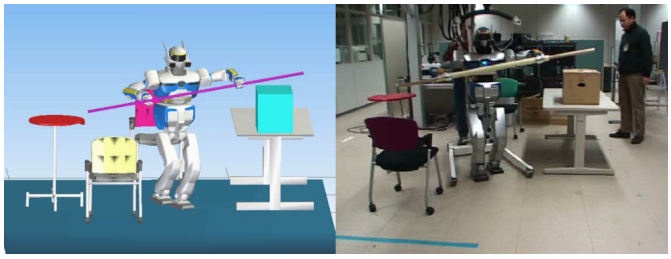


FIG. 4.7 – Modification du ZMP par le Preview Control

Chapitre 5

Expérimentations

5.1 La simulation : KineoWorks2

L'intégration des outils réalisée autour du moteur de planification KineoWorks2. Nous avons donc utilisé, afin de visualiser nos résultats, l'interface graphique fourni par KineoCam : KineoPathPlanner KPP.

KineoWorks2 est à la base un logiciel de planification pour des robots de type bras manipulateur. Il n'est donc pas conçu pour tenir compte de la dynamique des robots mais juste de la cinématique. L'intégration du générateur de marche permet de prendre en compte cette dynamique pour HRP-2.14. Donc toute les trajectoires que nous produisons sont produites en tenant de la dynamique du robot.

Malheureusement, le développement étant toujours en cours et les résultats étant trop récents, la plupart des images sont faites à partir de KineoWorks1. De la même manière, les outils pour l'évitement d'obstacles sont prêts mais l'intégration des obstacles et de l'environnement dans KineoWorks2 et KPP est en cours. Je ne peux donc présenter la totalité des tests dans ce rapport. En espérant avoir suffisamment avancé pour la soutenance (Fig. 5.1, et Fig. 5.2, Fig. 5.3, Fig. 5.4).

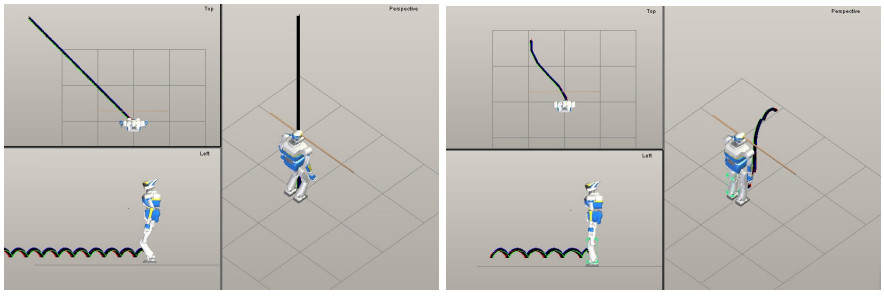


FIG. 5.1 – Trajectoires calculées à partir de la méthode locale - une droite - une coubre 0-0

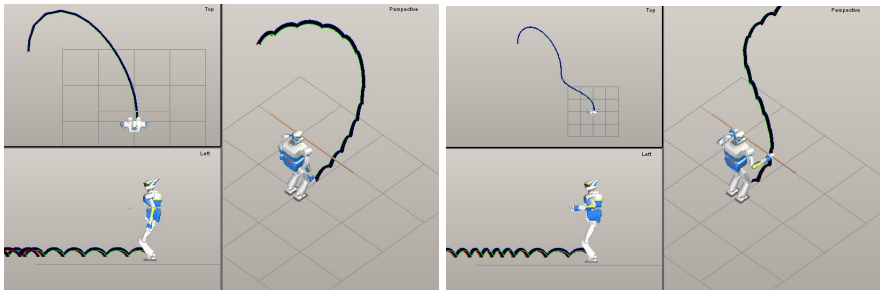


FIG. 5.2 – Trajectoires calculées à partir de la méthode locale - une courbe 0-π - une concaténation de deux courbes

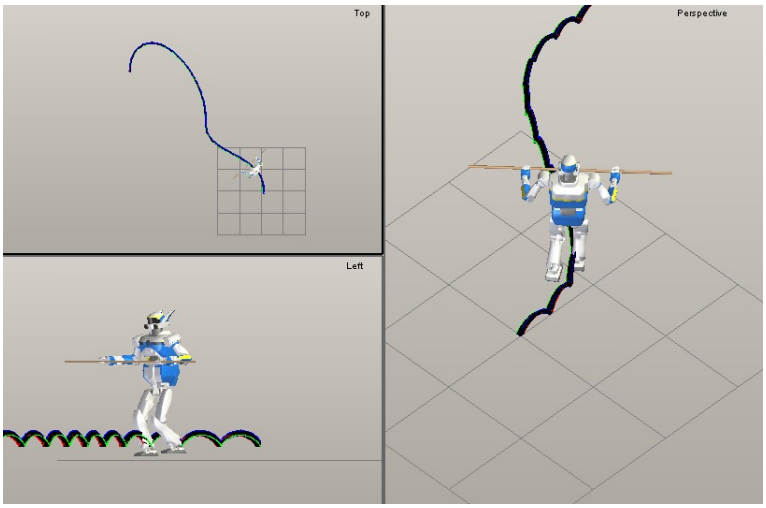


FIG. 5.3 – Exemple d'exécution sur la concaténation

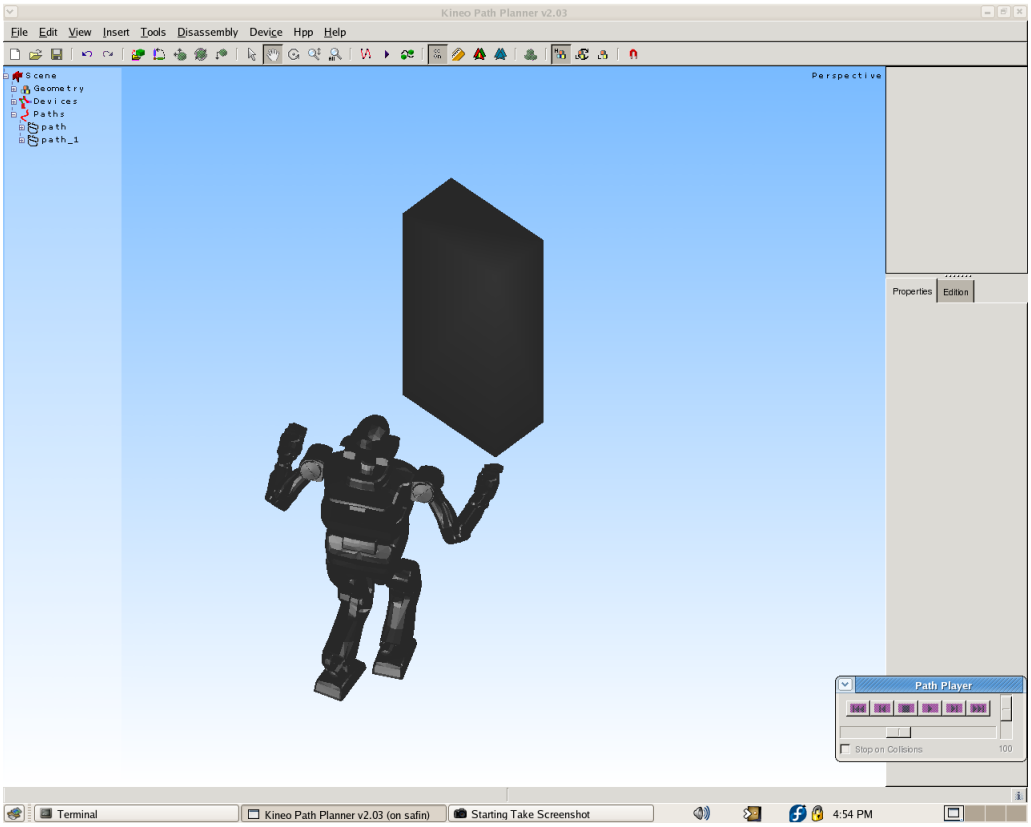


FIG. 5.4 – Exemple d'affichage KineoWorks2 et KPP en cours de développement - HRP 2 et sa boîte englobante. **On remarquera l'absence de couleurs, de visualisation pour la trajectoire ou encore l'abscence d'obstacle.**

5.2 La simulation : OpenHRP

Dans la finalité du projet, HPP communiquera directement avec le robot par server Corba mais étant en phase de développement et par simple mesure de sécurité tout ce qui est transmis à la plate-forme physique est d'abord validé dans le simulateur (cinématique et dynamique) de General Robotix : OpenHRP.

OpenHRP et HRP-2.14 utilisent la même interface Corba, donc les programmes écrits pour l'un fonctionnent sur l'autre. De ce fait, si la simulation s'est correctement effectuée on peut directement envoyer le programme au robot.

Malheureusement, il subsiste encore des erreurs dans la simulation. Des dérives au niveau physique, notamment dûes à la modélisation du contact du pied sur le sol. (Fig. 5.5)

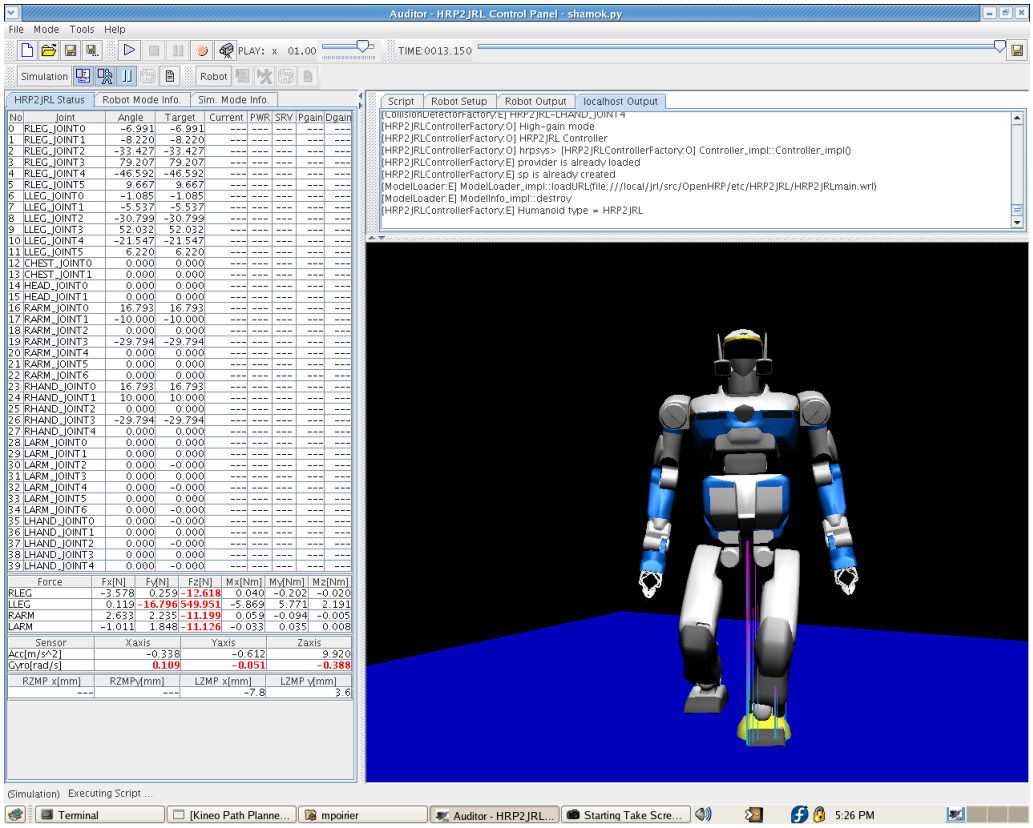


FIG. 5.5 – Exemple d'exécution sur OpenHRP. **Les traits représentent la simulation de la force de réaction au sol par le pied et la boule le calcul du ZMP**

5.3 Essais sur la plate-forme HRP-2.14

Précaution d'emploi Contrairement à ce que l'on pourrait espérer, il ne suffit pas d'appuyer sur un bouton pour mettre en marche le robot. La plate-forme HRP-2.14 reste un outil très fragile qui nécessite une grande attention.

Tous les matins un contrôle des articulations doit être effectué, afin de s'assurer que tout fonctionne correctement. Ensuite, il faut calibrer les capteurs de pression situés sous les pieds. Pour cela il faut poser la plate-forme sur une surface parfaitement plate. Ensuite, durant toute la manipulation, le robot doit être assuré par son lifteur [1] pour éviter toutes chutes malencontreuses. Ceci peut poser des problèmes lors des essais car le lifteur n'est pas pris en compte lors de la planification. Enfin, toutes les manipulations doivent se réaliser à 3 personnes et être filmée par caméra vidéo, pour pouvoir repasser la bande en cas d'accident.(Fig. 5.6)

FIG. 5.6 – HRP-2.14 prêt pour la calibration

Les tout premiers tests Les premiers résultats étant malheureusement trop récents, nous n'avons pas pu expérimenter suffisamment pour faire une bonne analyse des résultats. Etant aussi les premiers pas de HRP-2.14 à Toulouse, les petites mésaventures de débutant sont entrées en jeu pour l'équipe. Cependant nous avons déjà pu remarquer que le robot a tendance à taper un peu le sol en marchant. Nous l'avons donc pris en compte et en collaboration avec le JRL-Japon nous nous efforçons d'améliorer ceci le plus rapidement possible. Beaucoup de tests reste à faire ... cela commence à peine. (Fig. 5.7)

[1] Appareil utilisé dans le milieu hospitalier afin de mettre au lit ou sur fauteuil roulant les personnes ayant une mobilité réduite.

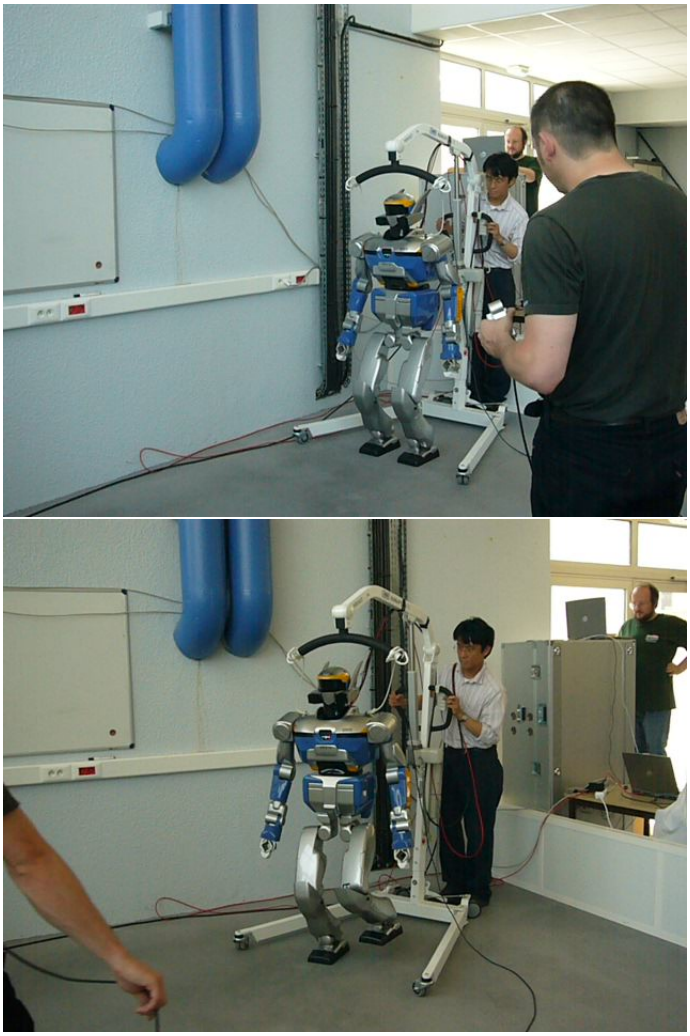


FIG. 5.7 – Les premiers pas de HRP-2.14 réalisant une trajectoire produite par la méthode locale

Chapitre 6

Conclusion

6.1 Synthèse et perspectives

Les différentes étapes de la planification humanoïde n'en sont qu'à leurs débuts et de nombreuses outils restent encore à développer.

La planification de la boite englobante, à l'aide de méthode locale, basée sur la platitude différentielle, issue du monde de la robotique mobile, reste un bon atout. Elle nous permet de simplifier dans un premier temps le problème complexe qu'est la planification de trajectoire pour un système avec un nombre de degrés de liberté important, en apportant son lots d'avantages tel la continuité de la vitesse le long de la courbe.

La transformation de cette trajectoire en une suite d'empreintes acceptables, à la fois par le générateur de marche et la plate-forme HRP-2.14, reste aussi un problème complexe où il faut souvent recourir à des méthodes haddock pour arriver aux résultats escompté. La simulation étant souvent un peu différente de la réalité.

Le générateur de marche ou PatternGenerator est lui aussi en perpétuelle extension. Beaucoup de critère existent pour définir la marche humanoïde, le ZMP n'étant qu'une facette. De grosses avancés ont été réalisées dans ce domaine, et de nombreuses sont encore en cours de développement. En plus du générateur de marche, le groupe s'intéresse fortement à la locomotion humanoïde "whole-Body" ou corps entier [7].

Les différents tests de simulation cinématique et dynamique ainsi que les tests réels sur la plate-forme HRP-2.14, ont permis de concrétiser mon travail. De nombreux problèmes inhérents aux systèmes physiques doivent maintenant être pris en compte afin d'améliorer au mieux nos performances.

En conclusion la plate-forme Humanoïde Path Planner : HPP progresse de jour en jour, la robotique humanoïde, au sein de la recherche française, ne fait que commencer et je pense qu'elle dispose de beaux jours devant elle.

6.2 Enrichissement humain

Ce stage m'a donné l'occasion de m'intégrer au sein d'une équipe travaillant dans un but commun, celui de rendre opérationnel la nouvelle plate-forme robotique HRP-2.14. Ce travail d'équipe a été très important car m'a permis d'être plus à l'aise dans mon travail.

J'ai également eu la possibilité d'assister à plusieurs séminaires sur des thématiques de la robotique humanoïde. Cela constitue un enrichissement important pour ma culture personnelle et scientifique. J'ai ainsi découvert un large panel des recherches menés actuellement en robotique humanoïde dans le monde.

6.3 Conclusion personnelle

De part son contenu scientifique unique et sa diversité, ce stage fut particulièrement enrichissant tant du point de vue technique que du point de vue humain. Il a été pour moi l'occasion de découvrir à la fois le monde de la recherche scientifique et le monde de la robotique humanoïde. Cette expérience de quelque mois n'a fait que confirmer mon souhait de travailler dans le domaine de la recherche en robotique. En espérant que cette expérience puisse continuer, tout d'abord, en thèse : Planification de mouvements et de tâches de manipulation pour systèmes redondants : application à la robotique humanoïde.

Bibliographie

[1] L.E. Kavraki, P. Svestka, J.-C. Latombe & M.H. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," in *IEEE Transaction on Robotics And Automation*, 1996, pp. 566–580.

[2] J.J. Kuffner & S.M. Lavalle. "RRT-Connect : An Efficient Approach to Single-Query Path Planning" in *Proc. IEEE Int. Conf. on Robotics And Automation*, 2000, pp. 995–1001.

[3] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *International Conference on Robotics And Automation, Taipei Taiwan*, September 2003, pp. 1620–1626.

[4] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa, "Resolved momentum control : Humanoid motion planning based on the linear and angular momentum," in *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Las Vegas, Nevada*. IEEE, 2003, pp. 1644–1650.

[5] Florent Lamiraux. *Robots mobiles à remorques : de la planification de chemins à l'exécution de mouvements*. PhD thesis, Institut National Polytechnique de Toulouse, Toulouse, September 1997.

[6] T. Siméon, J. P. Laumond, and F. Lamiraux. Move3d : a generic platform for path planning. In *4th International Symposium on Assembly and Task Planning*, 2001.

[7] E. Yoshida, I. Belousov, C. Esteves & J. P. Laumond. Humanoïde Motion Planning for Dynamic Tasks. In *Humanoïde 2005*, 2005.

[8] T. Siméon & C. Van Geem, "A Pratical collision detector for path planning in factory models," rapport LAAS 2001.

Annexe A

Démonstration de la majoration de $||\gamma'''(u)||$

$$\begin{aligned}\gamma'''(u) = (1-\alpha(u)).\gamma_1'''(u)\\ +\alpha(u).\gamma_2'''(u)\\ +3\alpha'(u).(\gamma_2''(u)-\gamma_1''(u))\\ +3\alpha''(u).(\gamma_2'(u)-\gamma_1'(u))\\ +\alpha'''(u).(\gamma_2(u)-\gamma_1(u))\end{aligned}$$

soit :

$$\begin{aligned}||\gamma'''(u)|| \leq |1-\alpha(u)|.||\gamma_1'''(u)||\\ +|\alpha(u)|.||\gamma_2'''(u)||\\ +|3\alpha'(u)|.||\gamma_2''(u)-\gamma_1''(u)||\\ +|3\alpha''(u)|.||\gamma_2'(u)-\gamma_1'(u)||\\ +|\alpha'''(u)|.||\gamma_2(u)-\gamma_1(u)||\end{aligned}$$

1 - Etude de $\alpha(u)$, $\alpha'(u)$, $\alpha''(u)$, $\alpha'''(u)$ et $\alpha^{(4)}(u)$

$\alpha(u) = 10.u^3 - 15.u^4 + 6u^5$

$\alpha'(u) = 30.u^2.(u-1)^2$

$\alpha''(u) = 60.u.(u-1)(2u-1)$

$\alpha'''(u) = 360(u - 0,211)(u - 0.788)$

$\alpha^{(4)}(u) = -360 + 720.u$

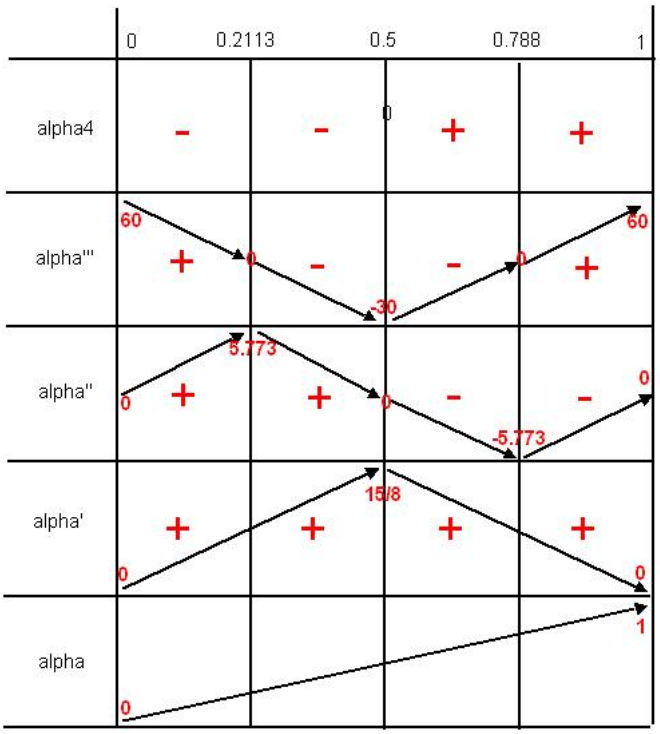


FIG. A.1 – Tableau récapitulatif de l'étude de $\alpha(u)$

2 - Majoration de $||\gamma_1'''(u)||$ et $||\gamma_2'''(u)||$

$\gamma_1(u) = \Gamma(q_1, v.u) = \Gamma_{q1}(v.u)$ pour $q1$ fixe

Rappel $\frac{d}{du} f(g(u)) = g'(u)f'(g(u))$

$\gamma_1'(u) = v.\Gamma_{q1}'(v.u)$

Mais $\Gamma_{q1}'(s) = \vec{u}(\tau_1 + \kappa_1 s)$ avec $\vec{u}(\theta) = \begin{pmatrix} cos(\theta) \\ sin(\theta) \end{pmatrix}$ et $||\vec{u}(\theta)|| = 1$

$\gamma_1'(u) = v.\vec{u}(\tau_1 + \kappa_1.v.u)$

–> $||\gamma_1'(u)|| = |v|$

$\gamma_1''(u) = v^2.\Gamma_{q1}''(v.u)$

Mais $\quad \Gamma_{q1}''(s) = \frac{d}{ds}\vec{u}(\tau_1 + \kappa_1 s) = K_1\vec{u}'(\tau_1 + \kappa_1 s) = K_1\vec{u}(\tau_1 + \kappa_1 s + \frac{\pi}{2})$

Car $\quad \vec{u}'(\theta) = \begin{pmatrix} -sin(\theta) \\ cos(\theta) \end{pmatrix} = \vec{u}(\theta + \frac{\pi}{2})$

$\gamma_1''(u) = v^2.\kappa_1.\vec{u}(\tau_1 + \kappa_1.v.u + \frac{\pi}{2})$

–> $||\gamma_1''(u)|| = |v^2.\kappa_1|$

$\gamma_1'''(u) = v^3.\Gamma_{q1}'''(v.u)$

$\gamma_1'''(u) = v^3.\kappa_1^2.\vec{u}(\tau_1 + \kappa_1.v.u + \pi)$

$\boxed{||\gamma_1'''(u)|| = |v^3.\kappa_1^2|}$

Idem pour $\gamma_2(u)$:

$\boxed{||\gamma_2'''(u)|| = |v^3.\kappa_2^2|}$

3 - Majoration de $(\gamma_2(u) - \gamma_1(u))$

$\gamma_1(u) = \Gamma_{q1}(v.u)$

$$\Gamma_{q1}(s) = \Gamma_{q1}(0) + \int_0^s \Gamma_{q1}'(\sigma)\, d\sigma \quad \text{avec} \quad f(b) = f(a) \int_a^b f'(\sigma)d\sigma$$

$$\text{et} \quad \Gamma_{q1}'(s) = \vec{u}(\tau_0 + \kappa s) \quad \text{donc} \quad \Gamma_{q1}(s) = \Gamma_{q1}(0) + \int_0^s \vec{u}(\tau_1 + \kappa_1\sigma)d\sigma$$

Alors

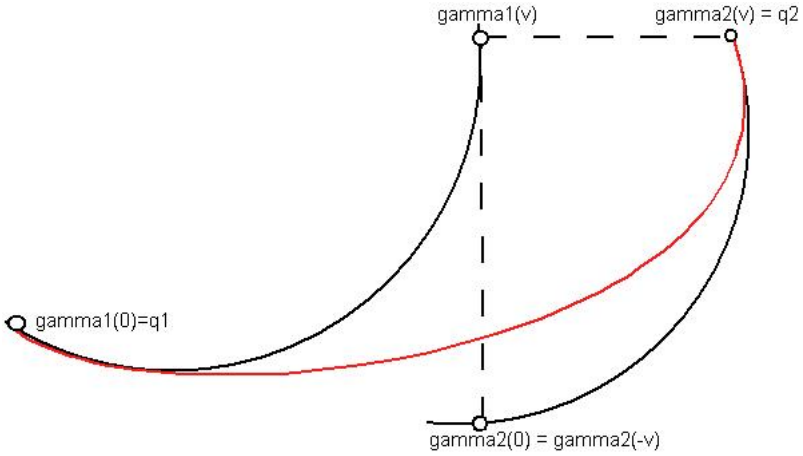
$$\gamma_1(u) = \Gamma_{q1}(v.u) = \Gamma_{q1}(0) + \int_0^{v.u} \Gamma_q'(\sigma)d\sigma$$

$$\gamma_1(u) = \Gamma_{q1}(v.u) = \gamma_1(0) + \int_0^{v.u} \vec{u}(\tau_1 + \kappa_1\sigma)d\sigma$$

Et

$$\gamma_2(u) = \Gamma_{q2}(v.(u-1)) = \Gamma_{q2}(0) + \int_0^{v.(u-1)} \vec{u}(\tau_2 + \kappa_2\sigma)d\sigma$$

$$\gamma_2(u) = \underbrace{\Gamma_{q2}(0) + \int_0^{-v} \vec{u}(\tau_2 + \kappa_2\sigma)d\sigma}_{\Gamma_{q2}(-|v|)=\gamma_2(0)} + \int_{-v}^{v.u-v)} \vec{u}(\tau_2 + \kappa_2\sigma)d\sigma$$



Changement de variable, on pose $\sigma = u - |v|$

$$\gamma_2(u) = \gamma_2(0) + \int_0^{|v|.u} \vec{u}(\tau_2 + \kappa_2(u-|v|)du = \gamma_2(0) + \int_0^{|v|.u} \vec{u}(\tau_2 - \kappa_2.v + \kappa_2.u)du$$

Avec $\Gamma'_{\overline{q2}}(|v|.u)$ et $\overline{q2}(\gamma_2(0), \overline{\tau_2}, \kappa_2)$ et $\overline{\tau_2} = \tau_2 - \kappa_2|v|$

Donc

$$(\gamma_2(u) - \gamma_1(u)) = \gamma_2(0) - \gamma_1(0) + \int_0^{|v|.u)} \vec{u}(\tau_1 + \kappa_1\sigma) - \vec{u}(\overline{\tau_2} + \kappa_2\sigma)d\sigma$$

$$||\gamma_2(u) - \gamma_1(u)|| \leq ||\gamma_2(0) - \gamma_1(0)|| + \int_0^{|v|.u)} ||\vec{u}(\tau_1 + \kappa_1\sigma) - \vec{u}(\overline{\tau_2} + \kappa_2\sigma)||d\sigma$$

$$||\gamma_2(u) - \gamma_1(u)|| \leq ||\gamma_2(0) - \gamma_1(0)|| + \int_0^{|v|.u)} |\overline{\tau_2} - \tau_1| + |\kappa_2 - \kappa_1|\sigma d\sigma$$

$$||\gamma_2(u) - \gamma_1(u)|| \leq ||\gamma_2(0) - \gamma_1(0)|| + |v|.|\overline{\tau_2} - \tau_1|.u + |\kappa_2 - \kappa_1|.\frac{|v|^2.u^2}{2}$$

Alors entre [0, 1]

$$\boxed{||\gamma_2(u) - \gamma_1(u)|| \leq ||\gamma_2(0) - \gamma_1(0)|| + |v|.|\overline{\tau_2} - \tau_1| + |\kappa_2 - \kappa_1|.\frac{|v|^2}{2}}$$

4 - Majoration de $(\gamma_2'(u) - \gamma_1'(u))$

$\gamma_1'(u) = v.\Gamma_{q1}'(v.u) = v.\vec{u}(\tau_1 + \kappa_1.v.u)$

$\gamma_2'(u) = v.\Gamma_{\overline{q2}}'(v.u) = v.\vec{u}(\overline{\tau_2} + \kappa_2.v.u)$

$\gamma_2'(u) - \gamma_1'(u) = v.\,(\vec{u}(\overline{\tau_2} + \kappa_2.v.u) - \vec{u}(\tau_1 + \kappa_1.v.u))$

Alors

$$\boxed{||\gamma_2'(u) - \gamma_1'(u)|| \leq |v|.\,(|\overline{\tau_2} - \tau_1| + |\kappa_2 - \kappa_1|.|v|)}$$

5 - Majoration de $(\gamma_2''(u) - \gamma_1''(u))$

$\gamma_1''(u) = \kappa_1 v^2.\vec{u}(\tau_1 + \kappa_1.v.u + \frac{\pi}{2})$

$\gamma_2''(u) = \kappa_2 v^2.\vec{u}(\overline{\tau_2} + \kappa_2.v.u + \frac{\pi}{2})$

$$(\gamma_2''(u) - \gamma_1''(u)) = v.\Big[\kappa_2(\vec{u}(\overline{\tau_2} + \kappa_2.v.u + \frac{\pi}{2}) - \vec{u}(\tau_1 + \kappa_1.v.u + \frac{\pi}{2})) + (\kappa_2 - \kappa_1).\vec{u}(\tau_1 + \kappa_1.v.u + \frac{\pi}{2})\Big]$$

Alors

$$\boxed{||\gamma_2''(u) - \gamma_1''(u)|| \leq v^2\Big[|\kappa_2|.(|\tau_1 - \overline{\tau_2}| + |\kappa_2 - \kappa_1|.|v|) + |\kappa_2 - \kappa_1|\Big]}$$

En résumé $||\gamma'''(u)||$ sur [0, 1]

$$\begin{aligned} ||\gamma'''(u)|| &\leq |\kappa_1^2 v^3| \\ &+|\kappa_1^2 v^3| \\ &+3.\frac{45}{8}.v^2.\Big[|\kappa_2|.(|\tau_1 - \overline{\tau_2}| + |\kappa_1 - \kappa_2|.|v|) + |\kappa_2 - \kappa_1|\Big] \\ &+51,957.|v|.(|\overline{\tau_2} - \tau_1| + |\kappa_2 - \kappa_1|.|v|) \\ &+60.\Big(||\gamma_2(0) - \gamma_1(0)|| + |v|.|\overline{\tau_2} - \tau_1| + |\kappa_2 - \kappa_1|.\frac{v^2}{2}\Big) \\ &\text{avec} \quad \overline{\tau_2} = \tau_2 - \kappa_2.|v| \end{aligned}$$

Annexe B

Schéma du PatternGenerator du JRL

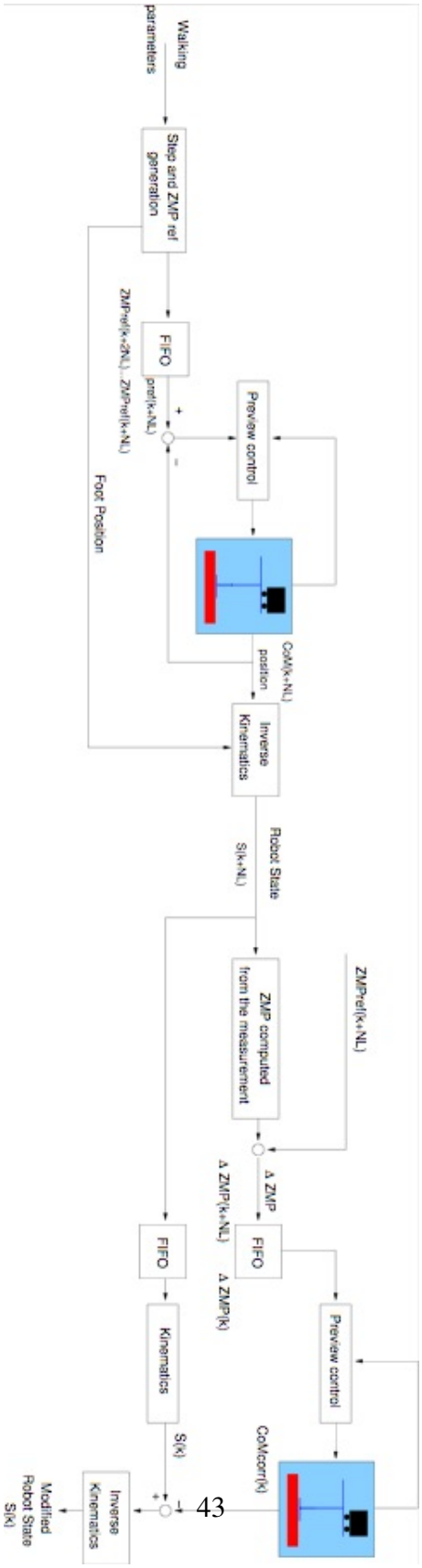


FIG. B.1 – Schéma du PatternGenerator développé a JRL-Japon

Table des figures