ROBOTFOOT: Conception d'une plateforme robotique humanoïde

M. Drapeau, D. Dumont, C. Hébert, M. Labonté, M. Paradis, A. Renquinha-Henri, A. Rioux, M. Tétrault, D. Trépanier

Résumé — Ce document traite du projet RobotFoot qui consiste à concevoir, construire et programmer un robot humanoïde autonome capable d'effectuer une marche bipède stable pour suivre une balle et la botter. Il est entre autres composé d'une caméra pour détecter la balle et de deux jambes comportant chacune six servomoteurs, donc six degrés de liberté par jambes. Ensuite, une courbe de Bézier du 3e degré est utilisée pour déterminer la trajectoire, en orientations et positions, entre le robot et la balle. Les équations du Zero-Moment-Point (ZMP) sont utilisées pour gérer l'équilibre du robot, puis le centre de masse (CoM) est trouvé à l'aide d'un algorithme de contrôle de stabilité par anticipation basé sur l'équation de Riccati. Finalement, une boucle de contrôle permet de contrôler le mouvement de chaque jambe et de rectifier les erreurs sur la position et l'orientation.

***Mots clés*— Robot humanoïdes, ZMP, Bézier 3e degré, Ricatti, marche bipède**

I. INTRODUCTION ET MOTIVATION

Les robots humanoïdes sont de plus en plus utilisés dans divers milieux. Comme leur nom l'indique, les robots humanoïdes sont des robots à ressemblance humaine. Le premier robot de ce genre a été développé par l'université de Waseda au Japon en 1973. Ce robot, nommé WABOT-1, était capable d'effectuer une marche bipède[1]. Plusieurs années plus tard, Honda développe P2, le premier robot humanoïde à pouvoir marcher sur deux jambes de façon autonome[2]. Ce robot fait partie de la série P de Honda et est un précurseur au célèbre modèle ASIMO de la même compagnie[2]. Aujourd'hui, les robots humanoïdes sont capables de marcher sur des surfaces inégales incluant des obstacles et peuvent maintenir leur équilibre même s'ils sont déstabilisés par une force externe[3].

Les robots humanoïdes peuvent être plus appropriés que des robots roulants dans plusieurs situations. En effet, de par leur forme se rapprochant de celle de l'homme, ils sont beaucoup plus adaptés aux environnements destinés aux humains (escaliers, échelles, valves, etc). C'est pourquoi la DARPA (Defense Advanced Research Projects Agency) a lancé le programme DARPA ROBOTICS CHALLENGE[4]. Cette compétition de robotique a pour but de promouvoir le développement de robots humanoïdes pouvant intervenir sur des sites affectés par une catastrophe. Ces robots ont pour but de remplacer les êtres humains sur des sites qui comportent un risque élevé, mais dont l'accès est trop difficile ou l'environnement est mal adapté pour des robots conventionnels.

Une compétition majeure qui promeut le développement de robots humanoïdes est la compétition robotique de soccer RoboCup. Le but ultime de cette initiative est de voir une équipe de robots humanoïdes autonomes jouer au soccer et gagner contre une équipe d'humains professionnels d'ici 2050[5]. Il existe plusieurs ligues classées selon les dimensions des robots, mais nous nous intéressons à la ligue humanoïde de taille enfant.

En effet, notre projet consiste à concevoir une plateforme robotique humanoïde pouvant marcher sur deux jambes de façon autonome sur une surface plane et sans obstacle pour se déplacer dans la direction d'une balle en mouvement. Cette balle sera détectée en temps réel par le robot à l'aide d'une caméra. Une fois la balle à portée du robot, il devra la frapper avec un de ses pieds. Le développement de ce robot a pour but ultime de participer d'ici quelques années à la compétition RoboCup. Étant donné que ce robot pourra être utilisé pour des fins académiques, celui-ci devra être à faible coût, soit environ 1000$ et devra être totalement libre de droit.

II. ÉTAT DE L'ART

Les robots participant à la compétition RoboCup dans la ligue humanoïde varient d'une équipe à l'autre. Certains de ces robots ont leur code source ouvert, soit libre de droits, et peuvent être achetés directement d'un fournisseur. C'est le cas des robots Darwin -OP et Nao.

Nao est un robot humanoïde de 58cm de hauteur développé par la compagnie française Aldebaran[6]. En 2011, la compagnie a publié le code source de Nao et la plateforme est maintenant libre de droits[6]. Le code de Nao utilise une distribution de Linux pour faire rouler son intelligence artificielle NaoQi[6]. Cependant, ce code est relativement lent et utilise beaucoup de ressource, soit plus de 15% du pouvoir du CPU[6]. C'est pourquoi plusieurs équipes, dont UNSW, ont décidé de créer leur propre code plutôt que d'utiliser NaoQi[6]. Nao utilise un CPU Atom Z530 qui peut analyser des images d'une résolution de 640x480 pixels à une vitesse de 30 images par secondes[7].Il est possible de se procurer Nao pour environ 16,000$[6].

Ensuite, un autre robot populaire autant pour les compétitions de robots humanoïdes telles que RoboCup ainsi que pour la recherche est le DARwIn-OP[10-12]. En effet, il a été développé en collaboration avec la compagnie ROBOTIS et plusieurs universités américaines. Il possède maintenant une communauté de laboratoires et d'universités à travers le monde l'utilisant pour leurs recherches[13]. Ceci est dû en bonne partie au fait que son code ainsi que ses plans mécaniques et son manuel de fabrication sont disponibles gratuitement et libre de droit[14]. En utilisant la plateforme DARwIn-OP, une équipe des États-Unis a également gagné deux fois la compétition RoboCup en 2011 et 2012 dans la catégorie Kid Size[15-16]. Ses performances sont comparables à celle du robot Nao présenté plus haut, car il possède lui aussi un Atom Z350 comme ordinateur embarqué et utilise un système d'exploitation Linux. Par contre, son coût s'élève quant à lui à 12,000$[17].

Bien qu'il soit possible de se procurer une plateforme de robot humanoïde pour ainsi n'avoir qu'à se préoccuper du code. La plupart des équipes développent leur propre plateforme robotique. C'est le cas de l'équipe CIT Brains Kid qui s'est classée 2e durant les deux dernières années dans la catégorie humanoïde taille enfant de RoboCup[8]. Ce robot est l'intégration de technologies développées à travers deux institutions. Le Hajime Research Institute au Japon a développé le système de contrôle du robot et le Chiba Institute of Technology a développé son système informatique[9]. Ce robot utilise un système d'exploitation Linux Ubuntu 12.04 LTS et contrôle 18 servomoteurs. Son CPU Atom D525 peut analyser 20 images par secondes d'une résolution de 640x480 pixels.

Une autre équipe fabriquant leur propre plateforme, celle du robot FUmanoids, participe à la compétition RoboCup chaque année depuis 2007. L'ensemble du code et la majorité des pièces mécaniques sont développés directement par le Freie Universität Berlin depuis 2009[18]. À leur première année de participation à la compétition RoboCup, ils arrivèrent en 3e place tout en ayant le robot le plus léger et le moins cher de leur catégorie, soit la catégorie Kid Size. Puis, ils ont atteint systématiquement la 2e position en 2008, 2009 et 2010[18]. Bien que le code soit disponible sur leur site internet[19] et qu'un article présente les différentes pièces mécaniques utilisées, aucun plan n'est fourni pour pouvoir reproduire ce robot ou ses pièces faites sur mesure[20]. En ce qui concerne la puissance de calcul, le FUmanoids possède un CPU Exynos4412 Quad-core ARM Cortex-A9 sur cadencé à 1.7 GHz.

Ces différents exemples comportent tous des avantages et désavantages bien distincts, mais aucun ne respecte complètement les contraintes de notre projet. En effet, les robots tels que Nao et DARwIn-OP, qui sont libres de droit et vendus par des compagnies, sont beaucoup trop dispendieux. De l'autre côté, les projets de recherche tels que CIT Brains et FUmaonids sont plus abordables, mais il n'est pas facile, voire impossible, d'obtenir les plans, les pièces, le code ou la documentation de ces robots.

III. DESCRIPTION DU SYSTÈME

La modélisation SolidWorks du squelette du robot ainsi que la disposition des différents servomoteurs dans celui-ci sont représentées dans la figure 1. On peut y voir que trois servomoteurs contrôlent l'articulation de la hanche, un pour le genou et deux pour la cheville, pour un total de 6 degrés de liberté par jambe.

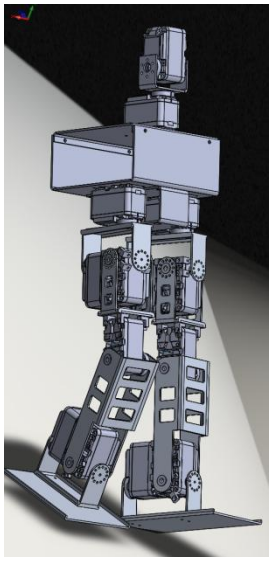


Figure 1: Squelette du robot

L'architecture du système de RobotFoot se divise en trois sections: la tête, le torse et les jambes. La tête comporte une caméra ainsi que deux servomoteurs permettant de la diriger. Le torse contient les différents capteurs ainsi que l'intelligence du robot. On y retrouve le système d'alimentation, le gyroscope, l'accéléromètre, l'ordinateur embarqué et la carte de contrôle. Les jambes, quant à elles, contiennent chacune six servomoteurs permettant de générer une marche à six degrés de liberté. Les différentes composantes de ces sections ainsi que leurs connexions sont représentées à la figure 2.

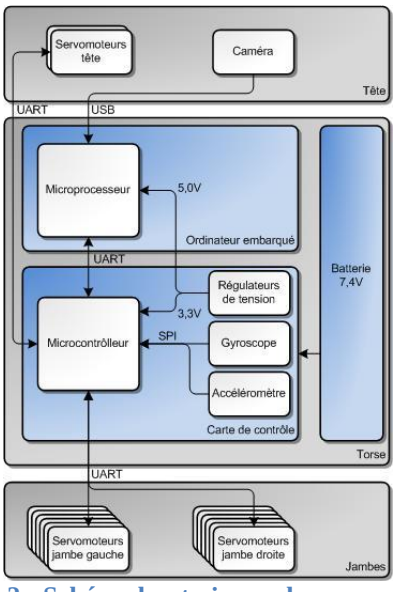


Figure 2 : Schéma haut niveau des composantes électroniques

Le tableau 1 liste les différentes pièces utilisées par le robot. Le facteur le plus déterminant pour leur sélection a été le prix, puisqu'un prix total environnant les 1000$ a été fixé. La caméra devait être suffisamment rapide pour bien suivre un mouvement tout en ayant une résolution assez faible pour être rapidement traité. Le type de batterie ainsi que sa puissance ont été déterminés selon les besoins des autres pièces et la carte de contrôle a dû être faite au complet sur un PCB afin de répondre à notre restriction de forme et d'espace à l'intérieur du torse. L'ordinateur embarqué a été sélectionné pour son bon rapport performance prix et est supérieur à la plupart des autres robots mentionnés plus haut en termes de puissance de calcul. Finalement, des servomoteurs intelligents plus dispendieux, mais très facile d'utilisation, ayant un torque élevé et une grande précision ont été favorisés à des servomoteurs moins chers pour les raisons qu'y ont été énumérés.

Tableau 1 : Choix des pièces

Composante	Choix
Caméra	Caméra Web USB 640 x 480 pixels, 30 fps
Batterie	Lithium Polymère Li-Po 7,4 V
Carte de contrôle	PCB fait sur mesure à partir du microcontrôleur STM32F4
Ordinateur embarqué	i.MX Quick Start Board, ARM Cortex-A8 1 GHz
Servomoteurs	Servomoteurs numériques intelligents DRS-0101

Au niveau logiciel, les tâches sont réparties entre l'ordinateur embarqué et la carte de contrôle. La carte de contrôle communique directement avec les moteurs pour les contrôler. L'ordinateur embarqué contient quant à lui les algorithmes de marche qui seront décrits plus loin. Il communique avec la caméra par une connexion USB et contient un algorithme de traitement d'image ainsi qu'un algorithme responsable du choix des comportements du robot. Une architecture multithreads a été implémentée pour que ces deux dernières tâches, la marche et la vue, soient traitées en parallèle et de façon indépendante. Les deux cartes communiquent entre elles de façon synchronisée pour exécuter les actions requises à l'aide d'une connexion USB. La répartition des modules entre les deux cartes est représentée dans la figure 3.

En ce qui concerne le traitement d'image pour reconnaitre une balle en mouvement, étant donné la quantité élevée de pixels à analyser pour chaque image et que le traitement d'images en temps réel demande une grande utilisation de mémoire vive, une bibliothèque logicielle graphique dont l'utilisation de ressources est limitée doit donc être utilisée. OpenCV, qui est une bibliothèque graphique spécialisée dans le traitement d'images en temps réel, libre de droits et dont le code est ouvert, a été choisie après une comparaison avec d'autres bibliothèques semblables telle que SimpleCV, CImg et libCVD. Grâce à la nature bas niveau de cette bibliothèque, constituée de fonctions écrites en C, son utilisation de ressources est faible, avoisinant les 25 Mo lors d'un test sur un ordinateur personnel. Toutefois, l'allocation et la libération de la mémoire ne sont pas effectuées automatiquement et doivent être gérées par les développeurs afin d'éviter les fuites de mémoires qui pourraient s'accumuler et causer des ralentissements et des plantages.

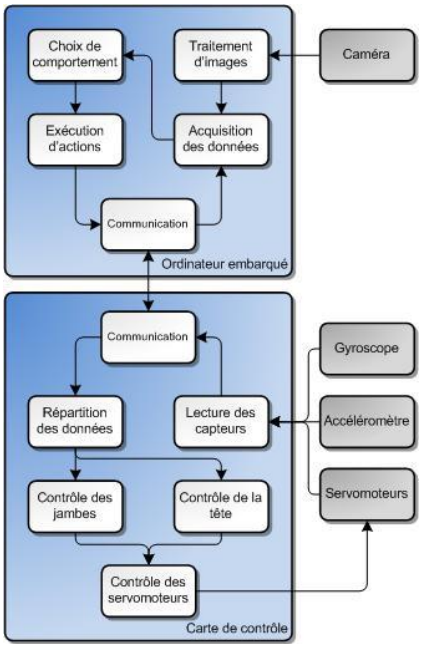


Figure 3: Schéma haut niveau des modules logiciels

Le défi majeur de notre projet consiste à concevoir un algorithme pour gérer la marche bipède du robot. Tout d'abord, il faut définir deux points dans l'espace: un premier référentiel qui représente la position et l'orientation initiale du robot et un second représentant la position de la balle par rapport au bassin du robot. Cette position est obtenue à l'aide de l'algorithme de traitement d'image.

Il faut ensuite générer une trajectoire entre la position actuelle du robot et la position actuelle de la balle. Dans la plupart des cas, le robot ne fera pas directement face à la balle. Plutôt que de faire pivoter le robot sur lui-même pour l'orienter dans la direction de la balle et le faire se diriger ensuite en ligne droite vers cette dernière, nous utilisons une courbe de Bézier cubique pour générer une trajectoire entre le bassin du robot et la balle. Une courbe de Bézier cubique est la courbe B(t) définie par les points de contrôle P_0, P_1, P_2 et P_3.[21]

$$B(t) = (1-t)^3 P_o + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3, \quad t \in [0,1]$$

L'étape suivante consiste à générer les empreintes de pas qui permettront au bassin du robot de suivre la courbe de Bézier générée lors de la marche. Plus précisément, ces empreintes représentent la position des pieds de chaque côté de la courbe qui permettront au robot de marcher de façon à ce que la projection de la trajectoire nominale du bassin sur le sol suive la courbe de Bézier (voir figure 4).

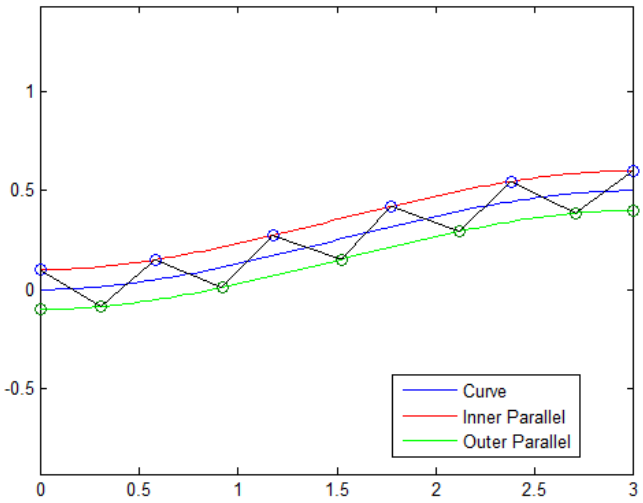


Figure 4: Trajectoire avec Bézier de troisième degré avec empreintes et la trajectoire du ZMP

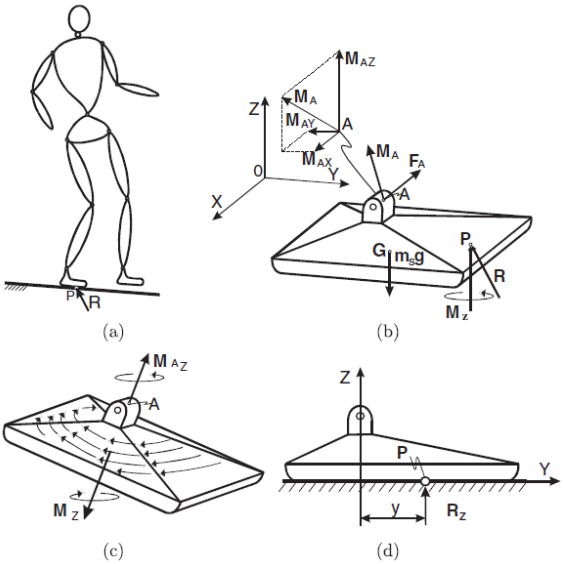


Figure 5: Forces et moments sur le pied du robot

Ensuite, la figure 5 représente les forces et moments appliqués sur la structure mécanique du pied du robot. Les sommes des forces et des moments agissants sur l'articulation de la cheville, soit le point A, sont représentées respectivement par F_A et M_A. Le point P représente le point où les forces R (la force de réaction du sol) et les moments M s'appliquent. Le *Zero-Moment-Point* (ZMP) est défini comme étant le point P sur la semelle à l'intérieur des limites du pied (polygone de support du pied) où les moments en X et en Y sont nuls, soit $M_X = M_Y = 0$[22]. En d'autres mots, le ZMP a été défini comme suis par le Dr. Miomir Vukobratovic, ingénieur mécanique et pionnier de la robotique humanoïde:

"*All the time the reaction of the ground due to the foot resting on it can be reduced to the force* ***R*** *and vertical component of the moment Mz; the point P at which the reaction force is acting represents ZMP*."[22]

Une traduction française de ce texte serait: « En tout temps où la réaction du sol sur le pied causé par son contact avec celui-ci peut être réduite à la force R et la composante verticale du moment Mz; le point P sur lequel la force de réaction agit représente le ZMP ».

Pour qu'un robot bipède reste en équilibre sur un seul pied, le ZMP doit absolument rester à l'intérieur du polygone de support de ce pied. Afin que cette condition soit vraie, les équations d'équilibre statique suivantes doivent être respectées.

$$R + F_A + m_s g = 0$$
$$\overrightarrow{OP} \times R + \overrightarrow{OG} \times m_s g + M_A + M_Z + \overrightarrow{OA} \times F_A = 0$$

où $\overrightarrow{OP}$, $\overrightarrow{OG}$ et $\overrightarrow{OA}$ sont les vecteurs de rayons entre l'origine du système de coordonnées O_{XYZ} et le point où les forces de réaction agissent (P), le centre de masse du pied (G), et le joint de la cheville (A), respectivement, et m_s est la masse du pied. M_Z représente la composante verticale du moment M.

$$M_Z = M_{friction} = -(M_A^Z + (\overrightarrow{OA})^Z)$$

Même si dans la plupart des cas le moment en Z n'est pas égal à zéro, on suppose que la friction entre les pieds et le sol est suffisamment grande pour éviter une rotation autour de l'axe des Z. En projetant maintenant l'équation (2) sur le plan horizontal (le sol), on obtient:

$$(\overrightarrow{OP} \times R)^H + \overrightarrow{OG} \times m_s g + M_A + M_Z + (\overrightarrow{OA} \times F_A)^H = 0$$

Cette équation permet ensuite de calculer la position du point P, qui représente alors un candidat au ZMP.

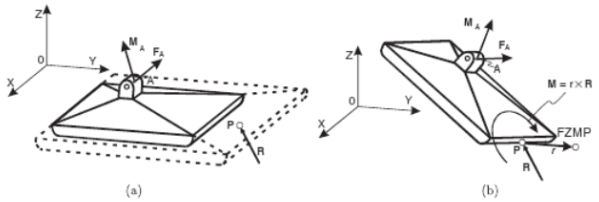


Figure 6: Point candidat P se situant à l'intérieur du polygone de support (a) et à l'extérieur (b)

Dans le cas où le point candidat P se situe à l'intérieur du polygone de support comme à la figure 6(a), il s'agit alors du ZMP recherché. Par contre, si ce point se situe à l'extérieur du support comme à la figure 6(b), on parle alors plutôt d'un FZMP où *Fictitious Zero-Moment Point*. Puisque les forces et moments au point P ne peuvent exister à l'extérieur du support du pied, avoir un FZMP se traduit en réalité par le point P se situant sur la limite du pied. Pour compenser l'erreur de position entre ce point P réel et le FZMP, un moment $M = r \times R$ est engendré en ce point, de grandeur proportionnelle à cette erreur. Le robot est alors en déséquilibre et va tomber si aucune action n'est prise.

En supposant que la hauteur Z du bassin ne varie pas ($z = z_c$ = constant) pendant la marche, le robot pourrait être modélisé

par un pendule inversé «Carte-table Model» dont les équations de ZMP sont :

$$p_x = x - \frac{z_c}{g}\ddot{x}$$

$$p_y = y - \frac{z_c}{g}\ddot{y}$$

Ensuite, lorsque le robot se tient sur ses deux pieds, le ZMP se situe entre ceux-ci. Avant d'effectuer un premier pas, on déplace le ZMP sous le pied, en bougeant le centre de masse (CoM) du robot au-dessus du pied, qui demeurera en contact avec le sol lors du pas. Lorsque le second pied aura terminé sa trajectoire à la fin du pas et sera de retour à plat sur le sol, il y aura à nouveau une transition du ZMP et du CoM vers la position de ce pied et ainsi de suite tel que représenté par le trait vert à la figure 4. On appelle ce type déplacement une marche statique, car le robot n'est pas en mouvement lors du déplacement du centre de masse et est complètement soutenu par ses deux pieds. Les effets dynamiques et moments créés peuvent alors être négligés.

Afin d'obtenir un déplacement plus rapide et plus fluide, il est important que le CoM ne suivre pas exactement le ZMP. Pour ce faire, le CoM est calculé par un algorithme de contrôle de stabilité par anticipation qui utilise le ZMP ainsi que les contraintes physiques du robot. L'équation principale utilisée dans cet algorithme est l'équation algébrique de Riccati à temps discret. Elle permet de trouver une solution au problème de régulateur linéaire-quadratique avec horizon infini et invariant dans le temps. En d'autres mots, cela permet de faire bouger le CoM en se basant sur plusieurs échantillons futurs prévus.

$$X = A^T XA - (A^T XB)(R + B^T XB)^{-1}(B^T XA) + Q$$

X est la matrice inconnue n par n symétrique et A, B, Q, R sont des matrices connus de coefficients réels basé sur les dimensions et contraintes physique du robot.

L'étape suivante pour implémenter la marche du robot consiste à contrôler le mouvement des jambes. Les 12 servomoteurs des jambes sont utilisés en chaîne. Dans la première partie de la chaîne, la jambe fixe sert à placer le CoM à l'endroit calculé précédemment à l'aide du ZMP. Dans la seconde partie, la jambe en mouvement commande le déplacement du pied, à partir de la nouvelle position du CoM. La chaine complète s'inversera à chaque pas, selon quel pied est le repère fixe et quel pied est en mouvement. Dans tous les cas, il est nécessaire de générer une trajectoire pour le chaque pied et le pelvis suivant le CoM.

Pour ce faire, il faut d'abord représenter chaque jambe du robot par ses paramètres Denavit-Hartenberg[23] (DH). Afin de caractériser la position relative de deux solides avec seulement quatre paramètres, au lieu de six, il faut suivre plusieurs règles précises en définissant les repères des joints.

1. L'axe $\overrightarrow{z_n - 1}$ est porté par l'axe de la liaison reliant le corps C_{n-1}au corps C_n. ($\overrightarrow{z_n}$ est donc porté par l'axe de la liaison reliant le corps C_n au corps C_{n+1}).
2. L'axe $\overrightarrow{x_n}$est porté par la normale commune à $\overrightarrow{z_n - 1}$ et $\overrightarrow{z_n}$ soit : $\overrightarrow{x_n} = \overrightarrow{z_{n-1}} \Lambda \overrightarrow{z_n}$.
3. L'axe $\overrightarrow{y_n}$ est choisi de manière à former un trièdre direct avec l'axe $\overrightarrow{z_n}$ et $\overrightarrow{x_n}$ soit : $\vec{y} = \overrightarrow{z_n} \Lambda \overrightarrow{x_n}$.

Lorsque les repères du système sont bien posés, on peut trouver les paramètres DH de la façon suivante:

a, la distance le long de l'axe $\overrightarrow{x_n - 1}$ entre les axes $\overrightarrow{z_n - 1}$ et $\overrightarrow{z_n}$. C'est donc également la longueur de la normale commune.

α, l'angle autour de l'axe $\overrightarrow{x_n - 1}$ entre les axes $\overrightarrow{z_n - 1}$ et $\overrightarrow{z_n}$.

d, la distance le long de l'axe $\overrightarrow{z_n}$ entre les axes $\overrightarrow{x_n - 1}$ et $\overrightarrow{x_n}$. C'est la variable articulaire dans le cas glissière.

θ, l'angle entre autour de l'axe $\overrightarrow{z_n}$ entre les axes $\overrightarrow{x_n - 1}$ et $\overrightarrow{x_n}$. C'est la variable articulaire dans le cas pivot.

Cette représentation nous permet d'obtenir facilement une matrice jacobienne définissant la position et l'orientation du pied par rapport à la hanche du robot. Sur la figure 7, les matrices $J_1(q)$ et $J_2(q)$ représentent respectivement la matrice jacobienne de position et la matrice jacobienne d'orientation. Les matrices $J_1^{\#}(q)$ et $J_2^{\#}(q)$ sont les pseudo-inverses avec amortissement des matrices $J_1(q)$ et $J_2(q)$.

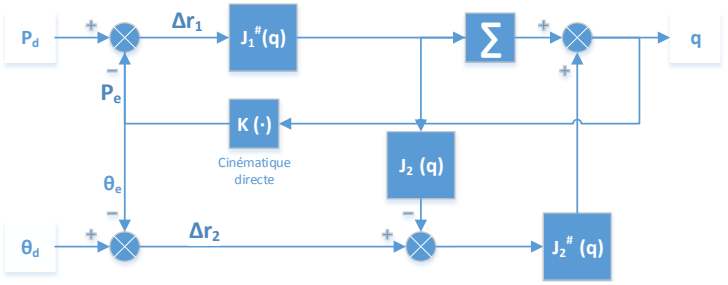


Figure 7: Architecture de contrôle

Il est ensuite nécessaire de définir des équations pour générer la trajectoire du pied avec une interpolation cubique:

$$a_0 = q_i$$
$$a_1 = \dot{q}_i$$
$$a_3 t_f^3 + a_2 t_f^2 + a_1 t_f + a_0 = q_f$$
$$3a_3 t_f^2 + 2a_2 t_f + a_1 = \dot{q}_f$$

où $q_i, q_f, \dot{q}_i, \dot{q}_f$ représentent respectivement la position initiale du pied, sa position finale, sa vitesse initiale et sa vitesse finale.

On obtient alors la position désirée P_d en calculant la position finale q_f. Cette position est mise à jour continuellement pour générer le mouvement du pied dans la boucle de contrôle de la figure 7. La position réelle P_e est calculée continuellement en

mettant à jour la matrice homogène. Celle-ci est obtenue à l'aide de la cinématique directe[23] K(·). La matrice homogène permet aussi de mettre à jour l'orientation du pied θe. On définit l'orientation du pied désirée θd comme 0 en x, y et z pour maintenir le pied parallèle au sol. Finalement, la boucle de contrôle de la figure 7 permettant de générer un pas peut se résumer par les équations suivantes:

$$\dot{q} = J_1^{\#}\dot{r}_1 + J_2^{\#}(\dot{r}_2 - J_2 J_1^{\#}\dot{r}_1) \equiv \Delta q$$
$$= J_1^{\#}\Delta r_1 + J_2^{\#}(\Delta r_2 - J_2 J_1^{\#}\Delta r_1)$$

Sachant que :

$$\Delta r_1 = P_{désiré} - P_{end\ effector}$$
$$\Delta r_2 = \theta_{désiré} - \theta_{end\ effector}$$

Et:

$$J^{\#} = J^T(JJ^T + k^2 I)^{-1}$$

où J^T est la matrice jacobienne transposée. L'ensemble des calculs de cette boucle de contrôle est implémenté de façon logicielle dans le code de notre robot.

IV. RÉSULTATS, VALIDATION ET TESTS

Finalement, le robot a été construit selon les plans initiaux et est maintenant utilisé pour tester les algorithmes de marches, de contrôle et de traitement d'images. Les algorithmes peuvent être testés soit directement avec du code C++ mis dans le robot ou encore à l'aide d'une interface implémentée pour pouvoir faire déplacer le robot avec une entrée pré-calculée en simulation. La première technique est celle qui sera utilisée pour son fonctionnement normal, alors que la seconde est plutôt utile pour tester rapidement les équations développées théoriquement sous MatLab. Le robot complet et assemblé est montré à la figure 8 suivante.

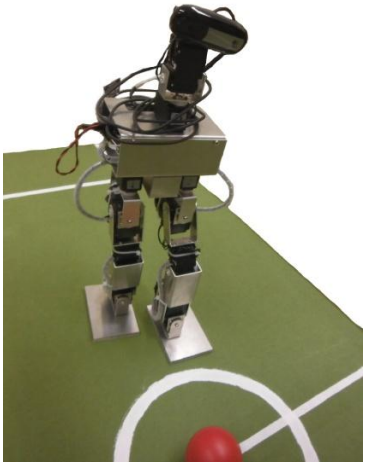


Figure 8: Robot complet et assemblé

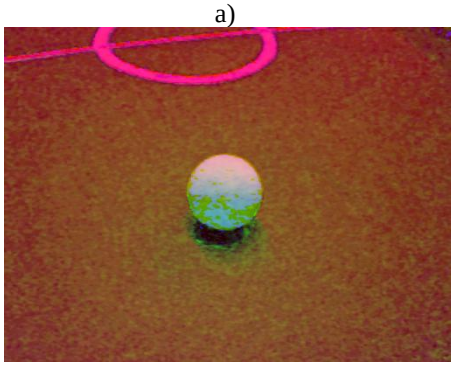
Ensuite, un des objectifs principaux du projet est qu'il soit reproductible à faible coup. Le tableau 2 détaille le prix de reproduction d'un exemplaire. Bien que le coup total soit légèrement au delà du prix ciblé au départ, soit environ 1000$, il reste tout de même très abordable lorsque comparé aux autres robots humanoïde précédemment énumérés. Aussi, ces prix sont ce qui a été payé pour créer un seul exemplaire et serait donc moindre dans le cas où le robot serait produit en plus grande quantité.

Tableau 2: Prix total d'un exemplaire

	Prix	Quantité	Totaux
Dépenses			**1 201,08 $**
Tête			
Caméra	22,88 $	1	22,88 $
Servomoteur	37,50 $	2	75,00 $
Torse			
Batterie	31,00 $	1	31,00 $
CPU	149,99 $	1	149,99 $
Carte controleur			
Micro-controlleur	7,16 $	1	7,16 $
PCB	41,31 $	1	41,31 $
Composants (résistances, condensateurs, etc.)	2,50 $	5	12,50 $
Régulateur 5V	1,19 $	3	3,57 $
Gyroscope/Accéléromètre	0,00 $	1	0,00 $
Jambe			
Servomoteur	37,50 $	12	450,00 $
Autres			
Fabrication des pièces	388,00 $	1	388,00 $
Poids	9,00 $	1	9,00 $
Lecteur USB<->SD	10,67 $	1	10,67 $
	Prix	Quantité	Totaux

Des tests unitaires sur le module de traitement d'images ont également été effectués pour identifier une couleur particulière dans l'environnement, comme à la figure 9, et déterminer la position de cette couleur dans l'image. Puis, à partir de la cette position et des dimensions du robot, il est possible de déterminer la position d'une balle dans l'espace, puis par rapport à la position du robot en utilisant simplement le théorème de Pythagore.

a)



b)

c)

Figure 9: Détection d'une balle rouge dans l'environnement. a) image non traité b) première phase de traitement c) isolation final

Des simulations MatLab ont été générées pour représenter le mouvement d'une jambe lorsque celle-ci effectue un pas. La figure 10 représente la modélisation physique de la jambe, à des temps différents. Les cercles rouges représentent les articulations rotatives de la jambe et le premier point à (0,0) représente la hanche du robot. Le point vert, quant à lui, représente son pied.

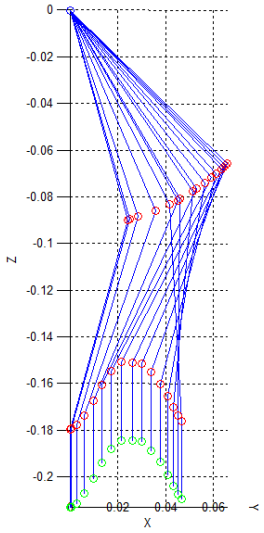


Figure 10: Trajectoire de la jambe lors d'un pas

Nous avons ensuite simulé dans MatLab une courbe de Bézier de degré 3 ainsi que les empreintes de pas du robot sur des courbes parallèles montrées à la figure 11.

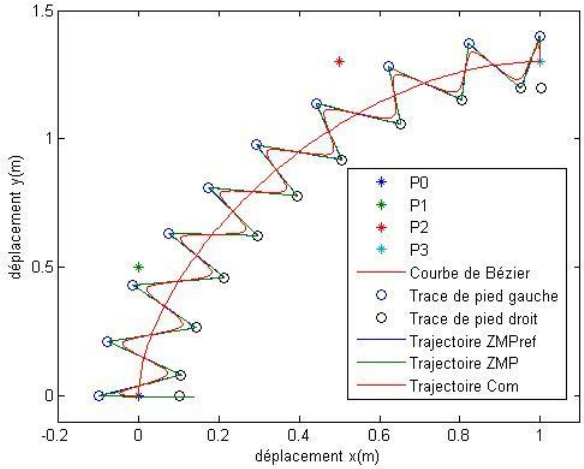
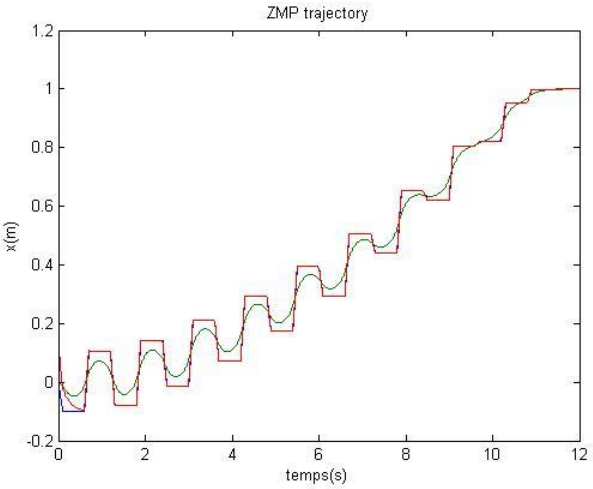
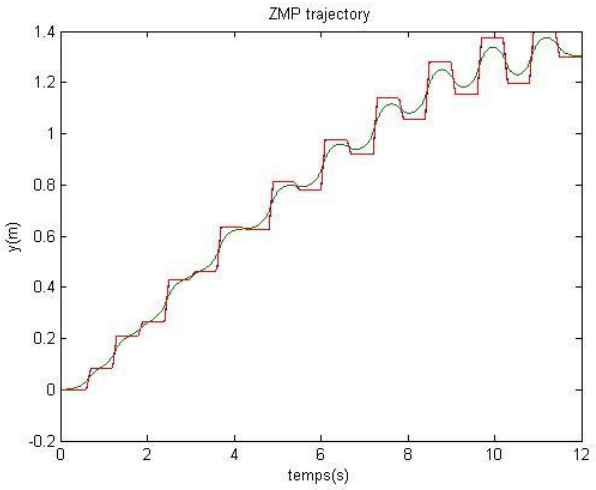


Figure 11: Génération des empreintes de pas du robot autour de la courbe de Bézier

On peut voir sur cette figure la courbe de Bézier de troisième degré représentant la trajectoire désirée du robot en bleu, la trajectoire du pied gauche en rouge et la trajectoire du pied droit en vert. Les cercles représentés tout au long de ces deux courbes externes représentent les empreintes de pas de chaque pied. La ligne noire reliant les empreintes de pas en zigzag représente le déplacement du ZMP lors de la transition entre chaque pas. Finalement, la courbe rouge suivant de près le ZMP est la trajectoire du centre de masse. Ce déplacement est représenté en X et en Y à travers le temps dans la figure 12(a) et la figure 12(b) respectivement:



a)



b)

Figure 12: Trajectoire du ZMP (rouge) et trajectoire du CoM (vert) lors de la marche a) en x et b) en y

Ces simulations nous permettent de déterminer la trajectoire du robot lorsqu'il suit une balle. Mises ensemble, ces trajectoires permettent de connaitre à tout moment la position désirée des pieds et du pelvis dans l'espace. En utilisant une architecture de contrôle des moteurs qui tente de continuellement minimiser l'erreur entre la position réelle et la position théorique, le robot suit de manière stable ces courbes tant qu'il n'est pas dérangé par une perturbation externe trop importante.

Le robot est donc maintenant capable de marcher de manière autonome et stable. Il peut également frapper la balle avec son pied lorsqu'elle est suffisamment proche et ensuite suivre son mouvement avec la caméra pour aller la botter de nouveau. Il cherche également à se positionner de façon à la frapper en direction du but, qui est marqué par une couleur particulière que le robot reconnaît comme étant son but. Malheureusement, il est susceptible aux débalancements causés par des forces externes, car la gestion de l'équilibre avec le gyroscope et l'accéléromètre n'a pas pu être implémentée, faute de temps et de ressources.

V. CONCLUSION ET TRAVAIL FUTUR

En conclusion, le projet consistant à concevoir, fabriquer et programmer un robot humanoïde pour qu'il effectue une marche bipède stable et se dirige vers une balle de couleur pour la frapper avec le pied est un succès. En effet, les résultats montrent que le traitement d'image fait permet bien de trouver une balle dans l'espace et de trouver sa position relativement à celle du robot dans le but de ci diriger. Aussi, les modèles théoriques et les simulations sous MatLab sont très précis quant aux positions désirées calculées pour guider le robot dans ses déplacements. Ensuite, il a été vérifié qu'il est possible de faire une marche bipède stable lorsque l'environnement est contrôlé et que le robot ne reçoit pas de bouleversement externe (sol en pente, obstacle, poussé, etc.) Finalement, le prix visé pour un exemplaire est légèrement plus cher que prévu, mais tout de même très abordable lorsque l'on compare à d'autres robots comparables.

Dans l'avenir, afin d'obtenir un coup encore moindre que ce a été présenté, plusieurs options sont disponibles. En effet, dans le cas où il est possible d'avoir accès à une imprimante 3D de qualité, imprimé, le squelette du robot avec une telle imprimante réduira de manière significative le prix, car le squelette constitue la plus grosse dépense du robot. Également, en fabriquant plusieurs exemplaires, il serait possible d'avoir des rabais en achetant en grande quantité, ce qui réduirait encore plus le prix final.

Afin d'améliorer la marche du robot et la rendre encore plus stable et robuste, le premier point à ajouter serait un algorithme de contrôle utilisant le gyroscope et l'accéléromètre du robot pour prendre en compte les forces extérieures et imprévues. Cette partie a été longuement considérée par notre équipe, mais un manque de temps et de ressource nous a forcés à renoncer à cette amélioration.

VI. RÉFÉRENCES

[1] Humanoid Robotics Institute, Waseda University. [En ligne] Disponible: http://www.humanoid.waseda.ac.jp/booklet/kato_2.html

[2] Honda Motor Co. [En ligne] Disponible: http://world.honda.com/ASIMO/history/p1_p2_p3.html

[3] Boston Dynamics. [En ligne] Disponible: http://www.bostondynamics.com/robot_petman.html

[4] DARPA. [En ligne] Disponible: http://www.darpa.mil/Our_Work/TTO/Programs/DARPA_Robotics_Challenge.aspx

[5] Auteur inconnu. [En ligne] Disponible: http://www.tzi.de/humanoid/bin/view/Website/WebHome

[6] COLLIEN, D., HUYNH, G. (2008), “From AIBO to NAO. The transition from 4Legged to 2Legged Robot Soccer” [En ligne] Disponible: http://www.cse.unsw.edu.au/~RoboCup/2008site/reports/DavidGaryThesis.pdf

[7] Aldebaran-Robotics. [En ligne] Disponible: https://developer.aldebaran-robotics.com/nao/

[8] Auteur inconnu. [En ligne] Disponible: http://wiki.RoboCup.org/wiki/Humanoid_League

[9] HAYASHIBARA, Y et al. “CIT Brains (Kid Size League)” [En ligne] Disponible: http://application2013.germanteam.org/upload/1defe33d991e5a91e269de315afdadc6955f2ae9/CITBrainsKid2013.pdf

[10] Compétition ICRA. [En ligne] Disponible: http://icra2012.org/program/robotChallenge.php

[11] Compétition RoboCup. [En ligne] Disponible: http://www.robocup2013.org/

[12] Compétition FIRA. [En ligne] Disponible: http://fira.net/?document_srl=3776#0

[13] Communauté DARwIn-OP. [En ligne] Disponible: http://www.robotsource.org/xe/

[14] Code source et manuel de fabrication. [En ligne] Disponible: http://sourceforge.net/projects/darwinop/

[15] Final RoboCup 2011. [En ligne] Disponible: http://news.cnet.com/8301-17938_105-20078777-1/u.s-droids-carry-the-day-at-2011-robocup-finals/

[16] Final RoboCup 2012. [En ligne] Disponible: http://romelarobocup.blogspot.mx/2012/06/team-darwin-repeats-win-at-robocup-in.html

[17] Achat en ligne de DARwIn-OP. [En ligne] Disponible: http://www.trossenrobotics.com/p/darwin-OP-Deluxe-humanoid-robot.aspx

[18] Historique et information de FUmanoide. [En ligne] Disponible: http://www.fumanoids.de/about/

[19] Code de FUmanoide. [En ligne] Disponible: http://www.fumanoids.de/code/

[20] Description générale de FUmanoide. [En ligne] Disponible: http://www.fumanoids.de/robots/

[21] Courbes de Bézier. [En ligne] Disponible: http://www.wolframalpha.com/input/?i=bezier+curve

[22] Vukobratovic, B. Borovac, *Zero-Moment Point - Thirty Five Years Of Its Life.* International Journal of Humanoid Robotics, Vol. 1, No. 1 (2004) 157–173

[23] B.Siciliano, L. Sciavicco, L. Villani, G. Oriolo, *Robotics Modelling, Planning and Control.* London:Springer-Verlag, 2009, pp. 58-68.